

UNIVERSIDADE DO PORTO
FACULDADE DE ENGENHARIA

Extracção Automática de Conhecimento a partir dos
Ficheiros de Incidentes dos Centros de Condução da
Rede Eléctrica Nacional

TEMPUS, uma ferramenta de Aprendizagem Automática

Fernando Jorge Ferreira Duarte

Março de 2000



**FACULDADE DE ENGENHARIA DA
UNIVERSIDADE DO PORTO**

**Departamento de Engenharia Electrotécnica e de
Computadores**

**Extracção Automática de Conhecimento a partir dos
Ficheiros de Incidentes dos Centros de Condução
da Rede Eléctrica Nacional**

**TEMPUS, uma ferramenta de Aprendizagem
Automática**

Fernando Jorge Ferreira Duarte

**Licenciado em Informática de Gestão pela Universidade
Portugalense**

**Dissertação submetida para satisfação parcial dos requisitos do grau de mestre em
Engenharia Electrotécnica e de Computadores**

(Área de especialização de Informática Industrial)

Dissertação realizada sob supervisão da

Professora Doutora Zita Maria Almeida do Vale

**Do Departamento de Engenharia Electrotécnica do Instituto Superior de Engenharia
do Porto**

Porto, Março de 2000

UNIVERSIDADE DO PORTO	
Faculdade de Engenharia	
BIBLIOTECA M	
N.º	51686
CDU	681.3(043)
da	8, 11, 20 00

Resumo

A Aprendizagem Automática é inerentemente um campo complexo de investigação e desenvolvimento, essencialmente multidisciplinar, envolvendo métodos provenientes da estatística, da inteligência artificial, da teoria da informação, da teoria de controlo, da teoria da complexidade computacional, da biologia, da ciência cognitiva, da filosofia e da psicologia.

A quantidade de dados disponíveis para análise é cada vez maior, resultado do seu arquivo sistemático e da sua geração por simuladores. Para fazer face a esta explosão de dados, a Aprendizagem Automática disponibiliza abordagens sistemáticas, que possibilitam uma interpretação mais fácil desses dados, apoiada nas potencialidades informáticas.

Nas duas últimas décadas, a Aprendizagem Automática registou progressos em várias direcções, tanto em termos teóricos como na sua aplicação em diversas áreas. Provavelmente, a razão principal desta evolução reside no aumento das capacidades computacionais, que proporcionam a aplicação de algoritmos de Aprendizagem Automática que necessitam de grandes recursos computacionais.

Hoje em dia, os métodos de Aprendizagem Automática são utilizados em Sistemas de Energia, em diagnósticos médicos, no reconhecimento de caracteres, no processamento de imagens, nas aplicações financeiras, no marketing, entre outras aplicações.

A sua aplicação a problemas de Engenharia, tais como os existentes nos Sistemas Eléctricos de Energia, tem-se revelado bastante promissora.

O SPARSE é um Sistema Baseado em Conhecimento para apoiar os operadores dos Centros de Controlo e Condução da Rede Nacional de Transporte de Energia Eléctrica na análise de incidentes e na reposição de serviço.

Os objectivos principais deste trabalho consistiram em apresentar um subconjunto representativo dos métodos de Aprendizagem Automática existentes, na concepção e desenvolvimento de uma ferramenta de Extração Automática de Conhecimento que fosse capaz de confirmar ou sugerir modificações ao Conhecimento já existente na Base de Conhecimento do Sistema Pericial SPARSE e na construção de um Editor de Regras que possibilitasse a alteração da Base de Regras do SPARSE. A ferramenta de Aprendizagem Automática tem a denominação TEMPUS, enquanto o Editor de Regras tem a denominação EDIREGRA.

O TEMPUS foi aplicado, com sucesso, ao sistema pericial SPARSE, tendo-se chegado a resultados que mostram efectivamente a necessidade e utilidade desta ferramenta.

Por sua vez, o EDIREGRA possibilitou a alteração dos tempos associados a algumas premissas das regras constituintes da Base de Regras do SPARSE e a criação de novas regras.

Abstract

Machine Learning is a complex field of research and development, essentially multidisciplinary, involving methods from statistics, artificial intelligence, information theory, control theory, computational complexity theory, biology, cognitive science, philosophy, and psychology.

The amount of available data for analysis is continuously increasing, as a result of its systematic storing and its generation by simulators. In order to face this explosion of data, Machine Learning offers systematic approaches giving an easier interpretation of these data, supported by computing power.

On the last two decades, Machine Learning had a significative evolution, in terms of theoretical understanding and actual applications in diverse areas. Probably the main reason of this evolution resides in the increase in the computing power, which enables the application of Machine Learning algorithms that need considerable computational resources.

Nowadays, Machine Learning methods are used in power systems, medical diagnosis, character recognition, image processing, financial applications, marketing, among other applications.

Its application to engineer problems, such as those existing in Power Systems, has revealed promising.

SPARSE is a Knowledge Base System for assisting Control Centre of the Portuguese Transmission Electrical Network in incident analysis and power restoration.

This work presented a representative subset of Machine Learning methods. A Knowledge Automatic Extraction tool was developed to confirm or suggest modifications in the Knowledge Base of SPARSE and in the construction of a Rules Editor, which could permit the alteration of the Rules Base of SPARSE. A Rule Editor module was developed. The Machine Learning tool is named TEMPUS, while the Rule Editor is named EDIREGRA.

TEMPUS was applied with success, to the expert system SPARSE. The results obtained so far show the need and utility of this tool.

On the other hand, EDIREGRA became possible the alteration of times associated to some premises of the SPARSE rules as well as the creation of new rules.

Résumé

L'Apprentissage Automatique est un champ complexe de recherche et de développement, surtout multidisciplinaire, avec des méthodes provenant de la statistique, de l'intelligence artificielle, de la théorie de l'information, de la théorie de contrôle, de la biologie, de la complexité de l'ordinateur, de la biologie, de la science cognitive, de la philosophie et de la psychologie.

Les données disponibles pour analyser sont de plus en plus nombreuses, grâce à leur classement systématique et à sa génération par simulateurs. Pour envisager cette explosion de données, l'Apprentissage Automatique offre des approches systématiques, permettant une interprétation plus facile de ces données-là, fondée sur les potentialités informatiques.

Pendant les deux dernières décades, l'Apprentissage Automatique a fait des progrès dans différents domaines soit théoriquement, soit au niveau de sa mise en place. Probablement, cette évolution est due au développement des capacités de l'ordinateur qui permettent l'utilisation d'algorithmes d'Apprentissage Automatique exigeant de grands moyens informatiques.

De nos jours, les méthodes d'Apprentissage Automatique sont utilisées dans les systèmes d'énergie, les diagnostics médicales, la reconnaissance des symboles, la construction d'images, les applications financières, le marketing, entre autres.

Son application dans les problèmes de Génie, tels que ceux des systèmes électriques d'énergie, semble assez promettante.

SPARSE est un Système Basé sur la Connaissance pour aider les opérateurs des Centres de Contrôle et Conduite du Réseau National de Transport d'Energie Electrique dans l'analyse d'incidents et dans le remplacement de service.

Les objectifs principaux de ce travail sont de présenter un ensemble représentatif de méthodes d'Apprentissage Automatique et de concevoir et développer un outil d'Extraction Automatique de connaissance, capable de confirmer ou de suggérer des changements dans la Connaissance qui existe déjà dans la Base de Connaissance du Système Expert SPARSE et dans la construction d'un Editeur de Règles qui permette la transformation de la Base de Règles du SPARSE. L'outil de l'Apprentissage Automatique s'appelle TEMPUS, et l'éditeur de règles s'appelle EDIREGRA.

TEMPUS a été appliqué, avec du succès, au système expert SPARSE, et les résultats montrent le besoin et l'utilité de cet outil.

EDIREGRA a permis le changement des temps relatifs à quelques prémisses des règles qui composent la Base de Règles du SPARSE et la création de nouvelles règles.

Prefácio

A realização deste trabalho foi acompanhada por alguns problemas de saúde e contrariedades de ordem pessoal.

No entanto, tive a sorte de ter tido o apoio de algumas pessoas, sem as quais, o resultado final deste trabalho não seria o mesmo.

De todos aqueles que me ajudaram gostaria de começar por agradecer à minha família pela compreensão e atenção que me dispensaram. Não posso, no entanto, deixar de sublinhar a dedicação da minha mãe, assim como algum trabalho de introdução de texto realizado pela minha irmã Virgínia.

À minha orientadora, a Professora Doutora Zita Vale, pela orientação, correcções, competência demonstrada e, muito especialmente, pela compreensão e amizade manifestada, o meu muito obrigado.

Ao meu co-orientador Professor Doutor Carlos Ramos, que em conjunto com a minha orientadora, demonstrou competência e percebeu a dimensão humana que este trabalho teve.

Ao Eng.º António Ferreira da Silva, que na fase inicial do trabalho, disponibilizou-me material para consolidar os meus conhecimentos em Prolog, e inclusive, apoiou-me na fase inicial de implementação.

A todos os elementos do centro CIM do ISEP que de alguma forma me ajudaram, nomeadamente, Avelino Marinho, Emanuel Silva, Sérgio Moreira, Luís Faria, Jorge Santos, Paulo Oliveira e Paulo Sousa. Um agradecimento muito especial é devido ao Nuno Malheiro pelos conhecimentos transmitidos e pelo apoio e disponibilidade dispensada na resolução de alguns problemas que surgiram durante a realização deste trabalho.

À Cristina Bastos que, na Faculdade de Engenharia da Universidade do Porto, me dedicou toda a atenção, resolveu problemas e demonstrou grande sensibilidade para o meu caso particular.

Ao Director da Faculdade de Engenharia da Universidade do Porto, Professor Doutor Marques dos Santos, aos ex-Directores do mestrado em Engenharia Electrotécnica e Computadores, Professor Doutor Raul Vidal, e Professor Doutor Borges Gouveia, pela compreensão e aceitação do pedido de adiamento para entrega desta tese de dissertação.

À minha amiga Paula Sá por toda a disponibilidade e preocupação demonstradas, nomeadamente na ajuda prestada na correcção gramatical da tese.

À minha amiga Paula Pacheco e ao meu amigo Rui Santos pelo incentivo manifestado e por estarem sempre presentes em alturas complicadas por que passei nos últimos tempos.

Ao meu amigo Raul Teles pelo apoio e compreensão demonstrados.

Gostaria, também de agradecer ao Centro CIM do ISEP, na pessoa do Professor Doutor Carlos Ramos, pela disponibilidade de equipamento, essencial à realização deste trabalho.

Agradeço ao ISEP o facto de ter apoiado a minha candidatura a uma bolsa PRODEP para a realização da presente tese e ao PRODEP que financiou as minhas propinas.

O meu intuito é referir todos quantos me ajudaram, mas caso isso não aconteça, gostaria desde já pedir as minhas desculpas.

Porto, Março de 2000

Índice

RESUMO.....	III
ABSTRACT.....	V
RÉSUMÉ.....	VII
PREFÁCIO.....	IX
ÍNDICE.....	XI
ÍNDICE DE FIGURAS.....	XV
ÍNDICE DE TABELAS.....	XVII
NOTAÇÃO UTILIZADA.....	XVII
CAPÍTULO 1.....	1
Introdução.....	1
1.1 Considerações Gerais.....	1
1.2 Guia de Leitura da Tese.....	2
1.3 Motivações do Trabalho.....	4
CAPÍTULO 2.....	5
Aprendizagem Indutiva.....	5
2.1 Introdução.....	5
2.2 Aprendizagem de Conceitos e Ordenação Geral-para-Específica.....	6
2.2.1 Introdução.....	6
2.2.2 A Tarefa de Aprendizagem de Conceitos.....	6
2.2.2.1 Notação.....	6
2.2.2.2 Exemplo ilustrativo.....	7
2.2.3 A Aprendizagem de Conceitos como Pesquisa.....	8
2.2.3.1 Ordenação Geral-para-Específica de Hipóteses.....	8
2.2.4 Algoritmo “FIND-S”: Procura da Hipótese Mais Específica.....	9
2.2.5 “Version Spaces” e Algoritmo “Candidate-Elimination”.....	11
2.2.5.1 Algoritmo “List-Then-Eliminate”.....	12
2.2.5.2 O Algoritmo de Aprendizagem “Candidate-Elimination”.....	13
2.2.5.3 Um exemplo ilustrativo.....	14
2.2.6 Observações sobre os “Versions Spaces” e o Algoritmo “Candidate-Elimination”.....	16
2.2.7 Limitação Indutiva.....	17
2.3 Aprendizagem Baseada em Árvores de Decisão.....	19
2.3.1 Introdução.....	19
2.3.2 Representação das Árvores de Decisão.....	19
2.3.3 Problemas Apropriados a Árvores de Decisão.....	20
2.3.4 Algoritmo de Aprendizagem de Árvores de Decisão ID3.....	20
2.3.4.1 Um exemplo ilustrativo.....	23

2.3.5 Pesquisa no Espaço de Hipóteses da Aprendizagem de Árvores de Decisão	26
2.3.6 Limitação Indutiva na Aprendizagem de Árvores de Decisão	27
2.3.7 Sobreajustamento dos Dados.....	28
2.3.7.1 Redução do Erro com a Eliminação de Ramos	29
2.3.7.2 Regra Pós-Eliminação de Ramos	29
2.3.8 Incorporação de Atributos com Valores Contínuos.....	29
2.3.9 Medidas Alternativas para Seleccionar Atributos.....	30
2.3.10 Tratamento de Exemplos de Treino com Falta de Valores nos Atributos	31
2.3.11 Tratamento de Exemplos de Treino com Custos Diferentes.....	31
2.4 Redes Neurais	33
2.4.1 Introdução.....	33
2.4.2 Representações de Redes Neurais.....	33
2.4.3 Problemas Apropriados para Aprendizagem de Redes Neurais	35
2.4.4 “Perceptrons”.....	35
2.4.4.1 Poder de Representação dos “Perceptrons”	36
2.4.4.2 Regra de Treino do “Perceptron”	37
2.4.4.3 Descida da Encosta	38
2.4.4.4 Descida Incremental da Encosta.....	40
2.4.5 Redes Multinível e o Algoritmo Backpropagation	41
2.4.5.1 Pesquisa do Espaço de Hipóteses e Limitação Indutiva.....	45
2.4.5.2 Representação do Nível Escondido.....	45
2.4.5.3 Generalização, Sobreajustamento e Critério de Fim	45
2.5 Aprendizagem Bayesiana.....	47
2.5.1 Introdução.....	47
2.5.2 Teorema de Bayes	47
2.5.3 Algoritmo de Aprendizagem da Hipótese Posterior Máxima (h_{PM}).....	48
2.5.4 A Hipótese de Probabilidade Máxima e a Hipótese do Menor Quadrado do Erro	51
2.5.5 Hipótese de Probabilidade Máxima para Predizer Probabilidades	53
2.5.6 Classificação Óptima de Bayes	54
2.5.7 Algoritmo Gibbs.....	55
2.5.8 Classificação Simples de Bayes.....	56
2.5.8.1 Um exemplo ilustrativo.....	57
2.5.8.1.1 Estimativa de Probabilidades.....	58
2.6 Aprendizagem Baseada em Instâncias.....	59
2.6.1 Introdução.....	59
2.6.2 Aprendizagem Baseada nos K-Vizinhos Mais Próximos	59
2.6.2.1 Algoritmo da Distância Pesada dos K-Vizinhos Mais Próximos.....	61
2.6.2.2 Aproximações de Funções Alvo Reais Localmente Pesadas.....	63
2.6.2.2.1 Aproximação de Funções Alvo Lineares e Reais Localmente Pesadas.....	63
2.7 Algoritmos Genéticos	66
2.7.1 Introdução.....	66
2.7.2 Algoritmos Genéticos.....	66
2.7.2.1 Representação de Hipóteses	68
2.7.2.2 Operadores Genéticos	69
2.7.2.3 Função Ajuste e Selecção.....	70
2.7.3 Pesquisa do Espaço de Hipóteses	70
2.7.4 Programação Genética.....	71
2.8 Aprendizagem de Conjuntos de Regras	73
2.8.1 Introdução.....	73
2.8.2 Algoritmos de Cobertura Sequencial.....	73
2.8.2.1 Algoritmos de Cobertura Sequencial Versus Algoritmos de Cobertura Simultânea.....	76
2.8.2.2 Funções Alternativas de Avaliação do Desempenho das Regras	77
2.8.3 Aprendizagem de Regras de Primeira Ordem	78
2.8.4 Algoritmo FOIL.....	79
2.8.5 Indução como Dedução Invertida.....	83
2.8.6 Resolução Invertida	85
2.8.6.1 Resolução de Primeira Ordem.....	87
2.8.6.2 Resolução Invertida nas Expressões de Primeira Ordem	88
2.8.6.3 Progol.....	89

CAPÍTULO 3	91
Aprendizagem Analítica.....	91
3.1 Introdução	91
3.2 Aprendizagem com Teorias do Domínio Perfeitas.....	92
3.2.1 Algoritmo Prolog-EBG	93
3.2.1.1 Explicação do Exemplo de Treino.....	94
3.2.1.2 Análise da Explicação.....	94
3.2.1.3 Refinamento da Hipótese Corrente.....	95
3.4 Aprendizagem Dedutiva.....	96
3.5 Limitação Indutiva na Aprendizagem Baseada em Explicações	97
3.6 Aprendizagem Baseada em Explicações aplicada ao Controlo da Pesquisa	98
 CAPÍTULO 4	 101
Combinação da Aprendizagem Indutiva e Analítica.....	101
4.1 Introdução	101
4.2. Pesquisa do Espaço de Hipóteses.....	103
4.3 Uso do Conhecimento Prévio para Inicializar a Hipótese	103
4.3.1 Algoritmo KBANN.....	103
4.4 Uso do Conhecimento Prévio para Alterar o Objectivo da Pesquisa.....	106
4.4.1 Algoritmo TangentProp	106
4.4.2 Um exemplo ilustrativo.....	107
4.5 Uso do Conhecimento Prévio para Aumentar os Operadores de Pesquisa.....	109
4.5.1 Algoritmo FOCL.....	109
 CAPÍTULO 5	 111
Avaliação de Hipóteses.....	111
5.1 Introdução	111
5.2 Estimativa da Exactidão de Uma Hipótese.....	111
5.3 Diferença no Erro de Duas Hipóteses.....	113
5.3.1 Teste de Hipóteses	114
5.4 Comparação de Algoritmos de Aprendizagem.....	114
 CAPÍTULO 6	 119
TEMPUS – Uma ferramenta de Aprendizagem Automática.....	119
6.1 Aplicação da Aprendizagem Automática aos Sistemas Eléctricos de Energia.....	119
6.1.1 Introdução	119
6.1.2 Aplicações por Área.....	119
6.1.3 Métodos da Aprendizagem Automática aplicados aos Sistemas Eléctricos de Energia.....	120
6.1.3.1 Método Híbrido Árvore de Decisão - Rede Neuronal	121
6.1.4 Resultados obtidos pelos Métodos de Aprendizagem Automática num problema de Sistemas Eléctricos de Energia	122
6.2 SPARSE	124
6.2.1 Introdução	124
6.2.2 Arquitectura do SPARSE.....	125
6.2.3 Objectivos do SPARSE.....	126
6.2.4 Base de Conhecimento.....	127
6.2.5 Motor de Inferência.....	129
6.2.5.1 Raciocínio Temporal	131
6.2.5.2 Raciocínio Não Monotónico.....	132
6.2.5.3 Manutenção do Conhecimento	133
6.2.6 Interface com o Utilizador	133
6.2.7 Explicações	134
6.2.8 Módulos Adicionais.....	135
6.2.8.1 Verificação e Validação.....	135
6.2.8.1.1 Validação do SPARSE.....	136

6.2.8.1.2 Verificação do SPARSE	136
6.2.8.2 Tutor Inteligente	137
6.3 TEMPUS.....	138
6.3.1 Introdução.....	138
6.3.2 Algoritmos de Aprendizagem Automática Versus TEMPUS.....	139
6.3.3 Objectivos do TEMPUS	140
6.3.4 Arquitectura do TEMPUS	141
6.3.5 Alterações ao Motor de Inferência do SPARSE	143
6.3.6 Algoritmo de Aprendizagem Automática – TEMPUS	144
6.3.7 Base de Conhecimento do TEMPUS.....	156
6.3.8 Estrutura do Relatório produzido pelo TEMPUS	156
6.4 EDIREGRA- Editor de Regras do SPARSE.....	161
CAPÍTULO 7.....	167
Resultados obtidos pelo TEMPUS	167
7.1 Introdução	167
7.2 Resultados Obtidos	168
7.2.1 Regra d1 – Disparo Trifásico Com Disjuntor Em Movimento	168
7.2.2 Regra d2 – Disparo Trifásico Sem Disjuntor Em Movimento.....	171
7.2.3 Regra d3 – Disparo Monofásico	172
7.2.4 Regra d4 – Religação Trifásica.....	174
7.2.5 Regra d5 – Religação Monofásica.....	176
7.2.6 Regra d6 – Religação Trifásica Mal Suceda.....	177
7.2.7 Regra d7 – Religação Monofásica Mal Suceda	178
7.2.8 Regra d21 – Disparo Nos Dois Extremos	180
7.2.9 Regra j2 – Fecho Sobre Defeito.....	182
CAPÍTULO 8.....	185
Conclusões.....	185
8.1 Objectivos Alcançados.....	185
8.2 Trabalho Futuro	187
REFERÊNCIAS	189
ANEXO A.....	1
Instalações.....	1
ANEXO B.....	1
Exemplo com Factos produzidos pelo SPARSE e Relatório produzido pelo TEMPUS.....	1
ANEXO C.....	1
Resultados	1

Índice de Figuras

Figura 2.1 – Algoritmo “Candidate-Elimination” aplicado aos dois primeiros exemplos de treino da tabela 2.1	14
Figura 2.2 - Algoritmo “Candidate-Elimination” aplicado ao terceiro exemplo de treino da tabela 2.1 ..	15
Figura 2.3 - Algoritmo “Candidate-Elimination” aplicado ao quarto exemplo de treino da tabela 2.1	15
Figura 2.4 – Um “Version Space” com os seus conjuntos fronteira S_4 e G_4	16
Figura 2.5 – Árvore de Decisão para o conceito alvo “Livro aconselhado a comprar”	19
Figura 2.6 – Ganho de informação do atributo Interpretabilidade.....	24
Figura 2.7 - Ganho de informação do atributo Preço.....	24
Figura 2.8 - Ganho de informação do atributo Conteúdo Técnico/Científico.....	25
Figura 2.9 - Ganho de informação do atributo Grafismo.....	25
Figura 2.10 – Cálculos do ganho de informação para a etapa seguinte da construção da Árvore de Decisão.....	26
Figura 2.11 – Representação típica de uma Rede Neuronal	34
Figura 2.12 – Um “Perceptron”	36
Figura 2.13 – Superfície de Decisão	37
Figura 2.14 – Espaço de Hipóteses de Vectors de Pesos e respectivos Erros	38
Figura 2.15 – Unidade Sigmóide	41
Figura 2.16 – Evolução das Probabilidades Posteriores com o aumento de dados de treino.....	50
Figura 2.17 – Aprendizagem de uma Função Linear	51
Figura 2.18 - Os K -Vizinhos Mais Próximos.....	60
Figura 2.19 – Representação em Árvore de uma Função na Programação Genética.....	72
Figura 2.20 - Operação de Crossover	72
Figura 2.21 – Pesquisa de pré-condições da regra	74
Figura 2.22 - (a) Inferência de C pela aplicação da Regra de Resolução.....	86
(b) Inversão do Operador de Resolução para a Inferência Indutiva de C_2	86
Figura 2.23 – Aplicação da Regra de Resolução Inversa.....	88
Figura 4.1 – Pesquisa do Espaço de Hipóteses no KBANN.....	105
Figura 4.2 – Ajuste de valores e derivadas com o Algoritmo TangentProp	106
Figura 4.3 – Pesquisa do Espaço de Hipóteses no TangenteProp.....	108
Figura 6.1 - Arquitectura do SPARSE.....	126
Figura 6.2 - Arquitectura do TEMPUS.....	142
Figura 6.3 (a) – Fluxograma do TEMPUS.....	154
Figura 6.3 (b) – Fluxograma do TEMPUS	155
Figura 6.3 – Janela principal do EDIREGRA.....	162
Figura 6.4 – Janela Alterar Regra do EDIREGRA (para regras com duas premissas temporais).....	162
Figura 6.5 – Janela Alterar Regra do EDIREGRA (para regras com uma premissa temporal)	163
Figura 6.6 – Janela Ver Alterações do EDIREGRA.....	164
Figura 6.7 – Janela Alterar Regra do EDIREGRA	164
Figura 6.8 – Janela Nova Regra do EDIREGRA.....	164
Figura 6.9 – Janela Ver Inserções do EDIREGRA	165
Figura 6.10 – Janela Alterar Regra do EDIREGRA	165

Índice de Tabelas

Tabela 2.1 – Exemplos de treino positivos e negativos para o conceito alvo “Filme aconselhado a ver” ..7	7
Tabela 2.2 – Tarefa de aprendizagem do conceito “Filme aconselhado a ver” ..8	8
Tabela 2.3 – Algoritmo “Find-S” ..10	10
Tabela 2.4 – Algoritmo “List-Then-Eliminate” ..12	12
Tabela 2.5 – Algoritmo “Candidate-Elimination” usando “Version Spaces” ..13	13
Tabela 2.6 – Algoritmo de Aprendizagem de Árvores de Decisão ID3 ..22	22
Tabela 2.7 – Exemplos de treino para o conceito alvo “Livro a Comprar” ..23	23
Tabela 2.8 – Algoritmo da Descida da Encosta para treinar uma unidade linear ..40	40
Tabela 2.9 – Descida Incremental da Encosta do Algoritmo “Backpropagation” (com dois níveis de unidades sigmóides).....43	43
Tabela 2.10 – Algoritmo da Hipótese Posterior Máxima ..49	49
Tabela 2.11 – Algoritmo dos K -Vizinhos Mais Próximos para aproximar uma função discreta.....60	60
Tabela 2.12 – Um protótipo de um Algoritmo Genético ..67	67
Tabela 2.13 – Operadores comuns para Algoritmos Genéticos.....69	69
Tabela 2.14 – Algoritmo de Cobertura Sequencial para aprender um conjunto disjuntivo de regras.....74	74
Tabela 2.15- O Algoritmo de Pesquisa Geral-para-Específica Aprender-uma-Regra.....75	75
Tabela 2.16 – Algoritmo FOIL.....80	80
Tabela 2.17 –Operador de Resolução (forma proposicional) 85	85
Tabela 2.18 – Operador de Resolução Inversa 86	86
Tabela 2.19 – Operador de Resolução (forma de primeira ordem).....87	87
Tabela 3.1 – Um Problema de Aprendizagem Analítica : <i>Pode_Empilhar(x,y)</i> 92	92
Tabela 3.2 – Algoritmo de Aprendizagem baseado em Explicações Prolog-EBG 94	94
Tabela 4.1 – Algoritmo KBANN.....104	104
Tabela 4.2 – Comparação de exactidão entre o Algoritmo TangentProp e o Algoritmo “Backpropagation” 108	108
Tabela 5.1- Valores de Z_N para diferentes Intervalos de Confiança.....112	112
Tabela 5.2 – Algoritmo para Estimar a Diferença de Erro entre dois Métodos de Aprendizagem (L_1 e L_2) 115	115
Tabela 5.3 – Níveis de Confiança de $t_{N,v}$ para intervalos de confiança com dois limites.....116	116

Notação Utilizada

Neste trabalho, as palavras não portuguesas para as quais não existe uma tradução apropriada encontram-se entre aspas. As variáveis e as expressões matemáticas encontram-se em *itálico*.

De seguida, encontra-se um sumário da notação usada na tese.

$(a,b]$: Parêntesis da forma $(,), [, e]$ são usados para representar intervalos, em que os parêntesis rectos representam intervalos que incluem a fronteira e os parêntesis curvos representam intervalos abertos.

$\sum_{i=1}^n x_i$: A soma $x_1 + x_2 + \dots + x_n$.

$\prod_{i=1}^n x_i$: O produto $x_1 \cdot x_2 \cdot \dots \cdot x_n$.

$\vdash$: O símbolo para o vínculo lógico. Por exemplo, $A \vdash B$ significa que B sucede dedutivamente de A .

$>_g$: O símbolo para a relação mais geral que.

$\arg \max_{x \in V} f(x)$: O valor de x que maximiza $f(x)$

$\hat{f}(x)$: A função que aproximação a função $f(x)$

η : O rácio de aprendizagem nas redes neuronais.

μ : O valor esperado de uma distribuição de probabilidade

σ : Desvio padrão de uma distribuição de probabilidade

$\nabla E(\vec{w})$: O gradiente de E relativamente ao vector $\vec{w}$

C : Classe de funções alvo possíveis

D : Dados de treino

$E[x]$: O valor esperado de x

$E[\vec{w}]$: A soma do quadrado dos erros de uma rede neuronal cujos pesos são dados pelo vector $\vec{w}$

Erro : O erro de uma hipótese ou predição

H : Espaço de hipóteses

$h(x)$: A predição da hipótese h para a instância x

$P(x)$: A probabilidade de x

$\Pr(x)$: A probabilidade do evento x

$p(x)$: A densidade de probabilidade de x

$\mathbb{R}$: Conjunto dos números reais

$V, S_{H,D}$: O "Version Space" (conjunto de hipóteses de H que são consistentes com D)

w_{ij} : O peso do nó i para o nó j , nas redes neuronais
 X : Espaço de instâncias

Capítulo 1

Introdução

1.1 Considerações Gerais

Os sistemas com Aprendizagem Automática têm a característica de apresentarem comportamentos e desempenhos que evoluem ao longo do tempo, com base na experiência adquirida.

Nos últimos anos, têm sido desenvolvidos algoritmos que são eficientes para certos tipos de tarefas de aprendizagem e está a emergir uma compreensão teórica da aprendizagem.

A compreensão detalhada dos algoritmos de processamento de informação para a Aprendizagem Automática levam a uma melhor compreensão das capacidades e limitações da aprendizagem humana.

Muitas aplicações práticas de Aprendizagem Automática têm sido desenvolvidas com sucesso em diferentes áreas.

No campo do “Data-Mining”, os algoritmos de Aprendizagem Automática são usados rotineiramente para descobrir conhecimento em grandes bases de dados em áreas tão diversas como os de Sistemas de Energia, finanças, medicina, entre outras.

Como exemplo, foram desenvolvidos programas que aprendem, com sucesso, a avaliar a segurança dos Sistemas Eléctricos de Energia [Wehenkel-98], a prever o rácio de recuperação de doentes com pneumonia [Cooper-97], a reconhecer a fala (palavras) [Waibel-89; Lee-1989], a conduzir autonomamente veículos em vias públicas [Pomerleau-89], a jogar alguns jogos como o “Backgammon” aproximando-se dos níveis de desempenho dos campeões mundiais humanos [Tesauro-92; Tesauro-95]), entre outros.

Na área de Sistemas de Energia, nos últimos anos, tem-se vindo a assistir a várias publicações relacionadas com a Aprendizagem Automática, nomeadamente, o reconhecimento de modelos, sistemas periciais, redes neuronais, conjuntos imprecisos, programação evolutiva, entre outros. No entanto, existem muito poucas aplicações reais relacionadas com essas publicações.

Algumas companhias eléctricas mostraram interesse na aplicação das abordagens da Aprendizagem Automática aos seus Sistemas de Energia, o que levou a que iniciassem projectos de investigação nesta área.

Hoje em dia, só algumas companhias eléctricas nos Estados Unidos da América e na Europa utilizam a Aprendizagem Automática na avaliação da segurança dos seus próprios Sistemas de Energia, pensando-se que num futuro próximo mais casos aparecerão. Efectivamente, a investigação na área da Aprendizagem Automática tem produzido métodos capazes de tratar a complexidade e a não linearidade dos problemas de segurança nos Sistemas Eléctricos de Energia.

Dada a evolução actual dos Sistemas de Energia, a segurança destes sistemas é uma questão primordial, existindo mesmo o risco desta diminuir se não forem tidas em linha de conta, metodologias apropriadas para a sua manutenção. A Aprendizagem Automática pode ser um bom auxílio na manutenção da segurança dos Sistemas de Energia e noutras aplicações deste tipo de sistemas.

No campo da teoria da aprendizagem computacional, foram desenvolvidos métodos teóricos que caracterizam as relações entre o número de exemplos de treino, o número de hipóteses sob consideração e o erro esperado em hipóteses aprendidas.

À medida que a nossa compreensão no campo da informática evolui, parece inevitável que a Aprendizagem Automática tem um papel cada vez mais relevante na ciência dos computadores.

1.2 Guia de Leitura da Tese

Esta tese é constituída por 8 capítulos e 3 anexos. O tema principal deste trabalho é a Aprendizagem Automática e a sua aplicação a um caso real no âmbito dos Sistemas de Energia.

O tema foi abordado de acordo com a estrutura que a seguir se apresenta. No primeiro capítulo (isto é, o presente capítulo), são incluídas considerações gerais sobre a Aprendizagem Automática e explicadas quais as motivações para a realização deste trabalho.

No segundo capítulo, Aprendizagem Indutiva, são apresentados por secção, os métodos de aprendizagem indutiva estudados.

Na secção, Conceitos e Ordenação Geral-para-Específica, é introduzida a tarefa de aquisição de conceitos gerais a partir de exemplos de treino específicos, sendo de seguida apresentada a ordenação geral-para-específica de hipóteses, que consiste numa estruturação útil de pesquisa do espaço de hipóteses. Neste capítulo são também incluídos dois algoritmos de aprendizagem de conceitos: "Find-S" e "Candidate-Elimination". Por fim são feitas algumas considerações em relação à necessidade da limitação indutiva na aprendizagem.

Na secção, Aprendizagem baseada em Árvores de Decisão, é apresentado o algoritmo ID3 para a aprendizagem de uma árvore de decisão, referida a limitação indutiva na aprendizagem de árvores de decisão, discutido o problema do sobreajustamento dos dados, e sugeridas abordagens para prevenir esse sobreajustamento. Na parte final da secção, é analisada a incorporação de atributos com valores contínuos, são referidas medidas alternativas para seleccionar atributos e é analisado o tratamento de exemplos de treino com falta de valores nos atributos ou com custos diferentes.

Na secção, Redes Neurais, são descritas as representações das redes neuronais, os "perceptrons", as redes multinível e o algoritmo "Backpropagation". Nesta secção são também incluídas considerações em relação ao espaço de pesquisa, limitação indutiva, generalização, sobreajustamento e critério de fim.

Na secção, Aprendizagem Bayesiana, é referida a perspectiva Bayesiana na Aprendizagem Automática que possibilita uma abordagem probabilística para a inferência.

Na secção, Aprendizagem baseada em Instâncias, são descritos os métodos de aprendizagem baseada em instâncias, que incluem o método de aprendizagem baseado nos K -vizinhos mais próximos e a aproximação de funções alvo lineares e reais localmente pesadas.

Na secção, Algoritmos Genéticos, são apresentados os algoritmos de aprendizagem modelados depois de uma evolução biológica, que incluem os algoritmos genéticos e a programação genética. Neste capítulo é feita também uma referência aos modelos de evolução e aprendizagem.

Na secção, Aprendizagem de Conjuntos de Regras, são descritos os algoritmos para a aprendizagem de um conjunto de regras, que incluem as abordagens da Programação Lógica Indutiva para aprender regras de primeira ordem.

No terceiro capítulo, Aprendizagem Analítica, é descrito o algoritmo baseado em explicações, Prolog-EBG, que usa o conhecimento prévio para explicar os exemplos de treino observados e que generaliza a partir dessas explicações. Neste capítulo são também feitas referências à

descoberta de novos atributos, à aprendizagem dedutiva e à limitação indutiva na aprendizagem baseada em explicações.

No quarto capítulo, Combinação da Aprendizagem Indutiva e Analítica, são apresentados algoritmos para combinar o conhecimento prévio e os dados de treino disponíveis de forma a melhorar a exactidão das hipóteses aprendidas.

No quinto capítulo, Avaliação de Hipóteses, é apresentada a estimativa da exactidão de uma hipótese, a diferença no erro de duas hipóteses e a comparação do desempenho dos algoritmos de aprendizagem.

O sexto capítulo, TEMPUS – Uma ferramenta de Aprendizagem Automática, é constituído por quatro secções.

Na secção, Aplicação da Aprendizagem Automática aos Sistemas Eléctricos de Energia, como introdução, são referidos os métodos de Aprendizagem Automática mais utilizados nesta área, incluindo considerações em relação aos rácios de exactidão, à interpretabilidade e tempo de CPU obtidos. Nesta secção é também apresentado o método híbrido Árvore de Decisão-Rede Neuronal, e os resultados obtidos pela aplicação dos métodos da Aprendizagem Automática a um problema de Sistemas Eléctricos de Energia.

Na secção, SPARSE, é descrito sucintamente o sistema pericial SPARSE, pois a Aprendizagem Automática a realizar pelo TEMPUS é sobre a Base de Conhecimento do SPARSE. No final da secção, é explicado o motivo da não utilização de nenhum dos métodos da Aprendizagem Indutiva e Analítica estudados, na análise realizada às premissas temporais das regras que constituem a Base de Regras do SPARSE. Por fim é referido que foi concebido e desenvolvido um novo algoritmo de Aprendizagem Automática para esse propósito – TEMPUS.

Na secção, TEMPUS, são apresentados os objectivos do TEMPUS, a sua arquitectura, as alterações realizadas ao motor de inferência do SPARSE, o fluxograma e o algoritmo de Aprendizagem Automática do TEMPUS, a Base de Conhecimento do TEMPUS e a estrutura tipo do relatório produzido pelo TEMPUS.

Na secção, Editor de Regras do SPARSE – EDIREGRA, é descrito e ilustrado o Editor de Regras do SPARSE, que possibilita a alteração da Base de Regras do SPARSE.

No sétimo capítulo, Resultados obtidos pelo TEMPUS, são apresentados e analisados os resultados obtidos pelo TEMPUS relativamente ao comportamento das premissas temporais das regras do SPARSE mediante o processamento de doze ficheiros mensais de incidentes da rede eléctrica nacional.

No oitavo capítulo, Conclusões, são apresentadas conclusões sobre os resultados obtidos no capítulo anterior, referidos os objectivos alcançados e o trabalho futuro.

Adicionalmente, a tese inclui ainda três anexos. O anexo A inclui uma lista com todas as instalações existentes e os respectivos painéis, indicando-se os respectivos códigos. O anexo B inclui um exemplo com os factos produzidos pelo SPARSE mediante o tratamento de um ficheiro de incidentes e o relatório produzido pelo TEMPUS, resultante do processamento efectuado a esses factos (mediante a escolha por parte do utilizador da análise específica das regras d1 e d21 e das instalações SPN e SVG). O anexo C inclui os resultados obtidos pelo TEMPUS na análise efectuada a 12 ficheiros mensais de incidentes. Os resultados são apresentados por regra e as situações de disparo da regra são apresentadas separadamente das situações de não disparo da regra.

1.3 Motivações do Trabalho

O Sistema Pericial SPARSE foi desenvolvido a partir do conhecimento adquirido a partir de peritos no Controlo e Condução da Rede Nacional de Transporte de Energia Eléctrica.

As aplicações de processamento de alarmes, como é o caso do SPARSE, têm uma grande complexidade, pois envolvem questões como o raciocínio temporal e não monotónico.

As regras constituintes da Base de Regras do SPARSE usam o conhecimento técnico e empírico recolhido dos peritos e operadores experientes.

No entanto, os testes feitos ao SPARSE demonstraram que, apesar da estrutura geral das regras estar correcta, os parâmetros temporais usados nas regras deveriam ser ajustados pela consideração de incidentes reais.

Por outro lado, é difícil efectuar sessões de aquisição de conhecimento demoradas, pois os peritos têm muitas tarefas dentro das empresas de Transporte de Energia Eléctrica.

Perante esta situação, considerou-se que seria adequado desenvolver um sistema capaz de Extrair Automaticamente Conhecimento a partir de análises efectuadas sobre os ficheiros de incidentes obtidos nos Centros de Condução da Rede Eléctrica Nacional.

Os objectivos principais do trabalho subjacente à presente tese consistiram em :

- Estudar e apresentar um subconjunto representativo dos métodos de Aprendizagem Automática existentes.
- Conceber e desenvolver uma ferramenta de Extração Automática de Conhecimento que fosse capaz de confirmar ou sugerir modificações nas premissas temporais das regras que constituem a Base de Regras do SPARSE.
- Construir um Editor de Regras que possibilitasse a alteração da Base de Regras do SPARSE.

Capítulo 2

Aprendizagem Indutiva

2.1 Introdução

A tarefa de Aprendizagem Indutiva consiste em determinar uma hipótese idêntica a um determinado conceito alvo para um conjunto de exemplos de treino, isto é, o “output” deve ser uma hipótese consistente com os exemplos de treino.

Os métodos de Aprendizagem Indutiva generalizam a partir de exemplos de treino observados e pela identificação de características que distinguem os exemplos positivos dos negativos.

A única informação disponível sobre o conceito alvo é o seu valor sobre os dados de treino. Assim, os algoritmos de Aprendizagem Indutiva podem apenas garantir que a hipótese resultante ajusta o conceito alvo sobre os dados de treino. Faltando alguma outra informação, assume-se que a melhor hipótese para as instâncias não vistas é a hipótese que melhor ajuste os dados de treino observados. Qualquer hipótese encontrada para aproximar bem o conceito alvo sobre um conjunto suficientemente grande de exemplos de treino, aproxima igualmente bem a função alvo sobre outros exemplos não vistos.

Estes métodos necessitam de um grande número de exemplos de treino para adquirirem um bom nível de exactidão na generalização. A limitação prática da aprendizagem indutiva é ter um desempenho pobre quando não estão disponíveis dados suficientes.

2.2 Aprendizagem de Conceitos e Ordenação Geral-para-Específica

2.2.1 Introdução

O problema da indução de funções gerais a partir de exemplos de treinos específicos é fundamental para a aprendizagem. A aprendizagem de conceitos é a aquisição da definição de uma categoria geral, dada uma amostra de exemplos de treino positivos e negativos da categoria. A aprendizagem de conceitos pode ser formulada como um problema de procura num espaço predefinido de potenciais hipóteses, das hipóteses que melhor ajustam os exemplos de treino.

A maior parte da aprendizagem envolve a aquisição de conceitos gerais a partir de exemplos de treino específicos. Cada conceito pode ser visto como a descrição de algum subconjunto de objectos ou eventos definido sobre um conjunto maior (ex: o subconjunto de animais que são considerados felinos). Alternativamente, cada conceito pode ser visto como uma função booleana definida sobre esse conjunto (ex: uma função definida sobre todos os animais, cujo valor é verdadeiro se for felino e falso se for outro animal). A aprendizagem de conceitos consiste em inferir automaticamente a definição geral de um conceito, a partir de exemplos membros e não membros desse conceito. Daí, a aprendizagem de conceitos ser entendida como a inferência de uma função booleana a partir de exemplos de treino.

2.2.2 A Tarefa de Aprendizagem de Conceitos

2.2.2.1 Notação

Nos problemas de aprendizagem de conceitos é seguida a seguinte terminologia.

O conjunto de itens sobre os quais o conceito é definido é chamado conjunto de instâncias, que é definido por X . Cada instância é constituída por um conjunto de atributos. O conceito ou função a ser aprendido(a) é chamado conceito alvo, que é definido por c . Geralmente c é uma função booleana definida sobre as instâncias X .

$$c : X \rightarrow \{0,1\}$$

Um exemplo de treino consiste numa instância x de X e o valor do seu conceito alvo $c(x)$. As instâncias em que $c(x) = 1$ são chamadas exemplos positivos, ou membros do conceito alvo. As instâncias em que $c(x) = 0$ são chamados de exemplos negativos, ou não membros do conceito alvo.

O conjunto de exemplos de treino disponíveis define-se por D . O par ordenado $(x, c(x))$ é utilizado para descrever o exemplo de treino consistindo na instância x e o valor do seu conceito alvo $c(x)$.

Dado um conjunto de exemplos de treino do conceito alvo c , o problema que se põe é estimar c . O símbolo H representa o conjunto de todas as hipóteses possíveis a considerar para identificar o conceito alvo. Geralmente H é determinado pela escolha da representação das hipóteses. Normalmente cada hipótese h em H representa uma função booleana definida sobre X . O objectivo é encontrar uma hipótese h tal que $h(x) = c(x)$ para todo x em X .

2.2.2.2 Exemplo ilustrativo

Considere-se o exemplo da tarefa de aprendizagem do conceito alvo “Filme aconselhado a ver”. A tabela 2.1 descreve um conjunto de filmes como exemplo, sendo cada filme constituído por um conjunto de atributos. O atributo Ver indica se é aconselhável ou não ver esse filme. A tarefa consiste em aprender a prever o valor do atributo Ver para um filme arbitrário a partir dos valores dos seus outros atributos.

Tabela 2.1 – Exemplos de treino positivos e negativos para o conceito alvo “Filme aconselhado a ver”

Filmes	Realização	Argumento	Banda Sonora	Efeitos Especiais	Guarda Roupas	Fotografia	Ver
1	Boa	bom	razoável	não tem	bom	boa	Sim
2	boa	bom	boa	não tem	bom	boa	Sim
3	má	mau	boa	não tem	bom	razoável	Não
4	boa	bom	boa	não tem	razoável	razoável	Sim

Uma representação possível e simples é cada hipótese consistir numa conjunção de restrições nos atributos da instância.

Cada hipótese é um vector de seis restrições, especificando os valores de seis atributos: Realização, Argumento, Banda Sonora, Efeitos Especiais, Guarda Roupas e Fotografia.

Para cada atributo a hipótese pode:

- indicar por um “?” que qualquer valor é aceitável para esse atributo,
- especificar um valor único requerido para o atributo (ex: bom), ou
- indicar por um “0” que nenhum valor é aceitável.

Se algum exemplo x satisfazer todas as restrições da hipótese h , então h classifica x como um exemplo positivo ($h(x)=1$).

A hipótese mais geral - que todos os filmes são exemplos positivos - é representada por (?,?,?,?,?) e a hipótese mais específica - que nenhum filme é um exemplo positivo - é representada por (0,0,0,0,0).

Geralmente qualquer tarefa de aprendizagem de conceitos pode ser descrita por um conjunto de instâncias sobre os quais a função alvo é definida, por uma função alvo, por um conjunto de hipóteses candidatas, e por um conjunto de exemplos de treino disponíveis.

A definição da tarefa de aprendizagem do conceito “Filme aconselhado a ver” na sua forma geral é dado pela tabela 2.2.

Tabela 2.2 – Tarefa de aprendizagem do conceito “Filme aconselhado a ver”

-
- Dado:
 - Instâncias X : filmes possíveis, cada um descrito pelos atributos.
 - Realização (com valores possíveis: má, média e boa)
 - Argumento (com valores possíveis: bom e mau)
 - Banda Sonora (com valores possíveis: razoável e boa)
 - Efeitos Especiais (com valores possíveis: tem e não tem)
 - Guarda Roupas (com valores possíveis: bom e razoável)
 - Fotografia (com valores possíveis: boa e razoável)
 - Hipóteses H : cada hipótese é descrita por uma conjunção de restrições nos atributos Realização, Argumento, Banda Sonora, Efeitos Especiais, Guarda Roupas e Fotografia. As restrições podem ser “?” (qualquer valor é aceitável), “0” (nenhum valor é aceitável), ou um valor específico.
 - Conceito alvo c : “Filme aconselhado a ver”: $X \rightarrow \{0, 1\}$

$$c(x) = 1 \text{ se Ver} = \text{Sim}$$

$$c(x) = 0 \text{ se Ver} = \text{Não}$$
 - Exemplos de treino D : exemplos positivos e negativos da função alvo.
 - Determina:
 - A hipótese h em H tal que $h(x) = c(x)$ para todo x em X .
-

2.2.3 A Aprendizagem de Conceitos como Pesquisa

A aprendizagem de conceitos pode ser vista como uma tarefa de pesquisa através de um grande espaço das hipóteses, implicitamente definidas por uma representação de hipóteses. O objectivo desta pesquisa é encontrar a hipótese que melhor ajuste os exemplos de treino. A escolha da representação das hipóteses define o espaço de todas as hipóteses que o programa poderá representar e assim poderá aprender. Considerando por exemplo, as instâncias X e as hipóteses H na tarefa de aprendizagem “Filme aconselhado a ver” é dado que o atributo Realização tem 3 valores possíveis, e todos os outros atributos têm dois valores possíveis, o espaço de instâncias X contém exactamente $3 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 96$ instâncias diferentes. Um calculo semelhante mostra que há $5 \cdot 4 \cdot 4 \cdot 4 \cdot 4 \cdot 4 = 5120$ hipóteses sintacticamente distintas em H . De notar, contudo que todas as hipóteses que contêm um ou mais “0” representam um conjunto vazio de instâncias, isto é, classifica todas as instâncias como negativas. Assim, o numero de hipóteses semanticamente diferentes é só $1 + (4 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3) = 973$.

2.2.3.1 Ordenação Geral-para-Específica de Hipóteses

Muitos algoritmos para aprendizagem de conceitos organizam a procura pelo espaço de hipóteses contando com uma estruturação útil existente em qualquer problema de aprendizagem de conceitos: a ordenação geral-para-específica de hipóteses. Tirando partido da vantagem da ocorrência natural dessa estruturação sobre o espaço de hipóteses, pode-se

construir algoritmos de aprendizagem que exaustivamente procuram mesmo espaços infinitos de hipóteses, sem explicitamente enumerar todas as hipóteses.

Para qualquer instância x em X e hipótese h em H , diz-se que x satisfaz h se e só se $h(x)=1$. Dadas duas hipóteses h_j e h_k ambas funções booleanas definidas sobre X , h_j é mais geral ou igual a h_k se e só se qualquer instância que satisfaz h_k também satisfaz h_j ($h_j \geq h_k$).

$$(\forall_{x \in X}) [h_k(x) = 1] \rightarrow (h_j(x) = 1)]$$

Diz-se que h_j é estritamente mais geral que h_k ($h_j >_g h_k$), se e só se $(h_j \geq_g h_k) \wedge (h_k \not\geq_g h_j)$. A hipótese h_j é mais específica que a hipótese h_k quando h_k é mais geral que h_j .

As relações $\geq_g$ e $>_g$ são definidas independentemente do conceito alvo. Estas relações dependem só da questão das instâncias satisfazerem as duas hipóteses e não na classificação dessas instâncias de acordo com o conceito alvo. Formalmente, a relação $\geq_g$ define uma ordenação parcial sobre as hipóteses do espaço H , pois podem haver pares de hipóteses em que $h_1 \not\geq_g h_2$ e $h_2 \not\geq_g h_1$.

Para ilustrar a ordenação geral-para-específica, considere-se as duas seguintes hipóteses para a tarefa de aprendizagem do conceito "Filme aconselhado a ver".

$$h_1 = (\text{boa}, ?, ?, \text{não tem}, ?, ?)$$

$$h_2 = (\text{boa}, ?, ?, ?, ?, ?)$$

A hipótese h_2 impõe menos restrições às instâncias, classificando mais instâncias como positivas do que h_1 . De facto, qualquer instância classificada como positiva por h_1 é classificada também de positiva por h_2 . Portanto, diz-se que h_2 é mais geral que h_1 .

A ordenação parcial geral-para-específica de hipóteses, que pode ser definida para qualquer problema de aprendizagem de conceitos, proporciona uma estrutura útil para organizar a pesquisa no espaço de hipóteses.

2.2.4 Algoritmo "FIND-S": Procura da Hipótese Mais Específica

Uma forma de organizar a pesquisa de uma hipótese consistente com os exemplos de treino consiste em começar com a hipótese possível mais específica em H , e depois generalizar essa hipótese sempre que um exemplo de treino positivo não é coberto pela hipótese. O algoritmo "Find-S" utiliza a ordenação parcial de hipóteses e é ilustrado na tabela 2.3 [Mitchell-97].

Para ilustrar este algoritmo, assume-se a sequência de exemplos de treino da tabela 2.1 para a tarefa "Filme aconselhado a ver".

A 1ª etapa do algoritmo "Find-S" consiste em inicializar h à hipótese mais específica de H

$$h \leftarrow (0, 0, 0, 0, 0, 0)$$

Tabela 2.3 – Algoritmo “Find-S”

-
1. Inicializar h à hipótese mais específica em H
 2. Para cada instância de treino positiva x
 - Para cada restrição de atributo a_i em h
 - Se a restrição a_i é satisfeita por x
 - Então não fazer nada
 - Senão substituir a_i em h pela restrição seguinte mais geral que é satisfeita por x
 3. Retornar a hipótese h
-

Pela observação do primeiro exemplo da tabela 2.1, que é positivo, fica claro que a hipótese h é muito específica. Como nenhuma das restrições “0” são satisfeitas por este exemplo, cada restrição é substituída pela seguinte hipótese mais geral que ajusta o exemplo. Assim, h fica

$$h \leftarrow (boa, bom, razoável, não\ tem, bom, boa)$$

A hipótese resultante é ainda bastante específica, pois assume que todas as instâncias são negativas excepto a última instância considerada. O segundo exemplo de treino (também positivo), força o algoritmo à generalização seguinte de h , colocando um ? no atributo Banda Sonora (este atributo não era satisfeito pela hipótese anterior h).

A hipótese refinada fica:

$$h \leftarrow (boa, bom, ?, não\ tem, bom, boa)$$

O algoritmo com o terceiro exemplo que é negativo não faz alterações a h , pois ignora os exemplos negativos.

Assumindo-se que o espaço de hipóteses H contém uma hipótese que descreve o verdadeiro conceito alvo c e que os dados de treino não contêm erros, pode-se concluir que a hipótese corrente h nunca requer uma revisão em resposta a um exemplo negativo. A hipótese corrente h é sempre a hipótese mais específica em H consistente com os exemplos positivos de treino já analisados. Como é assumido que conceito alvo c está em H e é consistente com os exemplos de treino positivos, c deve ser mais geral ou igual a h .

Assim, como o conceito alvo c nunca cobre um exemplo negativo, a hipótese h também não (pela definição de mais geral que), não sendo necessária nenhuma revisão a h em resposta a qualquer exemplo negativo.

O quarto e último exemplo de treino leva a outra generalização de h

$$h \leftarrow (boa, bom, ?, não\ tem, ?, ?)$$

O algoritmo “Find-S” ilustra uma forma pela qual a ordenação parcial mais geral que pode ser usada para organizar a pesquisa de uma hipótese aceitável. A pesquisa move-se de hipótese em hipótese, da mais específica para hipóteses progressivamente mais gerais. Em cada etapa, a hipótese é generalizada apenas o suficiente para cobrir o novo exemplo positivo. Assim, em cada etapa a hipótese é a hipótese mais específica consistente com os exemplos de treino observados até esse momento.

O algoritmo “Find-S” utiliza a ordenação geral-para-específica, realizando uma pesquisa específica-para-geral pelo espaço de hipóteses, para encontrar a hipótese mais específica consistente com os exemplos de treino.

Na literatura de aprendizagem de conceitos existem muitos algoritmos diferentes que utilizam esta mesma ordenação parcial mais geral que para organizar a pesquisa.

A propriedade chave do algoritmo “Find-S” é que para espaços de hipóteses descritas por conjunções de restrições de atributos, garante chegar à hipótese mais específica de H que é consistente com os exemplos de treino positivos. A hipótese final é também consistente com os exemplos negativos permitindo que o conceito alvo esteja contido em H . Contudo existem algumas questões não resolvidas por este algoritmo de aprendizagem tais como:

- Apesar do “Find-S” encontrar uma hipótese consistente com os dados de treino, não permite saber se encontrou a única hipótese em H consistente com os dados (isto é, o conceito alvo correcto), ou se existem outras hipóteses consistentes também.
- Exemplos de treino com erros ou ruído (inconsistentes) podem induzir em erro, dado que o algoritmo ignora os exemplos negativos.
- Para alguns espaços de hipóteses podem haver várias hipóteses o máximo específicas consistentes com os dados. Neste caso, o algoritmo “Find-S” tem que ser estendido para permitir o “backtrack” nas suas escolhas de generalização das hipóteses, para acomodar a possibilidade do conceito alvo encontrar-se ao longo de um ramo diferente daquele que foi seleccionado na ordenação parcial.

2.2.5 “Version Spaces” e Algoritmo “Candidate-Elimination”

Uma outra abordagem para a aprendizagem de conceitos é o algoritmo “Candidate-Elimination”, que resolve muitas das limitações do “Find-S”. A ideia chave neste algoritmo é encontrar o conjunto de todas as hipóteses consistentes com os exemplos de treino. Surpreendentemente este algoritmo obtém todas estas hipóteses sem explicitamente enumerar todos os seus membros. Isto é conseguido usando a ordenação parcial mais geral que, para manter uma representação compacta do conjunto de hipóteses consistentes e para refinar essa representação à medida que um novo exemplo de treino é encontrado.

Uma hipótese h é consistente com um conjunto de exemplos de treino D se e só se $h(x)=c(x)$ para cada exemplo $(x, c(x))$ em D .

$$\text{consistente}(h,D) \equiv (\forall (x, c(x)) \in D) \ h(x) = c(x)$$

De notar a diferença chave entre esta definição de consistente e a definição de satisfaz. Um exemplo x satisfaz a hipótese h quando $h(x) = 1$, indiferentemente se x é um exemplo positivo ou negativo do conceito alvo. Contudo, um exemplo para ser consistente com h depende do conceito alvo, e em particular, se $h(x) = c(x)$.

O subconjunto de todas as hipóteses consistentes com os exemplos de treino observados é chamado de “Version Space” ($VS_{H,D}$) relativamente ao espaço de hipóteses H e aos exemplos de treino D , pois contém todas as versões possíveis do conceito alvo.

$$VS_{H,D} \equiv \{h \in H \mid \text{consistente}(h, D)\}$$

2.2.5.1 Algoritmo “List-Then-Eliminate”

Uma forma de representar o “Version Space” é simplesmente listar todos os seus membros, que é conseguido com o algoritmo de aprendizagem “List-Then-Eliminate”, definido na tabela 2.4 [Mitchell-97].

O algoritmo “List-Then-Eliminate”, primeiro inicializa o “Version Space” para conter todas as hipóteses em H e depois elimina qualquer hipótese encontrada inconsistente com qualquer exemplo de treino.

Tabela 2.4 – Algoritmo “List-Then-Eliminate”

-
1. “Version Space” $\leftarrow$ lista contendo todas as hipóteses em H .
 2. Para cada exemplo de treino $(x, c(x))$
remover do “Version Space” qualquer hipótese h em que $h(x) \neq c(x)$
 3. Retornar lista de hipóteses no “Version Space”
-

O “Version Space” de hipóteses candidatas diminui à medida que mais exemplos são observados, até que se encontre só uma hipótese consistente com todos os exemplos observados, atingindo assim, presumivelmente o conceito alvo desejado. Se houver insuficiência de dados para levar o “Version Space” a uma hipótese única, então o algoritmo pode chegar ao conjunto de hipóteses consistentes com os dados observados.

O algoritmo “List-Then-Eliminate” pode ser aplicado sempre que o espaço de hipóteses H é finito. Este algoritmo tem a vantagem de garantir que se obtém todas as hipóteses consistentes com os dados de treino. No entanto, requer a enumeração exaustiva de todas as hipóteses em H .

O algoritmo “Candidate-Elimination” trabalha sob o mesmo princípio do algoritmo “List-Then-Eliminate”. Contudo, emprega uma representação mais compacta do “Version Space”. O “Version Space” é representado pelo armazenamento dos membros mais gerais (representados por G) e dos membros mais específicos (representados por S). Esses membros formam conjuntos fronteira gerais e específicos para delimitar o “Version Space” dentro do espaço ordenado de hipóteses. Dados somente os dois conjuntos S e G , é possível enumerar todos os membros pela generalização das hipóteses que se encontram entre esses dois conjuntos na ordenação parcial geral-para-específica de hipóteses. A fronteira geral G , respeitante ao espaço de hipóteses H e aos dados de treino D , é o conjunto de membros o máximo gerais de H consistentes com D .

$$G \equiv \{g \in H \mid \text{consistente}(g, D) \wedge (\neg \exists g' \in H) [(g' >_g g) \wedge \text{consistente}(g', D)]\}$$

A fronteira mais específica S , respeitante ao espaço de hipóteses H e dados de treino D , é o conjunto de membros o mínimo gerais (ou o máximo específicos) de H consistentes com D .

$$S \equiv \{s \in H \mid \text{consistente}(s, D) \wedge (\neg \exists s' \in H) [(s >_g s') \wedge \text{consistente}(s', D)]\}$$

Estando bem definidos G e S , estes especificam completamente o “Version Space”.

Sendo X um conjunto arbitrário de instâncias, H um conjunto de hipóteses booleanas definido sobre X , $c : X \rightarrow \{0, 1\}$ um conceito alvo arbitrário definido sobre X , D um conjunto arbitrário de exemplos de treino $\{(x, c(x))\}$ e S e G bem definidos então

$$VS_{H,D} = \{h \in H \mid (\exists s \in S) (\exists g \in G) (g \geq_g h \geq_g s)\}$$

2.2.5.2 O Algoritmo de Aprendizagem “Candidate-Elimination”

O algoritmo “Candidate-Elimination” obtém o “Version Space” contendo todas as hipóteses de H que são consistentes com uma sequência de exemplos de treino observados.

Na tabela 2.5 encontra-se o algoritmo “Candidate-Elimination” usando o “Version Space” [Mitchell-97]. De notar a dualidade em como os exemplos positivos e negativos influenciam S e G .

O algoritmo começa por inicializar o “Version Space” ao conjunto de todas as hipóteses em H , inicializando a fronteira G para conter a hipótese mais geral (G_o) e inicializando a fronteira S para conter a hipótese mais específica (menos geral) (S_o). Estas duas fronteiras delimitam todo o espaço de hipóteses, pois todas as outras hipóteses de H são mais gerais que S_o e mais específicas que G_o . De cada vez que cada exemplo de treino é considerado, a fronteira S é generalizada e G especializada, de forma a eliminar do “Version Space” qualquer hipótese inconsistente com o novo exemplo de treino. Depois de todos os exemplos terem sido processados, o “Version Space” contém todas as hipóteses consistentes com esses exemplos.

Tabela 2.5 – Algoritmo “Candidate-Elimination” usando “Version Spaces”

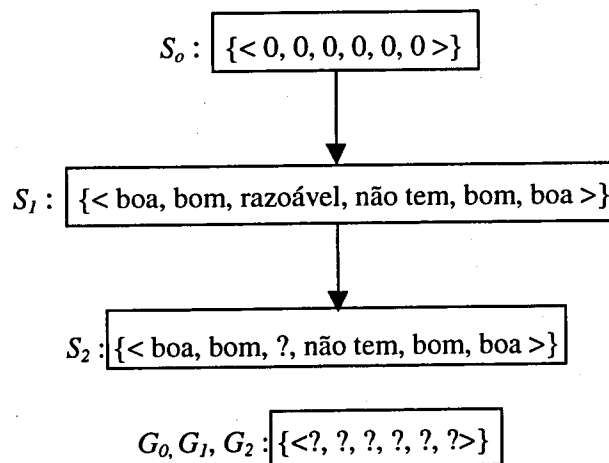
-
- Inicializar G ao conjunto de hipóteses o máximo gerais de H
 - Inicializar S ao conjunto de hipóteses o máximo específicas de H
 - Para cada exemplo de treino d , fazer
 - Se d é um exemplo positivo
 - Remover de G qualquer hipótese inconsistente com d
 - Para cada hipótese s em S que não é consistente com d
 - Remover s de S
 - Adicionar a S todas as generalizações mínimas h de s tal que
 - h é consistente com d , e algum membro de G é mais geral que h
 - Remover de S qualquer hipótese que é mais geral que outra hipótese em S
 - Se d é um exemplo negativo
 - Remover de S qualquer hipótese inconsistente com d
 - Para cada hipótese g em G que não é consistente com d
 - Remover g de G
 - Adicionar a G todas as especializações mínimas h de g tal que
 - h é consistente com d , e algum membro de S é mais específico que h
 - Remover de G qualquer hipótese que é menos geral do que outra hipótese em G
-

2.2.5.3 Um exemplo ilustrativo

A figura 2.1 ilustra o algoritmo “Candidate-Elimination” aplicado aos dois primeiros exemplos de treino da tabela 2.1. As fronteiras são inicializadas a G_0 e S_0 , a hipótese mais geral e a hipótese mais específica de H , respectivamente.

Quando o primeiro exemplo de treino é considerado (no caso um exemplo positivo), o algoritmo testa a fronteira S e verifica que é excessivamente específica. A fronteira S é revista movendo-se para a hipótese menos geral que cobre este novo exemplo (S_1). Nenhuma actualização da fronteira G é necessária em resposta a este exemplo de treino já que G_0 cobre correctamente este exemplo. Quando o segundo exemplo de treino (também positivo) é considerado tem um efeito similar de generalização de S para S_2 , deixando G outra vez inalterado (isto é $G_0 = G_1 = G_2$). De notar que o processamento destes dois exemplos positivos é muito semelhante ao processamento verificado pelo algoritmo “Find-S”.

O terceiro exemplo (no caso negativo) revela que a fronteira G do “Version Space” é excessivamente geral, isto é, a hipótese em G prediz incorrectamente que o novo exemplo é um exemplo positivo. A hipótese na fronteira G deve assim ser especializada até classificar correctamente esse novo exemplo negativo. Como ilustra a figura 2.2, há três hipóteses alternativas minimamente mais específicas. Todas elas tornam-se membros da nova fronteira G_3 .



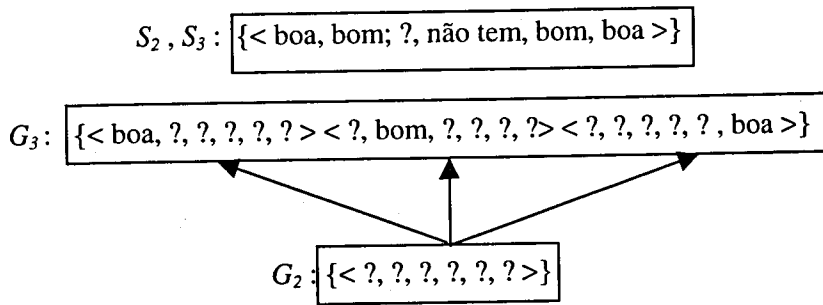
Exemplos de treino:

1. $\langle \text{boa}, \text{bom}, \text{razoável}, \text{não tem}, \text{bom}, \text{boa} \rangle$, Ver = Sim
2. $\langle \text{boa}, \text{bom}, \text{boa}, \text{não tem}, \text{bom}, \text{boa} \rangle$, Ver = Sim

Figura 2.1 – Algoritmo “Candidate-Elimination” aplicado aos dois primeiros exemplos de treino da tabela 2.1

A razão pela qual só há três novas hipóteses em G_3 , havendo seis atributos que podem ser especificados para especializarem G_2 , é que as outras três hipóteses, apesar de serem também especializações mínimas de G_2 que classificam correctamente o novo exemplo como negativo, são inconsistentes com os exemplos positivos anteriormente analisados. O algoritmo detecta esta situação, simplesmente por verificar que essas hipóteses não são mais gerais que a fronteira corrente S_2 . De facto, a fronteira S do “Version Space” é um sumário dos exemplos positivos analisados que pode ser usada para determinar se uma dada hipótese é consistente com esses exemplos.

Qualquer hipótese mais geral que S por definição, cobrirá qualquer exemplo que S cobre e assim cobrirá qualquer exemplo positivo passado.

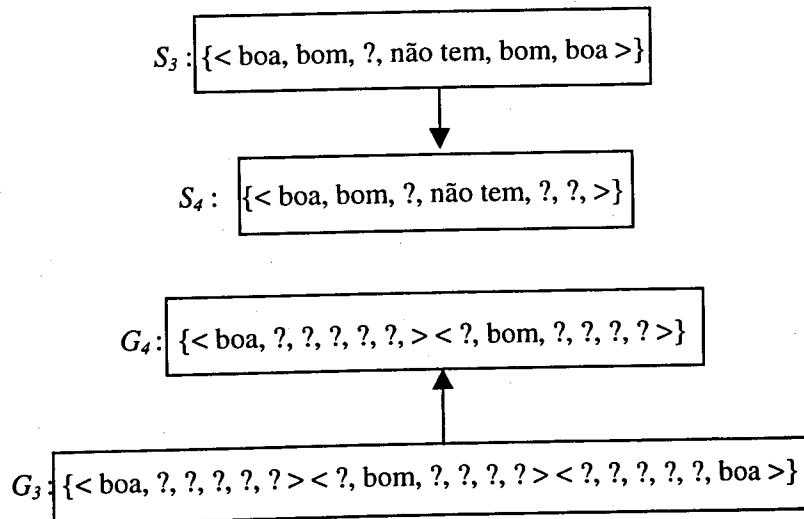


Exemplo de treino : 3. $\langle \text{má, mau, boa, não tem, bom, razoável} \rangle$, Ver = Não

Figura 2.2 - Algoritmo "Candidate-Elimination" aplicado ao terceiro exemplo de treino da tabela 2.1

De uma forma análoga a fronteira G sumaria a informação dos exemplos negativos analisados anteriormente. Com qualquer hipótese mais específica do que G é assegurada a consistência com os exemplos negativos tratados. Isto é verdade pois qualquer uma destas hipóteses, por definição, não pode cobrir exemplos que G não cobre.

Como é ilustrado por estes três exemplos, os exemplos de treino positivos forçam a fronteira S do "Version Space" a tornar-se cada vez mais geral, enquanto os exemplos negativos têm o papel complementar forçando a fronteira G a tornar-se cada vez mais específica.



Exemplo de treino: 4. $\langle \text{boa, bom, boa, não tem, razoável, razoável} \rangle$, Ver = Sim

Figura 2.3 - Algoritmo "Candidate-Elimination" aplicado ao quarto exemplo de treino da tabela 2.1

O quarto exemplo de treino, como mostra a figura 2.3, generaliza a fronteira S do "Version Space" e remove um membro da fronteira G , pois esse membro não cobre o novo exemplo positivo. Esse membro não pode ser especializado pois com a especialização não cobriria o novo exemplo, nem pode ser generalizado, pois pela definição de G , qualquer hipótese mais geral cobrirá pelo menos um exemplo negativo. Daí a necessidade da sua remoção de G .

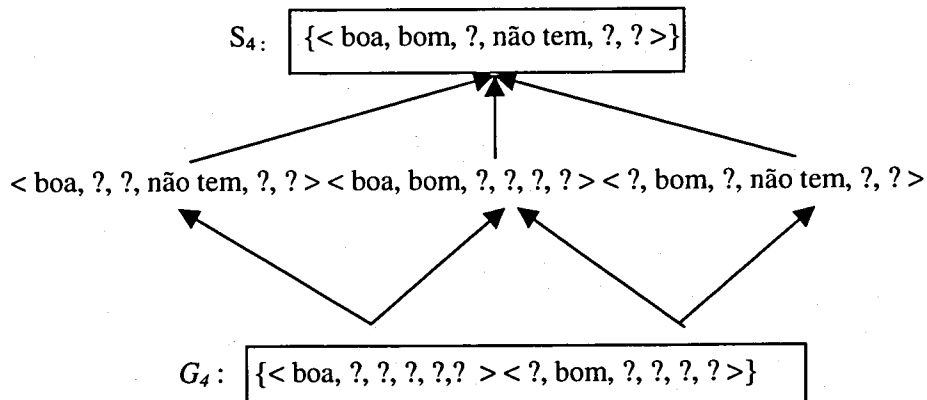


Figura 2.4 – Um “Version Space” com os seus conjuntos fronteira S_4 e G_4

Depois de processar estes quatro exemplos, as fronteiras S_4 e G_4 delimitam o “Version Space” de todas as hipóteses consistentes com os exemplos de treino. Todo o “Version Space” incluindo as hipóteses limitadas por S_4 e G_4 é representado na figura 2.4. Este “Version Space” aprendido é independente da sequência pela qual os exemplos de treino são apresentados.

2.2.6 Observações sobre os “Versions Spaces” e o Algoritmo “Candidate-Elimination”

O “Version Space” aprendido pelo algoritmo “Candidate-Elimination” converge para a hipótese que descreve correctamente o conceito alvo, se não existirem erros nos exemplos de treino, e se houver alguma hipótese h em H que descreve correctamente o conceito alvo. O conceito alvo é apreendido com exactidão quando as fronteiras S e G convergem para uma única hipótese. Quando os exemplos contêm erros, as hipóteses que são inconsistentes com esses exemplos são eliminadas, sendo eliminado o verdadeiro conceito alvo do “Version Space”. Com dados de treino suficientes, é eventualmente detectada uma inconsistência pela constatação que a fronteira S e a fronteira G convergem para um “Version Space” vazio. Um “Version Space” vazio indica que não há hipóteses em H consistentes com todos os exemplos de treino observados. O mesmo pode acontecer quando os exemplos de treino estão correctos, mas o conceito alvo não pode ser descrito na representação de hipóteses escolhida (por exemplo, se o conceito alvo é uma disjunção de atributos e o espaço de hipóteses suporta só conjunções de atributos).

Geralmente os exemplos de treino são disponibilizados e tratados na totalidade. Noutras situações, pode-se gerar a próxima instância a tratar, obtendo-se a classificação correcta para essa instância por algo externo. Uma boa estratégia para a escolha da próxima instância, passa pela escolha de uma instância que seria classificada como positiva por algumas hipóteses alternativas do “Version Space”, mas como negativa por outras hipóteses. A melhor estratégia na escolha da instância consiste em gerar instâncias que satisfazem exactamente metade das hipóteses no “Version Space” corrente. Quando isto é possível, o tamanho do “Version Space” é reduzido para metade com cada novo exemplo e o conceito alvo correcto pode assim ser encontrado só com $\lceil \log_2 |VS| \rceil$ experiências. Geralmente não é possível gerar uma instância que separe precisamente metade das hipóteses, e por isso, nesses casos é necessário gerar mais do que $\lceil \log_2 |VS| \rceil$ instâncias.

Quando o conceito alvo ainda não está completamente aprendido, havendo ainda no “Version Space” múltiplas hipóteses, é possível classificar certos exemplos com um determinado grau

de confiança. Se uma instância é classificada como positiva por todas as hipóteses no "Version Space" corrente, pode-se classificar a instância como positiva com a mesma confiança que teria se tivesse convergido a um único conceito alvo correcto. De notar que não é necessário enumerar todas as hipóteses no "Version Space" para classificar o exemplo como positivo, sendo só necessário que a instância satisfaça todos os membros da fronteira S .

Pela definição de mais geral que, se a nova instância satisfaz todos os membros de S , deve satisfazer cada uma das outras hipóteses mais gerais do "Version Space". Se por outro lado uma instância for classificada como negativa por todas as hipóteses do "Version Space" então pode ser classificada como negativa. Quando uma instância é classificada como positiva por metade das hipóteses do "Version Space" e como negativa pela outra metade, não é possível classificar esta instância com confiança até que mais exemplos de treino estejam disponíveis. Quando uma instância é classificada como negativa e como positiva por um diferente número de hipóteses, a instância pode ser classificada (com menos confiança) como positiva se o número de hipóteses positivas é maior que o número de hipóteses negativas, ou como negativas no caso contrário.

O algoritmo "Candidate-Elimination" utiliza a ordenação geral-para-específica para obter o "Version Space", refinando o conjunto de hipóteses o máximo específicas (S) e o conjunto de hipóteses o máximo gerais (G).

As fronteiras S e G delimitam o conjunto inteiro de hipóteses consistentes com os dados, proporcionando a descrição da incerteza do conceito alvo. O "Version Space" de hipóteses alternativas pode ser examinado para determinar se houve uma convergência para o conceito alvo, para verificar se os exemplos de treino são inconsistentes, e para verificar que instâncias não analisadas podem ser classificadas sem ambiguidade a partir do conceito parcial aprendido.

2.2.7 Limitação Indutiva

Para assegurar que o espaço de hipóteses contém o conceito alvo, deve-se enriquecer o espaço de hipóteses de forma a incluir todas as hipóteses possíveis. As limitações impostas à representação das hipóteses podem levar à impossibilidade de chegar à obtenção do conceito alvo. Como exemplo, o caso de se considerar apenas hipóteses conjuntivas, quando pode ser necessário hipóteses disjuntivas.

A solução para o problema de assegurar que o conceito alvo está no espaço de hipóteses H é ter um espaço de hipóteses capaz de representar todos os conceitos, ou seja, capaz de representar todos os subconjuntos das instâncias X . Geralmente o conjunto de todos os subconjuntos do conjunto X é chamado de "power set" de X . Na tarefa de aprendizagem "Filme aconselhado a ver", por exemplo, o tamanho do espaço de instâncias X descrito pelos seis atributos é 96 (3.2.2.2.2.2). De notar, que o espaço de hipóteses conjuntivas é capaz de representar só 973 conceitos alvos (1+(4.3.3.3.3.3)) - um espaço de hipóteses muito limitado. O número de subconjuntos distintos que podem ser definidos sob o conjunto X contendo $|X|$ elementos (isto é, o tamanho do "power set" de X) é $2^{|X|}$. Assim, há 2^{96} conceitos alvo distintos que podem ser definidos sob este espaço de instâncias.

Com o "power set" de X são permitidas disjunções, conjunções e negações de hipóteses. Contudo, enquanto este espaço de hipóteses elimina os problemas de expressividade, leva a outro problema difícil: a incapacidade de generalizar além dos exemplos observados. O problema reside no facto que com uma representação muito expressiva de hipóteses, a fronteira S é sempre simplesmente a disjunção dos exemplos positivos, enquanto que a fronteira G é sempre a negação da disjunção dos exemplos negativos. Assim, os únicos exemplos que são classificados sem ambiguidade por S e G são os próprios exemplos de treino. Qualquer instância não observada ainda é classificada de positiva por metade das hipóteses do "Version Space" e de negativa pela outra metade de hipóteses do "Version Space". Isto porque sendo H

o “power set” de X , por cada hipótese h do “Version Space” que cobre uma instância ainda não analisada, existe uma outra hipótese h' em H que é idêntica a h excepto para a classificação da instância. Se h está no “Version Space”, então h' está também, pois coincide com h em todos os exemplos de treino observados. Para convergir para um único, conceito alvo final, seria necessário ter presente todas as instâncias em X como exemplos de treino.

Tudo isto, ilustra uma propriedade da inferência indutiva, que refere que se não se faz à partida pressuposições relativamente à identidade do conceito alvo não existem bases racionais para classificar instâncias nas observadas. Nos casos em que a pressuposição (por exemplo, o conceito alvo ser só representado por conjunções de atributos) é correcta e não existem exemplos de treino com erros, a classificação de novas instâncias será também correcta. Se esta pressuposição é incorrecta, o algoritmo classificará mal pelo menos algumas instâncias de X . Pelo facto da aprendizagem indutiva requerer alguma forma de pressuposições, ou limitações indutivas, é útil caracterizar as diferentes abordagens de aprendizagem pelas limitações indutivas que elas empregam.

Considere-se um algoritmo de aprendizagem de conceitos L para um conjunto de instâncias X , um conceito arbitrário c definido sobre X , um conjunto arbitrário de exemplos de treino de c $D_c = \{(x, c(x))\}$, a classificação atribuída por L à instância x_i depois de analisar os exemplos de treino $L(x_i, D_c)$. Pode-se descrever a inferência indutiva realizada por L como

$$(D_c \wedge x_i) \succ L(x_i, D_c)$$

onde a notação $y \succ z$ indica que z é indutivamente inferido de y .

O resultado $L(x_i, D_c)$ que L infere, provavelmente não deve ser correcto, pois a classificação $L(x_i, D_c)$ não precisa de suceder dedutivamente dos dados de treino D_c e da descrição da nova instância x_i . Contudo, com a adição de $D_c \wedge x_i$ à limitação indutiva de L (o conjunto de pressuposições adicionais B , $L(x_i, D_c)$ já sucede dedutivamente de D_c , x_i e B :

$$(\forall x_i \in X) [(B \wedge D_c \wedge x_i) \vdash L(x_i, D_c)]$$

em que $y \vdash z$ indica que z sucede dedutivamente de y .

A limitação indutiva do algoritmo “Candidate-Elimination” consiste em assumir que o conceito alvo c está contido num dado espaço de hipóteses H . O algoritmo “Find-S” tem uma limitação indutiva mais forte. Adicionalmente, à pressuposição que o conceito alvo pode ser descrito nesse espaço de hipóteses, assume que todas as instâncias são negativas a não ser que o contrário seja vinculado por outro seu conhecimento.

O algoritmo “Candidate-Elimination” foi aplicado a problemas tais como a aprendizagem de regularidades no campo da espectroscopia [Mitchell-79] e aprendizagem de regras de controlo para a pesquisa heurística [Mitchell-83].

Contudo as aplicações práticas dos algoritmos “Find-S” e “Candidate-Elimination” são limitadas pelo facto de que ambos têm um desempenho pobre quando necessitam tratar dados de treino com ruído.

2.3 Aprendizagem Baseada em Árvores de Decisão

2.3.1 Introdução

A aprendizagem baseada em árvores de decisão é um dos métodos mais práticos e mais usados para a inferência indutiva. É um método representado por uma árvore de decisão para aproximar funções discretas, robusto contra dados com ruído e capaz de aprender expressões disjuntivas. As árvores aprendidas podem também ser representadas como conjuntos de regras Se-Então para melhorar a compreensão por parte dos humanos.

A aprendizagem baseada em árvores de decisão pesquisa num espaço de hipóteses muito expressivo, evitando as dificuldades dos espaços de hipóteses restritos.

Este método tem sido aplicado com sucesso num grande leque de tarefas, entre outras, em casos de avaliação da segurança em Sistemas Eléctricos de Energia [Wehenkel-98], diagnósticos médicos [Cooper-97] e na análise de risco de aplicações financeiras.

2.3.2 Representação das Árvores de Decisão

O processo de classificação de uma instância começa no nó raiz da árvore e termina num nó folha que classifica essa instância.

Cada nó na árvore corresponde a um teste de um atributo de uma instância e cada ramo que descende da raiz corresponde a um dos valores possíveis para esse atributo. A classificação de uma instância começa com o teste do atributo especificado no nó raiz, seguindo-se o encaminhamento para o ramo abaixo da árvore correspondente ao valor desse atributo na instância. Este processo é assim repetido para a sub-árvore com raiz no novo nó.

A figura 2.5 ilustra uma típica árvore de decisão aprendida que classifica se os livros são ou não aconselhados a comprar.

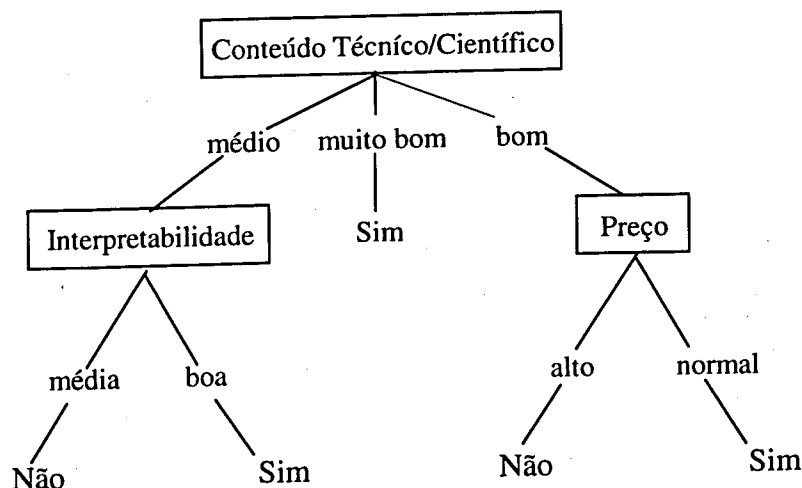


Figura 2.5 – Árvore de Decisão para o conceito alvo “Livro aconselhado a comprar”

Geralmente, as árvores de decisão representam uma disjunção de conjunções de restrições nos valores dos atributos das instâncias. Cada caminho a partir do nó raiz até às folhas corresponde a uma conjunção de testes de atributos e a árvore ela própria uma disjunção de conjunções.

Por exemplo, a árvore da figura 2.5 corresponde à expressão

Conteúdo Técnico/Científico = médio $\wedge$ Interpretabilidade = boa)

✓ *(Conteúdo Técnico/Científico = muito bom)*

✓ *(Conteúdo Técnico/Científico = bom $\wedge$ Preço = normal)*

2.3.3 Problemas Apropriados a Árvores de Decisão

A aprendizagem baseada em árvores de decisão é mais apropriada para problemas com as seguintes características:

- As instâncias são representadas por pares atributo-valor. Os atributos podem estar associados a uma disjunção de valores possíveis (exemplo: média ou boa) ou valores numéricos (exemplo : Temperaturas).
- A função alvo tem como resultado valores discretos. Frequentemente é atribuído um valor booleano a cada exemplo. Os métodos das árvores de decisão facilmente se estendem para funções de aprendizagem com mais do que dois valores como resultado. Uma extensão mais substancial permite aprender funções alvo com valores reais como “output” (menos comum).
- Problemas que necessitam descrições disjuntivas.
- Dados de treino com erros. Os métodos de aprendizagem de árvores de decisão são robustos a erros, tanto erros nas classificações dos exemplos de treino como erros nos valores dos atributos que descrevem esses exemplos.
- Dados de treino com falta de valores nos atributos.

Os problemas em que a tarefa consiste em classificar exemplos numa categoria, de um conjunto discreto de categorias, são frequentemente chamados de problemas de classificação.

2.3.4 Algoritmo de Aprendizagem de Árvores de Decisão ID3

Muitos algoritmos que têm sido desenvolvidos para aprendizagem de árvores de decisão são variações de um algoritmo que emprega uma pesquisa “top-down” pelo espaço de árvores de decisão possíveis. Esta abordagem é exemplificada pelo algoritmo ID3 [Quinlan-86]. Este algoritmo aprende as árvores de decisão construindo-as de cima para baixo, começando com a questão “que atributo deve ser testado na raiz da árvore?”. Para responder a esta questão, cada atributo da instância é avaliado usando um teste estatístico para determinar como o atributo sozinho classifica os exemplos de treino. O melhor atributo é seleccionado e usado como teste no nó raiz da árvore. Um descendente do nó raiz é então criado para cada valor possível desse atributo. Todo o processo é então repetido usando os exemplos de treino associados com cada nó descendente para seleccionar o melhor atributo para testar nesse ponto da árvore. Isto forma uma pesquisa numa árvore de decisão, em que o algoritmo nunca faz o backtrack para reconsiderar escolhas anteriores.

A escolha central no algoritmo ID3 é seleccionar o atributo para fazer o teste em cada nó da árvore, que melhor classifique os exemplos. Define-se para isso uma propriedade estatística, chamada ganho de informação, que mede como um dado atributo separa bem os exemplos de treino de acordo com a sua classificação alvo. O algoritmo ID3 usa esta medida do ganho de informação para seleccionar os atributos candidatos em cada etapa da construção da árvore.

O ganho de informação usa uma medida frequentemente usada na teoria da informação que é a entropia, que caracteriza a (im)pureza de um conjunto arbitrário de exemplos. Dado um conjunto S , contendo exemplos positivos e negativos de um determinado conceito alvo, a entropia de S relativamente a esta classificação booleana é

$$\text{Entropia}(S) = -p_+ \cdot \log_2 p_+ - p_- \cdot \log_2 p_-$$

em que p_+ é a proporção de exemplos positivos em S e p_- é a proporção de exemplos negativos em S . Nos cálculos envolvendo entropia define-se $0 \cdot \log_2 0$ como 0. A entropia é 0 se todos os membros de S pertencem à mesma classe. Por exemplo se todos os exemplos são positivos então

$$p_+ = 1$$

$$p_- = 0$$

$$\text{Entropia}(S) = -1 \cdot \log_2 1 - 0 \cdot \log_2 0 = -1 \cdot 0 - 0 \cdot \log_2 0 = 0$$

A entropia é 1 quando o conjunto contém um número igual de exemplos positivos e negativos. Se o conjunto contém um número desigual de exemplos positivos e negativos a entropia tem um valor entre 0 e 1.

A teoria da informação interpreta a entropia, como o número mínimo de “bits” de informação necessários para codificar a classificação de um membro arbitrário de S . Se p_+ é 1, sabe-se que o exemplo é positivo, a entropia é 0 e não é necessário enviar uma mensagem. Por outro lado, se p_+ é 0.5, a entropia é 1 e é necessário 1 “bit” para indicar se o exemplo é positivo ou negativo. Se p_+ é 0.8, então o conjunto de mensagens pode ser codificado usando em média menos do que 1 “bit” por mensagem, atribuindo códigos menores para os exemplos positivos e códigos maiores para os exemplos negativos.

Se o atributo alvo pode tomar c diferentes valores (mais do que 2), então a entropia de S relativa à classificação com tamanho c é definida por

$$\text{Entropia}(S) = \sum_{i=1}^c -p_i \cdot \log_2 p_i$$

em que p_i é a proporção de S pertencente à classe i . O logaritmo é ainda base 2 pois a entropia é uma medida do tamanho esperado da codificação medida em “bits”. Se o atributo puder tomar c valores possíveis, a entropia pode ser tão grande como $\log_2 c$ (maior que 1).

Enquanto a entropia é a medida da impureza dum conjunto de exemplos de treino, o ganho de informação é a medida da eficiência de um atributo na classificação dos dados de treino. O ganho de informação é a redução esperada na entropia, causada pela partição dos exemplos de acordo com esse atributo. Mais precisamente, o ganho de informação, $\text{Ganho}(S,A)$ de um atributo A , relativamente a um conjunto de exemplos S , é definida por

$$\text{Ganho}(S,A) = \text{Entropia}(S) - \sum_{v \in \text{Valores}(A)} \frac{|S_v|}{|S|} \cdot \text{Entropia}(S_v) \quad (2.1)$$

em que $\text{Valores}(A)$ é o conjunto de todos os valores possíveis para o atributo A , e S_v é o subconjunto de S em que o atributo A tem o valor v . O primeiro termo da equação (2.1) é a

entropia do S original, e o segundo termo é o valor esperado da entropia depois de S ser dividido usando o atributo A . A entropia esperada descrita pelo segundo termo é a soma das entropias de cada subconjunto S_v , pesado pela fracção de exemplos $\frac{|S_v|}{|S|}$ que pertencem a S_v .

Tabela 2.6 – Algoritmo de Aprendizagem de Árvores de Decisão ID3

ID3 (*Exemplos*, *Atributo_alvo*, *Atributos*)

Exemplos são os exemplos de treino. *Atributo_alvo* é o atributo cujo valor deve ser previsto pela árvore. *Atributos* é a lista dos outros atributos que devem ser testados pela árvore de decisão aprendida. Retorna uma árvore de decisão que correctamente classifica os exemplos dados

- Cria o nó raiz para a árvore
- Se todos os exemplos são positivos, retorna só o nó raiz, com valor +
- Se todos os exemplos são negativos, retorna só o nó raiz, com valor -
- Se *Atributos* está vazio, retorna só o nó raiz, com valor igual ao valor mais comum do *Atributo_alvo* nos exemplos
- Se não se verificar nenhuma das condições anteriores
 - Início
 - $A \leftarrow$ Atributo dos *Atributos* que melhor (maior ganho de informação) classifica os exemplos
 - O atributo de decisão para a raiz $\leftarrow A$
 - Para cada valor possível v_i de A
 - Adicionar um novo ramo da árvore abaixo da raiz, correspondendo ao teste $A = v_i$
 - Atribuir a $Exemplos_{v_i}$ o subconjunto dos *Exemplos* que têm valor v_i para A
 - Se $Exemplos_{v_i}$ está vazio
 - Então abaixo desse novo ramo adicionar um nó folha com valor igual ao valor mais comum do *Atributo_alvo* nos *Exemplos*
 - Senão abaixo desse novo ramo adicionar uma sub-árvore ID3 ($Exemplos_{v_i}$, *Atributo_alvo*, (*Atributos* - { A }))
 - Fim
- Retornar a raiz

O $Ganho(S, A)$ é assim a redução esperada na entropia, sabendo o valor do atributo A . Pode ser também interpretado como a informação proporcionada, relativamente ao valor da função alvo, dado o valor de um atributo A . O valor de $Ganho(S, A)$ é ainda, o número de “bits” necessários para codificar o valor alvo de um membro arbitrário de S , sabendo o valor do atributo A . O ganho de informação é precisamente a medida usada pelo algoritmo ID3 para seleccionar o melhor atributo em cada etapa da construção da árvore.

Uma versão do algoritmo especializada para aprender funções booleanas (aprendizagem de conceitos) é descrita na tabela 2.6 [Mitchell-97].

2.3.4.1 Um exemplo ilustrativo

Para ilustrar o funcionamento do algoritmo ID3, considere-se a tarefa de aprendizagem representada pelos exemplos de treino da tabela 2.7. O atributo Comprar pode ter os valores Sim ou Não e o seu valor é previsto a partir dos outros atributos da instância.

Tabela 2.7 – Exemplos de treino para o conceito alvo “Livro a Comprar”

Livro	Conteúdo Técnico/Científico	Grafismo	Interpretabilidade	Preço	Comprar
L1	médio	médio	média	normal	Não
L2	médio	médio	média	alto	Não
L3	muito bom	médio	Média	normal	Sim
L4	bom	muito bom	Média	normal	Sim
L5	bom	bom	Boa	normal	Sim
L6	bom	bom	Boa	alto	Não
L7	muito bom	bom	Boa	alto	Sim
L8	médio	muito bom	Média	normal	Não
L9	médio	bom	Boa	normal	Sim
L10	bom	muito bom	Boa	normal	Sim
L11	médio	muito bom	Boa	alto	Sim
L12	muito bom	muito bom	Média	alto	Sim
L13	muito bom	médio	Boa	normal	Sim
L14	bom	muito bom	Média	alto	Não

A 1ª etapa do algoritmo é saber qual o atributo que deve ser testado primeiro na árvore, ou seja, no nó raiz da árvore. O algoritmo ID3 determina o ganho de informação para cada atributo candidato (Conteúdo Técnico/Científico, Grafismo, Interpretabilidade e Preço), e escolhe o que tiver maior ganho de informação. As figuras 2.6, 2.7, 2.8 e 2.9 ilustram os cálculos da entropia de S e do ganho de informação para cada um dos atributos. Por convenção $n+$ representa n exemplos positivos e $n-$ representa n exemplos negativos.

De acordo com a medida do ganho de informação, o atributo Conteúdo Técnico/Científico proporciona uma melhor previsão do atributo alvo Comprar, sobre os exemplos de treino. Assim, o atributo Conteúdo Técnico/Científico é seleccionado como o atributo de decisão para o nó da raiz da árvore, e os ramos são criados abaixo da raiz para cada um dos valores possíveis (médio, muito bom e bom).

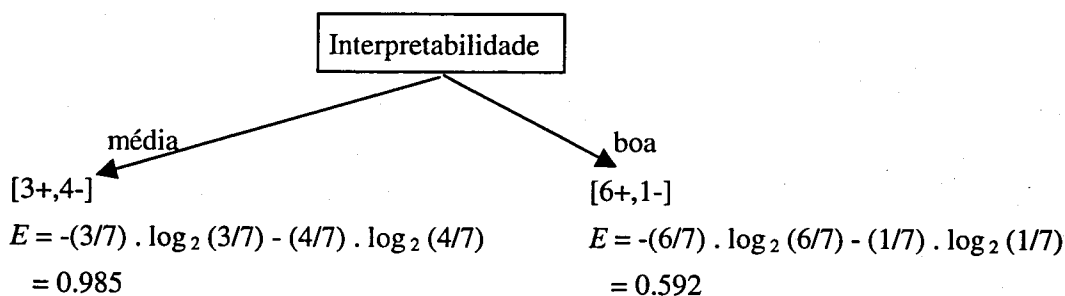
Todos os exemplos com *Conteúdo Técnico/Científico* = *muito bom* são também exemplos positivos de Comprar. Assim, este nó da árvore torna-se um nó folha com a classificação *Comprar* = *Sim*. Em contraste, os descendentes correspondentes a *Conteúdo Técnico/Científico* = *médio* e *Conteúdo Técnico/Científico* = *bom* têm uma entropia superior a 0, e uma árvore de decisão será construída abaixo desses nós. O processo de selecção de um

novo atributo e partição dos exemplos de treino é repetido para cada nó descendente não terminal, mas dessa vez usando só os exemplos de treino associados com esse nó.

Os atributos que foram incorporados mais acima na árvore são excluídos, daí que cada atributo só pode aparecer uma vez na árvore. Este processo continua para cada nova folha até uma das seguintes condições ser encontrada : todos os atributos estão já incluídos na árvore ou os exemplos de treino associados com esse nó folha têm todos o mesmo valor do atributo alvo.

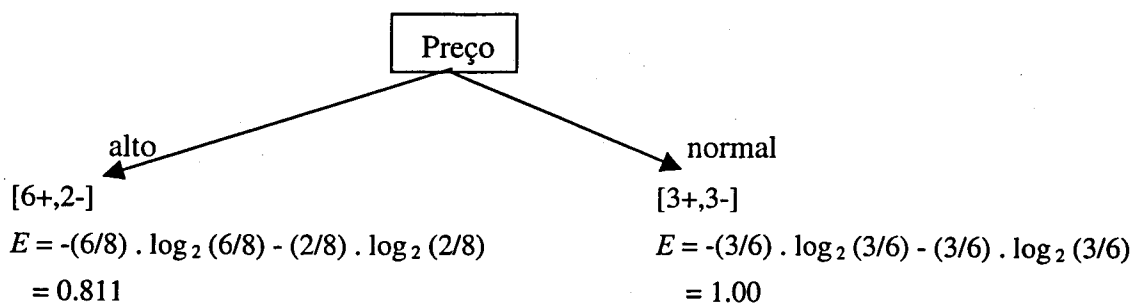
$S: [9+,5-]$

$$\begin{aligned} \text{Entropia}(S) &= \text{Entropia}([9+,5-]) = -p_+ \cdot \log_2 p_+ - p_- \cdot \log_2 p_- \\ &= -(9/14) \cdot \log_2 (9/14) - (5/14) \cdot \log_2 (5/14) \\ &= 0.940 \end{aligned}$$



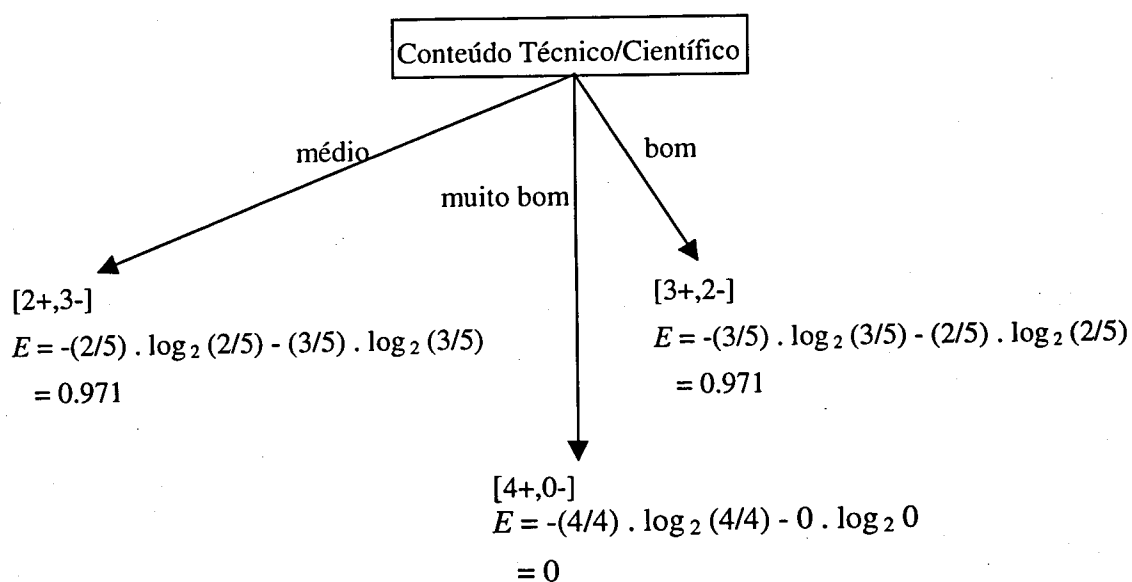
$$\begin{aligned} \text{Ganho}(S, \text{Interpretabilidade}) &= \text{Entropia}(S) - \sum_{v \in \{\text{média}, \text{boa}\}} \frac{|S_v|}{|S|} \cdot \text{Entropia}(S_v) \\ &= 0.940 - (7/14) \cdot 0.985 - (7/14) \cdot 0.592 \\ &= 0.151 \end{aligned}$$

Figura 2.6 – Ganho de informação do atributo Interpretabilidade



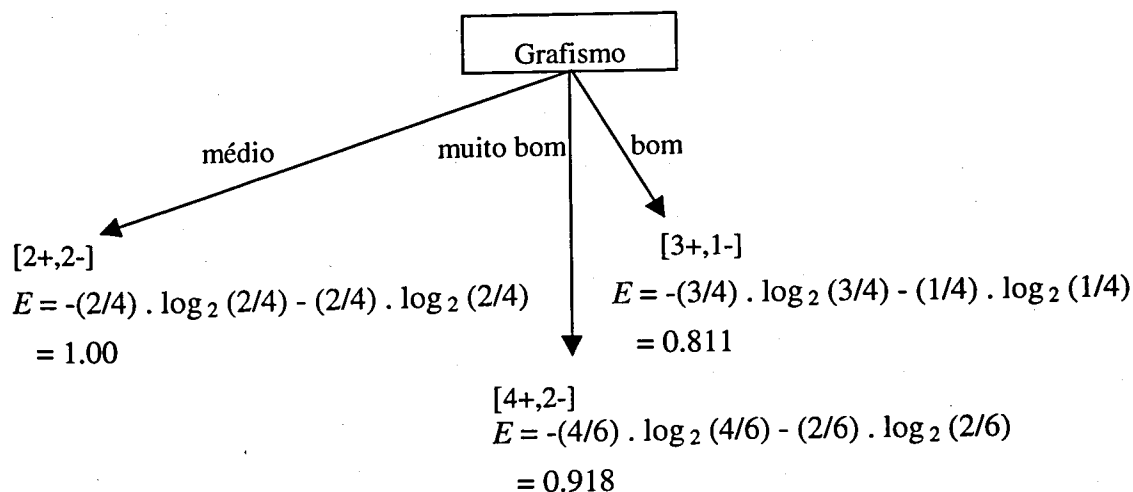
$$\begin{aligned} \text{Ganho}(S, \text{Preço}) &= \text{Entropia}(S) - \sum_{v \in \{\text{normal}, \text{alto}\}} \frac{|S_v|}{|S|} \cdot \text{Entropia}(S_v) \\ &= 0.940 - (8/14) \cdot 0.811 - (6/14) \cdot 1 \\ &= 0.048 \end{aligned}$$

Figura 2.7 - Ganho de informação do atributo Preço



$$\begin{aligned}
 \text{Ganho}(S, \text{Conteúdo Técnico/Científico}) &= \text{Entropia}(S) - \sum_{v \in \{\text{médio}, \text{bom}, \text{muito bom}\}} \frac{|S_v|}{|S|} \cdot \text{Entropia}(S_v) \\
 &= 0.940 - (5/14) \cdot 0.971 - (4/14) \cdot 0 - (5/14) \cdot 0.971 \\
 &= 0.246
 \end{aligned}$$

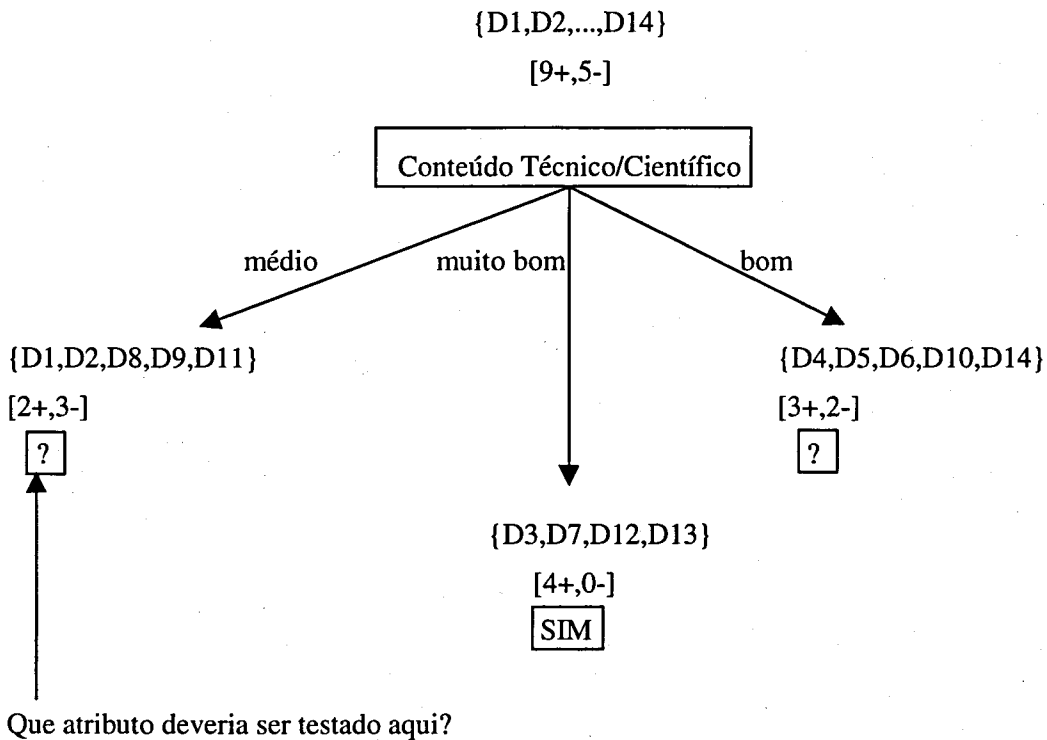
Figura 2.8 - Ganho de informação do atributo Conteúdo Técnico/Científico



$$\begin{aligned}
 \text{Ganho}(S, \text{Grafismo}) &= \text{Entropia}(S) - \sum_{v \in \{\text{médio}, \text{bom}, \text{muito bom}\}} \frac{|S_v|}{|S|} \cdot \text{Entropia}(S_v) \\
 &= 0.940 - (4/14) \cdot 1 - (6/14) \cdot 0.918 - (4/14) \cdot 0.811 \\
 &= 0.029
 \end{aligned}$$

Figura 2.9 - Ganho de informação do atributo Grafismo

A figura 2.10 ilustra os cálculos do ganho de informação para a etapa seguinte da construção da árvore. No nó relativo ao ramo da árvore Conteúdo Técnico/Científico, como os cálculos do ganho de informação demonstram, o atributo a ser testado neste nó deverá ser o atributo Interpretabilidade. A árvore de decisão final aprendida pelo algoritmo ID3 dos exemplos de treino da tabela 2.7 é ilustrada na figura 2.5.



$$S_{\text{médio}} = \{D1,D2,D8,D9,D11\}$$

$$\text{Ganho}(S_{\text{médio}}, \text{Interpretabilidade}) = 0.971 - (3/5) \cdot 0 - (2/5) \cdot 0 = 0.971$$

$$\text{Ganho}(S_{\text{médio}}, \text{Grafismo}) = 0.971 - (2/5) \cdot 0 - (2/5) \cdot 1 - (1/5) \cdot 0 = 0.571$$

$$\text{Ganho}(S_{\text{médio}}, \text{Preço}) = 0.971 - (2/5) \cdot 1 - (3/5) \cdot 0.918 = 0.020$$

Figura 2.10 – Cálculos do ganho de informação para a etapa seguinte da construção da Árvore de Decisão

2.3.5 Pesquisa no Espaço de Hipóteses da Aprendizagem de Árvores de Decisão

O algoritmo ID3 pode ser caracterizado pela pesquisa no espaço de hipóteses por uma hipótese que ajuste os exemplos de treino. O espaço de hipóteses pesquisado pelo algoritmo ID3 é o conjunto de árvores de decisão possíveis. O algoritmo ID3 realiza uma pesquisa pelo espaço de hipóteses, começando com uma árvore vazia, e depois considerando progressivamente hipóteses mais elaboradas na pesquisa de árvores de decisão que correctamente classificam os dados de treino. A função de avaliação que guia esta pesquisa é a medida do ganho de informação.

Em termos do espaço de pesquisa e estratégia de pesquisa do algoritmo ID3, podem-se fazer algumas considerações em relação às suas capacidades e limitações :

- O espaço de hipóteses de todas as árvores de decisão do algoritmo ID3, é um espaço completo de funções de valores discretos. Como todas as funções de valores discretos podem ser representadas por uma árvore de decisão, o algoritmo ID3 evita um dos maiores riscos dos métodos que pesquisam espaços de hipóteses (tal como os métodos que consideram só hipóteses conjuntivas), o de que o espaço de hipóteses poder não conter a função alvo.
- O algoritmo ID3 mantém só uma hipótese corrente à medida que pesquisa o espaço de árvores de decisões. Isto contrasta com o algoritmo "Candidate-Elimination" que mantém o conjunto de todas as hipóteses consistentes com os exemplos de treino disponíveis. Determinando só uma única hipótese, o algoritmo ID3 perde a capacidade de representar todas as hipóteses consistentes.
- O algoritmo ID3 na sua forma pura não realiza o backtrack na sua pesquisa. Uma vez seleccionado um atributo para testar num determinado nível da árvore, nunca mais faz o backtrack para reconsiderar essa escolha. É possível convergir para uma solução óptima local que não é uma solução óptima global. Esta solução óptima local é menos desejável do que outras árvores que poderiam ser encontradas ao longo de um ramo diferente da pesquisa. No entanto existe uma extensão ao algoritmo ID3 que inclui a possibilidade de fazer o backtrack.
- O algoritmo ID3 usa a estatística e todos os exemplos de treino em cada etapa da pesquisa de forma a refinar a hipótese. Isto contrasta com outros métodos que fazem decisões incrementalmente, a partir dos exemplos de treino ("Find-S" e "Candidate-Elimination"). Uma vantagem de usar propriedades estatísticas de todos os exemplos (como por exemplo o ganho de informação) é que o resultado da pesquisa é muito menos sensível a erros nos exemplos de treino. O ID3 pode ser facilmente estendido para tratar dados de treino com ruído modificando o critério de forma a aceitar hipóteses que ajustam de uma forma imperfeita os dados de treino.

2.3.6 Limitação Indutiva na Aprendizagem de Árvores de Decisão

A política pela qual o algoritmo ID3 generaliza a partir de exemplos de treino observados para classificar instâncias não observadas, ou seja, a base da escolha entre várias árvores de decisão consistentes com os exemplos de treino, baseia-se na escolha de árvores de decisão mais pequenas e na escolha de árvores de decisão que colocam os atributos com maior ganho de informação perto da raiz da árvore.

O algoritmo ID3 pesquisa num espaço de hipóteses completo (capaz de representar uma função qualquer), mas a pesquisa é feita de uma forma incompleta através desse espaço, de hipóteses simples para complexas, até ser encontrada uma hipótese consistente com os dados. A sua limitação indutiva é só uma consequência da ordenação das hipóteses efectuada pela estratégia de pesquisa. O espaço de hipóteses não inclui mais limitações adicionais.

O "Version Space" do algoritmo "Candidate-Elimination" procura num espaço de hipóteses incompleto (só um subconjunto dos potenciais conceitos é que podem ser representados). O algoritmo procura este espaço completamente, encontrando todas as hipóteses consistentes com os exemplos de treino. A sua limitação indutiva é só uma consequência do poder expressivo da sua representação de hipóteses. A sua estratégia de pesquisa não introduz limitações adicionais. Enquanto que a limitação indutiva do ID3 decorre da sua estratégia de pesquisa (limitação preferencial), a limitação indutiva do "Candidate-Elimination" decorre da

definição do seu espaço de pesquisa (limitação restritiva). Tipicamente, uma limitação preferencial é mais desejável do que uma limitação restritiva, pois disponibiliza um espaço de hipóteses completo que assegura a inclusão da função alvo.

2.3.7 Sobreajustamento dos Dados

Com o algoritmo ID3 pode haver um sobreajustamento dos exemplos de treino. Isto pode acontecer no caso de existir ruído nos dados, ou se o número de exemplos de treino for muito pequeno para produzir uma amostra representativa da função alvo.

Uma hipótese $h \in H$ sobreajusta os dados de treino, se existir uma hipótese alternativa $h' \in H$, tal que h tem um erro menor que h' sobre os exemplos de treino, mas h' tem um erro menor do que h sobre a distribuição de todas as instâncias. O sobreajustamento é uma dificuldade para a aprendizagem de árvores de decisão. Num estudo experimental do ID3 [Mingers-89], o sobreajustamento levou a uma diminuição da exactidão das árvores de decisão aprendidas entre 10 a 25%.

Existem várias abordagens para evitar o sobreajustamento na aprendizagem de árvores de decisão, que podem ser agrupadas em duas classes:

- abordagens que param a construção da árvore antes de atingir o ponto em que classificam perfeitamente os dados de treino,
- abordagens que permitem à árvore sobreajustar os dados, e depois retiraram alguns ramos na árvore ("post-pruning").

Apesar das primeiras abordagens parecerem muito directas, as segundas abordagens revelam-se com mais sucesso na prática. Isto deve-se à dificuldade de nas primeiras abordagens se estimar precisamente quando parar a construção da árvore.

Independentemente da abordagem a utilizar é necessário ter um critério para determinar o tamanho correcto da árvore. As abordagens incluem :

- o uso de um conjunto de exemplos, distintos dos exemplos de treino, para avaliar a utilidade da eliminação de ramos na árvore,
- aplicação de um teste estatístico para estimar a melhoria da expansão de um nó (ou eliminação de ramos) no ajustamento de dados ainda não observados. Por exemplo, [Quinlan-86] usa um teste para estimar se a expansão de um nó melhora o desempenho sobre toda a distribuição de instâncias, ou só na amostra de dados de treino,
- o uso de uma medida da complexidade de codificação dos exemplos de treino e da árvore de decisão, parando a construção quando o tamanho da codificação é minimizado. Esta abordagem é baseada numa heurística chamada de princípio do tamanho da descrição mínimo [Quinlan-89;Mehta-95].

O uso de dois conjuntos de exemplos, um de treino e outro de validação é mais comum. Os dados disponíveis são separados em dois conjuntos de exemplos: o conjunto de treino, que é usado para formar a hipótese aprendida, e um conjunto separado de validação, que é usado para avaliar a exactidão dessa hipótese sobre dados subsequentes e em particular, para avaliar o impacto da eliminação de ramos na árvore. A razão do uso do conjunto de validação baseia-se no facto de que as flutuações nos erros do conjunto de treino não devem aparecer da mesma forma no conjunto de validação. Daí, poder-se esperar do conjunto de validação um teste seguro contra o sobreajustamento dos dados de treino. É importante que o conjunto de validação seja suficientemente grande para possibilitar uma amostra estatística significativa de instâncias.

2.3.7.1 Redução do Erro com a Eliminação de Ramos

Uma abordagem para prevenir o sobreajustamento usando o conjunto de validação [Quinlan-87] é considerar cada nó da árvore de decisão como candidato à eliminação das suas sub-árvores, ficando esse nó como nó folha e atribuindo-lhe a classificação mais comum dos exemplos de treino relacionados com esse nó. As sub-árvores só são removidas se o resultado dessa remoção não provocar um desempenho pior do que a anterior sobre o conjunto de validação. Assim, alguns erros nos dados de treino que levaram à construção incorrecta da árvore são reparados. As sub-árvores são iterativamente eliminadas até que a sua eliminação diminua a exactidão da árvore sobre o conjunto de validação. A maior limitação desta abordagem é quando a quantidade de dados é escassa para fazer a separação dos exemplos no conjunto de treino e no conjunto de validação.

2.3.7.2 Regra Pós-Eliminação de Ramos

Outro método alternativo que se revela útil quando a quantidade de dados é escassa é a regra pós-eliminação de ramos ("post-pruning").

Este método envolve as seguintes etapas:

1. Inferir a árvore de decisão dos exemplos de treino, construindo a árvore até que os dados de treino sejam ajustados e permitindo que ocorra sobreajustamento dos exemplos de treino.
2. Converter a árvore aprendida num conjunto de regras equivalente criando uma regra para cada caminho desde o nó raiz até ao nó folha.
3. Remover qualquer pré-condição nas regras que resultem num melhoramento na estimativa da exactidão.
4. Ordenar as regras alteradas pela estimativa da exactidão, e considerá-las nessa sequência para classificar instâncias subsequentes.

Um método para estimar a exactidão da regra consiste em usar o conjunto de exemplos de validação disjunto do conjunto de exemplos de treino.

Existem algumas razões pelas quais converte-se a árvore de decisão em regras antes da eliminação das condições:

- Permite a distinção dos diferentes contextos em que os nós de decisão são usados.
- Remove a distinção entre atributos de teste que ocorrem perto da raiz da árvore e aqueles que ocorrem perto dos nós folha.
- Melhora a compreensão dos humanos do conceito alvo.

2.3.8 Incorporação de Atributos com Valores Contínuos

A definição inicial do algoritmo ID3, refere que o valor do atributo alvo que é previsto deve ter um valor discreto, e os atributos testados nos nós de decisão devem ter também valores discretos. Esta segunda restrição pode ser removida de forma a que atributos de decisão com valores contínuos possam ser incorporados na árvore aprendida. Isto é conseguido pela definição dinâmica de novos atributos com valores discretos que fazem a partição do atributo

contínuo num conjunto discreto de intervalos. Em particular, para um atributo contínuo, o algoritmo pode dinamicamente criar um novo atributo booleano A_c que é verdade se $A < c$ (ou $A > c$) e falso caso contrário. O limite c escolhido terá que ser o que produz o maior ganho de informação. Ordenando os exemplos de acordo com o atributo contínuo A , e identificando exemplos adjacentes que diferem na sua classificação alvo, pode-se gerar um conjunto de limites candidatos que têm o valor a meio caminho dos correspondentes valores de A . Estes limites candidatos podem ser avaliados calculando o ganho de informação associado com cada um [Fayyad-91] sendo seleccionado o que tiver maior ganho de informação. Este atributo booleano criado dinamicamente, poderá assim competir com os outros atributos para a construção da árvore. [Fayyad-93] propõe uma extensão a esta abordagem que divide um atributo contínuo em múltiplos intervalos em vez de dois intervalos baseados num único limite.

2.3.9 Medidas Alternativas para Seleccionar Atributos

Existe uma característica inerente à medida do ganho de informação, que é favorecer atributos com muitos valores sobre aqueles que têm poucos valores. Considerando o caso extremo de adicionar o atributo data à tabela 2.7 (que tem um grande número de valores possíveis), verificava-se que tinha o maior ganho de informação de todos os atributos. Este atributo só por si previa perfeitamente o atributo alvo sobre os dados de treino. Assim este atributo seria seleccionado para o nó raiz da árvore com profundidade um, que perfeitamente classificava os exemplos de treino. Esta árvore de decisão classificaria mal exemplos subsequentes ainda não observados. Uma forma de evitar esta dificuldade é seleccionar atributos de decisão baseados noutra medida que não o ganho de informação. Uma medida alternativa que foi usada com sucesso é o rácio de ganho [Quinlan-86]. O rácio de ganho penaliza atributos como a data pela incorporação de um atributo chamado de informação de separação, que é sensível à forma como o atributo largamente e uniformemente separa os dados:

$$\text{InformaçãoSeparação}(S,A) = - \sum_{i=1}^c \frac{|S_i|}{|S|} \cdot \log_2 \frac{|S_i|}{|S|}$$

em que S_1 até S_c são os subconjuntos de exemplos resultantes da partição de S pelos valores c do atributo A . A informação de separação é a entropia de S relativamente aos valores do atributo A . Isto contrasta com o uso anterior da entropia, em que se considera a entropia S relativamente ao atributo alvo cujo valor é para ser previsto pela árvore aprendida.

A medida do rácio de ganho é definida em termos da medida do ganho de informação e da medida da informação de separação :

$$\text{RacioGanho}(S,A) = \frac{\text{Ganho}(S,A)}{\text{InformaçãoSeparação}(S,A)}$$

A informação de separação desencoraja a selecção de atributos com muitos valores distribuídos uniformemente. No caso de n exemplos que são completamente separados pelo atributo A (por exemplo, data) a informação de separação é $\log_2 n$. Em contraste, um atributo booleano B que separa os mesmos n exemplos exactamente a meio tem informação de separação 1. Se A e B produzem o mesmo ganho de informação, então B claramente terá um valor superior do rácio de ganho.

Uma questão importante no uso do rácio de ganho é que o denominador pode ser zero ou muito pequeno quando $|S_i| \approx |S|$ para um dos S_i . Isto faz com que o rácio de ganho seja indefinido ou

muito grande para atributos que têm o mesmo valor para quase todos os membros de S . Para evitar a selecção de atributos puramente nesta base, podemos utilizar uma heurística que consiste em primeiro calcular o ganho de informação de cada atributo, e depois aplicar o rácio de ganho considerando só os atributos com a média do ganho de informação [Quinlan-86].

2.3.10 Tratamento de Exemplos de Treino com Falta de Valores nos Atributos

Em certos casos, os dados disponíveis podem ter falta de valores em alguns atributos. Em alguns casos, é comum estimar o valor de um atributo em falta a partir de outros exemplos em que esse atributo tem um valor conhecido.

Considere-se a situação em que o ganho de informação tem que ser calculado no nó n da árvore de decisão para avaliar se o atributo A é o melhor atributo a ser testado nesse nó de decisão. Supondo que $(x, c(x))$ é um dos exemplos de treino em S e que o valor $A(x)$ é desconhecido.

Uma estratégia para lidar com o valor de um atributo em falta é atribuir-lhe o valor mais comum entre os exemplos de treino no nó n . Alternativamente, deve-se atribuir o valor mais comum entre os exemplos de treino no nó n que tem a classificação $c(x)$. O exemplo de treino elaborado usando este valor estimado para $A(x)$ pode então ser usado directamente pelo algoritmo de aprendizagem da árvore de decisão.

Um procedimento mais complexo é atribuir uma probabilidade a cada um dos possíveis valores de A em vez de simplesmente atribuir o valor mais comum de $A(x)$. Estas probabilidades podem ser estimadas a partir das frequências observadas dos vários valores de A entre os exemplos no nó n .

Estas fracções de exemplos são usadas para calcular o ganho de informação e podem ser aplicadas depois de aprendidas, para classificar novas instâncias cujos valores dos atributos não são conhecidos. Neste caso, a classificação de novas instâncias é simplesmente a classificação mais provável, calculada pela soma dos pesos (frequências observadas) das diferentes classificações nos nós folha da árvore. Este método é usado no algoritmo C4.5 [Quinlan-93], que foi desenvolvido a partir do algoritmo ID3.

2.3.11 Tratamento de Exemplos de Treino com Custos Diferentes

Em algumas tarefas de aprendizagem os atributos das instâncias têm custos associados (custos monetários, custos no conforto do paciente, entre outros). Nestas tarefas, é preferível árvores de decisão que usam atributos de baixo custo onde for possível, recorrendo a atributos de alto custo só quando necessário para produzir classificações. O algoritmo ID3 pode ser modificado (tomando em atenção os custos dos atributos) com a introdução de um termo de custo na medida para a selecção do atributo. Um exemplo possível, é dividir o ganho de informação pelo custo do atributo, possibilitando assim a preferência por atributos com menores custos. Apesar desta medida sensível ao custo não garantir a obtenção de uma árvore de decisão óptima sensível aos custos, possibilita a pesquisa em favor dos atributos de baixo custo.

[Tan90] e [Tan-93] numa aplicação de aprendizagem da classificação de objectos por um robot aplicou com sucesso em vez do ganho de informação a seguinte medida

$$\frac{\text{Ganho}^2(S, A)}{\text{Custo}(A)}$$

Nunes (1988) numa aplicação de aprendizagem de regras de diagnóstico médico usou também com sucesso a seguinte medida em vez do ganho de informação

$$\frac{2^{\text{Ganho}(S, A)} - 1}{(\text{Custo}(A) + 1)^w}$$

em que $w \in [0, 1]$ e é uma constante que determina a importância relativa do custo versus o ganho de informação.

2.4 Redes Neurais

2.4.1 Introdução

As redes neuronais (RNs) possibilitam um método prático para aprender funções com valores reais, valores discretos e vectores a partir dos exemplos de treino. A aprendizagem de RNs é robusta a erros nos dados de treino e foi aplicada com sucesso a problemas de interpretação, com sensores de dados do mundo real, como a interpretação visual de cenários (reconhecimento de faces – [Cottrell-90]), reconhecimento da fala [Lang-90], aprendizagem de estratégias de controlo de robots [Pomerleau-93], entre outros. Outra área de grande aplicabilidade é na avaliação da segurança dos Sistemas Eléctricos de Energia [Wehenkel-98].

O estudo das RNs foi inspirado em parte pela constatação de que os sistemas de aprendizagem neurobiológicos são constituídos por redes muito complexas de neurónios interligados. As RNs são constituídas por um denso conjunto interligado de unidades simples, em que cada unidade recebe como “input” um determinado número de valores reais (possivelmente o “output” de outras unidades) e produz um valor real único como “output” (que provavelmente se tornará num “input” para outras unidades).

Estima-se que o cérebro humano contenha uma densa rede interligada de aproximadamente 10^{11} neurónios, cada um ligado, em média, a mais 10^4 neurónios. A actividade neurológica é tipicamente excitada ou inibida através das ligações a outros neurónios. O tempo mais rápido do “switch” de um neurónio é na ordem dos 10^{-3} segundos, um pouco mais lento comparado com um switch de um computador, de 10^{-10} segundos. Os humanos são capazes de fazer decisões complexas surpreendentemente rápidas. Por exemplo, necessitam de aproximadamente 10^{-1} segundos para visualmente reconhecerem a sua mãe. De notar que a sequência de disparos dos neurónios que podem ocorrer durante o intervalo de 10^{-1} segundos não possibilita ir além de poucas centenas de etapas, dada a velocidade do “switch” dos neurónios. Isto levou a especular-se que a capacidade do processamento de informação dos sistemas neurobiológicos devia seguir um processamento paralelo operando em representações distribuídas sobre muitos neurónios.

O intuito dos sistemas de RNs é adquirir informaticamente este tipo de processamento paralelo a partir de representações distribuídas. A maior parte do “software” de RNs corre em máquinas emulando processos distribuídos, apesar das versões mais rápidas dos algoritmos terem sido também implementadas em máquinas paralelas e com “hardware” especializado, concebido especificamente para aplicações de RNs. Apesar das RNs terem sido motivadas pelos sistemas neurobiológicos, existem muitos sistemas neurobiológicos que não são modelados por RNs, e muitas características das RNs são inconsistentes com os sistemas neurobiológicos. Historicamente, dois grupos de investigadores trabalharam com redes neuronais. Um grupo foi motivado pelo objectivo de usar as RNs para estudar e modelar processos de aprendizagem biológica. Um segundo grupo foi motivado pelo objectivo de obter algoritmos bastante eficientes de Aprendizagem Automática independentemente dos algoritmos espelharem processos biológicos.

2.4.2 Representações de Redes Neurais

Os sistemas de redes neuronais são constituídos por três tipos de unidades:

- unidades de “input”,
- unidades “escondidas”,

- unidades de “output”.

Cada um destes tipos de unidades formam os respectivos níveis. Numa rede neuronal (RN) existe um nível de “input”, um nível de “output” e podem existir um ou mais níveis “escondidos”. A figura 2.11 ilustra a representação típica de muitas redes neuronais.

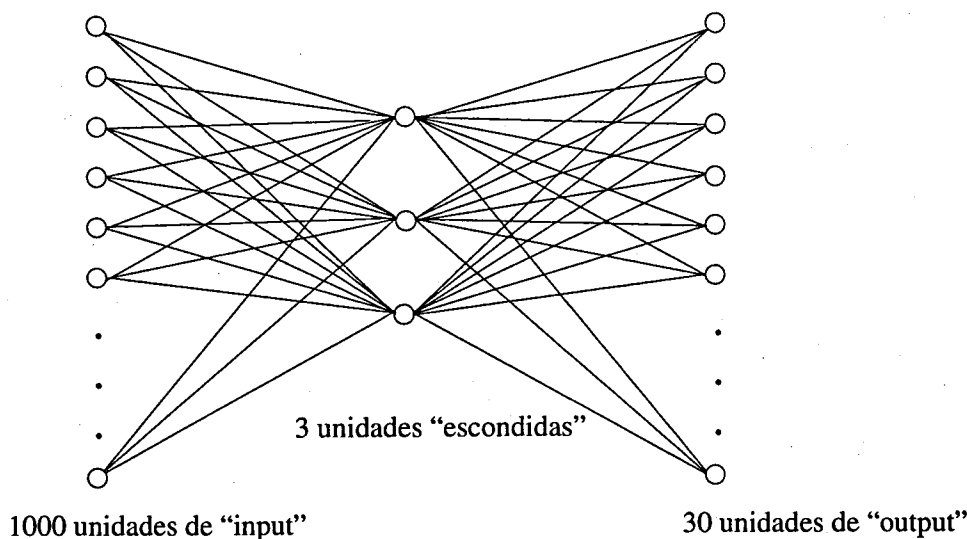


Figura 2.11 – Representação típica de uma Rede Neuronal

Neste caso a RN é constituída por um nível de “input”, um nível “escondido” e um nível de “output”. As unidades são interligadas em níveis que formam um grafo directo acíclico. Cada nó na rede (círculo), corresponde a uma unidade, e as linhas que chegam às unidades “escondidas” e às unidades de “output” são os “inputs” de cada uma dessas unidades. As unidades “escondidas” têm esse nome porque o seu “output” é disponível só dentro da rede e não está disponível como “output” global da rede. Cada uma das unidades “escondidas” calcula um valor real como “output”, a partir do peso combinado dos seus “inputs”.

Os “outputs” das unidades “escondidas” são usados como “inputs” de outras unidades de um outro nível “escondido, ou mais frequentemente, como “inputs” das unidades do nível de “output”. Os valores de “output” das unidades do nível de “output” determinam a decisão a tomar pela rede neuronal. O sistema Alvin construído por [Pomerleau-93] é um exemplo de um protótipo de aprendizagem de RNs que usa uma RN para conduzir um veículo a uma velocidade de cerca de 120km por hora numa auto-estrada num percurso com cerca de 150 kms. O “output” da rede é a direcção em que o veículo é guiado.

A RN foi treinada para imitar a condução de um humano durante 5 minutos. A RN é constituída por quatro unidades que recebem “inputs” directamente dos 30x32 “pixels” da imagem. Cada uma destas quatro unidades “escondidas” calcula um valor real como “output” a partir do peso combinado dos seus 960 “inputs”. Os “outputs” destas unidades “escondidas” são usadas como “inputs” de um nível com 30 unidades de “output”. Cada unidade de “output” corresponde a uma direcção de condução, e os valores de “output” destas unidades determinam qual a direcção de condução mais recomendada. O sistema Alvin utiliza o algoritmo “Backpropagation”.

2.4.3 Problemas Apropriados para Aprendizagem de Redes Neurais

A aprendizagem de RNs é apropriada para problemas em que os dados de treino correspondem a dados de sensores, como por exemplo, de câmaras e microfones. As RNs são também aplicáveis a problemas em que representações simbólicas são frequentemente usadas. O algoritmo "Backpropagation" é o mais usado na técnica de aprendizagem de RNs. É apropriado para problemas com as seguintes características :

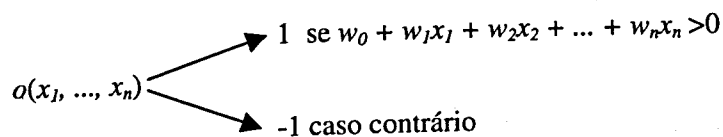
- As instâncias são representadas por muitos pares de atributo-valor. A função alvo a ser aprendida é definida sobre instâncias que podem ser descritas por um vector de características predefinidas (como por exemplo o valor dos "pixels" no exemplo Alvin).
- A função alvo pode ser discreta, real, ou um vector de vários atributos discretos ou reais. Por exemplo, no sistema Alvin o "output" é um vector de 30 atributos, cada um correspondendo a uma indicação da direcção de condução a ser tomada.
- Os exemplos de treino podem conter erros.
- A avaliação rápida da função alvo aprendida deve ser fundamental para aplicá-la a instâncias subsequentes.
- O interesse dos humanos em compreenderem a função alvo aprendida não é importante.

Existem vários tipos de unidades alternativas para as RNs. Entre outras, as mais utilizadas são os "perceptrons", as unidades lineares e as unidades sigmóides.

2.4.4 "Perceptrons"

Alguns sistemas de RNs são baseados numa unidade chamada "perceptron", ilustrada na figura 2.12 [Mitchell-97].

Um "perceptron" recebe como "input" um vector de valores reais e realiza uma combinação linear desses "inputs", emitindo o valor 1 se o resultado é maior que um limite e -1 caso contrário. Mais precisamente, dados os "inputs" x_1 até x_n , o "output" $o(x_1, \dots, x_n)$ calculado pelo "perceptron" é



em que cada w_i é uma constante real, ou peso que determina o contributo do "input" x_i para o "perceptron" de "output"; $-w_0$ é um limite que a combinação pesada dos "inputs" $w_1x_1 + \dots + w_nx_n$ deve superar para o "perceptron" ter como "output" 1. Para simplificar a notação, inclui-se uma constante de "input" adicional $x_0=1$, permitindo escrever a desigualdade acima como

$$\sum_{i=0}^n w_i \cdot x_i > 0$$

ou na forma vectorial como $\vec{w} \cdot \vec{x} > 0$. A função do “perceptron” pode-se escrever como

$$o(\vec{x}) = \text{sgn}(\vec{w} \cdot \vec{x}) \text{ em que } \text{sgn}(y) \begin{cases} 1 & \text{se } y > 0 \\ -1 & \text{caso contrário} \end{cases}$$

Aprender um “perceptron” envolve escolher vários valores para os pesos $w_0, \dots, w_n$. Assim, o espaço das hipóteses candidatas H , considerado na aprendizagem do “perceptron” é o conjunto de todos os vectores possíveis de pesos reais.

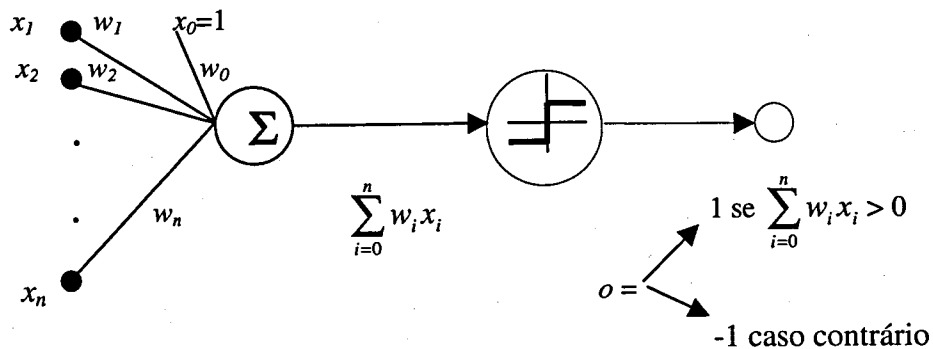


Figura 2.12 – Um “Perceptron”

2.4.4.1 Poder de Representação dos “Perceptrons”

Um “perceptron” pode ser visto como uma superfície de decisão de um hiperplano num espaço de instâncias (pontos) com dimensão n . O “perceptron” tem “output” 1 para instâncias que se encontram de um dos lados do hiperplano e tem “output” -1 para instâncias que se encontram do outro lado como ilustrado na figura 2.13 [Mitchell-97]. A equação para este hiperplano de decisão é $\vec{w} \cdot \vec{x} = 0$.

Alguns conjuntos de exemplos positivos e negativos não podem ser separados por nenhum hiperplano, pois isto só acontece com conjuntos de exemplos linearmente separáveis.

Um “perceptron” pode ser usado para representar muitas funções booleanas. Uma forma de usar um “perceptron” para implementar uma função E (assumindo os valores booleanos 1 (verdade) e -1 (falso)), consiste em utilizar os pesos $w_0 = -0.8$ e $w_1 = w_2 = 0.5$. Este “perceptron” pode representar a função **OU** substituindo o limite w_0 por -0.3. As funções E e **OU** podem ser vistas como casos especiais de funções m -de- n : funções em que pelo menos m dos n “inputs” do “perceptron” devem ser verdadeiros. A função **OU** corresponde a $m=1$ e a função E a $m=n$. Qualquer função m -de- n é facilmente representada usando um “perceptron”, com a atribuição do mesmo valor aos pesos dos “inputs” e um limite w_0 apropriado.

Os “perceptrons” podem representar todas as funções booleanas primitivas E , **OU**, **NE** ($\neg E$) e **NOU** ($\neg \text{OU}$). Algumas funções booleanas não podem ser representadas por um único “perceptron” único, como a função **XOU** (“**XOR**”) cujo valor é 1 se e só se $x_1 \neq x_2$. O conjunto de exemplos de treino não linearmente separáveis da figura 2.13 (b) corresponde à função **XOU**.

A possibilidade dos “perceptrons” poderem representar as funções *E*, *OU*, *NE*, e *NOU* é importante pois todas as funções booleanas podem ser representadas por uma rede de unidades (“perceptrons”) interligadas baseadas nestas primitivas, com dois níveis de profundidade, em que os “inputs” são fornecidos a múltiplas unidades, e os “outputs” dessas unidades são “inputs” de um segundo estado. Uma forma é representar a função booleana numa forma disjuntiva (isto é, como uma disjunção (*OU*) de conjunções (*Es*) de “inputs” e suas negações). Um “input” para um “perceptron” *E* nega-se mudando o sinal do peso do “input” correspondente.

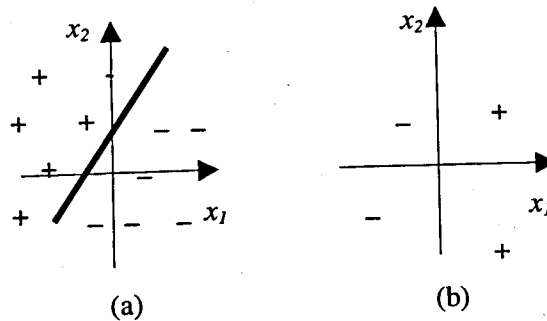


Figura 2.13 – Superfície de Decisão

A aprendizagem de redes multinível de unidades é útil, pois podem representar uma variedade rica de funções, enquanto as unidades isoladas não.

2.4.4.2 Regra de Treino do “Perceptron”

Para o “perceptron” produzir o “output” correcto (± 1) para cada um dos seus exemplos de treino é necessário aprender o vector dos pesos. Existem algoritmos para resolver esta questão, nomeadamente, a regra do “perceptron” e a regra delta.

Uma forma de aprender um vector aceitável de pesos consiste em começar com pesos aleatórios, e aplicar iterativamente o “perceptron” a cada exemplo de treino, modificando os pesos do “perceptron” sempre que este classifique incorrectamente um exemplo. Este processo é repetido iterativamente pelos exemplos de treino tantas vezes quantas necessárias até o “perceptron” classificar todos os exemplos correctamente. Os pesos são modificados em cada etapa de acordo com a regra de treino do “perceptron”, que revê o peso de w_i associado com o “input” x_i de acordo com a regra

$$w_i \leftarrow w_i + \Delta w_i$$

em que $\Delta w_i = \eta (t - o) x_i$, t é “output” alvo para o exemplo de treino corrente, o é o “output” gerado pelo “perceptron” e η é uma constante positiva chamada rácio de aprendizagem. O papel do rácio de aprendizagem é moderar o grau em que o peso é alterado em cada etapa, sendo-lhe frequentemente atribuído um valor pequeno (por exemplo, 0.1).

Esta regra pode ser aplicada a problemas em que os dados são linearmente separáveis, não sendo assegurada a convergência se os dados não são linearmente separáveis.

2.4.4.3 Descida da Encosta

A regra delta resolve o problema da convergência ao conceito alvo quando os exemplos de treino não são linearmente separáveis. A ideia chave da regra delta é o uso da descida da encosta para pesquisar o espaço de hipóteses de vectores de pesos, de forma a encontrar o vector de pesos que melhor ajuste os exemplos de treino. A descida da encosta é a base do algoritmo “Backpropagation”, que aprende redes com muitas unidades interligadas.

A regra delta treina um “perceptron” sem limite, isto é, uma unidade linear que o “output” o é dado por

$$o(\vec{x}) = \vec{w} \cdot \vec{x}.$$

Esta unidade linear corresponde ao primeiro estado do “perceptron”, sem o limite.

O erro de treino de uma hipótese (vector de pesos) relativo aos exemplos de treino pode ser medido por

$$E(\vec{w}) = \frac{1}{2} \cdot \sum_{d \in D} (t_d - o_d)^2$$

em que D é o conjunto de exemplos de treino, t_d é o “output” alvo para o exemplo de treino d , e o_d é o “output” da unidade linear para o exemplo d .

Sob certas condições a hipótese que minimiza E é também a hipótese de H que melhor ajusta os dados de treino. O algoritmo da descida da encosta calcula um vector de pesos que minimiza E , começando com um vector inicial de pesos arbitrário, que é depois repetidamente alterado em pequenas etapas. Em cada etapa, o vector de pesos é alterado com o intuito de se atingir um erro global mínimo. A figura 2.14 ilustra todo o espaço de hipóteses de vectores de pesos possíveis e os seus E valores [Mitchell-97].

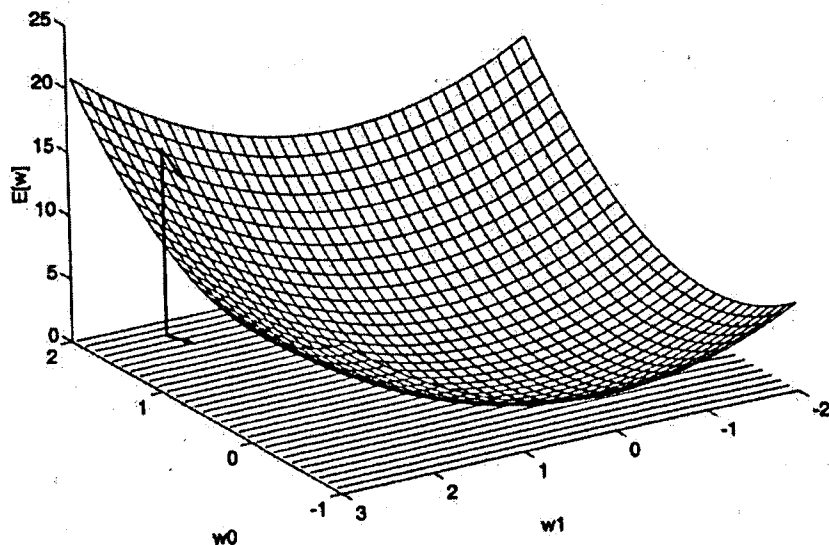


Figura 2.14 – Espaço de Hipóteses de Vectores de Pesos e respectivos Erros

Os eixos w_0 e w_1 representam valores possíveis para os pesos de uma unidade linear. Com dois pesos o plano w_0, w_1 representa todo o espaço de hipóteses. O eixo vertical indica o erro E relativo a um determinado conjunto de exemplos de treino. A superfície de erro da figura 2.14 evidencia os erros de todos os vectores de pesos considerados. A superfície de erro é sempre uma parabólica com um mínimo global, dependendo a sua posição dos exemplos de treino. A direcção do mínimo global na superfície de erro é calculada derivando E em relação a cada componente do vector $\vec{w}$. A derivada deste vector chama-se gradiente de E relativamente a $\vec{w}$ ($\nabla E(\vec{w})$).

$$\nabla E(\vec{w}) = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right]$$

$\nabla E(\vec{w})$ é também um vector cujos componentes são derivadas parciais de E relativamente a cada w_i . Quando interpretado como um vector no espaço de pesos, a descida especifica a direcção que aumenta E . Assim, o negativo do gradiente dá-nos a direcção da diminuição de E . A regra de treino para o gradiente da encosta é então

$$\vec{w} \leftarrow \vec{w} + \Delta \vec{w} \quad \text{em que} \quad \Delta \vec{w} = -\eta \cdot \nabla E(\vec{w}).$$

Esta regra pode também ser escrita na forma de componente

$$w_i \leftarrow w_i + \Delta w_i \quad \text{em que} \quad \Delta w_i = -\eta \cdot \frac{\partial E}{\partial w_i} \quad \text{e} \quad \frac{\partial E}{\partial w_i} = \sum_{d \in D} (t_d - o_d)(-x_{id})$$

$$\text{daí que } \Delta w_i = \eta \cdot \sum_{d \in D} (t_d - o_d)x_{id}$$

em que x_{id} é um componente de “input” do exemplo de treino d .

Na tabela 2.8 ilustra-se o algoritmo de descida da encosta para treinar uma unidade linear [Mitchell-97].

Em virtude da superfície de erro conter apenas um único mínimo global, este algoritmo converge para o vector de pesos com mínimo erro, independentemente dos exemplos de treino serem ou não linearmente separáveis (utilizando-se um rácio de aprendizagem η suficientemente pequeno). Se η for grande demais, corre-se o risco de sobreajustar o mínimo erro, daí ser comum alterar o algoritmo de modo a reduzir gradualmente o valor de η assim que o número de etapas de descida da encosta aumenta.

Tabela 2.8 – Algoritmo da Descida da Encosta para treinar uma unidade linear

Descida_da_Encosta (*Exemplos de treino*, η)

Cada *exemplo de treino* é um par da forma $(\vec{x}, t)$, em que $\vec{x}$ é um vector de valores de “input”, e t é o valor do “output” alvo. η é o rácio de aprendizagem.

- Inicializar cada w_i a um valor aleatório pequeno
- Até a condição terminal ser encontrada, Fazer
 - Inicializar cada Δw_i a zero
 - Para cada $(\vec{x}, t)$ nos *exemplos de treino*, Fazer
 - A unidade recebe como “input” a instância $\vec{x}$ e calcula o “output” o
 - Para cada peso (w_i) da unidade linear, Fazer

$$\Delta w_i \leftarrow \Delta w_i + \eta \cdot (t - o)x_i \quad (2.2)$$

- Para cada peso (w_i) da unidade linear, Fazer

$$w_i \leftarrow w_i + \Delta w_i \quad (2.3)$$

2.4.4.4 Descida Incremental da Encosta

As dificuldades práticas em aplicar a descida da encosta consistem no facto de que a convergência para o mínimo local pode por vezes ser bastante lento (pode requerer muitos milhares de etapas de descida da encosta), e se existirem múltiplos mínimos locais na superfície de erro, não há garantia que se encontre o mínimo global.

Uma variação comum à descida da encosta de forma a aliviar estas dificuldades é a descida incremental da encosta. Enquanto que a regra de treino da descida da encosta calcula as actualizações aos pesos depois de calcular o Δw_i de todos os exemplos de treino em D , a descida incremental da encosta actualiza os pesos incrementalmente, calculando o erro para cada exemplo individualmente, apoiando-se na regra delta :

$$\Delta w_i = \eta \cdot (t - o)x_i$$

Para modificar o algoritmo da descida da encosta da tabela 2.8, de forma a implementar a descida incremental da encosta, a equação (2.3) é eliminada e a equação (2.2) é substituída por

$$w_i \leftarrow w_i + \eta \cdot (t - o)x_i$$

As diferenças chave entre a descida da encosta e a descida incremental da encosta são :

- Na descida da encosta o erro é somado sobre todos os exemplos antes da actualização

dos pesos, enquanto que na descida incremental da encosta os pesos são actualizados cada exemplo de treino.

- A soma sobre múltiplos exemplos na descida da encosta requer mais cálculos por etapa de actualização de pesos.
- Em casos que há múltiplos mínimos locais relativamente a $\bar{E}(\bar{w})$, a descida incremental da encosta pode por vezes evitar a falha nesse mínimo local pois usa os vários $\nabla E_d(\bar{w})$ em vez de $\nabla E(\bar{w})$ para conduzir a pesquisa.

A regra de treino do “perceptron” e a regra delta são os dois algoritmos utilizados para iterativamente alterar os pesos.

A regra de treino do “perceptron” actualiza os pesos a partir do erro do “output” do “perceptron” (limitado com w_0), enquanto que a regra delta actualiza os pesos a partir do erro de uma combinação linear dos “inputs” (sem limite w_0).

Desde que os exemplos de treino sejam linearmente separáveis, a regra de treino do “perceptron” converge depois de um número finito de iterações para uma hipótese que classifica perfeitamente os dados de treino. A regra delta converge para a hipótese de erro mínimo, indiferentemente dos dados de treino serem linearmente separáveis ou não.

Ambos os algoritmos são usados na prática e aprendem iterativamente os pesos de um “perceptron”.

2.4.5 Redes Multinível e o Algoritmo Backpropagation

O tipo de rede multinível aprendido pelo algoritmo “Backpropagation” é capaz de expressar uma grande variedade de superfícies de decisão não lineares.

As unidades lineares não podem ser utilizadas para representar funções não lineares pois múltiplos níveis de unidades lineares em cascata produzem também funções lineares.

O “perceptron” tem um limite descontínuo, tornando-o indiferenciável e inadequado para a descida da encosta. É necessária uma unidade cujo “output” é uma função não linear e diferenciável dos seus “inputs”. Uma solução reside na unidade sigmóide (“sigmoid unit”), que é uma unidade parecida a um “perceptron” mas baseada numa função regular limitada (com w_0) e diferenciável. A unidade sigmóide é ilustrada na figura 2.15 [Mitchell-97].

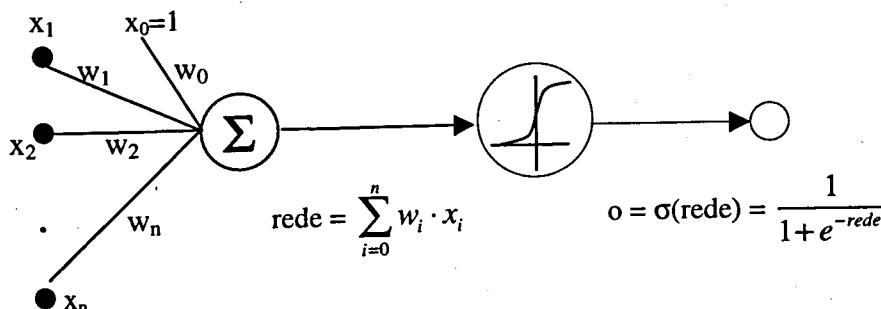


Figura 2.15 – Unidade Sigmóide

Como acontece com o “perceptron”, a unidade sigmóide primeiro realiza uma combinação linear dos seus “inputs”, aplicando depois o limite ao resultado. No caso da unidade sigmóide, contudo, o “output” é uma função contínua dos seus “inputs”. Mais precisamente, a unidade sigmóide calcula o “output” da seguinte forma

$$o = \sigma(\vec{w} \cdot \vec{x}) \quad \text{em que} \quad \sigma(y) = \frac{1}{1 + e^{-y}}$$

e σ é chamada de função sigmóide ou logística. O “output” varia entre 0 e 1, aumentando com os seus “inputs” (ver figura 2.15).

O algoritmo “Backpropagation” aprende os pesos para uma rede multinível, dada uma rede com um conjunto fixo de unidades e interligações. Este algoritmo emprega a descida da encosta para minimizar o quadrado do erro entre os valores de “output” da rede e os valores alvo para esses “outputs”. O cálculo de E tem que ser redefinido (já que necessitamos de considerar redes com múltiplos “outputs”) com a soma dos erros de todas as unidades de “output”:

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in \text{outputs}} (t_{kd} - o_{kd})^2$$

em que os *outputs* são o conjunto de unidades de “output” na rede, e t_{kd} e o_{kd} são o valor alvo e o valor de “output” associados à k^{a} unidade de “output” e ao exemplo de treino d .

O problema de aprendizagem do “Backpropagation” consiste em pesquisar um grande espaço de hipóteses definido por todos os valores de pesos possíveis para todas as unidades na rede. A situação pode ser visualizada em termos de uma superfície de erro similar à das unidades lineares. O erro dessas unidades é substituído por uma nova definição de E , e as outras dimensões do espaço correspondem a todos os pesos associados com todas as unidades da rede. A descida da encosta no “Backpropagation” garante convergir para um mínimo local e não necessariamente o mínimo erro global. Apesar deste obstáculo, o “Backpropagation” tem produzido excelentes resultados em muitas aplicações do mundo real.

A versão da descida incremental da encosta do algoritmo “Backpropagation” é apresentado na tabela 2.9. O algoritmo aplica-se a redes contendo dois níveis de unidades sigmóides, com as unidades de cada nível ligadas a todas as unidades do nível precedente.

O algoritmo, para cada exemplo de treino, propaga o seu “input” na rede, calcula o erro do “output” da rede para esse exemplo, calcula o gradiente relativo ao erro desse exemplo e depois actualiza todos os pesos na rede. Esta etapa de descida de encosta é iterada (frequentemente milhares de vezes, usando os mesmos exemplos de treino múltiplas vezes) até a rede ter um desempenho aceitável.

O valor δ_k é calculado para cada unidade k de “output” da rede (equação (2.4) no algoritmo), e corresponde a $(t_k - o_k)$ da regra delta multiplicando pelo factor $o_k(1-o_k)$ que é a derivada da função sigmóide

$$\frac{d\sigma(y)}{dy} = \sigma(y) \cdot (1 - \sigma(y))$$

O valor de δ_h para cada unidade escondida h tem um cálculo similar (equação (2.5) no algoritmo). Contudo, os exemplos de treino possibilitam só os valores alvo t_k para os “outputs” da rede, não havendo valores alvo directamente disponíveis para indicar o valor do erro das unidades “escondidas”. Assim, o erro para a unidade escondida h é calculado somando os erros δ_k de cada unidade de “output” influenciada por h pesando cada um dos δ_k com w_{kh} , o peso da unidade escondida h para a unidade de “output” k . Este peso caracteriza o grau com que a unidade escondida h é responsável pelo erro na unidade de “output” k . A regra de actualização do peso na descida da encosta (equação (2.6) no algoritmo) é similar à regra de treino delta ($\Delta w_i = \eta(t - o) x_i$). Tal como a regra delta, actualiza cada peso na proporção do rácio de

aprendizagem η , o valor de “input” x_{ji} ao qual o peso é aplicado, e o erro no “output” da unidade. A única diferença reside no facto do erro ($t-o$) na regra delta ser substituído por δ_j .

O algoritmo da tabela 2.9 actualiza os pesos incrementalmente, correspondendo à descida incremental da encosta [Mitchell-97]. Para obter o gradiente de E relativo à descida da encosta deve-se somar os valores de δx_{ji} sobre todos os exemplos de treino antes de alterar os valores dos pesos. No algoritmo “Backpropagation” o ciclo de actualização de pesos deve ser iterado milhares de vezes numa aplicação típica. Podem ser usadas várias condições terminais para parar o procedimento. Nomeadamente, poderá parar depois de um determinado número de iterações do ciclo, ou quando o erro nos exemplo de treino descer abaixo de um limite, ou quando o erro num conjunto separado de exemplos de validação atinge um determinado valor.

Tabela 2.9 – Descida Incremental da Encosta do Algoritmo “Backpropagation” (com dois níveis de unidades sigmóides)

“Backpropagation” (*Exemplos de treino*, η , n_{input} , n_{output} , $n_{escondido}$)

Cada *exemplo de treino* é um par da forma $(\vec{x} \cdot \vec{t})$, onde $\vec{x}$ é o vector de valores de “input” da rede, e $\vec{t}$ é o vector dos valores alvo de “output” da rede; η é o rácio de aprendizagem (por exemplo, 0.05), n_{input} é o número de “inputs” da rede, $n_{escondido}$ o número de unidades no nível escondido e n_{output} o número de unidades de “output”.

x_{ij} é o “input” da unidade i para a unidade j e w_{ji} é o peso do “input” da unidade i para a unidade j .

- Criar uma rede com n_{inputs} “inputs”, $n_{escondido}$ unidades “escondidas” e n_{output} unidades de “output”
- Inicializar todos os pesos da rede a valores aleatórios pequenos (entre -0.05 e 0.05)
- Até a condição terminal ser encontrada, Fazer

- Para cada $(\vec{x} \cdot \vec{t})$ dos exemplos de treino

Propagar o “input” para a frente na rede:

1. Fazer o “input” da instância $\vec{x}$ para a rede e calcular o “output” o_u de todas as unidades u na rede

Propagar os erros para trás na rede:

2. Para cada unidade de “output” da rede k , calcular o seu erro δ_k

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k) \quad (2.4)$$

3. Para cada unidade h , calcular o seu erro δ_h

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{kh} \delta_k \quad (2.5)$$

4. Actualizar cada peso da rede w_{ji}

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji} \quad \text{em que} \quad \Delta w_{ji} = \eta \delta_j x_{ji} \quad (2.6)$$

A escolha do critério de fim do ciclo é importante, pois poucas iterações podem levar a uma redução insuficiente do erro, enquanto muitas iterações podem levar ao sobreajustamento dos dados.

O algoritmo “Backpropagation” apresentado na tabela 2.9 aplica-se só a redes com dois níveis de unidades sigmóides. Contudo, o algoritmo facilmente é generalizado para redes de profundidade arbitrária. A regra de actualização de pesos da equação (2.6) do algoritmo é retida, e a única mudança verifica-se no procedimento de cálculo dos valores de δ . O valor δ_r para uma unidade r no nível m é calculado a partir dos valores δ no nível seguinte $m+1$ de acordo com

$$\delta_r = o_r(1 - o_r) \sum_{s \in \text{nível}_{m+1}} w_{sr} \delta_s$$

Esta etapa é idêntica à da equação (2.5) do algoritmo da tabela 2.9, isto é, esta etapa deve ser repetida para qualquer número de níveis escondidos da rede.

O algoritmo “Backpropagation” é amplamente usado e foram desenvolvidas muitas variações. A mais comum é alterar a regra de actualização de pesos na equação (2.6) do algoritmo de forma a que a actualização do peso na n^{a} iteração dependa parcialmente da actualização ocorrida na $(n-1)^{\text{a}}$ iteração:

$$\Delta w_{ji}(n) = \eta \delta_j x_{ji} + \alpha \Delta w_{ji}(n-1)$$

em que $0 \leq \alpha \leq 1$ e é chamado “momentum”. O efeito de α é o de adicionar velocidade na pesquisa durante a descida da encosta de forma a que a pesquisa tenha a mesma direcção de iteração em iteração. Por vezes tem o efeito de fazer avançar a pesquisa (não a deixando parar) em mínimos locais da superfície de erro, ou em regiões sem variação da superfície de erro (onde a pesquisa não avançaria se não existisse o “momentum”).

Devido ao facto da superfície de erro para redes multinível poder conter muitos mínimos locais diferentes, a descida da encosta pode parar em algum destes mínimos locais. Assim, o algoritmo “Backpropagation” sobre redes multinível só garante a convergência a um mínimo local de E , e não necessariamente o mínimo erro global. Apesar da falta de segurança na convergência para o erro mínimo global, o “Backpropagation” na prática é um método altamente eficaz de aproximação de funções. Em muitas aplicações práticas do algoritmo, verificou-se que o problema do mínimo local não é tão severo como se temia. Isto deve-se ao facto de que quantos mais pesos existirem na rede, mais caminhos de fuga existem para que a descida da encosta saia do mínimo local relativo a um determinado peso.

A descida da encosta sobre superfícies de erro ainda não é bem entendida, e não existem métodos para prever com certeza quando um mínimo local causará dificuldades. Existem heurísticas que tentam aliviar o problema do mínimo local que incluem:

- a adição do “momentum” à regra de actualização de pesos,
- o uso da descida incremental da encosta em vez da descida da encosta,
- o treino de múltiplas redes usando os mesmos dados, mas inicializando cada rede com valores de pesos diferentes. Se as diferentes redes levam a diferentes mínimos locais, então deve ser seleccionada a rede com melhor desempenho sobre um conjunto de dados de validação separados. Alternativamente, poder-se-á considerar todas as redes calculando o “output” a partir da média (possivelmente pesada) dos “outputs” individuais de cada uma das redes.

2.4.5.1 Pesquisa do Espaço de Hipóteses e Limitação Indutiva

No algoritmo “Backpropagation” todas as atribuições possíveis aos pesos da rede representam uma hipótese sintacticamente distinta. O espaço de hipóteses é contínuo, em contraste com o espaço de hipóteses da aprendizagem de árvores de decisão e outros métodos baseados em representações discretas. O facto do espaço de hipóteses ser contínuo, permite a obtenção do erro do gradiente, possibilitando assim uma forma para organizar a pesquisa da melhor hipótese. Esta forma de pesquisa é diferente da ordenação geral-para-específica usada para organizar a pesquisa em algoritmos de aprendizagem de conceitos simbólicos e da ordenação sobre as árvores de decisão usada pelo algoritmo ID3.

É difícil caracterizar a limitação indutiva do algoritmo “Backpropagation”, contudo pode-se caracterizá-la como uma interpolação entre os pontos de dados. Dados dois exemplos positivos sem nenhum exemplo negativo entre eles, o “Backpropagation” tende a classificar os pontos entre eles como exemplos positivos também.

2.4.5.2 Representação do Nível Escondido

Uma propriedade do algoritmo “Backpropagation” é a sua capacidade de descobrir representações intermédias úteis no nível de unidades “escondidas”. Os exemplos de treino fornecem os “inputs” e os “outputs” da rede, sendo o procedimento de actualização dos pesos livre de atribuir os pesos às unidades “escondidas” que mais eficientemente minimizem o quadrado do erro E . Isto leva o “Backpropagation” a definir características novas no nível escondido que não são explícitas na representação do “input”, mas que são adquiridas das propriedades das instâncias de “input”. Em contraste com os métodos de aprendizagem que são limitados a usar somente características predefinidas, o algoritmo “Backpropagation” possibilita um grau importante de flexibilidade que permite descobrir características não explicitamente adquiridas.

2.4.5.3 Generalização, Sobreajustamento e Critério de Fim

Uma escolha óbvia para terminar o ciclo de actualizações de pesos no algoritmo “Backpropagation” é terminar o ciclo quando o erro E nos exemplos de treino toma um valor abaixo de um limite predefinido. No entanto, este critério é pobre pois é susceptível de sobreajustar os exemplos de treino ainda não observados. A actualização de pesos a partir de determinado número de iterações ajusta idiossincrasias dos exemplos de treino que não são representativas da distribuição geral de exemplos. Um grande número de parâmetros (pesos) nas RNs possibilita um grande número de graus de liberdade para ajustar essas idiossincrasias. Uma abordagem com sucesso para resolver o problema do sobreajustamento dos dados de treino no algoritmo “Backpropagation” consiste em simplesmente ter um conjunto de validação de dados em adição ao conjunto de dados de treino. A actualização de pesos deve terminar quando se verificar o menor erro sobre o conjunto de dados de validação. Em implementações típicas desta abordagem, são guardadas duas cópias dos pesos da rede: uma cópia para o treino e uma cópia separada do melhor desempenho dos pesos desde o início do treino, medida pelo erro sobre o conjunto de validação. Quando os pesos de treino atingem um erro significativamente alto sobre o conjunto de validação é retornada a cópia com o melhor desempenho dos pesos.

Esta abordagem funciona bem quando existem dados extra disponíveis para o conjunto de validação. Contudo, o problema do sobreajustamento é mais problemático em conjuntos de treino pequenos. Nestes casos, é utilizada uma abordagem chamada de validação cruzada. Os

m exemplos disponíveis são particionados em k conjuntos disjuntos, cada um com o tamanho m/k . O procedimento de validação cruzada é realizado k vezes, usando em cada iteração um subconjunto diferente de dados de validação e de treino. Assim, cada exemplo é usado no conjunto de dados de validação para uma das experiências e no conjunto de dados de treino para as restantes $k-1$ experiências. Em cada experiência esta abordagem de validação cruzada é usada para obter o número de iterações i correcto que atinge o melhor desempenho no conjunto de dados de validação. A média destas estimativas de i é então calculada, e é realizada uma execução final do “Backpropagation” com todos os n exemplos e com i iterações, sem nenhum conjunto de validação.

2.5 Aprendizagem Bayesiana

2.5.1 Introdução

O raciocínio Bayesiano possibilita uma abordagem probabilística para a inferência.

Este raciocínio é baseado na pressuposição que as quantidades de interesse são determinadas por distribuições probabilísticas e que decisões óptimas podem ser realizadas raciocinando sobre essas probabilidades em conjunto com os dados observados. Esta perspectiva é importante para a Aprendizagem Automática pois possibilita uma abordagem quantitativa para analisar hipóteses alternativas. O raciocínio Bayesiano disponibiliza as bases para os algoritmos de aprendizagem que manipulam directamente probabilidades, assim como uma estrutura para analisar a operação de outros algoritmos que não manipulam probabilidades explicitamente.

Os métodos de aprendizagem Bayesiana incluem as seguintes características:

- Cada exemplo de treino observado pode incrementalmente diminuir ou aumentar a probabilidade estimada da hipótese ser correcta. Isto possibilita uma abordagem mais flexível para a aprendizagem do que os algoritmos que eliminam hipóteses que não sejam consistentes com um determinado exemplo.
- O conhecimento prévio pode ser combinado com os dados observados para determinar a probabilidade final de uma hipótese. Na aprendizagem Bayesiana o conhecimento prévio é possibilitado pela probabilidade prévia de cada hipótese candidata, e pela distribuição da probabilidade sobre os dados observados para cada hipótese candidata.
- Os métodos Bayesianos podem incluir hipóteses que permitem previsões probabilísticas (por exemplo, “um doente de pneumonia tem 93% de possibilidades de recuperar completamente”).
- Novas instâncias podem ser classificadas combinando as previsões de múltiplas hipóteses, pesando as suas probabilidades.

Uma dificuldade prática na aplicação dos métodos Bayesianos reside no facto que estes tipicamente requerem conhecimento prévio de muitas probabilidades. Quando estas probabilidades não são conhecidas, são frequentemente estimadas a partir do conhecimento prévio de dados disponíveis previamente e pressuposições relativas à forma da distribuição subjacente. Uma segunda dificuldade prática é o custo computacional significativo requerido para determinar a hipótese óptima de Bayes. Em certas situações especiais, este custo computacional pode ser significativamente reduzido.

2.5.2. Teorema de Bayes

O teorema de Bayes possibilita um método directo para encontrar a hipótese que melhor ajuste os dados de treino, isto é, procura a hipótese mais provável, a partir dos dados D e mais algum conhecimento prévio das probabilidades prévias das várias hipóteses em H . O teorema de Bayes possibilita uma forma de calcular a probabilidade de uma hipótese a partir da sua probabilidade prévia, da probabilidade de se observarem dados dada essa hipótese e da probabilidade dos próprios dados serem observados.

A $P(h)$ é frequentemente chamada probabilidade prévia de h e consiste em qualquer conhecimento prévio relativo à possibilidade da hipótese h ser uma hipótese correcta. Se não existir este conhecimento prévio deve-se atribuir a mesma probabilidade prévia para cada uma das hipóteses candidatas. $P(D)$ é a probabilidade prévia dos dados de treino D serem observados (sem conhecimento de que hipótese os ajusta). $P(D|h)$ é a probabilidade de os dados D serem observados, dado um universo em que a hipótese h ajusta esses dados. Mais genericamente, diz-se que $P(x|y)$ é a probabilidade de x dado y . Nos problemas de Aprendizagem Automática é importante a probabilidade $P(h|D)$ que consiste na probabilidade da hipótese h ajustar os dados de treino observados D . Esta probabilidade chama-se probabilidade posterior de h , pois reflecte a confiança de h ajustar os dados não observados depois de serem observados os dados de treino D . Enquanto que a probabilidade posterior $P(h|D)$ reflecte a influência dos dados de treino D , a probabilidade prévia $P(h)$ é independente de D .

Teorema de Bayes :

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Em muitos cenários de aprendizagem, considera-se um conjunto de hipóteses candidatas H e pretende-se encontrar a hipótese mais provável $h \in H$ a partir dos dados observados D .

Esta hipótese é chamada de hipótese posterior máxima (h_{PM}). Pode-se determinar a hipótese posterior máxima usando o teorema de Bayes para calcular a probabilidade posterior de cada hipótese candidata.

$$h_{PM} \equiv \arg \max_{h \in H} P(h|D) = \arg \max_{h \in H} \frac{P(D|h)P(h)}{P(D)} = \arg \max_{h \in H} P(D|h)P(h) \quad (2.7)$$

No último passo o termo $P(D)$ é eliminado pois é uma constante independente de h .

Em alguns casos assume-se que todas as hipóteses em H têm igual probabilidade prévia ($P(h_i) = P(h_j)$ para todas h_i e h_j em H). Neste caso pode-se simplificar a equação (2.7) considerando só o termo $P(D|h)$ para encontrar a hipótese mais provável. $P(D|h)$ é frequentemente chamada de probabilidade dos dados D dada a hipótese h , e qualquer hipótese que maximiza $P(D|h)$ é chamada de hipótese de probabilidade máxima, h_{PRM} .

$$h_{PRM} \equiv \arg \max_{h \in H} P(D|h) \quad (2.8)$$

2.5.3 Algoritmo de Aprendizagem da Hipótese Posterior Máxima (h_{PM})

O teorema de Bayes possibilita uma forma de calcular a probabilidade posterior de cada hipótese dados os dados de treino, e de determinar a mais provável (h_{PM}).

A tabela 2.10 ilustra o algoritmo de aprendizagem da hipótese posterior máxima, baseado no teorema de Bayes.

Tabela 2.10 – Algoritmo da Hipótese Posterior Máxima

Algoritmo de aprendizagem da hipótese posterior máxima (h_{PM})

1. Para cada hipótese h em H , calcular a probabilidade posterior

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

2. Retornar a hipótese posterior máxima (h_{PM}), isto é, a que tiver maior probabilidade posterior.

$$h_{PM} = \arg \max_{h \in H} P(h|D)$$

Este algoritmo pode requerer um esforço computacional significativo, pois aplica o teorema da Bayes a cada hipótese em H para calcular $P(h|D)$. Se não houver conhecimento prévio da probabilidade das várias hipóteses, é razoável atribuir à mesma probabilidade prévia a todas as hipóteses h em H . Dado que assumimos que o conceito alvo está contido em H , a soma das probabilidades prévias é 1, daí escolher-se

$$P(h) = \frac{1}{|H|} \text{ para todas as hipóteses } h \text{ em } H$$

A $P(D|h)$ é a probabilidade de observar os valores alvo $D = (d_1, \dots, d_m)$ para um conjunto de instâncias $(x_1, \dots, x_m)$, num universo em que h é a descrição correcta do conceito alvo c . Assumindo que os dados de treino não têm erros, a probabilidade de observar a classificação d_i é 1, dado h , se $d_i = h(x_i)$ e 0 se $d_i \neq h(x_i)$.

Assim

$$P(D|h) = \begin{cases} 1 & \text{se } d_i = h(x_i) \text{ para todos } d_i \text{ em } D \\ 0 & \text{caso contrário} \end{cases} \quad (2.9)$$

Isto é, a probabilidade de D dada a hipótese h é 1 se D é consistente com h e 0 caso contrário.

Na primeira etapa do algoritmo de aprendizagem da h_{PM} é necessário calcular a probabilidade posterior ($P(h|D)$) das hipóteses em H .

Considerando primeiro os casos em que h é inconsistente com os dados de treino D , pela equação (2.9)

$$P(h|D) = \frac{0 \cdot P(h)}{P(D)} = 0$$

ou seja, a probabilidade posterior de uma hipótese inconsistente é 0.

Considerando os casos em que h é consistente com D , pela equação (2.9)

$$P(h|D) = \frac{1 \cdot \frac{1}{|H|}}{P(D)} = \frac{1 \cdot \frac{1}{|H|}}{\frac{|VS_{H,D}|}{|H|}} = \frac{1}{|VS_{H,D}|}$$

em que $VS_{H,D}$ é o subconjunto das hipóteses de H que são consistentes com D .

Sumariando, o teorema de Bayes permite calcular a probabilidade posterior $P(h|D)$ a partir de $P(h)$ e $P(D|h)$:

$$P(h|D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{se } h \text{ é consistente com } D \\ 0 & \text{caso contrário} \end{cases}$$

em que $|VS_{H,D}|$ é o número de hipóteses de H consistentes com D .

A evolução das probabilidades associadas às hipóteses é ilustrado esquematicamente na figura 2.16. Inicialmente todas as hipóteses têm a mesma probabilidade (figura 2.16a) [Mitchell-97]. À medida que mais dados de treino são tratados (figura 2.16b e 2.16c), a probabilidade posterior para as hipóteses inconsistentes torna-se zero enquanto que a probabilidade total (1) é partilhada igualmente entre as restantes hipóteses consistentes. Todas as hipóteses consistentes com D são hipóteses posteriores máximas.

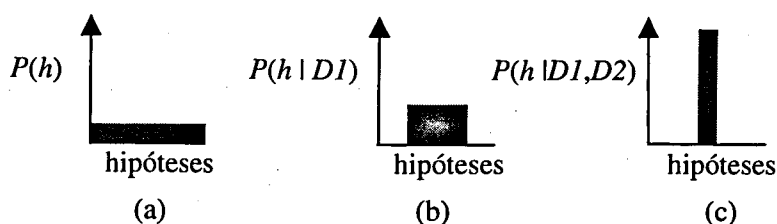


Figura 2.16 – Evolução das Probabilidades Posteriores com o aumento de dados de treino

O algoritmo “Find-S” pesquisa o espaço de hipóteses H , de hipóteses específicas para gerais, retornando a hipótese consistente o máximo específica. Como o algoritmo “Find-S” retorna uma hipótese consistente, sabemos que retorna a hipótese posterior máxima sob a distribuição de probabilidades de $P(h)$ e $P(D|h)$. O “Find-S” não manipula explicitamente probabilidades, simplesmente retorna o membro o máximo específico do “Version Space”. Contudo, identificando distribuições para $P(h)$ e $P(D|h)$ em que as suas hipóteses de “output” sejam hipóteses posteriores máximas, encontra-se uma forma útil de caracterizar o conhecimento do “Find-S”.

Qualquer distribuição de probabilidades $P(h)$ sobre H atribui $P(h_1) \geq P(h_2)$ se h_1 é mais específica do que h_2 . Assim, pode ser demonstrado que o “Find-S” retorna uma hipótese posterior máxima assumindo a probabilidade prévia $P(h)$ e a distribuição $P(D|h)$ referida anteriormente.

A estrutura Bayesiana disponibiliza uma forma de caracterizar o conhecimento dos algoritmos

de aprendizagem (por exemplo, o "Find-S"), mesmo quando os algoritmos de aprendizagem não manipulam explicitamente probabilidades.

2.5.4 A Hipótese de Probabilidade Máxima e a Hipótese do Menor Quadrado do Erro

Sob certas pressuposições, qualquer algoritmo de aprendizagem que minimiza o quadrado do erro entre o "output" previsto pela hipótese e os dados de treino retornará uma hipótese de probabilidade máxima (h_{PRM}). Isto permite uma justificação Bayesiana para muitas redes neurais e outros métodos de aprendizagem que minimizem a soma do quadrado dos erros sobre os dados de treino.

Considere-se um espaço de instâncias X e um espaço de hipóteses H constituído por uma classe de funções reais definidas sobre X (cada hipótese h em H é uma função da forma $h : X \rightarrow \mathbb{R}$, em que $\mathbb{R}$ é o conjunto de números reais). Pretende-se aprender uma função alvo desconhecida $f : X \rightarrow \mathbb{R}$ a partir de H . Estão disponíveis m exemplos de treino, em que o valor de cada exemplo está corrompido por ruído aleatório com uma distribuição de probabilidades normal. Mais precisamente, cada exemplo de treino é um par da forma (x, d_i) em que $d_i = f(x_i) + e_i$, em que $f(x_i)$ é o valor da função alvo sem erro e e_i é uma variável aleatória representando o ruído. É assumido que os valores de e_i são independentes e que têm uma distribuição normal com média zero.

Um exemplo simples para este problema é a aprendizagem de uma função linear. A figura 2.17 ilustra a função linear alvo f com linha sólida, e um conjunto de exemplos de treino com erro [Mitchell-97]. A linha a tracejado corresponde à hipótese que minimiza a soma do quadrado dos erros, por isso a hipótese de probabilidade máxima (h_{PRM}). A hipótese de probabilidade máxima não é necessariamente idêntica à hipótese correcta f , pois foi inferida de uma amostra limitada de dados de treino com erro.

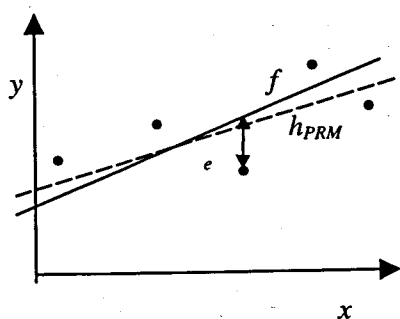


Figura 2.17 – Aprendizagem de uma Função Linear

Utilizando a equação (2.8) com p para referenciar a densidade de probabilidade (já que é uma variável contínua)

$$h_{PRM} = \arg \max_{h \in H} p(D | h)$$

e assumindo um conjunto fixo de instâncias de treino $(x_1 \dots x_m)$, a sequência dos respectivos valores alvo $D = (d_1 \dots d_m)$ e que os exemplos de treino são mutuamente independentes dado h , pode-se escrever $P(D | h)$ como o produto dos vários $p(d_i | h)$

$$h_{PRM} = \arg \max_{h \in H} \prod_{i=1}^m p(d_i | h)$$

Dado que o ruído e_i segue uma distribuição normal com média zero e uma variância desconhecida σ^2 , cada d_i deve também seguir uma distribuição normal com variância σ^2 centrada à volta do valor alvo verdadeiro $f(x_i)$. Assim, $p(d_i | h)$ pode ser escrita como uma distribuição normal com variância σ^2 e média $\mu = f(x_i)$. Uma distribuição normal é uma distribuição em forma de sino definida pela função densidade de probabilidade

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Daí que

$$h_{PRM} = \arg \max_{h \in H} \prod_{i=1}^m \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i - \mu)^2}$$

$$= \arg \max_{h \in H} \prod_{i=1}^m \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i - h(x_i))^2}$$

Em vez de maximizar a expressão acima, pode-se maximizar o seu logaritmo. Isto é possível pois o $\ln p$ é uma função monótona de p .

Assim maximizando $\ln p$ também se maximiza p .

$$h_{PRM} = \arg \max_{h \in H} \sum_{i=1}^m \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} (d_i - h(x_i))^2$$

O primeiro termo na expressão é uma constante independente de h , e assim pode ser eliminado, ficando

$$h_{PRM} = \arg \max_{h \in H} \sum_{i=1}^m -\frac{1}{2\sigma^2} (d_i - h(x_i))^2$$

Maximizar esta quantidade negativa é equivalente a minimizar a quantidade positiva correspondente

$$h_{PRM} = \arg \min_{h \in H} \sum_{i=1}^m \frac{1}{2\sigma^2} (d_i - h(x_i))^2$$

Finalmente, pode-se eliminar a constante independente de h e

$$h_{PRM} = \arg \min_{h \in H} \sum_{i=1}^m (d_i - h(x_i))^2 \quad (2.10)$$

A equação anterior mostra que a hipótese de probabilidade máxima minimiza a soma do quadrado dos erros entre os dados de treino observados d_i e as previsões da hipótese $h(x_i)$. Isto acontece assumindo que os valores dos dados de treino d_i são gerados adicionando um ruído

aleatório ao valor alvo verdadeiro, e em que este ruído aleatório é atribuído independentemente a cada exemplo a partir duma distribuição normal com média zero.

Esta análise considera ruído só no valor alvo dos exemplos de treino, não considerando ruído nos atributos que descrevem as instâncias. A análise tornaria-se significativamente mais complexa se estas pressuposições fossem removidas.

2.5.5 Hipótese de Probabilidade Máxima para Predizer Probabilidades

Considere-se o problema da necessidade de aprender uma função não determinista (probabilística) $f : X \rightarrow \{0,1\}$, que tem dois valores de “output” discretos. O espaço de instâncias X representam pacientes com os seus sintomas, e a função alvo $f(x)$ deve ser um se o paciente sobrevive à doença e zero se não sobrevive. Entre um conjunto de pacientes exibindo o mesmo conjunto de sintomas, 92% devem sobreviver e 8% não. A função alvo $f(x)$ tem como “output” uma função probabilística do “input”. A partir deste problema, pode-se aprender uma rede neuronal (ou outra função real) em que o seu “input” é a probabilidade de $f(x)=1$.

Pretende-se aprender uma função $f' : X \rightarrow [0,1]$, tal que $f'(x) = P(f(x)=1)$. No exemplo se x é um dos pacientes dos 92% que sobrevivem, então $f'(x) = 0.92$ e a função probabilística $f(x)$ será 1 em 92% dos casos e igual a 0 nos 8% restantes. Para aprender f' usando uma rede neuronal, será necessário saber as frequências de 1's e 0's para cada valor possível de x e depois treinar a rede neuronal para retornar a frequência alvo para cada x .

Outra forma, consiste em derivar a hipótese de probabilidade máxima para f' em vez de treinar a rede neuronal directamente dos exemplos de treino de f . Primeiro, é necessário obter $P(D|h)$. Assumindo que os dados de treino D são da forma $D = \{(x_1, d_1) \dots (x_m, d_m)\}$, em que d_i é o valor 0 ou 1 para $f(x_i)$, que as variáveis x_i e d_i são aleatórias, e que cada exemplo de treino é independente dos outros exemplos, pode-se escrever $P(D|h)$ como

$$P(D|h) = \prod_{i=1}^m P(x_i, d_i | h)$$

É razoável assumir, que a probabilidade de encontrar uma instância particular x_i é independente da hipótese h . Quando x é independente de h pode-se escrever a anterior expressão como

$$P(D|h) = \prod_{i=1}^m P(x_i, d_i | h) = \prod_{i=1}^m P(d_i | h, x_i) P(x_i) \quad (2.11)$$

A probabilidade $P(d_i | h, x_i)$ de se observar $d_i = 1$ para uma instância, é dada por $h(x)$ ($P(d_i = 1 | h, x_i) = h(x)$) e assim,

$$P(d_i | h, x_i) = \begin{cases} h(x_i) & \text{se } d_i = 1 \\ (1-h(x_i)) & \text{se } d_i = 0 \end{cases}$$

ou $P(d_i | h, x_i) = h(x_i)^{d_i} (1-h(x_i))^{1-d_i}$

Usando a última equação para substituir $P(d_i | h, x_i)$ na equação (2.11) obtém-se

$$P(D | h) = \prod_{i=1}^m h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} P(x_i)$$

e pode-se escrever a seguinte expressão para a hipótese de probabilidade máxima

$$h_{PRM} = \arg \max_{h \in H} \prod_{i=1}^m h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} P(x_i)$$

Retirando o último termo já que é independente de h , fica

$$h_{PRM} = \arg \max_{h \in H} \prod_{i=1}^m h(x_i)^{d_i} (1 - h(x_i))^{1-d_i}$$

ou trabalhando com o logaritmo da probabilidade

$$h_{PRM} = \arg \max_{h \in H} \sum_{i=1}^m d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i))$$

Esta equação indica a quantidade que deve ser maximizada com o intuito de obter a hipótese de probabilidade máxima.

2.5.6 Classificação Óptima de Bayes

Considere-se um espaço de hipóteses contendo três hipóteses h_1 , h_2 e h_3 e as probabilidades posteriores destas hipóteses dados os exemplos de treino, 0.4, 0.3 e 0.3 respectivamente. Assim, a hipótese h_1 é a hipótese posterior máxima. Ao ser encontrada uma nova instância x , h_1 classifica-a como positiva mas h_2 e h_3 como negativa. Tomando todas as hipóteses em consideração a probabilidade de x ser positiva é 0.4 e a probabilidade de ser negativa é 0.6. A classificação mais provável neste caso é diferente da classificação gerada pela hipótese posterior máxima. Geralmente, a classificação mais provável de uma nova instância é obtida combinando as predições de todas as hipóteses, pesadas pelas suas probabilidades posteriores. Se a classificação de uma nova instância puder ser um qualquer valor v_j de um conjunto V , então a probabilidade $P(v_j | D)$ de que a classificação correcta para a nova instância é v_j é

$$P(v_j | D) = \sum_{h_i \in H} P(v_j | h_i) P(h_i | D)$$

A classificação óptima de Bayes da nova instância é o valor v_j , para o qual $P(v_j | D)$ é máximo:

$$\arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j | h_i) P(h_i | D)$$

Para ilustração considere-se o exemplo acima, em que o conjunto de classificação da nova instância é $V = \{+, -\}$ e

$$P(h_1 | D) = 0.4, P(- | h_1) = 0, P(+ | h_1) = 1$$

$$P(h_2 | D) = 0.3, P(- | h_2) = 1, P(+ | h_2) = 0$$

$$P(h_3 | D) = 0.3, P(- | h_3) = 1, P(+ | h_3) = 0$$

e portanto

$$\sum_{h_i \in H} P(+ | h_i) P(h_i | D) = 0.4$$

$$\sum_{h_i \in H} P(- | h_i) P(h_i | D) = 0.6$$

$$e \quad \arg \max_{v_j \in \{+, -\}} \sum_{h_i \in H} P(v_j | h_i) P(h_i | D) = -$$

Nenhum outro método de classificação usando o mesmo espaço de hipóteses e o mesmo conhecimento prévio tem em média resultados tão bons. Este método maximiza a probabilidade da nova instância ser classificada correctamente, dados os exemplos de treino, o espaço de hipóteses, e as probabilidades prévias das hipóteses.

2.5.7 Algoritmo Gibbs

Apesar da classificação óptima de Bayes obter o melhor desempenho que pode ser obtido dos dados de treino, pode ser onerosa de aplicar. Isto porque calcula a probabilidade posterior de todas as hipóteses em H e combina as predições de cada hipótese para classificar cada nova instância.

Um método alternativo, menos eficiente é o algoritmo Gibbs [Oppen-91] definido da seguinte forma:

1. Escolher aleatoriamente uma hipótese h de H , de acordo com a distribuição de probabilidade posterior sobre H
2. Usar h para prever a classificação da nova instância x

Dada uma nova instância a classificar, o algoritmo Gibbs simplesmente aplica uma hipótese construída aleatoriamente de acordo com a distribuição da probabilidade posterior assumida. Surpreendentemente, pode ser demonstrado que sob certas condições, o erro esperado por uma má classificação pelo algoritmo Gibbs, é no máximo o dobro do erro esperado da classificação óptima de Bayes. O valor esperado é obtido sobre conceitos alvo construídos aleatoriamente de acordo com a distribuição de probabilidade prévia assumida.

2.5.8 Classificação Simples de Bayes

A classificação simples de Bayes é aplicada em tarefas de aprendizagem onde cada instância x é descrita por uma conjunção de valores dos atributos e onde a função alvo $f(x)$ pode tomar um valor qualquer do conjunto V .

Considere-se um conjunto de exemplos de treino da função alvo disponíveis e que é apresentada uma nova instância, descrita pelos valores dos seus atributos $(a_1, a_2, \dots, a_n)$. A abordagem Bayesiana para classificar a nova instância consiste em atribuir o valor alvo mais provável, V_{SB} , dados os valores dos atributos $(a_1, a_2, \dots, a_n)$ que descrevem a instância.

$$V_{SB} = \arg \max_{V_j \in V} P(v_j | a_1, a_2, \dots, a_n)$$

Pode-se usar o teorema de Bayes para escrever a expressão como

$$\begin{aligned} V_{SB} &= \arg \max_{V_j \in V} \frac{P(a_1, a_2, \dots, a_n | v_j) P(v_j)}{P(a_1, a_2, \dots, a_n)} \\ &= \arg \max_{V_j \in V} P(a_1, a_2, \dots, a_n | v_j) P(v_j) \end{aligned} \quad (2.12)$$

De seguida é necessário estimar os dois termos na última equação a partir dos dados de treino. É fácil estimar cada um dos $P(v_j)$ simplesmente contando a frequência com que cada valor alvo v_j ocorre nos dados de treino. Contudo, estimar os diferentes termos $P(a_1, a_2, \dots, a_n | v_j)$ desta forma não é razoável a não ser que se tenha um grande conjunto de dados de treino. O problema é que o número destes termos é igual ao número de instâncias possíveis vezes o número de valores alvo possíveis.

A classificação simples de Bayes é baseada na pressuposição simples que o valor dos atributos são condicionalmente independentes dado o valor alvo. A pressuposição consiste no facto de que dado um valor de uma instância, a probabilidade de observar a conjunção $a_1, a_2, \dots, a_n$ é o produto das probabilidades de cada atributo individualmente:

$$P(a_1, a_2, \dots, a_n | v_j) = \prod_i P(a_i | v_j)$$

Substituindo-se na equação (2.12) obtém-se a abordagem usada na classificação simples de Bayes :

$$V_{SB} = \arg \max_{V_j \in V} P(v_j) \prod_i P(a_i | v_j) \quad (2.13)$$

em que V_{SB} é o valor alvo retornado pela classificação simples de Bayes. O número de termos distintos $P(a_i | v_j)$ que devem ser estimados a partir dos dados de treino é o número de valores do atributo distintos vezes o número de valores alvo distintos. Este número é muito menor do que o anterior, relativo ao caso de ser necessário estimar os $P(a_1, a_2, \dots, a_n | v_j)$ termos.

Sumariando, o método de aprendizagem simples de Bayes envolve uma etapa de aprendizagem em que os vários $P(v_j)$ e $P(a_i | v_j)$ termos são estimados a partir das suas frequências sobre os

dados de treino. O conjunto destas estimativas corresponde à hipótese aprendida. Esta hipótese é usada para classificar cada nova instância aplicando a regra da equação (2.13). Sempre que a pressuposição da classificação simples de Bayes relativa à independência condicional dos atributos é satisfeita, a classificação simples de Bayes V_{SB} é idêntica à classificação da hipótese posterior máxima.

Neste método de aprendizagem não há uma pesquisa explícita pelo espaço de hipóteses (espaço de valores possíveis que podem ser atribuídos aos vários $P(v_j)$ e $P(a_i | v_j)$ termos). Em vez disso, a hipótese é formada sem pesquisa, simplesmente calculando a frequência das várias combinações de dados nos exemplos de treino.

2.5.8.1 Um exemplo ilustrativo

Considere-se os exemplos de treino da tabela 2.7 de classificação de livros de acordo com algumas das suas características [Mitchell-97]. Os livros são analisados de forma a concluir-se se são ou não aconselháveis a comprar. Cada exemplo é descrito pelos atributos Conteúdo Técnico/Científico, Grafismo, Interpretabilidade e Preço. Pretende-se que perante a nova instância :

(Conteúdo Técnico/Científico=médio, Grafismo=bom, Interpretabilidade=média, Preço=alto)

a classificação simples de Bayes, a partir dos dados de treino classifique o livro associado a esta instância como aconselhável ou não a comprar.

Instanciando a equação (2.13), o valor alvo V_{SB} é dado por

$$\begin{aligned}
 V_{SB} &= \arg \max_{V_j \in \{Sim, Não\}} P(v_j) \prod_i P(a_i | v_j) \\
 &= \arg \max_{V_j \in \{Sim, Não\}} P(v_j) P(\text{Conteúdo Técnico / Científico} = \text{médio} | v_j) P(\text{Grafismo} = \text{bom} | v_j) \\
 &\quad P(\text{Interpretabilidade} = \text{média} | v_j) P(\text{Preço} = \text{alto} | v_j) \quad (2.14)
 \end{aligned}$$

As probabilidades dos valores alvo podem ser facilmente estimadas a partir das frequências sobre os 14 exemplos de treino.

$$P(\text{Comprar} = \text{Sim}) = 9/14 = 0.64$$

$$P(\text{Comprar} = \text{Não}) = 5/14 = 0.36$$

Similarmente, podem-se estimar as probabilidades condicionais. Por exemplo, para $\text{Preço}=\text{alto}$

$$P(\text{Preço} = \text{alto} | \text{Comprar} = \text{Sim}) = 3/9 = 0.33$$

$$P(\text{Preço} = \text{alto} | \text{Comprar} = \text{Não}) = 3/5 = 0.60$$

Usando estas estimativas de probabilidade e estimativas similares para os restantes valores dos

atributos, pode-se calcular V_{SB} de acordo com a equação (2.14) como se segue (omitindo os nomes dos atributos para abreviar):

$$P(\text{Sim}) = P(\text{médio} | \text{Sim})P(\text{bom} | \text{Sim})P(\text{média} | \text{Sim})P(\text{alto} | \text{Sim}) = 0.0053$$

$$P(\text{Não}) = P(\text{médio} | \text{Não})P(\text{bom} | \text{Não})P(\text{média} | \text{Não})P(\text{alto} | \text{Não}) = 0.0206$$

Assim, a classificação simples de Bayes atribui o valor alvo $\text{Comprar} = \text{Não}$ para esta nova instância, a partir das estimativas de probabilidade aprendidas a partir dos dados de treino.

Pode-se ainda, normalizar as quantidades acima de forma a somarem um, calculando assim a probabilidade condicional para que o valor alvo seja Não. Para este exemplo, a probabilidade é $0.0206/(0.0206+0.0053) = 0.795$

2.5.8.1.1 Estimativa de Probabilidades

As probabilidades podem ser estimadas pela fracção de vezes que um evento ocorre sobre o número total de oportunidades (n_c/n).

Esta estimativa proporciona uma boa estimativa da probabilidade para muitos casos, proporcionando uma estimativa pobre quando o n_c é muito pequeno. Supondo que o valor de $P(\text{Preço} = \text{alto} | \text{Comprar} = \text{Não})$ é 0.08 e que se tem uma amostra só com cinco exemplos em que $\text{Comprar} = \text{Não}$, o valor mais provável para n_c é 0. A estimativa n_c/n produz uma estimativa da probabilidade 0. Este termo domina a classificação simples de Bayes num futuro “query” com $\text{Preço} = \text{alto}$, já que a equação (2.13) requer a multiplicação de todos os outros termos de probabilidade por este valor 0. Para evitar esta dificuldade pode-se adoptar outra abordagem Bayesiana para estimar a probabilidade, usando m como se segue

$$\frac{n_c + mp}{n + m}$$

em que n_c é o número de eventos que ocorrem no número total de oportunidades (n), p é a estimativa prévia da probabilidade que se pretende determinar, e m é uma constante chamada tamanho equivalente da amostra, que determina o peso de p relativo aos dados observados. Um método típico para escolher p , à falta de outra informação, consiste em assumir prioridades prévias uniformes.

Se m for zero, esta estimativa é equivalente à simples fracção n_c/n . Se n e m forem diferentes de zero, então a fracção n_c/n e p são combinados, de acordo com o peso m . A razão pela qual m é chamado tamanho equivalente deve-se ao aumento às n observações, de m amostras virtuais distribuídas de acordo com p .

2.6 Aprendizagem Baseada em Instâncias

2.6.1 Introdução

Em contraste com os métodos de aprendizagem que constróem uma descrição geral explícita da função alvo a partir dos exemplos de treino, os métodos de aprendizagem baseados em instâncias, simplesmente armazenam os exemplos de treino. A generalização para além desses exemplos é adiada até que uma nova instância tenha de ser classificada. De cada vez que uma nova instância necessite ser classificada, a sua relação com os exemplos anteriores armazenados é examinada, de forma a atribuir-se um valor à função alvo para essa instância. Os métodos baseados em instâncias retardam o processamento até que uma nova instância deva ser classificada (por isso chamados de “preguiçosos”). Estes métodos têm a vantagem de em vez de estimar a função alvo para todo o espaço de instâncias, estimam a função alvo localmente e diferenciadamente para cada nova instância a ser classificada.

A aprendizagem baseada em instâncias inclui o método dos k -vizinhos mais próximos e o método de regressão localmente pesada que aproxima funções alvo de valores reais. Estes métodos assumem que as instâncias podem ser representadas por pontos no espaço euclidiano.

Uma desvantagem das abordagens baseadas em instâncias reside no facto que o custo de classificar novas instâncias pode ser grande. Isto deve-se ao facto de quase todo o processamento ter lugar na altura da classificação e não quando os exemplos de treino estão disponíveis. Uma segunda desvantagem para muitas abordagens baseadas em instâncias é que tipicamente consideram todos os atributos das instâncias quando realizam a pesquisa de exemplos de treino similares na memória. Se o conceito alvo depende só de alguns de muitos atributos disponíveis, verifica-se a utilização desnecessária dos restantes atributos sem interesse para a classificação.

2.6.2 Aprendizagem Baseada nos K-Vizinhos Mais Próximos

Um dos métodos baseados em instâncias é o algoritmo dos k -vizinhos mais próximos. Este algoritmo é utilizado em várias áreas, nomeadamente na área dos Sistemas Eléctricos de Energia [Wehenkel-98], e assume que todas as instâncias correspondem a pontos no espaço $\mathcal{R}^n$ de dimensão n . Os vizinhos mais próximos de uma instância são definidos em termos da distância euclidiana standard.

Assumindo que uma instância arbitrária x é descrita por

$$(a_1(x), a_2(x), \dots, a_n(x))$$

onde $a_n(x)$ é o valor do atributo n da instância x , então a distância entre duas instâncias x_i e x_j é definida por $d(x_i, x_j)$ em que

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^n (a_r(x_i) - a_r(x_j))^2}$$

Considere-se funções alvo de aprendizagem discretas da forma $f: \mathcal{R}^n \rightarrow V$, em que V é um

conjunto finito $\{v_1, \dots, v_s\}$. O algoritmo dos k -vizinhos mais próximos para aproximar funções alvo discretas encontra-se na tabela 2.11 [Mitchell-97].

Tabela 2.11 – Algoritmo dos K-Vizinhos Mais Próximos para aproximar uma função discreta

Algoritmo de treino :

- Para cada exemplo de treino $(x, f(x))$, adicionar o exemplo à lista de exemplos de treino

Algoritmo de classificação :

- A partir de um “query” de uma instância x_q a ser classificada,
 - considerando $x_1 \dots x_k$ as k instâncias dos exemplos de treino que são mais próximas de x_q
 - retornar

$$\hat{f}(x_q) \leftarrow \arg \max_{v \in V} \sum_{i=1}^k \delta(v, f(x_i))$$

em que $\delta(a,b)=1$ se $a=b$, caso contrário $\delta(a,b)=0$

O valor $\hat{f}(x_q)$ retornado por este algoritmo é a estimativa de $\hat{f}(x_q)$, que consiste no valor mais comum de f entre os k exemplos de treino mais próximos de x_q . Se $k=1$, então o algoritmo atribui a $\hat{f}(x_q)$ o valor $f(x_i)$ em que x_i é a instância de treino mais próxima de x_q . Para valores maiores de k , o algoritmo atribui o valor mais comum entre os k exemplos de treino mais próximos.

A figura 2.18 ilustra a operação do algoritmo dos k -vizinhos mais próximos para o caso em que as instâncias são pontos num espaço bidimensional e em que a função alvo é um valor booleano [Mitchell-97]. Os exemplos de treino positivos e negativos são representados por ‘+’ e ‘-’ respectivamente. Na figura, o algoritmo classifica x_q como um exemplo positivo considerando o vizinho mais próximo, enquanto que considerando os 5 vizinhos mais próximos o algoritmo classifica-o como um exemplo negativo.

O algoritmo dos k -vizinhos mais próximos nunca gera uma hipótese geral $\hat{f}$ pois simplesmente classifica cada nova instância assim que é necessário.

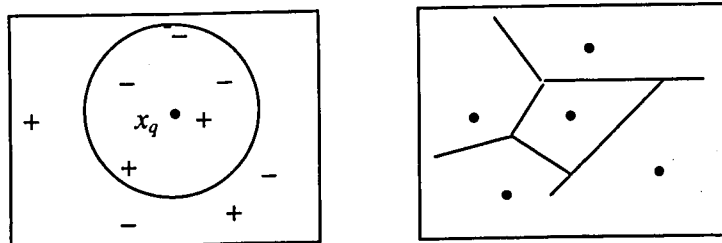


Figura 2.18 - Os K-Vizinhos Mais Próximos

O diagrama do lado direito da figura 2.18 mostra a forma da superfície de decisão induzida pelo algoritmo do vizinho mais próximo sobre o espaço inteiro de instâncias. A superfície de decisão é um poliedro convexo circundante de cada um dos exemplos de treino. Para cada exemplo de treino, o poliedro indica o conjunto de pontos de “query” cuja classificação é completamente determinada pelo exemplo de treino. Os pontos fora do poliedro estão mais perto de outro qualquer exemplo de treino.

O algoritmo dos k -vizinhos mais próximos pode ser facilmente adaptado para aproximar funções alvo contínuas. Para isso ser possível, o algoritmo em vez de determinar o valor mais comum, calcula o valor médio dos k exemplos mais próximos. Mais precisamente, para aproximar uma função alvo real

$$f: \mathcal{R}^n \rightarrow \mathcal{R}$$

substitui-se a última linha do algoritmo da tabela 2.11 por

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^k f(x_i)}{k}$$

2.6.2.1 Algoritmo da Distância Pesada dos K-Vizinhos Mais Próximos

Um refinamento ao algoritmo dos k -vizinhos mais próximos consiste em pesar a contribuição de cada um dos k vizinhos de acordo com a sua distância ao ponto de “query” x_q , dando pesos maiores aos k vizinhos mais próximos. Por exemplo, no algoritmo da tabela 2.11, pode-se pesar o voto de cada vizinho de acordo com o inverso do quadrado da sua distância a x_q . Isto pode ser conseguido substituindo a última linha do algoritmo por

$$\hat{f}(x_q) \leftarrow \arg \max_{v \in V} \sum_{i=1}^k w_i \delta(v, f(x_i))$$

$$\text{em que } w_i = \frac{1}{d(x_q, x_i)^2}$$

Para acomodar o caso em que o ponto de “query” coincide com uma das instâncias x_i e em que o denominador $d(x_q, x_i)^2$ é zero, atribui-se $f(x_i)$ a $\hat{f}(x_q)$.

Pode-se pesar as instâncias para funções alvo reais de uma forma similar. Isto é possível substituindo a última linha do algoritmo por

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^k w_i f(x_i)}{\sum_{i=1}^k w_i}$$

em que w_i é igual ao anterior. O denominador na equação anterior é uma constante e normaliza os contributos dos vários pesos (assegura que se $f(x_i) = c$ para todos os exemplos de treino, então $\hat{f}(x_q) \leftarrow c$ também).

Com a introdução dos pesos dos vizinhos no algoritmo, é viável a inclusão de todos os exemplos de treino na classificação de x_q , pois exemplos muito distantes têm um pequeno efeito em $\hat{f}(x_q)$. A única desvantagem de considerar todos os exemplos é que a classificação torna-se mais lenta.

O algoritmo da distância pesada dos k -vizinhos mais próximos é um método de inferência indutiva altamente eficaz para muitos problemas práticos. É um método robusto a dados de treino com ruído e eficaz quando existe um conjunto suficientemente grande de dados de treino. De notar, que tomando a média pesada dos k vizinhos mais próximos do ponto de “query”, atenua-se o impacto dos exemplos de treino isolados com ruído.

A limitação indutiva do algoritmo dos k -vizinhos mais próximos corresponde à pressuposição de que a classificação de uma instância x_q é similar à classificação de outras instâncias que estão mais próximas (distância euclidiana) no espaço euclidiano.

Uma particularidade dos algoritmos dos K -vizinhos mais próximos reside no facto que a distância entre as instâncias é calculado a partir de todos os atributos da instância (em todos os eixos no espaço Euclidiano que contêm as instâncias). Isto contrasta com métodos como o método de aprendizagem de árvores de decisão que selecciona um subconjunto dos atributos da instância quando forma a hipótese final.

Considere-se um problema em que cada instância é descrita por vinte atributos mas em que só dois destes são relevantes para determinar a classificação para a função alvo. Neste caso, instâncias que têm idênticos valores para os dois atributos relevantes não deveriam estar distantes no espaço de instâncias de dimensão vinte, o que não acontece. A distância entre os vizinhos será dominada por um grande número de atributos irrelevantes.

Uma abordagem interessante para ultrapassar este problema consiste em pesar cada atributo de forma diferente quando se calcula a distância entre duas instâncias. Isto corresponde a alongar os eixos no espaço euclidiano que correspondem a atributos mais relevantes e encurtar os eixos que correspondem a atributos menos relevantes. O valor por que cada eixo deve ser alongado ou incurtado pode ser automaticamente determinado usando uma abordagem de validação cruzada. Para alongar ou incurtar os eixos é necessário multiplicar os atributos por um factor z_i em que os valores $z_1, \dots, z_n$ são escolhidos para minimizar o erro verdadeiro de classificação do algoritmo de aprendizagem. O erro pode ser estimado usando a validação cruzada. Selecciona-se um subconjunto aleatório de dados disponíveis para usar como exemplos de treino, e determina-se os valores $z_1, \dots, z_n$ que levam ao mínimo erro na classificação dos restantes exemplos. Com a repetição deste processo múltiplas vezes a estimativa para os pesos pode ser feita com maior exactidão. Este processo, de alongar ou incurtar os eixos de forma a otimizar o algoritmo dos K -vizinhos mais próximos possibilita um mecanismo de superar o impacto dos atributos irrelevantes. Uma alternativa ainda mais drástica é eliminar completamente os atributos menos importantes do espaço de instâncias. Isto é equivalente a atribuir a alguns factores z_i o valor zero.

Este algoritmo retarda todo o processo de classificação até que um novo “query” seja recebido, podendo ser requerido processamento significativo para cada “query”. Uma forma de melhorar o desempenho deste algoritmo é indexar eficientemente os exemplos de treino. Existem vários métodos de indexação de exemplos de treino. Um destes métodos de indexação é o “kd-tree” [Bentley-75; Friedman-77], em que as instâncias são armazenadas nas folhas de uma árvore,

com as instâncias vizinhas armazenadas no mesmo nó ou em nós próximos. Os nós internos da árvore ordenam o novo "query" x_q até uma folha da árvore. Esta ordenação é realizada pelo teste aos atributos de x_q .

2.6.2.2 Aproximações de Funções Alvo Reais Localmente Pesadas

A abordagem dos K -vizinhos mais próximos pode ser vista como uma aproximação de uma função alvo $f(x)$ num ponto de "query" único x_q . A aproximação de funções alvo reais localmente pesadas é uma generalização da abordagem anterior, pois constrói uma aproximação explícita de f sobre uma região circundante de x_q a partir da proximidade (distância pesada) dos exemplos de treino. A aproximação é local pois a função é baseada só nos pontos próximos do ponto de "query".

A aproximação à função alvo na vizinhança circundante de x_q pode ser feita usando uma função linear, uma função quadrática, uma rede neuronal multinível, ou outra função qualquer.

Dada uma nova instância de "query" x_q , a abordagem da aproximação de funções reais localmente pesadas constrói uma aproximação $\hat{f}$ que ajusta os exemplos de treino na vizinhança circundante de x_q . Esta aproximação é depois usada para calcular o valor $\hat{f}(x_q)$,

que é a estimativa do valor alvo para a instância de "query". A descrição de $\hat{f}$ deve ser eliminada, pois uma aproximação local diferente deve ser calculada para cada instância de "query" distinta.

2.6.2.2.1 Aproximação de Funções Alvo Lineares e Reais Localmente Pesadas

A aproximação de uma função alvo linear e real f localmente pesada pode ser feita por uma função linear da forma

$$\hat{f}(x) = w_0 + w_1 a_1(x) + \dots + w_n a_n(x)$$

em que $a_i(x)$ é o valor do i^o atributo da instância x .

Existem métodos como a descida da encosta para encontrar os coeficientes $w_0 \dots w_n$ de forma a minimizar o erro no ajuste de funções lineares a um dado conjunto de exemplos de treino. Nestes métodos, como se pretende uma aproximação global à função alvo, escolhe-se os pesos que minimizem a soma do quadrado do erro sobre o conjunto D de exemplos de treino

$$E = \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2$$

que leva à regra de treino de descida da encosta

$$\Delta w_j = \eta \sum_{x \in D} (f(x) - \hat{f}(x)) a_j(x) \quad (2.16)$$

em que η é uma constante de aprendizagem.

Nesta notação $f(x)$ corresponde a t , $\hat{f}(x)$ a o e $a_i(x)$ a x_j na anterior notação.

Com o intuito de adaptar este procedimento para obter uma aproximação local é necessário redefinir o critério de erro E para ajustar os exemplos de treino. Existem três tipos possíveis de critérios de erros:

1. Minimizar o quadrado do erro sobre os K -vizinhos mais próximos

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \in k\text{-vizinhos_mais_próximos_de_}x_q} (f(x) - \hat{f}(x))^2$$

2. Minimizar o quadrado do erro sobre o conjunto inteiro D de exemplos de treino, pesando o erro de cada exemplo de treino com uma função decrescente k da sua distância a x_q :

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 k(d(x_q, x))$$

3. Combinando 1 e 2

$$E_3(x_q) \equiv \frac{1}{2} \sum_{x \in k\text{-vizinhos_mais_próximos_de_}x_q} (f(x) - \hat{f}(x))^2 k(d(x_q, x))$$

O segundo critério de erro permite que todos os exemplos de treino tenham impacto na classificação de x_q . Contudo, esta abordagem requer processamento que cresce linearmente com o número de exemplos de treino. O terceiro critério é uma boa aproximação ao segundo critério e tem a vantagem de que o custo de processamento é independente do número total de exemplos de treino. Este custo depende só do número dos K -vizinhos considerados.

Escolhendo o terceiro critério de erro e utilizando a regra da descida da encosta obtêm-se a seguinte regra de treino

$$\Delta w_j = \eta \sum_{x \in k\text{-vizinhos_mais_próximos_de_}x_q} k(d(x_q, x)) (f(x) - \hat{f}(x)) a_j(x) \quad (2.17)$$

A única diferença entre a equação (2.17) e a equação (2.16) reside no facto que a contribuição da instância x à actualização do peso é multiplicada pela distância $k(d(x_q, x))$, e que o erro é assumido só sobre os K -vizinhos mais próximos.

Na aproximação linear localmente pesada de funções alvo reais, podem-se utilizar métodos alternativos para pesar a distância dos exemplos de treino, e métodos alternativos para localmente aproximar a função alvo. Na maioria dos casos, a função alvo é aproximada por uma função linear ou uma função quadrática. Outras funções mais complexas não são

frequentemente utilizadas pois o custo de ajustar funções complexas para cada instância de "query" é altamente proibitivo, ao passo que as anteriores aproximações modelam a função alvo suficientemente bem sobre uma pequena região do espaço de instâncias.

Existe uma grande diferença nas limitações indutivas entre os métodos baseados em instâncias e os outros métodos tratados anteriormente.

O algoritmo de aproximação de funções alvo lineares e reais quando surge um "query" x_q generaliza uma hipótese a partir dos dados de treino próximos de x_q . Em contraste, os algoritmos que não são baseados em instâncias e que usam o mesmo espaço de hipóteses de funções lineares devem chegar a uma hipótese antes dos "queries" serem observados. Esta hipótese é uma função linear que cobre todo o espaço de instâncias e futuros "queries". Os algoritmos baseados em instâncias usam um espaço de hipóteses rico pois usam muitas funções lineares locais diferentes para formar uma aproximação global à função alvo.

2.7 Algoritmos Genéticos

2.7.1 Introdução

Os algoritmos genéticos possibilitam uma abordagem de aprendizagem que é baseada na evolução simulada. As hipóteses são frequentemente descritas por “bit strings” cuja interpretação depende da aplicação, apesar das hipóteses também poderem ser descritas por expressões simbólicas ou até programas de computador. A pesquisa de uma hipótese apropriada começa com uma população, conjunto de hipóteses iniciais. Os membros da geração actual dão origem à população da nova geração por meio de operações tal como a mutação aleatória e o “crossover”.

Em cada etapa, as hipóteses na população corrente são avaliadas por uma determinada medida do ajuste e a melhor hipótese é seleccionada probabilisticamente para produzir a geração seguinte. Os algoritmos genéticos foram aplicados com sucesso em várias tarefas de aprendizagem e noutros problemas de optimização. Como exemplo, os algoritmos genéticos foram usados para aprender regras para o controlo de robots e para optimizar as topologias e os parâmetros de aprendizagem de redes neuronais.

Os algoritmos genéticos disponibilizam um método de aprendizagem análogo à evolução biológica. Os algoritmos genéticos geram hipóteses sucessivamente, pela repetição das mutações e recombinações de partes das melhores hipóteses correntes. Em cada etapa, um conjunto de hipóteses chamada de população corrente é actualizada pela substituição de uma fracção da população pela descendência das melhores hipóteses correntes.

O processo forma uma pesquisa de geração-e-teste de hipóteses, em que variantes das melhores hipóteses correntes devem ser consideradas a seguir. A popularidade dos algoritmos genéticos é motivada por um conjunto de factores que incluem:

- A evolução é reconhecida como um método com sucesso e robusto para a adaptação dentro dos sistemas biológicos.
- Os algoritmos genéticos podem pesquisar espaços de hipóteses contendo partes mais complexas e em que o impacto de cada parte no ajuste global da hipótese é difícil de modelar.

2.7.2. Algoritmos Genéticos

Os algoritmos genéticos pesquisam o espaço de hipóteses candidatas para identificar a melhor hipótese. A melhor hipótese é a que optimiza uma medida numérica predefinida para o problema chamado ajuste. Se a tarefa de aprendizagem consiste em aproximar uma função a partir de exemplos de treino então o ajuste pode ser definido como a exactidão da hipótese sobre os dados de treino. Se por outro lado, a tarefa consiste em aprender uma estratégia para jogar xadrez, o ajuste pode ser definido como o número de jogos ganhos por um indivíduo quando joga com outros indivíduos na população corrente.

Apesar das diferentes implementações dos algoritmos genéticos variarem em alguns detalhes, tipicamente realizam várias iterações de forma a actualizar o conjunto de hipóteses (população). Em cada iteração todos os membros da população são avaliados de acordo com a função ajuste. A nova população é gerada pela selecção probabilística das hipóteses com melhor ajuste da população corrente. Algumas destas hipóteses seleccionadas chegam à geração seguinte da população, intactas. Outras são usadas para criarem novas hipóteses descendentes pela aplicação de operações genéticas tais como o “crossover” e a mutação.

Um protótipo de um algoritmo genético é descrito na tabela 2.12 [Mitchell-97].

Tabela 2.12 – Um protótipo de um Algoritmo Genético

Algoritmo Genético (*Ajuste*, *Limite_Ajuste*, *p*, *r*, *m*)

Ajuste : uma função que a partir de uma hipótese chega a um resultado de avaliação

Limite_Ajuste : um limite especificando o critério de término

p : o número de hipóteses a serem incluídas na população

r : a fracção da população a ser substituída pelo “crossover” em cada etapa

m : o rácio da mutação

- Inicialização da população : $P \leftarrow$ gera p hipóteses aleatoriamente
- Avaliação : para cada h em P , calcula $Ajuste(h)$
- Enquanto $[\max_h Ajuste(h)] < Limite_Ajuste$ Fazer

 Criar uma nova geração, P_s :

1. Selecção : Selecção probabilística de $(1-r)p$ membros de P para adicionar a P_s . A probabilidade da hipótese h seleccionada de P é dada por

$$\Pr(h_i) = \frac{Ajuste(h_i)}{\sum_{j=1}^p Ajuste(h_j)}$$

2. “Crossover” : Selecção probabilística de $\frac{rp}{2}$ pares de hipóteses de P , de acordo com $\Pr(h_i)$. Para cada par, (h_1, h_2) , produz dois descendentes pela aplicação da operação de “crossover”. Adicionar todos os descendentes a P_s
3. Mutação : Escolher m por cento de membros de P_s com probabilidade uniforme. Para cada um, inverter um “bit” seleccionado aleatoriamente na sua representação
4. Actualizar : $P \leftarrow P_s$
5. Avaliação : Para cada h em P , calcular $Ajuste(h)$

- Retornar a hipótese de P que tem melhor $Ajuste$

Em cada iteração, a população P_s é formada pela selecção probabilística das hipóteses correntes de acordo com o seu ajuste e pela adição de novas hipóteses. As novas hipóteses são criadas pela aplicação de uma operação de “crossover” aos pares de hipóteses com melhor ajuste e pela criação de pontos de mutação na geração de hipóteses resultantes. Este processo é iterado até serem descobertas hipóteses aceitáveis.

Em cada iteração é feita uma selecção probabilística em que a probabilidade de uma hipótese ser seleccionada é proporcional ao seu próprio ajuste e inversamente proporcional ao ajuste das outras hipóteses na população corrente.

2.7.2.1 Representação de Hipóteses

As hipóteses nos algoritmos genéticos são frequentemente representados por “bit strings”, sendo por isso facilmente manipuladas por operadores genéticos tais como a mutação e o “crossover”. As hipóteses representadas por estas “bit strings” podem ser complexas. Um conjunto de regras se-então pode ser facilmente representado desta forma, escolhendo uma codificação de regras que aloca “substrings” específicas para cada pré-condição ou pós-condição da regra.

Considere-se o atributo Conteúdo Técnico/Científico, que pode tomar os valores médio, bom e muito bom. Uma forma óbvia de representar uma restrição no atributo Conteúdo Técnico/Científico é usar uma “bit string” de tamanho três, em que cada bit corresponde a um dos três possíveis valores. O número 1 numa posição indica que o atributo pode assumir o valor correspondente. Por exemplo, a string 010 representa a restrição *Conteúdo Técnico/Científico = bom*. Similarmente a string 011 representa uma restrição mais geral que permite dois valores possíveis (*Conteúdo Técnico/Científico = bom ∨ muito bom*). De notar que 111 representa a restrição mais geral possível, indicando que não é importante qual o valor que o atributo toma.

Com este método de representação de restrições num atributo, as conjunções de restrições em múltiplos atributos podem facilmente ser representadas pela concatenação das “bit strings” correspondentes. Por exemplo, considere-se um segundo atributo, Preço, que pode tomar os valores normal ou alto. Uma pré-condição de uma regra como

$$(Conteúdo\ Técnico/Científico = bom \vee muito\ bom) \wedge (Preço = normal)$$

pode ser representada pela seguinte “bit string” de comprimento cinco:

Conteúdo Técnico/Científico	Preço
011	10

As pós-condições das regras (tal como *Comprar=Sim*) podem ser representadas de uma forma similar. Assim, uma regra pode ser representada pela concatenação de “bit strings” que descrevem as pré-condições da regra, e a “bit string” que descreve a pós-condição da regra. Por exemplo, a regra

$$Se\ Preço = normal\ Então\ Comprar = Sim$$

seria representada pela string

Conteúdo Técnico/Científico	Preço	Comprar
111	10	10

A “bit string” que representa a regra contém uma “substring” para cada atributo no espaço de hipóteses, mesmo se esse atributo não é restringido pelas pré-condições da regra. Isto leva a uma representação de “bit strings” de regras de tamanho fixo, em que “substrings” em locais específicos descrevem restrições em atributos específicos. Dada esta representação para as

regras, pode-se representar conjuntos de regras concatenando similarmente as representações das “bit strings” de cada regra.

2.7.2.2 Operadores Genéticos

A geração de descendentes nos algoritmos genéticos é determinada por um conjunto de operadores que recombina e mutua membros seleccionados da população corrente. Os operadores típicos dos algoritmos genéticos para manipular hipóteses de “bit strings” estão ilustrados na tabela 2.13. Estes operadores correspondem a versões de operações genéticas encontradas na evolução biológica. Os dois operadores mais comuns são o “crossover” e a mutação.

O operador “crossover” a partir de duas hipóteses (“bit strings”) da geração corrente cria duas hipóteses descendentes copiando “bits” seleccionados das “bit strings” da geração corrente. O bit na posição i em cada descendente é copiado do bit da posição i de uma das duas “bit strings” da geração corrente. A escolha de qual das “bit strings” contribui com o bit para a posição i é determinada por uma string adicional chamada máscara de “crossover”. Como ilustração, considere-se o operador de “crossover” com um ponto na tabela 2.13 [Mitchell-97].

	Strings Iniciais	Máscara do “crossover”	Descendentes
“Crossover” com um ponto :	11101001000 00001010101	11111000000	11101010101 00001001000
“Crossover” com dois pontos :	11101001000 00001010101	00111110000	11001011000 00101000101
“Crossover” uniforme :	11101001000 00001010101	10011010011	10001000100 01101011001
Mutação com um ponto :	11101001000		11101011000

Tabela 2.13 – Operadores comuns para Algoritmos Genéticos

O primeiro descendente tem os cinco (n) primeiros “bits” de uma das “bit strings” e os restantes seis “bits” da outra, pois a máscara de “crossover” 11111000000 determina estas escolhas para cada uma das posições dos “bits”. O segundo descendente usa a mesma máscara de “crossover”, mas troca os papéis das “bit strings” da geração corrente. O segundo descendente contém os “bits” que não foram usados pelo primeiro descendente. De cada vez que o operador “crossover” com um ponto é aplicado, o ponto n de “crossover” é escolhido aleatoriamente e a máscara de “crossover” é criada e aplicada.

No “crossover” com dois pontos, os descendentes são criados pela substituição de segmentos intermédios de uma das “bit strings” da geração corrente no meio da outra. Isto é, a máscara de “crossover” é uma string com n_0 zeros, seguida por uma string contígua de n_1 uns, seguida pelo número de zeros necessários para completar a string. De cada vez que o operador “crossover” com dois pontos é aplicado, uma máscara é gerada escolhendo aleatoriamente n_0 e n_1 . No exemplo da tabela 2.13 os descendentes são criados usando uma máscara com $n_0=2$ e

$n_i=5$. De novo, os dois descendentes são criados trocando os papéis das “bit strings” da geração corrente.

No “crossover” uniforme a máscara de “crossover” é gerada como uma “bit string” aleatória em que cada bit é escolhido aleatoriamente e independentemente dos outros.

O operador mutação produz um descendente de uma única “bit string” da geração corrente.

Mais concretamente, o operador mutação produz pequenas mudanças aleatórias na “bit string” escolhendo um bit aleatoriamente, e mudando o seu valor. O operador mutação é frequentemente usado depois do “crossover” ter sido aplicado.

Alguns sistemas de algoritmos genéticos empregam operadores adicionais, especialmente operadores que são especializados em representações particulares de hipóteses usadas pelo sistema.

2.7.2.3 Função Ajuste e Selecção

A função ajuste define o critério de avaliação das potenciais hipóteses, de forma a seleccioná-las probabilisticamente e inclui-las na população da geração seguinte. Se a tarefa consiste em aprender regras de classificação, então a função ajuste tipicamente tem um componente que avalia a exactidão da classificação da regra sobre um conjunto de exemplos de treino disponíveis. Outros critérios são frequentemente incluídos, tal como a complexidade ou a generalidade da regra. Quando a hipótese (“bit string”) é interpretada como um procedimento complexo (por exemplo, quando a “bit string” representa um conjunto de regras se-então que serão todas encadeadas para controlar um robot), a função ajuste deve medir o desempenho global do procedimento resultante em vez do desempenho das regras individualmente.

No protótipo do algoritmo genético da tabela 2.12, a probabilidade de uma hipótese ser seleccionada é dada pelo rácio do seu ajuste sobre o ajuste dos outros membros da população corrente. No entanto, existem outros métodos que usam o ajuste para seleccionar as hipóteses. Na selecção por “torneio” são inicialmente escolhidas duas hipóteses aleatoriamente da população corrente. Com uma probabilidade predefinida p , é seleccionada a hipótese de melhor ajuste dessas duas hipóteses e a outra hipótese é seleccionada com probabilidade $(1-p)$.

Este tipo de selecção frequentemente leva a uma população mais diversificada do que a anterior selecção.

Noutro método com a denominação selecção por “ordenação”, as hipóteses na população corrente são primeiro ordenadas pelo seu ajuste. A probabilidade de uma hipótese ser seleccionada é proporcional ao seu lugar nessa lista ordenada (e não proporcional ao seu ajuste).

2.7.3 Pesquisa do Espaço de Hipóteses

A pesquisa da melhor hipótese nos algoritmos genéticos é diferente da de outros métodos de aprendizagem já considerados. Por exemplo, a pesquisa do espaço de hipóteses no “Backpropagation” utiliza a descida da encosta, passando da hipótese corrente para uma nova hipótese que é muito similar à hipótese corrente. Em contraste, a pesquisa nos algoritmos genéticos pode ser muito mais abrupta, substituindo uma hipótese por outra que pode ser radicalmente diferente. Por isso, a pesquisa nos algoritmos genéticos tem menos tendência a cair no mesmo tipo de mínimos locais do que o método da descida da encosta.

Uma dificuldade prática em algumas aplicações dos algoritmos genéticos é o problema do “crowding”. O “crowding” é um fenómeno que consiste no facto que uma hipótese com

melhor ajuste do que outras, depressa se reproduz, verificando-se que cópias desta hipótese e de outras muito similares fazem parte de uma grande fracção da população. O impacto negativo do “crowding” reside em reduzir a diversidade da população, atrasando a evolução dos algoritmos genéticos. Muitas estratégias foram exploradas para reduzir o “crowding”. Uma estratégia consiste em alterar a função selecção usando outro critério, tal como a selecção por “torneio” ou por “ordenação” em vez da selecção probabilística do ajuste das hipóteses. Uma outra estratégia é a chamada “partilha do ajuste”, em que a medida do ajuste de uma hipótese é reduzida pela presença de outras hipóteses similares na população.. Muitas destas técnicas são inspiradas por analogia à evolução biológica.

2.7.4 Programação Genética

A programação genética é uma forma de computação evolutiva em que os indivíduos (hipóteses) na evolução da população são programas de computador em vez de “bit strings”. [Koza-92] descreveu a abordagem básica de programação genética e apresenta um leque de programas simples que podem ser aprendidos com sucesso pela programação genética.

Os programas manipulados por programação genética são tipicamente representados por árvores correspondendo à árvore de “parse” do programa. Cada função é representada por um nó na árvore, e os argumentos para a função são dados pelos seus nós descendentes. Por exemplo, a figura 2.19 ilustra esta representação em árvore para a função $\text{seno}(x) + \sqrt{x^2 + y}$ [Mitchell-97].

Para aplicar a programação genética a um domínio particular, devem-se definir as funções primitivas a serem consideradas (por exemplo, *seno*, *coseno*, $\sqrt{}$, +, -), assim como os terminais (por exemplo, *x*, *y*, constantes como por exemplo o 2). O algoritmo de programação genética usa depois uma pesquisa evolutiva para explorar o vasto espaço de programas que pode ser descrito usando estas primitivas.

Tal como nos algoritmos genéticos, o protótipo do algoritmo de programação genética faz a manutenção da população de hipóteses (neste caso, árvores dos programas). Em cada iteração produz uma nova geração de hipóteses usando a selecção, o “crossover” e a mutação. O ajuste de um dado programa da população é tipicamente determinado executando o programa num conjunto de dados de treino. As operações de “crossover” são realizadas substituindo uma sub-árvore escolhida aleatoriamente de um programa por uma sub-árvore de outro programa. A figura 2.20 ilustra uma operação típica de “crossover” [Mitchell-97].

[Koza-92] descreve um conjunto de experiências aplicando a programação genética a determinadas aplicações. Nas suas experiências, 10% da população corrente, seleccionada probabilisticamente de acordo com o ajuste, é retida sem alteração na próxima geração. A parte restante da nova geração é criada pela aplicação do “crossover” a pares de programas da geração corrente, outra vez seleccionada probabilisticamente de acordo com o seu ajuste.

Um exemplo ilustrativo apresentado por [Koza-92] envolve a aprendizagem de um algoritmo para o empilhamento de blocos de forma a construir a palavra universal (cada bloco contém uma letra).

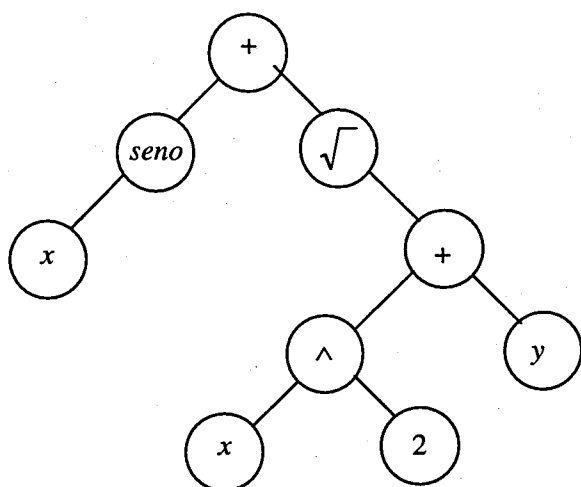


Figura 2.19 – Representação em Árvore de uma Função na Programação Genética

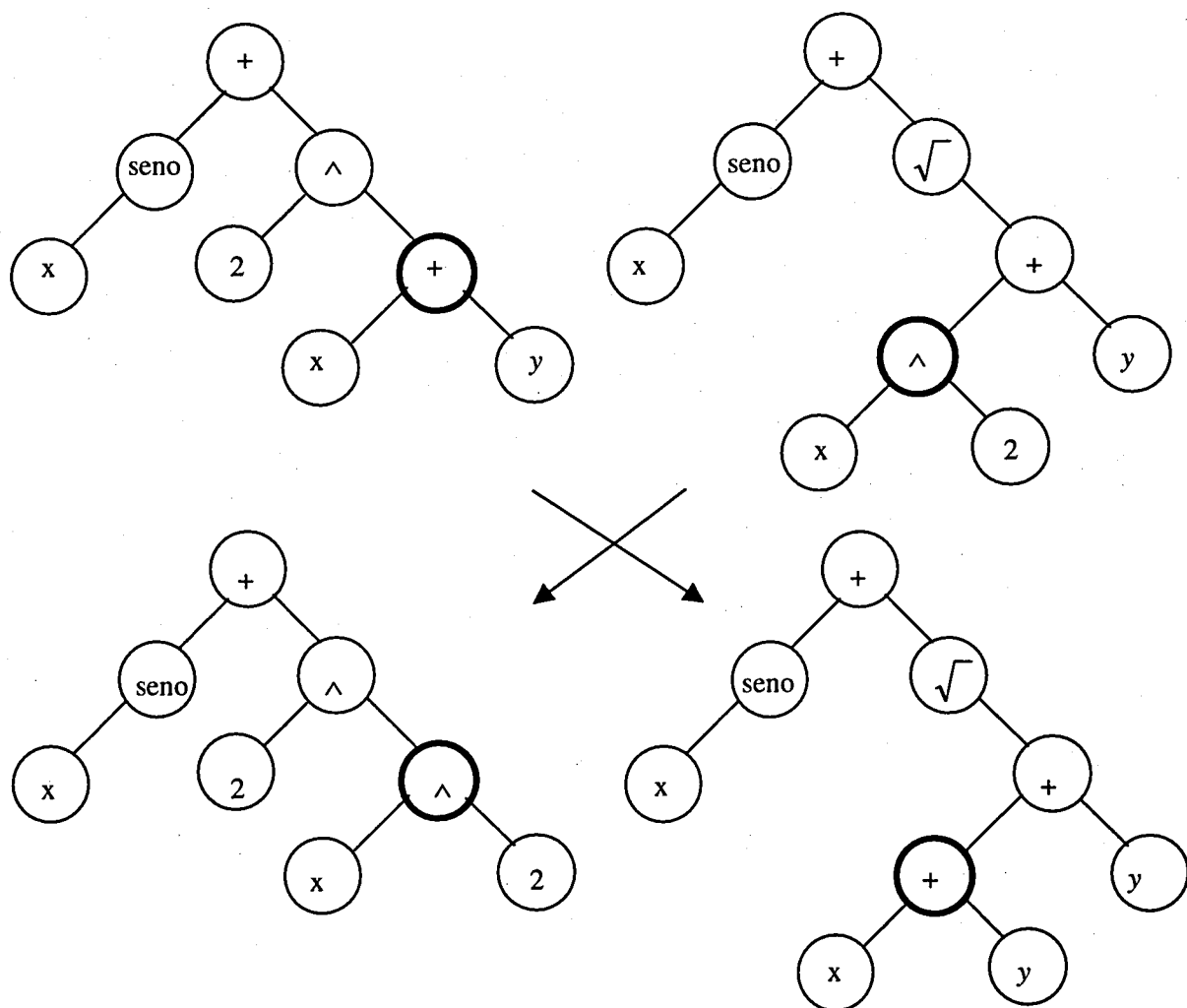


Figura 2.20 - Operação de Crossover

2.8 Aprendizagem de Conjuntos de Regras

2.8.1 Introdução

Uma das representações mais expressivas para aprender hipóteses são os conjuntos de regras se-então. Em muitos casos é útil aprender a função alvo representada como um conjunto de regras se-então que conjuntamente definem a função. Uma forma de aprender um conjunto de regras consiste em aprender uma árvore de decisão e transformar a árvore num conjunto de regras equivalente (uma regra para cada nó folha na árvore). Outra forma consiste em usar um algoritmo genético que codifica cada conjunto de regras como uma “bit string” e usar os operadores de pesquisa genética para explorar esse espaço de hipóteses.

No entanto, existem outros algoritmos (algoritmos de cobertura sequencial) que aprendem directamente conjuntos de regras. Estes algoritmos diferem dos anteriores pois foram construídos para aprender conjuntos de regras de primeira ordem (contêm variáveis) e usam a cobertura sequencial para aprender uma regra de cada vez para incrementalmente construir o conjunto de regras final.

2.8.2 Algoritmos de Cobertura Sequencial

Os algoritmos de cobertura sequencial aprendem conjuntos de regras baseados na estratégia de aprender uma regra, remover os dados que ela cobre, e voltar a iterar este processo.

Considere-se a subrotina *Aprender-uma-regra*, que aceite um conjunto de exemplos de treino positivos e negativos como “input”, e que retorne uma regra que cubra alguns dos exemplos positivos. Dada esta subrotina, uma abordagem óbvia para aprender um conjunto de regras consiste em invocar a subrotina *Aprender-uma-regra* para todos os exemplos de treino disponíveis. Cada vez que o algoritmo aprende uma nova regra deve remover qualquer exemplo positivo coberto por essa regra, e invocar novamente a subrotina para aprender outra regra a partir do resto dos exemplos de treino. Este procedimento pode ser iterado tantas vezes quantas desejado para aprender um conjunto disjuntivo de regras que conjuntamente cobrem todo o conjunto de exemplos positivos. O conjunto final de regras pode ser ordenado de forma a que as regras com maior exactidão sejam consideradas primeiro quando uma nova instância deva ser classificada. Um protótipo de um algoritmo de cobertura sequencial é descrito na tabela 2.14 [Mitchell-97].

Este algoritmo é uma das abordagens mais divulgadas para aprender conjuntos disjuntivos de regras e reduz o problema de aprender um conjunto disjuntivo de regras a uma sequência de problemas mais simples. Uma vez que realiza uma pesquisa que forma uma sequência de regras sem “backtracking”, não garante encontrar o mais pequeno ou o melhor conjunto de regras que cobrem os exemplos de treino. A subrotina *Aprender-uma-regra* deve formular uma regra com grande exactidão, mas que não necessita cobrir todos os exemplos positivos.

Uma abordagem para implementar a subrotina *Aprender-uma-regra* consiste em organizar a pesquisa do espaço de hipóteses da mesma forma que o algoritmo ID3, mas em cada etapa seguir só o ramo mais promissor da árvore. Como é ilustrado na árvore de pesquisa da figura 2.21, a pesquisa começa por considerar a pré-condição mais geral possível (que cobre todas as instâncias), adicionando-se de seguida o teste ao atributo que permita obter o melhor desempenho da regra sobre os exemplos de treino. Uma vez adicionado este teste, o processo é repetido adicionando-se um segundo teste de atributo, e assim sucessivamente. Tal como com o ID3, este processo constrói a hipótese adicionando novos testes de atributo até a hipótese atingir um nível de desempenho aceitável. Contrariamente ao ID3, esta implementação de

Aprender-uma-regra segue só um único descendente em cada etapa de pesquisa (o atributo que tem melhor desempenho) em vez de construir uma sub-árvore que cobre todos os valores possíveis de um atributo seleccionado.

Tabela 2.14 – Algoritmo de Cobertura Sequencial para aprender um conjunto disjuntivo de regras

Cobertura_Sequencial (*Atributo_alvo*, *Atributos*, *Exemplos*, *Limite*)

- $Regras_aprendidas \leftarrow \{\}$
- $Regra \leftarrow \text{Aprender-uma-regra} (Atributo_alvo, Atributos, Exemplos)$
- Enquanto $Desempenho (Regra, Exemplos) > Limite$, Fazer
 - $Regras_aprendidas \leftarrow Regras_aprendidas + Regra$
 - $Exemplos \leftarrow Exemplos - \{Exemplos \text{ correctamente classificados pela regra}\}$
 - $Regra \leftarrow \text{Aprender-uma-regra} (Atributo_alvo, Atributos, Exemplos)$
- $Regras_aprendidas \leftarrow \text{Ordenar } Regras_aprendidas \text{ de acordo com o desempenho sobre os exemplos}$
- Retornar $Regras_aprendidas$

Esta abordagem para implementar a subrotina Aprender-uma-regra realiza uma pesquisa geral-para-específica pelo espaço de regras possíveis de forma a encontrar uma regra com uma boa exactidão. Assim, como na aprendizagem de árvores de decisão, existem muitas formas de definir uma medida para seleccionar o “melhor” descendente.

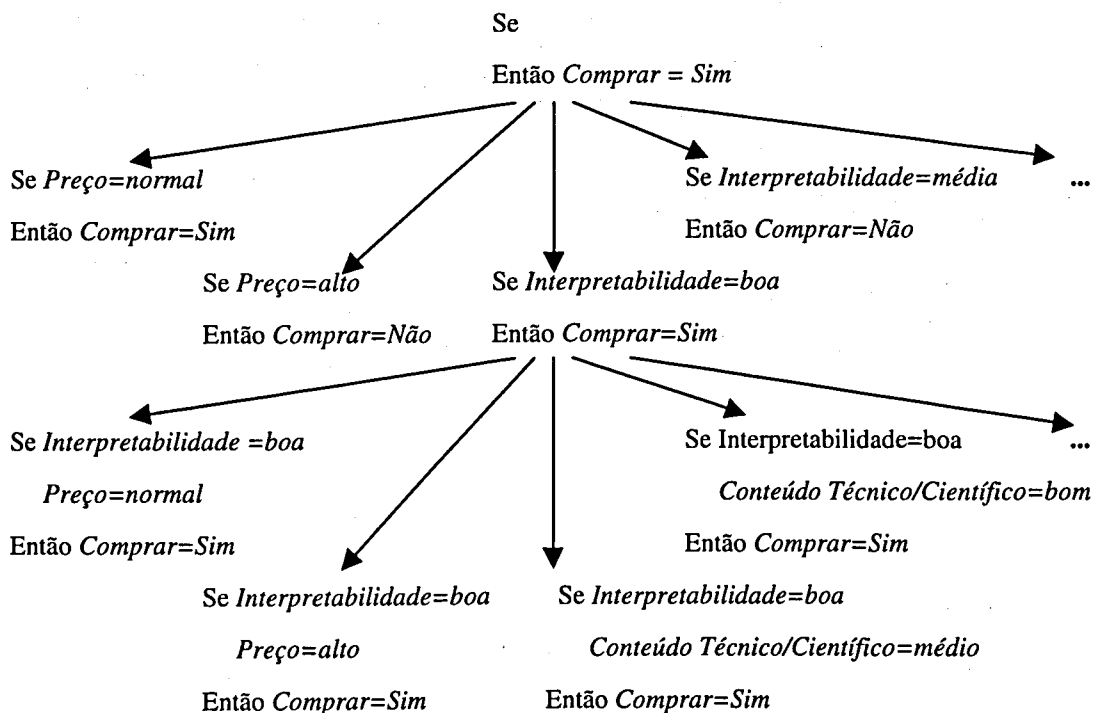


Figura 2.21 – Pesquisa de pré-condições da regra

Se seguirmos o algoritmo ID3, o melhor descendente é o que cujos exemplos cobertos tenham a menor entropia.

Esta pesquisa é uma pesquisa de 'profundidade um sem backtracking. Como em todas as pesquisas deste tipo, existe o perigo de em cada etapa ser feita uma escolha que não é a ótima. Para reduzir este risco, pode-se estender o algoritmo para realizar uma pesquisa em que o algoritmo mantém uma lista dos k melhores candidatos em cada etapa, em vez de um único (melhor candidato).

**Tabela 2.15- O Algoritmo de Pesquisa Geral-para-Específica
Aprender-uma-Regra**

Aprender-uma-Regra (*Atributo_alvo*, *Atributos*, *Exemplos*, k)

Retorna uma regra que cobre alguns exemplos. Realiza uma pesquisa geral-para-específica para a obtenção da melhor regra (a que tiver melhor desempenho).

- Inicializar *Melhor_hipótese* à hipótese mais geral $\emptyset$
- Inicializar *Hipóteses_candidatas* ao conjunto {*Melhor_hipótese*}
- Enquanto *Hipóteses_candidatas* não estiver vazia, Fazer
 1. Gerar as próximas *Hipóteses_candidatas* mais específicas
 - *Todas_restrições* $\leftarrow$ o conjunto de todas as restrições da forma $(a=v)$, onde a é um membro dos *Atributos*, e v é um valor de a que ocorre no actual conjunto de exemplos
 - *Novas_hipóteses_candidatas* $\leftarrow$
 - Para cada h nas *Hipóteses_candidatas*
 - Para cada restrição c em *Todas_restrições*
 - Criar uma especialização de h adicionando a restrição c
 - Remover das *Novas_hipóteses_candidatas* alguma hipótese que esteja duplicada, ou que seja inconsistente, ou que não é o máximo específica
 2. Actualizar *Melhor_hipótese*
 - Para todas as hipóteses h nas *Novas_hipóteses_candidatas* Fazer
 - Se (*Desempenho*(h , *Exemplos*, *Atributo_alvo*)
 $>$ *Desempenho*(*Melhor_hipótese*, *Exemplos*, *Atributo_alvo*))
 - Então *Melhor_hipótese* $\leftarrow h$
 3. Actualizar *Hipóteses_candidatas*
 - *Hipóteses_candidatas* $\leftarrow$ Os k melhores membros das *Novas_hipóteses_candidatas* de acordo com a medida do desempenho
- Retornar uma a regra da forma

"Se *Melhor_hipótese* Então *Predição*"

em que *Predição* é o valor mais frequente do *Atributo_alvo* entre os exemplos que instânciam *Melhor_hipótese*

Desempenho(h , *Exemplos*, *Atributo_alvo*)

- *H_exemplos* $\leftarrow$ O subconjunto de exemplos que instânciam h
- Retornar – *Entropia*(*H_exemplos*), em que *Entropia* é respeitante ao *Atributo_alvo*

Em cada etapa de pesquisa, são geradas especializações (descendentes) para cada um dos k melhores candidatos, e o conjunto resultante é novamente reduzido aos k membros mais promissores. Este algoritmo é usado no programa CN2 descrito por [Clark-89]. O algoritmo é descrito na tabela 2.15 [Mitchell-97].

Cada hipótese considerada no ciclo principal do algoritmo é uma conjunção de restrições aos valores dos atributos. Cada uma destas hipóteses conjuntivas correspondem a um conjunto candidato de pré-condições para a regra que se pretende aprender e cada uma delas é avaliada pela entropia dos exemplos que cobre. A pesquisa considera hipóteses candidatas cada vez mais específicas até obter uma hipótese o máximo específica que contém todos os atributos disponíveis. A regra que o algoritmo retorna é a regra encontrada durante a pesquisa cujo desempenho é maior. A pós-condição da regra é escolhida só na etapa final do algoritmo, depois das pré-condições (representadas por *Melhor_hipótese*) serem determinadas. O algoritmo constrói uma regra para prever o valor do atributo alvo que é mais comum entre os exemplos cobertos pelas pré-condições da regra.

O algoritmo de cobertura sequencial conjuntamente com o algoritmo Aprender-uma-regra, aprende um conjunto de regras se-então que cobrem os exemplos de treino. No entanto foram exploradas muitas variações a esta abordagem. Em alguns casos é desejável que o programa aprenda só regras que cobrem exemplos positivos e que atribua por defeito uma classificação negativa a instâncias não cobertas por nenhuma regra. Esta abordagem corresponde à estratégia de “negação-como-falha” do Prolog, em que qualquer expressão que não seja provada como verdadeira é assumida por defeito como falsa. De forma a que o algoritmo aprenda estas regras que predizem um valor alvo, pode-se modificá-lo para aceitar um argumento de “input” adicional especificando o valor alvo. A pesquisa geral-para-específica é conduzida como anteriormente, mudando só a subrotina *Desempenho* que avalia as hipóteses. A avaliação do desempenho com a entropia negativa não é apropriada neste caso, pois atribui um resultado máximo a hipóteses que cobrem exclusivamente exemplos negativos, assim como as que cobrem exclusivamente exemplos positivos. O uso de uma medida que avalia a fracção de exemplos positivos cobertos pela hipótese seria mais apropriado neste caso.

2.8.2.1 Algoritmos de Cobertura Sequencial Versus Algoritmos de Cobertura Simultânea

Os algoritmos de cobertura sequencial aprendem uma regra de cada vez, removendo os exemplos cobertos e repetindo o processo nos restantes exemplos, enquanto os algoritmos baseados em árvores de decisão, tal como o ID3, aprendem todo o conjunto de disjunções simultaneamente.

Os algoritmos com uma estratégia de pesquisa igual à do algoritmo ID3 são considerados algoritmos de cobertura simultânea. A diferença chave entre estes algoritmos e os algoritmos de cobertura sequencial reside na escolha realizada na pesquisa. Em cada etapa, o ID3 escolhe entre atributos alternativos pela comparação das partições de dados que eles geram. Em contraste, o CN2 escolhe entre pares de atributo-valor alternativos, pela comparação dos subconjuntos de dados que eles cobrem. Uma forma de analisar o significado desta diferença é comparar o número de escolhas distintas feitas pelos dois algoritmos de forma a aprender o mesmo conjunto de regras. Para aprender um conjunto de n regras, cada uma contendo k atributos nas suas pré-condições, os algoritmos de cobertura sequencial realizam $n \cdot k$ etapas de pesquisa, realizando escolhas independentes na selecção de cada pré-condição de cada regra.

Em contraste, os algoritmos de cobertura simultânea realizam muito menos escolhas independentes, pois cada escolha de um nó de decisão na árvore de decisão corresponde a escolher a pré-condição para as múltiplas regras associadas com esse nó. Se um nó de decisão testa um atributo que tem m valores possíveis, a escolha desse nó de decisão corresponde à escolha de uma pré-condição para cada uma das m regras correspondentes. Assim, pode-se

concluir que os algoritmos de cobertura sequencial fazem um maior número de escolhas independentes do que os algoritmos de cobertura simultânea.

Se os dados forem abundantes, então possibilitam o grande número de escolhas independentes requeridas pelos algoritmos de cobertura sequencial, enquanto que se forem escassos, deve ser mais eficiente a partilha de decisões relativas às pré-condições de diferentes regras. Nos algoritmos de aprendizagem de árvores de decisão é desejável que regras diferentes testem os mesmos atributos, enquanto que nos algoritmos de cobertura sequencial isso não é necessário.

O algoritmo Aprender-uma-regra pode ter dois sentidos de pesquisa, o sentido das hipóteses gerais-para-específicas (por exemplo, o CN2) e o sentido das hipóteses específicas-para-gerais (por exemplo, o "Find-S"). Uma vantagem da pesquisa geral-para-específica é que existe só uma única hipótese o máximo geral a partir da qual se começa a pesquisa, existindo em contrapartida muito mais hipóteses o máximo específicas na maior parte dos espaços de hipóteses (uma por cada instância). Este facto pode dificultar a escolha da hipótese inicial na pesquisa específica-para-geral.

Na presença de muitas hipóteses o máximo específicas, não é claro qual delas seleccionar como ponto de partida da pesquisa. Um programa que conduz uma pesquisa específica-para-geral, chamado Golem [Muggleton-90], resolve esta questão escolhendo vários exemplos positivos aleatoriamente para inicializar e conduzir a pesquisa. A melhor hipótese obtida entre as múltiplas escolhas aleatórias é então seleccionada.

A pesquisa no algoritmo Aprender-uma-regra pode ser uma pesquisa de gerar-e-testar (como aqui referida) as hipóteses sintacticamente legais ou uma pesquisa guiada por exemplos de treino em que os exemplos de treino restringem a geração de hipóteses ("Find-S", "Candidate-Elimination", Cigol, entre outros). Em cada um dos algoritmos em que a pesquisa é guiada por exemplos, a geração ou correcção das hipóteses faz-se a partir da análise de cada exemplo de treino, e em que o resultado é uma hipótese corrigida com bom desempenho para esse exemplo. Isto contrasta com a pesquisa de geração-e-teste em que as sucessivas hipóteses são geradas baseadas só na sintaxe da representação da hipótese. Os dados de treino são só considerados depois dessas hipóteses candidatas serem geradas. As hipóteses candidatas são escolhidas a partir do seu desempenho sobre todo o conjunto de exemplos de treino.

Uma vantagem importante da abordagem de geração-e-teste reside no facto que cada escolha na pesquisa é baseada no desempenho das hipóteses sobre muitos exemplos, sendo o impacto do ruído nos dados minimizado. Contrariamente, os algoritmos conduzidos por exemplos que refinam a hipótese a partir dos exemplos individualmente, são mais facilmente afectados pelo ruído num exemplo de treino e são assim menos robustos aos erros. Com métodos de aprendizagem baseados em árvores de decisão, é possível a subrotina Aprender-uma-regra formular regras que têm um desempenho muito bom nos dados de treino, mas menor desempenho nos dados subsequentes. Uma forma de resolver esta questão consiste no uso do "post-pruning" de cada regra depois de ser aprendida. Mais propriamente, podem ser removidas pré-condições da regra sempre que isto leve a melhoria no desempenho sobre um conjunto de exemplos diferente dos exemplos de treino.

2.8.2.2 Funções Alternativas de Avaliação do Desempenho das Regras

Têm sido utilizadas funções alternativas de avaliação do desempenho das regras na subrotina Aprender-uma-Regra. As funções de avaliação mais comuns são:

- Frequência relativa. Sendo n o número de exemplos que a regra cobre e n_c o número destes que a regra classifica correctamente, a estimativa da frequência relativa do desempenho da regra é

$$\frac{n_c}{n}$$

- A estimativa m da exactidão. Esta estimativa da exactidão é preferível quando os dados são escassos e a regra deve ser avaliada baseada em poucos exemplos. Considere-se n o número de exemplos cobertos pela regra, n_c o número de exemplos correctamente classificados pela regra e p a probabilidade prévia de um exemplo aleatório do conjunto de dados ter a classificação atribuída pela regra (por exemplo, se 12 dos 100 exemplos têm o valor predito pela regra, então $p = 0.12$). Considerando m o peso, ou o número equivalente de exemplos para pesar a probabilidade prévia p , a estimativa m da exactidão da regra é

$$\frac{n_c + mp}{n + p}$$

Se for atribuído zero a m então a estimativa m é equivalente à estimativa da frequência relativa. A medida da estimativa de m [Cestnik-91] foi usada em algumas versões do algoritmo CN2 e é usada também na classificação simples de Bayes.

- Entropia. Esta é a medida usada pela subrotina *Desempenho* no algoritmo da tabela 2.15. Considere-se S o conjunto de exemplos que são cobertos pelas pré-condições da regra. A entropia mede a uniformidade dos valores da função alvo para esse conjunto de exemplos. Considera-se a entropia negativa, pelo que regras melhores têm valores maiores também.

$$-Entropia(S) = \sum_{i=1}^c p_i \log_2 p_i$$

em que c é o número de valores distintos que a função alvo pode tomar, e p_i é a proporção de exemplos de S em que a função alvo toma o valor de i . A medida da entropia é usada no algoritmo CN2 [Clark-89]. Constitui também a base da medida do ganho de informação usada em muitos algoritmos de aprendizagem de árvores de decisão.

2.8.3 Aprendizagem de Regras de Primeira Ordem

Uma regra de primeira ordem é uma regra que contém variáveis, contrariamente às regras proposicionais. Estas regras são também chamadas de “*Horn clauses*” de primeira ordem e são muito mais expressivas do que as regras proposicionais. A aprendizagem indutiva de regras de primeira ordem é frequentemente referida como Programação Lógica Indutiva (PLI), pois este processo pode ser visto como a inferência automática de programas Prolog a partir de exemplos.

Considere-se a tarefa de aprender o conceito alvo simples $Filha(x,y)$, definido sobre pares da pessoa x e y . O valor de $Filha(x,y)$ é verdadeiro se a filha de x é y e falso caso contrário. Supondo que cada pessoa é descrita pelos atributos Nome, Mãe, Pai, Masculino e Feminino, cada exemplo de treino consiste na descrição de duas pessoas em termos dos seus atributos, e o valor do atributo alvo Filha. Considere-se o exemplo positivo em que Maria é filha do António:

$(Nome_1 = Maria,$	$Mãe_1 = Luísa,$	$Pai_1 = António,$
$Masculino_1 = Falso,$	$Feminino_1 = Verdade,$	
$Nome_2 = António,$	$Mãe_2 = Marta,$	$Pai_2 = Vitor,$
$Masculino_2 = Verdade,$	$Feminino_2 = Falso,$	$Filha_{1,2} = Verdade)$

e em que os números em cada atributo indicam qual das duas pessoas está a ser descrita. Um algoritmo de regras proposicionais tal como o CN2, a partir um conjunto de exemplos de treino para o conceito alvo *Filha_{1,2}* obtém um conjunto de regras muito específicas, tal como

Se ($Pai_1 = \text{António}$) $\wedge$ ($Nome_2 = \text{António}$) $\wedge$ ($Feminino_1 = \text{Verdade}$)
Então $Filha_{1,2} = \text{Verdade}$

Apesar de ser correcta, esta regra é tão específica que raramente, senão nunca, será útil na classificação de pares de pessoas futuras. O problema consiste que as representações proposicionais não oferecem uma forma geral de descrever relações essenciais entre os valores dos atributos. Em contraste, um programa usando representações de primeira ordem pode aprender a seguinte regra geral:

Se $pai(y, x) \wedge Feminino(y)$ Então $Filha(x, y)$

em que x e y são variáveis que podem estar ligadas a qualquer pessoa.

As “*Horn Clauses*” de primeira ordem podem ter variáveis nas pré-condições que não ocorrem nas pós-condições. Por exemplo, uma regra para Neta pode ser

Se $Pai(y, z) \wedge Mãe(z, x) \wedge Feminino(y)$ Então $Neta(x, y)$

2.8.4. Algoritmo FOIL

Vários algoritmos foram propostos para aprender regras de primeira ordem. O programa FOIL [Quinlan-90] emprega uma abordagem muito similar aos algoritmos de cobertura sequencial e Aprender-uma-regra. O programa FOIL é uma extensão destes algoritmos para representações de primeira ordem, e como tal, as hipóteses aprendidas pelo FOIL são conjuntos de regras de primeira ordem. O algoritmo FOIL é sumariado na tabela 2.16 [Mitchell-97].

O ciclo exterior corresponde a uma variante do algoritmo de cobertura sequencial, pois aprende novas regras, uma de cada vez, removendo os exemplos positivos cobertos pela última regra antes de aprender a próxima regra.

O ciclo interno corresponde a uma variante do algoritmo Aprender-uma-regra, estendido para acomodar as regras de primeira ordem. O algoritmo FOIL tem a particularidade de só procurar regras que só fazem predições quando o valor alvo é verdadeiro. A pesquisa do espaço de hipóteses realizada pelo FOIL é mais facilmente entendido se for visto de um ponto de vista hierárquico. Cada iteração do ciclo exterior do programa FOIL adiciona uma nova regra à disjunção de hipóteses *Regras_aprendidas*. O efeito de cada nova regra é generalizar a hipótese disjuntiva corrente (aumentando o número de instâncias que classifica como positiva), com a adição de uma nova disjunção. Vista desta forma, a pesquisa é específica-para-geral pelo espaço de hipóteses, começando com a disjunção o mais específica (vazia) e terminando quando a hipótese é suficientemente geral para cobrir todos os exemplos de treino positivos.

O ciclo interno do FOIL realiza uma pesquisa para determinar a definição exacta de cada nova regra. Este ciclo interno procura um espaço de hipóteses (conjunções de literais), de forma a encontrar uma conjunção que forme as pré-condições para a nova regra. Dentro deste espaço de hipóteses, conduz uma pesquisa geral-para-específica, começando com as pré-condições o

mais gerais possíveis (a pré-condição vazia), adicionando depois os literais (um a um) de forma a especializar a regra até evitar todos os exemplos negativos.

Tabela 2.16 – Algoritmo FOIL

FOIL (*Predicado alvo*, *Predicados*, *Exemplos*)

- $Pos \leftarrow$ Exemplos em que o *predicado alvo* é verdadeiro
- $Neg \leftarrow$ Exemplos em que o *predicado alvo* é falso
- $Regras_aprendidas \leftarrow \{\}$
- Enquanto Pos , Fazer

Aprender uma *Regra_nova*

- $Regra_nova \leftarrow$ A regra que prediz o *predicado alvo* sem pré-condições
- $Regra_nova_neg \leftarrow Neg$
- Enquanto $Regra_nova_neg$, Fazer

Adicionar um novo literal para especificar *Regra_nova*

- $Literais_candidatos \leftarrow$ gerar novos literais candidatos para *Regra_nova*, baseado nos predicados
 - $Melhor_literal \leftarrow \underset{L \in Literais_Candidatos}{\arg \max} \quad Ganho_FOIL(L, Regra_nova)$
 - Adicionar *Melhor_literal* a pré-condições da *regra_nova*
 - $Regra_nova_neg \leftarrow$ Subconjunto de *Regra_nova_neg* que satisfazem as pré-condições da *regra_nova*
 - $Regras_aprendidas \leftarrow Regras_aprendidas + Regra_nova$
 - $Pos \leftarrow Pos - \{\text{membros de } Pos \text{ cobertos pela } regra_nova\}$
 - Retornar *Regras_aprendidas*
-

As duas diferenças substanciais entre o FOIL e os algoritmos de Cobertura Sequencial e Aprender-uma-regra devem-se ao facto da necessidade de acomodar as regras de primeira ordem e são:

- Na pesquisa geral-para-específica para aprender cada nova regra, o FOIL utiliza etapas diferentes para gerar as especializações da regra. Esta diferença deve-se à necessidade de acomodar variáveis nas pré-condições da regra.
- O FOIL utiliza uma medida de desempenho, *Ganho_FOIL*, que difere da entropia, devido à necessidade de distinguir as diferentes ligações entre as variáveis das regras e ao facto do FOIL procurar só regras que cobrem exemplos positivos.

O FOIL, para aprender especializações candidatas da regra corrente, gera uma variedade de novos literais, cada um dos quais deve ser individualmente adicionado às pré-condições da regra.

Supondo que a regra corrente a ser considerada é

$$P(x_1, x_2, \dots, x_k) \leftarrow L_1 \dots L_n$$

em que $L_1 \dots L_n$ são literais que fôrnam as pré-condições da regra corrente e em que $P(x_1, x_2, \dots, x_k)$ é o literal que forma a cabeça da regra, ou pós-condições. FOIL gera especializações candidatas desta regra considerando novos literais L_{n+1} que tomam uma das seguintes formas:

- $Q(v_1, \dots, v_r)$ em que Q é qualquer nome de predicado que ocorre em *predicados* e em que v_i são novas variáveis ou variáveis já presentes na regra. Pelo menos uma das v_i no literal criado tem que existir já como variável na regra.
- $Igual(x_j, x_k)$ em que x_j e x_k são variáveis já presentes na regra.
- A negação de ambas as formas anteriores de literais.

Como ilustração, considere-se as regras aprendidas para predizer o literal alvo $Neta(x, y)$, em que os outros predicados usados para descrever os exemplos são *Pai* e *Feminino*. A pesquisa geral-para-específica no FOIL começa com a regra mais geral

$$Neta(x, y) \leftarrow$$

que assume que $Neta(x, y)$ é verdade para qualquer x e y . Para especializar esta regra inicial, o procedimento acima gera os seguintes literais como condições adicionais para as pré-condições da regra: $Igual(x, y)$, $Feminino(x)$, $Feminino(y)$, $Pai(x, y)$, $Pai(y, x)$, $Pai(x, z)$, $Pai(z, x)$, $Pai(y, z)$, $Pai(z, y)$ e as negações de cada um destes literais (por exemplo, $\neg Igual(x, y)$). A variável z é uma nova variável, enquanto x e y já existiam dentro da regra corrente.

Supondo que entre os literais acima, o programa FOIL selecciona $Pai(y, z)$ como o mais promissor, obtemos a uma regra mais específica:

$$Neta(x, y) \leftarrow Pai(y, z)$$

Na geração de literais candidatos para especializarem esta regra, o FOIL considera todos os literais mencionados na etapa anterior, mais os literais $Feminino(z)$, $Igual(z, x)$, $Igual(z, y)$, $Pai(z, w)$, $Pai(w, z)$ e as suas negações. Estes novos literais são considerados neste ponto pois a variável z foi adicionada à regra na etapa anterior. Assim, o FOIL considera uma nova variável adicional w .

Se o FOIL neste ponto seleccionar o literal $Pai(z, x)$ e na próxima iteração seleccionar o literal $Feminino(y)$, obtém a regra seguinte que cobre só exemplos positivos, terminando por isso a pesquisa de outras especializações da regra.

$$Neta(x, y) \leftarrow Pai(y, z) \wedge Pai(z, x) \wedge Feminino(y)$$

Neste ponto, o FOIL remove todos os exemplos positivos cobertos por esta regra. Se os exemplos positivos adicionais necessitam ser cobertos, então é iniciada uma nova pesquisa geral-para-específica de uma regra adicional.

Para seleccionar o literal mais promissor dos candidatos gerados em cada etapa, o FOIL considera o desempenho da regra sobre os dados de treino. Ao fazer isto, considera todas as ligações possíveis de cada variável na regra corrente. Para ilustrar este processo, considere-se de novo o exemplo em que se procura aprender um conjunto de regras para o literal alvo $Neta(x, y)$. Considere-se que os dados de treino incluem o seguinte conjunto de asserções, e é usada a convenção que $P(x, y)$ pode ser lido “o P de x é y ”:

Neta(Paulo, Sara)

Pai(Sara, António)

Pai(José, António)

Feminino(Sara)

Pai(António, Paulo)

Considere-se também que a pressuposição de que qualquer literal envolvendo o predicado *Neta*, *Pai*, ou *Feminino* e as constantes *Paulo*, *Sara*, *António* e *José* que não estão estejam listados acima podem ser assumidos como falsos (por exemplo, pode-se implicitamente assumir que $\neg Neta(José, António)$, $\neg Neta(Paulo, Paulo)$, etc.).

Para seleccionar a melhor especialização da regra corrente, o FOIL considera todas as formas possíveis das variáveis da regra poderem ligar-se a constantes nos exemplos de treino. Por exemplo, na etapa inicial

$$Neta(x,y) \leftarrow$$

as variáveis da regra x e y não são restringidas por nenhuma pré-condição e podem por isso ligar-se a qualquer uma das quatro constantes *Paulo*, *Sara*, *António* e *José*.

Dadas as quatro constantes possíveis existem 16 ligações de variáveis possíveis para esta regra inicial. A ligação $\{x/Paulo, y/Sara\}$ corresponde a uma ligação a um exemplo positivo (ligação “positiva”), pois os dados de treino incluem a asserção $Neta(Paulo, Sara)$. As outras 15 ligações permitidas pela regra não são consideradas pois não existem asserções equivalentes nos dados de treino (ligações “negativas”).

Em cada estádio, a regra é avaliada a partir destas ligações “positivas” e “negativas”. À medida que novos literais são adicionados, o conjunto das ligações vai-se alterando. Se é adicionado um literal que introduz uma nova variável, então as ligações para a regra aumentam de comprimento (por exemplo, se $Pai(y,z)$ é adicionado à regra acima, então a ligação original $\{x/Paulo, y/Sara\}$ ficará $\{x/Paulo, y/Sara, z/António\}$.

A função de avaliação usada pelo FOIL para estimar a utilidade de adicionar um novo literal é baseada no número de ligações “positivas” e “negativas” cobertos pela hipótese antes e depois de adicionar o novo literal.

Considerando uma regra R , um literal candidato L que deve ser adicionado ao corpo de R , R' a regra criada pela adição do literal L à regra R , então o valor $Ganho_FOIL(L,R)$ de adicionar L a R é definido como

$$Ganho_FOIL(L,R) \equiv t(\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0})$$

em que p_0 é o número de ligações “positivas” da regra R , n_0 é o número de ligações “negativas” de R , p_1 é o número de ligações “positivas” da regra R' , n_1 é o número de ligações “negativas” de R' e t é o número de ligações “positivas” da regra R que são ainda cobertas depois de adicionar o literal L a R .

De acordo com a teoria da informação, $-\log_2 \frac{p_0}{p_0 + n_0}$ é o número mínimo de “bits” necessários para codificar a classificação de uma qualquer ligação “positiva” entre as ligações cobertas pela regra R . Analogamente, $-\log_2 \frac{p_1}{p_1 + n_1}$ é o número de “bits” requerido se a ligação é uma das cobertas pela regra R' . Já que t é o número de ligações “positivas” cobertas por R que permanecem cobertas por R' , o $Ganho_FOIL(L,R)$ pode ser visto como a redução esperada para L no número total de “bits” necessários para codificar a classificação de todas as ligações “positivas” de R .

Os novos literais adicionados ao corpo da regra podem referir-se ao próprio predicado alvo (ao predicado da cabeça da regra). Contudo, se for colocado o predicado alvo na lista de “input” de predicados, o FOIL considera-o também quando gera novos literais. Isto permitirá formar regras recursivas que usam o mesmo predicado no corpo e na cabeça da regra.

Considere-se o seguinte conjunto de regras que possibilita uma definição recursiva da relação Antepassado.

Se $\text{Pais}(x,y)$ Então $\text{Antepassado}(x,y)$
 Se $\text{Pais}(x,z) \wedge \text{Antepassado}(z,y)$ Então $\text{Antepassado}(x,y)$

Dado um conjunto apropriado de exemplos de treino, estas duas regras podem ser aprendidas seguindo um caminho similar ao da regra antes analisada (Neta).

A aprendizagem de regras recursivas introduz alguma complexidade ao FOIL, de forma a evitar conjuntos de regras que resultam em recursividade infinita. No entanto, o FOIL tem demonstrado ser eficiente na aprendizagem de conjuntos de regras recursivas em vários casos.

O FOIL adiciona novos literais à regra até esta não cobrir exemplos negativos. O programa FOIL usa a abordagem do comprimento mínimo da descrição para interromper o crescimento das regras, verificando-se que novos literais são adicionados à regra só quando o seu comprimento de descrição é menor do que o comprimento de descrição dos dados de treino que eles explicam [Quinlan-90]. O programa FOIL utiliza o "post-pruning" de cada regra que aprende.

2.8.5. Indução como Dedução Invertida

Uma outra abordagem da PLI é baseada na simples constatação que a indução é o inverso da dedução. Geralmente, a Aprendizagem Automática envolve a construção de teorias que justificam os dados observados.

Considere-se D o conjunto de dados de treino, cada um da forma $(x_i, f(x_i))$, em que x_i é a i^a instância de treino e $f(x_i)$ o valor alvo. A aprendizagem consiste no problema de descobrir uma hipótese h , tal que a classificação $f(x_i)$ de cada instância de treino x_i sucede dedutivamente da hipótese h , da descrição de x_i e de qualquer outro conhecimento prévio B conhecido pelo sistema.

$$(\forall (x_i, f(x_i)) \in D) (B \wedge h \wedge x_i) \vdash f(x_i) \quad (2.18)$$

A expressão $x \vdash y$ é lida como "y sucede dedutivamente a partir de x" ou alternativamente "x vincula y". A expressão anterior descreve a restrição que deve ser satisfeita pela hipótese aprendida h que consiste em que para todas as instâncias de treino x_i , a classificação alvo $f(x_i)$ deve suceder dedutivamente a partir de B , h e x_i .

Como exemplo, considere-se o caso em que o conceito alvo a ser aprendido, é "pares de pessoas (u,v) tal que a criança de u é v ", representado pelo predicado $\text{Criança}(u,v)$. Assumindo o exemplo positivo $\text{Criança}(\text{Paulo}, \text{Glória})$, em que a instância é descrita pelos literais $\text{Masculino}(\text{Paulo})$, $\text{Feminino}(\text{Glória})$ e $\text{Pai}(\text{Glória}, \text{Paulo})$. Supondo que existe o seguinte conhecimento prévio $\text{Pais}(u,v) \leftarrow \text{Pai}(u,v)$ pode-se descrever esta situação em termos da equação (2.18):

x_i : $\text{Masculino}(\text{Paulo}), \text{Feminino}(\text{Glória}), \text{Pai}(\text{Glória}, \text{Paulo})$
 $f(x_i)$: $\text{Criança}(\text{Paulo}, \text{Glória})$
 B : $\text{Pais}(u,v) \leftarrow \text{pai}(u,v)$

Neste caso, duas de muitas outras hipóteses que satisfazem a restrição $(B \wedge h \wedge x_i) \vdash f(x_i)$ são :

$$h_1 : Criança(u,v) \leftarrow Pai(v,u)$$

$$h_2 : Criança(u,v) \leftarrow Pais(v,u)$$

O literal alvo $Criança(Paulo, Glória)$ é vinculado por $h_1 \wedge x_i$ sem necessidade do conhecimento prévio B . No caso da hipótese h_2 , o literal alvo $Criança(Paulo, Glória)$ sucede dedutivamente a partir de $B \wedge h_2 \wedge x_i$ mas não a partir de $h_2 \wedge x_i$. Este exemplo ilustra o papel do conhecimento prévio na expansão do conjunto de hipóteses aceitáveis para um dado conjunto de dados de treino.

Ilustra também como novos predicados (por exemplo, Pais) podem ser introduzidos nas hipóteses, mesmo quando o predicado não está presente na descrição original da instância x_i . Este processo de aumentar o conjunto de predicados, baseado no conhecimento prévio é frequentemente referido como indução construtiva.

A equação (2.18) lança o problema da aprendizagem a partir da estrutura da inferência dedutiva e da lógica formal. A partir dos algoritmos de dedução automática de regras proposicionais e de primeira ordem, é possível desenvolver inversos destes procedimentos de modo a automatizar o processo de generalização indutiva.

Os operadores de vínculo inverso são fundamentais na questão da exploração da indução como o inverso da dedução.

O operador de vínculo inverso $O(B,D)$ toma os dados de treino $D=\{(x_i, f(x_i))\}$ e o conhecimento prévio B como “input” e produz como “output” uma hipótese h satisfazendo a equação (2.18)

$$O(B,D) = h \text{ tal que } (\forall (x_i, f(x_i)) \in D) (B \wedge h \wedge x_i) \vdash f(x_i)$$

Uma heurística comum na PLI para escolher entre as diferentes hipóteses h que satisfazem $(\forall (x_i, f(x_i)) \in D) (B \wedge h \wedge x_i) \vdash f(x_i)$ é o comprimento de descrição mínimo.

Existem várias características atractivas para formular a tarefa de aprendizagem como uma hipótese h que respeita a relação $(\forall (x_i, f(x_i)) \in D) (B \wedge h \wedge x_i) \vdash f(x_i)$:

- Esta formulação respeita a definição comum de aprendizagem como um conceito geral que cobre um conjunto de exemplos de treino.
- Pela incorporação do conhecimento prévio B , esta formulação permite aprender métodos que usam o conhecimento prévio para conduzir a pesquisa de h , em vez de meramente procurar o espaço de hipóteses sintacticamente legais.

Paralelamente, a investigação na PLI seguindo esta formulação encontrou várias dificuldades práticas:

- A formulação $(\forall (x_i, f(x_i)) \in D) (B \wedge h \wedge x_i) \vdash f(x_i)$ não acomoda os dados de treino com ruído. O problema reside em que a expressão não permite a possibilidade de haver erros na descrição observada da instância x_i ou no seu valor alvo $f(x_i)$. Tais erros podem produzir um conjunto de restrições inconsistentes em h .
- A linguagem da lógica de primeira ordem é tão expressiva e o número de hipóteses que satisfazem $(\forall (x_i, f(x_i)) \in D) (B \wedge h \wedge x_i) \vdash f(x_i)$ é tão grande, que de uma forma geral a pesquisa pelo espaço de hipóteses é intratável de uma forma geral. Trabalhos mais recentes permitiram restringir a forma das expressões de primeira ordem, ou adicionar novo conhecimento (“second-order knowledge”), para melhorar a pesquisa do espaço de hipóteses.

- Apesar da nossa intuição de que o conhecimento prévio B deve ajudar a restringir a pesquisa de uma hipótese, na maior parte de sistemas de PLI a complexidade da pesquisa do espaço de hipóteses aumenta à medida que o conhecimento prévio B aumenta também. No entanto, existem algoritmos que usam o conhecimento prévio para diminuir em vez de aumentar a complexidade, como o caso da aprendizagem analítica e a aprendizagem baseada na combinação da aprendizagem indutiva e analítica.

2.8.6 Resolução Invertida

Um método geral para a dedução automática é a regra de resolução introduzida por [Robinson-65]. A regra de resolução é uma regra para a inferência dedutiva na lógica de primeira ordem. É possível inverter a regra de resolução para formar um operador de vínculo inverso. Este operador forma a base do programa Cigol introduzido por [Muggleton-88].

Considere-se L um literal arbitrário proposicional e P e R duas disjunções de literais arbitrárias proposicionais. A regra de resolução é

$$\begin{array}{c} P \vee L \\ \neg L \vee R \\ \hline P \vee R \end{array}$$

que pode ser lida como: dadas as duas disjunções de literais acima da linha, conclui a disjunção de literais abaixo da linha. Dadas duas asserções $P \vee L$ e $\neg L \vee R$, como é obvio que L ou $\neg L$ deve ser falso, P ou R deve ser verdade. Assim, a conclusão $P \vee R$ da regra de resolução é intuitivamente satisfeita. A forma geral do operador de resolução (forma proposicional) é descrita na tabela 2.17 [Mitchell-97].

Tabela 2.17 –Operador de Resolução (forma proposicional)

1. Dadas duas disjunções de literais iniciais C_1 e C_2 , encontrar um literal L da disjunção de literais C_1 tal que $\neg L$ ocorre na disjunção de literais C_2 .
2. Formar o resolvente C pela inclusão de todos os literais de C_1 e C_2 , excepto para L e $\neg L$. O conjunto de literais que ocorrem na conclusão C é

$$C = (C_1 - \{L\}) \cup (C_2 - \{\neg L\})$$

em que $\cup$ é a união e “-” o conjunto diferença

Considere-se a aplicação do operador de resolução ilustrado pela figura 2.22 (a). Dadas as disjunções de literais C_1 e C_2 , a primeira etapa do procedimento identifica o literal $L = \neg A$, que está presente em C_1 , e cuja negação $\neg(\neg A) = A$ está presente em C_2 . Assim a conclusão é a disjunção de literais formada pela união dos literais $C_1 - \{L\} = I$ e $C_2 - \{\neg L\} = \neg C$. O resultado da aplicação da regra de resolução à disjunção de literais $C_1 = A \vee I \vee C \vee \neg G$ e $C_2 = \neg I \vee E \vee F$ é a disjunção de literais $A \vee C \vee \neg G \vee E \vee F$.

É necessário inverter o operador de resolução para formar um operador de vínculo inverso $O(C, C_1)$ que realiza a inferência indutiva. O operador de vínculo inverso deve derivar uma disjunção de literais iniciais, C_2 , dado o resolvente C e outra disjunção de literais inicial C_1 .

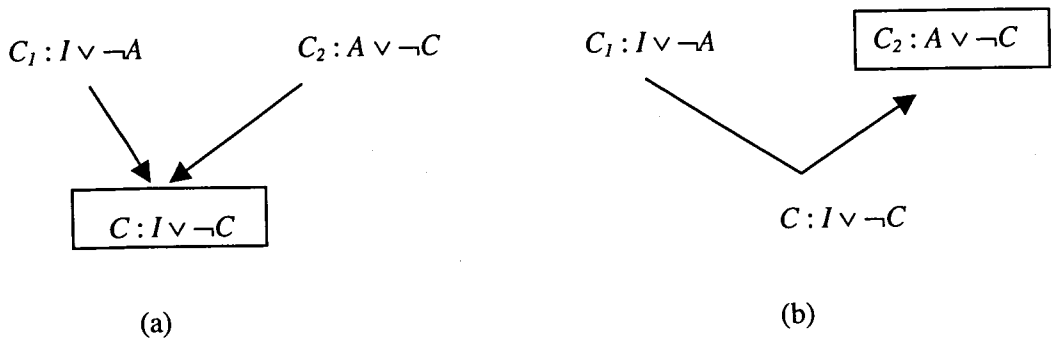


**Figura 2.22 - (a) Inferência de C pela aplicação da Regra de Resolução
(b) Inversão do Operador de Resolução para a Inferência Indutiva de C_2**

Considere-se um exemplo em que é dado o resolvente $C = I \vee \neg C$ e a disjunção de literais inicial $C_1 = I \vee \neg A$. É possível derivar a disjunção de literais C_2 tal que $C_1 \wedge C_2 \vdash C$. Por definição do operador de resolução, qualquer literal que ocorre em C mas não em C_1 deve estar presente em C_2 . No exemplo, isto indica que C_2 deve conter o literal $\neg C$. O literal que ocorre em C_1 mas não em C deve ser o literal removido pela regra de resolução, e portanto a sua negação deve ocorrer em C_2 . No exemplo C_2 deve conter o literal A . Assim $C_2 = A \vee \neg C$. Aplicando a regra de resolução a C_1 e C_2 produz o resolvente desejado C .

De notar, que existe uma segunda solução para C_2 no exemplo que consiste na disjunção $A \vee \neg C \vee I$. A diferença está na inclusão em C_2 de um literal que ocorre em C_1 . O ponto essencial é que a resolução inversa não é determinista, já que devem existir múltiplas disjunções de literais C_2 tais que C_1 e C_2 produzem o resolvente C . Uma heurística para escolher entre as alternativas consiste em preferir disjunções de literais mais pequenas (equivalentemente, assumir que C_2 não partilha literais em comum com C_1). Na tabela 2.18 está descrito o operador de resolução inversa assumindo a pressuposição de preferência de disjunções de literais mais pequenas [Mitchell-97].

Tabela 2.18 – Operador de Resolução Inversa

1. Dadas duas disjunções de literais C_1 e C , encontrar um literal L que ocorre na disjunção de literais C_1 mas não na disjunção de literais C_2 .
2. Formar a segunda disjunção de literais C_2 incluindo os seguintes literais

$$C_2 = (C - (C_1 - \{L\})) \cup \{\neg L\}$$

É possível desenvolver algoritmos para a aprendizagem de regras a partir de operadores de vínculo inverso tal como com a resolução inversa. O algoritmo de aprendizagem pode usar o operador de vínculo inverso para construir hipóteses que conjuntamente com o conhecimento prévio, vinculam os dados de treino. Uma estratégia consiste em usar um algoritmo de cobertura sequencial para iterativamente aprender um conjunto de regras desta forma. Em cada iteração, o algoritmo selecciona um exemplo de treino $(x_i, f(x_i))$ que ainda não está coberto por regras aprendidas previamente. A regra de resolução inversa é então aplicada para gerar hipóteses candidatas h_i que satisfazem $(B \wedge h_i \wedge x_i) \vdash f(x_i)$, em que B é o conhecimento prévio mais quaisquer regras aprendidas em iterações anteriores. Esta pesquisa é uma pesquisa conduzida por exemplos, pois cada hipótese candidata é construída para cobrir um exemplo particular. Se existirem múltiplas hipóteses candidatas, uma estratégia para seleccioná-las é escolher a hipótese com maior exactidão sobre os outros exemplos de treino. O programa Cigol usa a resolução inversa com o algoritmo de cobertura sequencial, interagindo com o utilizador para obter os exemplos de treino e para obter orientação na sua pesquisa pelo vasto

espaço de etapas de inferência indutiva possíveis. Contudo, o programa Cigol usa uma representação de primeira ordem em vez de uma representação proposicional.

2.8.6.1 Resolução de Primeira Ordem

A regra de resolução estende-se facilmente para expressões de primeira ordem. Como no caso proposicional, recebe duas disjunções de literais como "input" e produz uma disjunção de literais como "output". A diferença chave do caso proposicional consiste no facto que o processo é baseado na noção das substituições (unificações). Uma substituição é um mapeamento das variáveis para os termos. Por exemplo, a substituição $\theta = \{x/Jorge, y/z\}$ indica que a variável x é substituída pelo termo Jorge, e a variável y é substituída pelo termo z . Usa-se a notação $w\theta$ para o resultado da aplicação da substituição θ para uma expressão W . Por exemplo, se L é o literal $Pai(x, Rui)$ e θ é a substituição definida acima, então $L\theta = Pai(Jorge, Rui)$.

O símbolo θ é uma substituição para os dois literais L_1 e L_2 se $L_1\theta = L_2\theta$. Por exemplo se $L_1 = Pai(x, y)$, $L_2 = Pai(Rui, z)$ e $\theta = \{x/Rui, z/y\}$, então θ é uma substituição para L_1 e L_2 pois $L_1\theta = L_2\theta = Pai(Rui, y)$. Na forma proposicional de resolução, o resolvente de duas disjunções de literais C_1 e C_2 é encontrado pela identificação de um literal L que aparece em C_1 e que $\neg L$ aparece em C_2 . Na resolução de primeira ordem, realiza-se uma generalização para encontrar um literal L_1 da disjunção de literais C_1 e um literal L_2 de C_2 , de forma que alguma substituição θ possa ser encontrada para L_1 e $\neg L_2$ ($L_1\theta = \neg L_2\theta$). A regra de resolução constrói o resolvente C de acordo com a equação

$$C = (C_1 - \{L_1\})\theta \cup (C_2 - \{L_2\})\theta \quad (2.19)$$

A forma geral da regra de resolução de primeira ordem é ilustrada na tabela 2.19 [Mitchell-97].

Tabela 2.19 – Operador de Resolução (forma de primeira ordem)

1. Encontrar um literal L_1 na disjunção de literais C_1 , um literal L_2 na disjunção de literais C_2 e a substituição θ tal que $L_1\theta = \neg L_2\theta$.
2. Formar o resolvente C pela inclusão de todos os literais de $C_1\theta$ e $C_2\theta$, excepto para $L_1\theta = \neg L_2\theta$. Mais precisamente, o conjunto de literais que ocorrem na conclusão C é

$$C = (C_1 - \{L_1\})\theta \cup (C_2 - \{L_2\})\theta$$

Como exemplo, considere-se $C_1 = A(x) \leftarrow I(x)$ e $C_2 = I(Paula)$. Para aplicar a regra de resolução, expressa-se C_1 na forma de disjunção de literais $C_1 = A(x) \vee \neg I(x)$.

Inicialmente encontra-se o literal $L_1 = \neg I(x)$ de C_1 e o literal $L_2 = I(Paula)$ de C_2 . Se for escolhida a substituição $\theta = \{x/Paula\}$ então estes dois literais satisfazem $L_1\theta = \neg L_2\theta = \neg I(Paula)$. Assim, a conclusão C é a união de $(C_1 - \{L_1\})\theta = A(Paula)$ e $(C_2 - \{L_2\})\theta = \emptyset$, ou $C = A(Paula)$.

2.8.6.2 Resolução Invertida nas Expressões de Primeira Ordem

O operador de resolução inversa pode ser obtido analiticamente, pela manipulação algébrica da equação (2.19) que define a regra de resolução. A substituição θ na equação (2.19) pode ser factorizada em θ_1 e θ_2 , em que $\theta = \theta_1\theta_2$ e em que θ_1 contém todas as substituições envolvendo variáveis da disjunção de literais C_1 , e θ_2 contém todas as substituições envolvendo variáveis da disjunção de literais C_2 . Esta factorização é possível porque C_1 e C_2 começam sempre com nomes de variáveis distintos. Usando esta factorização de θ , a partir da equação (2.19) pode-se dizer que

$$C = (C_1 - \{L_1\})\theta_1 \cup (C_2 - \{L_2\})\theta_2$$

Se a resolução inversa for restringida para inferir só disjunções de literais C_2 que não contêm literais em comum com C_1 (correspondendo a uma preferência por disjunções de literais C_2 pequenas), pode ainda ser re-exprimida como

$$C - (C_1 - \{L_1\})\theta_1 = (C_2 - \{L_2\})\theta_2$$

Finalmente pode-se partir do facto de que por definição da regra de resolução, $L_2 = \neg L_1\theta_1\theta_2^{-1}$, e obter a equação de resolução inversa

$$C_2 = (C - (C_1 - \{L_1\})\theta_1)\theta_2^{-1} \cup \{\neg L_1\theta_1\theta_2^{-1}\}$$

Com esta equação obtém-se a regra de resolução inversa da lógica de primeira ordem. Como no caso proposicional, este operador de vínculo inverso não é determinista, pois é possível encontrar múltiplas escolhas para C_1 e para as substituições θ_1 e θ_2 , verificando-se que cada uma destas escolhas leva a uma solução diferente C_2 .

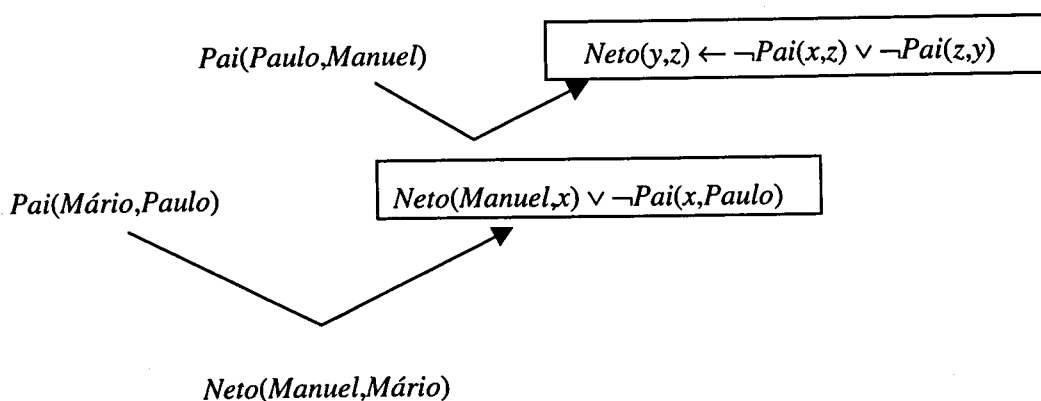


Figura 2.23 – Aplicação da Regra de Resolução Inversa

A figura 2.23 ilustra uma aplicação desta regra para um exemplo simples. Na figura, pretende-se aprender regras para o predicado alvo $Neto(y,x)$, a partir do dado de treino $D=Neto(Manuel,Mário)$ e o conhecimento prévio $B=\{Pai(Mário,Paulo), Pai(Paulo,Manuel)\}$.

Na etapa mais abaixo da figura 2.23 atribui-se à conclusão C o exemplo de treino $Neto(Manuel,Mário)$ e selecciona-se $C_1=Pai(Mário,Paulo)$ do conhecimento prévio. Para aplicar o operador de resolução inversa existe só uma escolha para o literal L_1 ,

$Pai(Mário, Paulo)$. Supondo que se escolhe as substituições inversas $\theta_1^{-1} = \{ \}$ e $\theta_2^{-1} = \{ Mário/x \}$, C_2 é a união de $(C - (C_1 - \{ L_1 \}) \theta_1) \theta_2^{-1} = (C \theta_1) \theta_2^{-1} = Neto(Manuel, x)$ e $\{ \neg L_1 \theta_1 \theta_2^{-1} \} = \neg Pai(x, Paulo)$. Assim, o resultado é $Neto(Manuel, x) \vee \neg Pai(x, Paulo)$ ou equivalentemente $(Neto(Manuel, x) \leftarrow Pai(x, Paulo))$. Esta regra geral, conjuntamente com C_1 vincula o exemplo de treino $Neto(Manuel, Mário)$. De uma forma análoga, esta disjunção de literais pode ser usada como conclusão C para uma segunda etapa de resolução inversa como ilustrado na figura 2.23. Em cada etapa, existem vários resultados possíveis, dependendo das escolhas para as substituições.

Na figura, cada caixa é o resultado de uma etapa de inferência. Em ambas etapas de inferência, θ_1 é a substituição vazia $\{ \}$, e θ_2^{-1} é a substituição abaixo de C_2 . No exemplo um conjunto particular de escolhas produziu a disjunção de literais $Neto(y, x) \leftarrow Pai(x, z) \wedge Pai(z, y)$.

A regra de resolução inversa tem a vantagem de gerar só hipóteses que satisfazem a restrição $(B \wedge h \wedge x_i) \vdash f(x_i)$. Em contraste, a pesquisa de geração-e-teste do FOIL gera muitas hipóteses em cada etapa de pesquisa, incluindo algumas que não satisfazem esta restrição. O algoritmo FOIL depois considera os dados D para escolher entre as hipóteses. Dada esta diferença é de esperar que a pesquisa baseada na resolução inversa seja mais focalizada e eficiente. Contudo, não é necessariamente verdade, pois o operador de resolução inversa pode considerar só uma pequena fracção dos dados disponíveis quando em cada etapa gera uma hipótese, enquanto o algoritmo FOIL considera todos os dados disponíveis para seleccionar as hipóteses geradas.

2.8.6.3 Progol

A resolução inversa é um método que pode gerar hipóteses candidatas de uma forma explosiva. Uma abordagem alternativa é usar o operador de vínculo inverso para gerar a hipótese mais específica que conjuntamente com o conhecimento prévio vincula os dados observados. Esta hipótese mais específica pode ser usada para limitar a pesquisa geral-para-específica pelo espaço de hipóteses, similar à usada pelo FOIL, mas com a restrição adicional de que só hipóteses mais gerais do que a mais específica são consideradas.

Esta abordagem é utilizada pelo sistema Progol, cujo algoritmo pode ser sumariado como:

1. O utilizador especifica uma linguagem restrita de exemplos de primeira ordem para ser usada como espaço de hipóteses H . Ao utilizador é dada a possibilidade de especificar os predicados a serem considerados e os tipos e formatos dos argumentos de cada predicado.
2. O Progol usa o algoritmo de cobertura sequencial para aprender um conjunto de expressões de H que cobrem os dados. Para cada exemplo $(x_i, f(x_i))$ que ainda não está coberto por estas expressões aprendidas, primeiro procura a hipótese mais específica h_i dentro de H , tal que $(B \wedge h \wedge x_i) \vdash f(x_i)$. Para isso, obtém a hipótese mais específica entre aquelas que vinculam $f(x_i)$ dentro das k aplicações da regra de resolução (em que k é um parâmetro especificado pelo utilizador).
3. O Progol realiza depois uma pesquisa geral-para-específica do espaço de hipóteses limitada pela hipótese o mais geral possível e pela hipótese h_i calculada em 2. Dentro deste conjunto de hipóteses, procura a hipótese com comprimento de descrição mínimo (medido pelo número de literais).

Capítulo 3

Aprendizagem Analítica

3.1 Introdução

A tarefa de Aprendizagem Indutiva consiste em determinar uma hipótese idêntica a um determinado conceito alvo para um conjunto de exemplos de treino,

Os métodos de aprendizagem indutiva (tais como as redes neuronais, a aprendizagem de árvores de decisão, PLI e algoritmos genéticos) generalizam a partir de exemplos de treino observados e pela identificação de características que distinguem os exemplos positivos dos negativos. Estes métodos necessitam de um grande número de exemplos de treino para adquirirem um nível aceitável de exactidão na generalização.

A limitação prática da aprendizagem indutiva reside no facto que tem um desempenho pobre quando não estão disponíveis dados suficientes.

A aprendizagem analítica usa o conhecimento prévio e o raciocínio dedutivo para aumentar a informação possibilitada pelos exemplos de treino, não estando sujeita às mesmas limitações no número de exemplos disponíveis. O conhecimento prévio é aceite explicitamente como “input” adicional aos dados de treino.

A aprendizagem baseada em explicações (ABE) é um exemplo desta abordagem. Na ABE, o conhecimento prévio é usado para analisar, ou explicar, como cada exemplo de treino observado satisfaz o conceito alvo. Mais precisamente, usa o conhecimento prévio para analisar ou explicar cada exemplo de treino de forma a inferir que características do exemplo são relevantes para a função alvo. Desta forma, os exemplos podem ser generalizados a partir de um raciocínio lógico em vez de estatístico.

Os sistemas de PLI tal como o Cigol usam também o conhecimento prévio para conduzir a aprendizagem, contudo, usam o conhecimento prévio para inferir características que aumentam as descrições de “input” das instâncias aumentando desse modo a complexidade do espaço de hipóteses a ser pesquisado.

A ABE usa o conhecimento prévio para reduzir a complexidade do espaço de hipóteses a ser pesquisado, reduzindo a complexidade da amostra e melhorando a exactidão na generalização.

A ABE foi aplicada com sucesso para aprender regras de controlo da pesquisa para uma variedade de tarefas de planificação e calendarização.

Na aprendizagem indutiva o “output” deve ser uma hipótese h de H que é consistente com os exemplos de treino, enquanto na aprendizagem analítica o “output” deve ser uma hipótese h de H que é consistente com os exemplos de treino e com a teoria do domínio B (conhecimento prévio). O conhecimento prévio pode ser usado para explicar os exemplos de treino. Uma hipótese h é consistente com a teoria do domínio B quando B não vincula a negação de h . Esta restrição adicional de que a hipótese deve ser consistente com B reduz a ambiguidade encontrada quando os dados só por si não possibilitam a escolha entre todas as hipóteses em H , e aumenta a exactidão das hipóteses.

3.2 Aprendizagem com Teorias do Domínio Perfeitas

Uma teoria do domínio é considerada perfeita se for correcta e completa. Uma teoria do domínio perfeita é considerada correcta se cada uma das asserções é uma instrução verdadeira e é considerada completa relativamente a um determinado conceito alvo e um espaço de instâncias, se a teoria do domínio cobre todos os exemplos positivos do espaço de instâncias. Vista de outra forma, é completa se todas as instâncias que satisfazem o conceito alvo possam ser provadas pela teoria do domínio. A teoria do domínio não prova que os exemplos negativos não satisfazem o conceito alvo, contudo, se for seguida a convenção usual do Prolog, segundo a qual asserções improváveis são assumidas como falsas, então a definição inclui a cobertura completa dos exemplos positivos e negativos pela teoria do domínio.

**Tabela 3.1 – Um Problema de Aprendizagem Analítica :
*Pode_Empilhar(x,y)***

Dado:

- Espaço de Instâncias X : cada instância descreve características de objectos representados pelos predicados *Tipo*, *Cor*, *Volume*, *Dono*, *Material*, *Densidade* e *Em*.
- Espaço de Hipóteses H : cada hipótese é um conjunto de regras ("Horn clauses"). A cabeça de cada regra é um literal contendo o predicado alvo *Pode_Empilhar*. O corpo de cada regra é uma conjunção de literais baseados nos mesmos predicados usados para descrever as instâncias assim como os predicados *Menor_que*, *Igual*, *Maior_que*, e as funções *mais*, *menos* e *vezes*. Como exemplo, a regra seguinte está no espaço de hipóteses

$$Pode_Empilhar(x,y) \leftarrow Volume(x,vx) \wedge Volume(y,vy) \wedge Menor_que(vx,vy)$$

- Conceito alvo : *Pode_Empilhar(x,y)*
- Exemplos de treino : um exemplo positivo típico, *Pode_Empilhar(obj1,obj2)* é:

• <i>Em(obj1,obj2)</i>	<i>Dono(obj1,Liliana)</i>
• <i>Tipo(obj1,Caixa)</i>	<i>Dono(obj2,Carla)</i>
• <i>Tipo(obj2,Mesa)</i>	<i>Densidade(obj1,1,0.3)</i>
• <i>Cor(obj1,Vermelho)</i>	<i>Material(obj1,Cartão)</i>
• <i>Cor(obj2,Azul)</i>	<i>Material(obj2,Madeira)</i>
• <i>Volume(obj1,2)</i>	
- Teoria do domínio B :
 - *Pode_Empilhar(x,y) ← ¬Frágil(y)*
 - *Pode_Empilhar(x,y) ← Transporta(x,y)*
 - *Transporta(x,y) ← Peso(x,wx) ∧ Peso(y,wy) ∧ Menor_que(wx,wy)*
 - *Peso(x,w) ← Volume(x,v) ∧ Densidade(x,d) ∧ Igual(w,vezes(v,d))*
 - *Peso(x,5) ← Tipo(x,Mesa)*
 - *Frágil(x) ← Material(x,Vidro)*

Determina :

- Uma hipótese de H consistente com os exemplos de treino e com a teoria do domínio

Mesmo em casos da existência de uma teoria do domínio perfeita é comum formular uma descrição útil do conceito alvo com a generalização a partir dos exemplos de treino. Por exemplo, no problema do jogo do xadrez, as movimentações legais das peças formam uma teoria do domínio perfeita mas a estratégia óptima de cada jogada pode ser inferida.

Como ilustração de um exemplo de aprendizagem analítica, considere-se o problema *Pode_Empilhar* ilustrado na tabela 3.1 [Mitchell-97].

Considere-se um espaço de instâncias X , em que cada objecto físico nas instâncias é descrito pelos predicados *cor*, *volume*, *dono*, *material*, *tipo* e *densidade*, e a relação entre dois objectos é descrita pelo predicado *Em*. Dado este espaço de instâncias a tarefa consiste em aprender o conceito alvo “pares de objectos físicos, tal que um objecto possa ser empilhado em cima do outro” (predicado *Pode_empilhar(x,y)*). Por exemplo, para um robot que tem a tarefa de armazenar vários objectos físicos dentro de um espaço de trabalho limitado deve ser útil aprender este conceito alvo.

A teoria do domínio explica porque é que certos pares de objectos podem ser empilhados. A teoria do domínio é descrita por um conjunto de regras. O sistema incorpora qualquer hipótese aprendida em teorias de domínio subsequentes. A teoria do domínio utiliza predicados adicionais tais como *Transporta* e *Frágil*, que não estão presentes nas descrições dos exemplos de treino, mas que podem ser inferidos dos atributos das instâncias usando outras regras da teoria do domínio.

3.2.1 Algoritmo Prolog-EBG

O algoritmo Prolog-EBG [Kedar-Cabelli-87] é representativo de muitos algoritmos de aprendizagem baseados em explicações, e é um algoritmo de cobertura sequencial que considera os dados de treino de forma incremental e que assume que a teoria do domínio é correcta e completa.

Dada uma teoria do domínio correcta e completa, o algoritmo Prolog-EBG garante obter uma hipótese (conjunto de regras) correcta que cobre os exemplos de treino positivos observados. Para qualquer conjunto de exemplos de treino, a hipótese gerada constitui um conjunto de condições lógicas para o conceito alvo, de acordo com a teoria do domínio. O algoritmo Prolog-EBG é um refinamento do algoritmo EBG introduzido por [Mitchell-86]. Se a teoria do domínio é incorrecta ou incompleta, o conceito aprendido poderá também ser incorrecto.

O algoritmo Prolog-EBG encontra-se na tabela 3.2. Para cada novo exemplo de treino positivo que ainda não está coberto por uma regra, o algoritmo Prolog-EBG cria uma nova regra seguindo os seguintes passos:

- Explica o novo exemplo de treino positivo
- Analisa esta explicação para determinar uma generalização apropriada
- Refina a hipótese corrente adicionando uma nova regra para cobrir este exemplo de treino positivo, assim como instâncias similares.

Tabela 3.2 – Algoritmo de Aprendizagem baseado em Explicações Prolog-EBG

Prolog-EBG (*Conceito_Alvo*, *Exemplos_Treino*, *Teoria_Domínio*)

- $Regras_Aprendidas \leftarrow \{ \}$
 - $Pos \leftarrow$ Exemplos positivos dos *Exemplos_Treino*
 - Para cada *Exemplo_Positivo* em *Pos* que não está coberto por *Regras_Aprendidas*, Fazer
 1. Explicar :
 - $Explicação \leftarrow$ demonstração em termos da *Teoria_Domínio* que o *Exemplo_Positivo* satisfaz o *Conceito_Alvo*
 2. Análise :
 - $Condições_Suficientes \leftarrow$ o conjunto mais geral de características do *Exemplo_Positivo* suficientes para satisfazer o *Conceito_Alvo* de acordo com a explicação
 3. Refinamento
 - $Regras_Aprendidas \leftarrow Regras_Aprendidas + Nova_Regra$, em que *Nova_Regra* é da forma

$$Conceito_Alvo \leftarrow Condições_Suficientes$$
 - Retornar *Regras_Aprendidas*
-

3.2.1.1 Explicação do Exemplo de Treino

A primeira etapa no processamento de cada exemplo de treino, é construir uma explicação em termos da teoria do domínio, mostrando como um exemplo positivo satisfaz o conceito alvo. A explicação indica quais os atributos relevantes para o conceito alvo.

Quando a teoria do domínio é correcta e completa esta explicação constitui a prova que esse exemplo satisfaz o conceito alvo. Quando se lida com conhecimento prévio imperfeito, a notação da explicação deve ser estendida para permitir argumentos aproximados chegando-se a explicações plausíveis em vez de provas exactas.

3.2.1.2 Análise da Explicação

Apesar da explicação ter sido criada para cobrir o exemplo de treino observado, a mesma explicação aplicar-se-á a qualquer instância que cumpra as pré-condições da regra. Esta regra constitui uma generalização significativa do exemplo de treino, pois omite muitas propriedades do exemplo que são irrelevantes para o conceito alvo.

Contudo, uma regra ainda mais geral pode ser obtida por uma análise mais cuidada da explicação. O Prolog-EBG obtém a regra mais geral que pode ser justificada pela explicação, usando um procedimento denominado regressão [Waldinger-77]. A regra mais geral de uma conclusão *C* relativamente à prova *P* é o conjunto mais geral de asserções iniciais *A*, tal que *A* vincula *C* de acordo com *P*.

Por exemplo, a regra mais geral associada ao conceito alvo *Pode_Empilhar(x,y)* da tabela 3.1 é dada pelo corpo da regra seguinte:

$$\begin{aligned} \text{Pode_Empliar}(x,y) \leftarrow & \text{Volume}(x,vx) \wedge \text{Densidade}(x,dx) \wedge \text{Igual}(wx,\text{vezes}(vx,dx)) \\ & \wedge \text{Menor_que}(wx,5) \wedge \text{Tipo}(y,\text{Mesa}) \end{aligned}$$

Esta regra não requer os valores específicos para o volume e densidade que eram requeridas pela regra inicial obtida pela explicação do exemplo de treino:

$$\text{Pode_Empliar}(x,y) \leftarrow \text{Volume}(x,2) \wedge \text{Densidade}(x,0.3) \wedge \text{Tipo}(y,\text{Mesa})$$

Em vez disso, determina uma restrição mais geral nos valores destes atributos.

3.2.1.3 Refinamento da Hipótese Corrente

A hipótese corrente em cada estágio consiste no conjunto de regras aprendidas até aí. Em cada estágio, o algoritmo de cobertura sequencial, considera um novo exemplo positivo que ainda não está coberto pelas regras correntes, explica este novo exemplo, e formula uma nova regra de acordo com o procedimento descrito anteriormente.

No algoritmo só os exemplos positivos são cobertos e as regras aprendidas predizem só os exemplos positivos. Uma nova instância é classificada como negativa se as regras correntes falham a predição de que é positiva.

A aprendizagem baseada em explicações (ABE) usa a teoria do domínio para generalizar racionalmente a partir de exemplos, distinguindo os atributos relevantes dos irrelevantes, permitindo assim evitar as limitações inerentes da aprendizagem puramente indutiva relativamente à complexidade da amostra. O algoritmo Prolog-EBG pode ser visto como um método para reformular a teoria do domínio de forma a torná-la mais operacional. De facto, a teoria do domínio original é reformulada pela criação de novas regras que são entretanto deduzidas e que permitem classificar os exemplos de treino numa única etapa de inferência..

Os algoritmos de ABE re-exprimem o conhecimento já existente na teoria do domínio, revelando-se uma forma importante de aprendizagem, pois permitem mais eficientemente a classificação de uma instância. Enquanto que com a teoria do domínio inicial são requeridas muitas etapas de inferência e uma pesquisa considerável para classificar uma instância arbitrária, com o uso das regras aprendidas, as instâncias observadas são classificadas numa única etapa de inferência.

Uma capacidade interessante do Prolog-EBG reside na capacidade de formular novas características que não estão explícitas na descrição dos exemplos de treino, mas que são importantes para descrever a regra geral subjacente aos exemplos de treino. No exemplo já analisado verificou-se a existência de uma restrição essencial no volume e na densidade de x que restringe que o seu produto seja menor que 5.

Esta característica aprendida é similar às características representadas pelas unidades “escondidas” das redes neuronais. Tal como no algoritmo “Backpropagation”, o algoritmo Prolog-EBG deriva analiticamente a característica $\text{volume} \times \text{densidade} > 5$ a partir de uma instanciação da teoria de domínio durante a explicação de um exemplo de treino.

3.4 Aprendizagem Dedutiva

O Prolog-EBG na sua forma pura é um processo de aprendizagem dedutivo. Ao obter a regra mais geral da explicação produz uma hipótese h que é deduzida a partir da teoria do domínio B , e que cobre os dados de treino D . O Prolog-EBG obtém uma hipótese h que satisfaz as seguintes duas restrições:

$$(\forall (x_i, f(x_i)) \in D) (h \wedge x_i) \vdash f(x_i) \quad (3.1)$$

$$D \wedge B \vdash h \quad (3.2)$$

em que os dados de treino D consistem num conjunto de exemplos de treino, x_i é a i^a instância de treino e $f(x_i)$ é o valor alvo (f é a função alvo). A primeira destas restrições é a formalização da restrição da Aprendizagem Automática, de que a hipótese h prediz correctamente o valor alvo $f(x_i)$ para cada instância x_i nos dados de treino. Geralmente existem muitas hipóteses alternativas que satisfazem esta restrição. A segunda restrição descreve o impacto da teoria do domínio no Prolog-EBG. A hipótese obtida é restringida pela teoria do domínio e pelos dados de treino. Esta segunda restrição reduz a ambiguidade quando é necessário escolher uma hipótese, pois o impacto da teoria do domínio consiste em reduzir o tamanho efectivo do espaço de hipóteses e assim reduzir a complexidade da amostra de aprendizagem.

O Prolog-EBG assume que a teoria do domínio B vincula as classificações das instâncias nos dados de treino:

$$(\forall (x_i, f(x_i)) \in D) (B \wedge x_i) \vdash f(x_i) \quad (3.3)$$

Esta restrição na teoria do domínio B assegura que uma explicação pode ser construída para cada exemplo positivo.

A PLI usa também conhecimento prévio (B'). É utilizada a notação B' em vez de B pois o conhecimento prévio não satisfaz a restrição da equação (3.3). A PLI é um sistema de aprendizagem indutiva enquanto o Prolog-EBG é dedutivo.

A PLI usa o conhecimento prévio B' para alargar o conjunto de hipóteses a serem consideradas, enquanto o Prolog-EBG usa a teoria do domínio B para reduzir o conjunto de hipóteses aceitáveis. Os sistemas de PLI obtêm uma hipótese h que satisfaz a seguinte restrição:

$$(\forall (x_i, f(x_i)) \in D) (B' \wedge h \wedge x_i) \vdash f(x_i)$$

Esta restrição da PLI em h é uma forma enfraquecida da restrição dada pela equação (3.1), pois só requer que $(B' \wedge h \wedge x_i) \vdash f(x_i)$, enquanto que a restrição do Prolog-EBG requer a restrição mais estrita $(h \wedge x_i) \vdash f(x_i)$. A PLI não impõe uma restrição correspondente à restrição do Prolog-EBG da equação (3.2).

Em algumas situações, o algoritmo Prolog-EBG pode deduzir a hipótese h a partir somente da teoria do domínio B independentemente dos dados de treino D . Isto é possível pela enumeração de todas as árvores de prova que concluem o conceito alvo a partir das asserções da teoria do domínio B . Para cada árvore de prova é construída uma regra, gerando-se um conjunto muito grande de hipóteses. O papel dos exemplos de treino observados é muito importante pois focalizam a geração de regras que cobrem a distribuição de instâncias, prescindindo de regras menos relevantes.

Por outro lado, a hipótese aprendida pelo Prolog-EBG não vai além do conhecimento já implícito na teoria do domínio, isto é, não classifica uma instância que não pudesse ser classificada pela teoria do domínio original. Se $B \vdash h$, então qualquer classificação vinculada por h é também vinculada por B .

No entanto, existem outras situações em que a partir somente de B não se consegue deduzir h (só com $B \wedge D \vdash h$).

Para ilustração, considere-se um exemplo de aprendizagem do conceito alvo “Livro a Comprar”. Imagine-se que cada livro é descrito somente pelo atributo Interpretabilidade, e a teoria do domínio inclui a asserção “se é aconselhável comprar um livro quando a Interpretabilidade é x (quantificação da interpretabilidade), então é também aconselhável comprar um livro quando a Interpretabilidade é melhor (maior) que x ” que pode ser formalizado como

$$(\forall x) \text{ Se } ((\text{Comprar} = \text{Sim}) \leftarrow (\text{Interpretabilidade} = x))$$

$$\text{Então } ((\text{Comprar} = \text{Sim}) \leftarrow (\text{Interpretabilidade} \geq x))$$

A teoria do domínio não vincula nenhuma conclusão relativamente ao facto de as instâncias serem positivas ou negativas. Contudo, uma vez que se observe um exemplo positivo para $\text{Interpretabilidade} = 0.5$, a teoria do domínio conjuntamente com este exemplo positivo vincula a seguinte hipótese h :

$$(\text{Comprar} = \text{Sim}) \leftarrow (\text{Interpretabilidade} \geq 0.5)$$

3.5 Limitação Indutiva na Aprendizagem Baseada em Explicações

Como já foi referido, a limitação indutiva de um algoritmo é um conjunto de pressuposições que conjuntamente com os exemplos de treino, vincula dedutivamente predições subsequentes. A limitação indutiva caracteriza como se generaliza para além dos exemplos de treino observados.

No Prolog-EBG a hipótese obtida h é deduzida a partir de $D \wedge B$. Assim, a teoria do domínio B é um conjunto de asserções que conjuntamente com os exemplos de treino, vinculam a hipótese h . Dado que as predições são determinadas pela hipótese h , percebe-se que a limitação indutiva do Prolog-EBG é a teoria do domínio B . No entanto existem muitos conjuntos de regras alternativos vinculados pela teoria do domínio. Os restantes componentes da limitação indutiva são as bases pelas quais o Prolog-EBG escolhe entre estes conjuntos de regras alternativos. Assim, pode-se caracterizar a limitação indutiva do Prolog-EBG como a teoria do domínio e a preferência pelos conjuntos de regras gerados mais pequenos. A limitação indutiva do Prolog-EBG é essencialmente determinada pela teoria do domínio, que contrasta com a maior parte dos outros algoritmos de aprendizagem (redes neuronais, árvores de decisão, entre outros) em que a limitação indutiva é uma propriedade fixa do algoritmo de aprendizagem tipicamente determinada pela sintaxe da representação da hipótese.

3.6 Aprendizagem Baseada em Explicações aplicada ao Controlo da Pesquisa

A aplicabilidade do algoritmo Prolog-EBG é restringida pelo facto da teoria do domínio ter que ser correcta e completa. Uma classe importante de problemas de aprendizagem em que esta restrição é facilmente satisfeita consiste em aprender a aumentar a velocidade de programas complexos de pesquisa. De facto, a ABE é aplicada em larga escala na aprendizagem do controlo da pesquisa. Por exemplo, alguns jogos tais como o xadrez envolvem a pesquisa num vasto espaço de posições do tabuleiro e movimentações de peças possíveis para encontrar a melhor movimentação da peça. Muitos problemas de calendarização e optimização são formulados como grandes problemas de pesquisa, em que a tarefa consiste em encontrar uma forma de atingir o objectivo. Em tais problemas as definições dos operadores legais de pesquisa, conjuntamente com a definição do objectivo da pesquisa, possibilitam uma teoria do domínio correcta e completa para controlar a pesquisa do conhecimento.

Considere-se um problema de pesquisa em que S é um conjunto de estados de pesquisa possíveis, O é o conjunto de operadores de pesquisa legais que transformam um estado de pesquisa noutro e G um predicado definido sobre S que indica quais os estados objectivo. O problema consiste em encontrar uma sequência de operadores que transformam um estado inicial arbitrário e_i num estado final e_f que satisfaça o predicado objectivo G . Uma forma de formular este problema de aprendizagem consiste em ter um sistema de aprendizagem com um conceito alvo separado para cada um dos operadores em O . Cada um dos operadores o em O deve tentar aprender o conceito alvo "o conjunto de estados para os quais o operador leva a um estado objectivo".

A escolha de quais os conceitos alvo a aprender depende da estrutura interna do sistema de resolução do problema. Por exemplo, se o sistema de resolução do problema é um sistema de planeamento que trabalha para estabelecer e resolver sub-objectivos então é necessário aprender conceitos alvo tais como "o conjunto de estados de planeamento em que os sub-objectivos do estado A deverão ser resolvidos antes dos sub-objectivos do estado B ".

Um sistema que emprega a aprendizagem baseada em explicações para melhorar a sua pesquisa é o Prodigy [Carbonell-90]. O Prodigy é um sistema de planeamento independente do domínio que aceita a definição do domínio do problema em termos do espaço de estados S e operadores O . Depois resolve os problemas da forma "encontra uma sequência de operadores que leva de um estado inicial e_i a um estado que satisfaz o predicado alvo G ". O Prodigy usa uma planificação que decompõe os problemas em sub-objectivos, resolve-os, e depois combina as suas soluções numa solução para todo o problema. Assim, durante a sua pesquisa por soluções para o problema, o Prodigy repetidamente enfrenta questões como "que sub-objectivo deverá ser resolvido a seguir?" e "que operador deve ser considerado para resolver este sub-objectivo?". [Minton-88] descreveu a integração da ABE no Prodigy definindo um conjunto de conceitos alvo apropriados para este tipo de decisões de controlo. Um exemplo de uma regra aprendida pelo Prodigy para o conceito alvo "conjunto de estados em que o sub-objectivo A deve ser resolvido antes do sub-objectivo B " num problema de empilhamento de peças é

Se um sub-objectivo a ser resolvido é $Em(x,y)$ e

um sub-objectivo b ser resolvido é $Em(y,z)$

Então resolver o sub-objectivo $Em(y,z)$ antes de $Em(x,y)$

O Prodigy aprende da seguinte forma: primeiro encontra um conflito, depois explica a razão para esse conflito e cria em seguida uma regra como a anterior para o resolver. O Prodigy usa conhecimento independente do domínio relativo a possíveis conflitos de sub-objectivos, conjuntamente com o conhecimento específico dos operadores, de forma a aprender regras de planeamento específicas do domínio, tal como a acima ilustrada.

O uso da aprendizagem baseada em explicações para adquirir conhecimento de controlo para o Prodigy foi demonstrado numa variedade de domínios de problemas incluindo, passando pelo simples problema simples de empilhagem, assim como nas calendarizações mais complexas e problemas de planeamento. [Minton-88] realizou experiências em três domínios de problemas em que as regras de controlo aprendidas melhoraram a eficiência da resolução de problemas na razão de 2 para 4.

Com o Prodigy demonstrou-se que os métodos de ABE podem ser sucessivamente aplicados para adquirirem conhecimento de controlo de pesquisa numa variedade de domínios de problemas. Contudo, muitas ou a maior parte das heurísticas de pesquisa usam funções de avaliação numérica em vez de regras adquiridas pela ABE. Isto deve-se à existência de problemas práticos significativos na aplicação da ABE para aprendizagem de controlo de pesquisa. Em muitos casos o número de regras de controlo que devem ser aprendidas é muito grande, na ordem de vários milhares.

À medida que o sistema aprende mais regras de controlo para melhorar a sua pesquisa, o custo de fazer o “match” entre o conjunto de regras aprendidas e o estado corrente é cada vez maior. Foram desenvolvidos alguns algoritmos para fazer o “match” de forma a aliviar este problema. [Minton-88] adoptou estratégias para estimar empiricamente o custo e o benefício computacional de cada regra, aprendendo só regras em que os benefícios estimados superam os custos estimados, e eliminando regras que têm utilidade negativa. Minton usando este tipo de análise da utilidade para determinar o que deve ser aprendido e o que deve ser eliminado, demonstrou como aumentar significativamente a eficiência da ABE no Prodigy. Por exemplo, numa série de problemas de empilhagem de objectos realizadas por um robot, o Prodigy encontrou 328 oportunidades de aprender uma nova regra, mas escolheu para explorar só 69 delas, reduzindo as regras aprendidas para 19, uma vez que as regras com baixa utilidade foram eliminadas.

Outro problema prático com a aplicação do ABE à aprendizagem de controlo de pesquisa é que em muitos casos é mesmo intratável construir as explicações para o conceito alvo desejado. Por exemplo, considere-se o caso de no jogo do xadrez pretender-se aprender um conceito alvo, tal como, “estados em que o operador A leva à solução óptima”. Infelizmente, para provar ou explicar porque o operador A leva à solução óptima é necessário explicar que todos os operadores alternativos levam a um resultado pior. Isto tipicamente requer esforço exponencial na profundidade da pesquisa.

Capítulo 4

Combinação da Aprendizagem Indutiva e Analítica

4.1 Introdução

Os métodos de aprendizagem puramente indutiva formulam hipóteses pela obtenção de regularidades empíricas sobre os exemplos de treino. Os métodos de aprendizagem puramente analítica usam o conhecimento prévio para derivarem dedutivamente hipóteses. Estes dois paradigmas de aprendizagem oferecem vantagens e desvantagens complementares. A sua combinação oferece a possibilidade de se obterem métodos de aprendizagem mais potentes. Os métodos de aprendizagem puramente analíticos oferecem a vantagem de generalizarem com mais exactidão a partir de menos dados, usando o conhecimento prévio para conduzir a aprendizagem. Contudo, podem ser enganosos quando o conhecimento prévio é insuficiente ou incorrecto. Os métodos puramente indutivos oferecem a vantagem de não requererem conhecimento prévio explícito e aprenderem regularidades baseados somente nos dados de treino. Contudo, podem falhar quando não existem dados de treino suficientes, e podem ser induzidos em erro pela limitação indutiva que devem adoptar para generalizarem para além dos dados observados.

A diferença entre os métodos de aprendizagem indutiva e analítica pode ser vista pela natureza das justificações que podem ser dadas para as hipóteses aprendidas. Enquanto que as hipóteses obtidas com métodos de aprendizagem puramente analíticos têm uma justificação lógica, isto é, as hipóteses são deduzidas a partir da teoria do domínio e dos exemplos de treino, as hipóteses obtidas com métodos de aprendizagem puramente indutivos têm uma justificação estatística, isto é, as hipóteses são obtidas a partir de argumentos estatísticos e com dados de treino suficientes, de forma a serem representativos da distribuição de exemplos subjacente.

As justificações lógicas dos métodos analíticos são tão credíveis quanto as pressuposições, ou o conhecimento prévio em que são construídas e são suspeitas se o conhecimento prévio é incorrecto ou não está disponível. As justificações estatísticas dos métodos indutivos são tão credíveis quanto os dados e as pressuposições estatísticas em que assentam, e são suspeitas quando as pressuposições à cerca da distribuição subjacente são incertas ou quando os dados são escassos.

O espectro dos problemas de aprendizagem varia com a disponibilidade do conhecimento prévio e dos dados de treino. Enquanto num extremo está disponível um grande volume de dados de treino, mas o conhecimento prévio não, no outro extremo está disponível um conhecimento prévio excelente, mas poucos dados de treino. A maior parte dos problemas de aprendizagem permanecem entre estes dois extremos do espectro.

Os algoritmos que combinam a aprendizagem indutiva e a aprendizagem analítica consideram o conhecimento prévio como um “input” explícito, da mesma forma que os dados de treino o são. Por outro lado, pretendem-se algoritmos que sejam independentes do domínio e que empreguem explicitamente conhecimento de “input” dependente do domínio.

Algumas características que os métodos que combinam a aprendizagem indutiva e analítica deveriam incluir são:

- Sem teoria do domínio, deverão aprender pelo menos tão eficientemente como os métodos puramente indutivos.
- Dada uma teoria do domínio perfeita, deverão aprender pelo menos tão eficientemente como os métodos puramente analíticos.
- Dada uma teoria do domínio imperfeita e dados de treino imperfeitos deverão combiná-los para obter métodos puramente indutivos ou puramente analíticos.
- Deverão acomodar um qualquer nível de erro nos dados de treino.
- Deverão acomodar um qualquer nível de erro na teoria do domínio.

Esta lista de propriedades desejáveis é um pouco ambiciosa, pois a combinação destes dois métodos de aprendizagem é ainda uma área de investigação activa, não existindo ainda algoritmos que satisfaçam todas estas restrições.

A partir de um qualquer conjunto de exemplos de treino D (possivelmente com erros), uma teoria do domínio B (possivelmente com erros) e um espaço de hipóteses candidatas H pretende-se um algoritmo que determine a hipótese que ajuste melhor os exemplos de treino e a teoria do domínio. Poder-se-á pretender uma hipótese que ajuste os dados melhor do que a teoria do domínio, ou vice-versa, ou ainda que os ajuste da mesma forma.

Considerado que o erro $erro_D(h)$ de h é a proporção de exemplos de D que são mal classificados por h , o erro $erro_B(h)$ de h relativamente à teoria do domínio B como a probabilidade de h entrar em desacordo com B na classificação de uma instância criada aleatoriamente. Poder-se-á caracterizar a hipótese desejada em termos destes erros, pretendendo-se a hipótese que minimize uma medida combinada destes erros, tal como

$$\arg \min_{h \in H} K_D erro_D(h) + K_B erro_B(h)$$

Não é claro que valores atribuir a K_D e K_B para especificar a importância relativa do ajuste dos dados, versus o ajuste da teoria do domínio. Se existir uma teoria do domínio pobre e um grande número de dados sem ruído, será melhor pesar mais $erro_D(h)$. Se por sua vez existir uma boa teoria do domínio e uma pequena amostra de dados com ruído, os melhores resultados são obtidos pesando mais $erro_B(h)$. Se não se souber à partida qual a qualidade da teoria do domínio e dos dados de treino, não será claro como pesar estes dois erros.

Uma perspectiva alternativa na questão de como pesar o conhecimento prévio e os dados de treino é a perspectiva Bayesiana. Pode-se pensar em $P(h)$, $P(D)$ e $P(D|h)$ como uma forma de conhecimento prévio ou teoria do domínio e pode-se pensar no teorema de Bayes como um método de pesar esta teoria do domínio conjuntamente com os dados observados D , para atribuir uma probabilidade posterior $P(h|D)$ a h . Segundo a estratégia Bayesiana a melhor hipótese é a hipótese cuja probabilidade posterior é maior. No entanto, o teorema de Bayes implicitamente assume conhecimento perfeito relativamente às distribuições de probabilidade $P(h)$, $P(D)$ e $P(D|h)$. Uma abordagem possível nos casos em que estas distribuições de probabilidade não são perfeitamente conhecidas, consiste em assumir as distribuições da probabilidade prévia sobre $P(h)$, $P(D)$ e $P(D|h)$ e depois calcular o valor esperado da probabilidade posterior $P(h|D)$. Contudo, isto requer conhecimento adicional à cerca das probabilidades prévias sobre $P(h)$, $P(D)$ e $P(D|h)$.

4.2. Pesquisa do Espaço de Hipóteses

Algumas abordagens foram desenvolvidas para combinar a teoria do domínio e os dados de treino, de forma a restringir a pesquisa de uma hipótese aceitável. A maior parte destas abordagens são extensões aos métodos indutivos (“Backpropagation”, FOIL, entre outros). Estas abordagens usam o conhecimento prévio para alterar a pesquisa realizada pelos métodos puramente indutivos:

- Uso do conhecimento prévio para obter uma hipótese inicial a partir da qual começa a pesquisa.
- Uso do conhecimento prévio para alterar o objectivo da pesquisa do espaço de hipóteses.
- Uso do conhecimento prévio para alterar as etapas de pesquisa disponíveis.

4.3 Uso do Conhecimento Prévio para Inicializar a Hipótese

Uma abordagem consiste em usar o conhecimento prévio para inicializar uma hipótese que ajuste perfeitamente a teoria do domínio, e depois refinar indutivamente esta hipótese inicial para ajustar os dados de treino. Esta abordagem é usada pelo algoritmo KBANN (“Knowledge-Based Artificial Neuronal Network”) para aprender redes neuronais. No KBANN é construída uma rede inicial, de tal forma, que para todas as instâncias possíveis, a classificação atribuída pela rede é igual à atribuída pela teoria do domínio. O algoritmo “Backpropagation” é depois aplicado para ajustar os pesos desta rede inicial de forma a ajustar os exemplos de treino.

Se a teoria do domínio estiver correcta, a hipótese inicial classificará correctamente todos os exemplos de treino não havendo necessidade de revisão. Contudo, se a hipótese inicial é imperfeita para a classificação dos exemplos de treino, então será refinada indutivamente para melhorar o ajuste desses exemplos. Como no algoritmo KBANN, a inicialização da rede é feita com valores de forma a ajustar a teoria do domínio, consegue-se uma melhor aproximação inicial à função alvo do que uma inicialização da rede com pesos iniciais aleatórios. Este facto verifica-se mesmo em casos em que a teoria do domínio é aproximadamente correcta, levando por certo a uma melhor exactidão na generalização da hipótese final.

Esta abordagem foi explorada por muitos investigadores como por exemplo, [Shavlik-89; Towell-94; Fu-89; Fu-93; Pratt-93a; Pratt-93b].

4.3.1 Algoritmo KBANN

O algoritmo KBANN assume a teoria do domínio representada por um conjunto de regras proposicionais e não recursivas.

O algoritmo KBANN é constituído por dois estádios, primeiro a criação de uma rede neuronal que ajuste perfeitamente a teoria do domínio e depois o uso do algoritmo “Backpropagation” para refinar esta rede de forma a ajustar os exemplos de treino. Os detalhes deste algoritmo encontram-se na tabela 4.1 [Mitchell-97].

No primeiro estádio do algoritmo KBANN (etapas 1-3), é construída uma rede inicial que é consistente com a teoria do domínio.

Tabela 4.1 – Algoritmo KBANN

KBANN (*Teoria_Domínio*, *Exemplos_Treino*)*Teoria_Domínio* : Conjunto de regras proposicionais não recursivas*Exemplos_Treino* : Conjunto de pares (“input”, “output”) da função alvo

Etapa Analítica : Criar uma rede inicial equivalente à teoria do domínio

1. Para cada atributo da instância, criar um “input” da rede
2. Para cada regra na *Teoria_Domínio*, criar uma unidade de rede que:
 - Ligue os “inputs” desta unidade aos atributos testados pelos antecedentes (pré-condições) da regra.
 - Para cada antecedente não negado da regra, atribua um peso de W ao input da unidade sigmóide correspondente.
 - Para cada antecedente negado da regra, atribua um peso de $-W$ ao input da unidade sigmóide correspondente.
 - Atribua $-(n-0.5)W$ ao peso limite w_0 para essa unidade, em que n é o número de antecedentes não negados da regra.
3. Acrescentar ligações adicionais entre as unidades da rede, ligando cada unidade da rede com profundidade i do nível de “input” a todas as unidades da rede com profundidade $i+1$. Atribuir pesos aleatórios perto de zero a estas ligações adicionais.

Etapa Indutiva : Refinar a rede inicial

4. Aplicar o algoritmo “Backpropagation” para ajustar os pesos da rede inicial de forma a ajustarem os exemplos de treino.

O primeiro passo da etapa analítica consiste na criação de um “input” da rede por cada atributo das instâncias. De seguida, é criada uma unidade sigmóide limitada, para cada regra da teoria do domínio. O KBANN segue a convenção que um valor de “output” do sigmóide maior que 0.5 é interpretado como verdadeiro e um valor menor que 0.5 como falso. Assim, cada unidade é construída, de forma que o seu “output” seja maior que 0.5 no caso da regra correspondente seja aplicada. Para cada antecedente da regra é criado um input para cada unidade sigmóide correspondente. Os pesos das unidades sigmóides são assim atribuídos calculando-se então o E lógico dos seus “inputs”. Para cada “input” correspondente a um antecedente não negado, é atribuído um peso que é uma constante positiva W enquanto que para cada “input” correspondente a um antecedente negado, é atribuído o peso $-W$. Ao peso limite da unidade (w_0) é atribuído $-(n-0.5)W$, em que n é o número de antecedentes não negados.

Quando os valores de “input” da unidade são 1 ou 0, assegura-se que a sua soma pesada mais w_0 é um valor positivo (e o “output” do sigmóide será maior que 0.5) se e só se todas as regras antecedentes foram satisfeitas. Para as unidades sigmóides no segundo nível e subsequentes, as unidades de “input” não têm necessariamente que ser 1 e 0 e o anterior argumento não pode aplicar-se. Contudo, se for escolhido um valor suficientemente grande para W , o algoritmo KBANN pode codificar a teoria do domínio para uma rede de profundidade arbitrária. [Towell-94] usou $W=4$ em muitas das suas experiências.

Cada “input” da unidade sigmóide é ligado a um “input” da rede apropriado ou ao “output” de outra unidade sigmóide, para repercutir as dependências entre os atributos correspondentes na teoria do domínio. Numa etapa final são adicionados “inputs” a cada unidade sigmóide com pesos próximos de zero. O papel destas ligações adicionais consiste em permitir à rede aprender indutivamente dependências adicionais além das sugeridas pela teoria do domínio.

O segundo estágio do KBANN (etapa 4 no algoritmo da tabela 4.1) usa os exemplos de treino e o algoritmo “Backpropagation” para refinar os pesos iniciais da rede.

No caso da teoria do domínio e os exemplos de treino serem inconsistentes, esta etapa altera os pesos iniciais da rede. A rede resultante pode ter pesos muito positivos, pesos muito negativos e pesos insignificantes. Durante a etapa indutiva podem ser descobertas novas dependências.

A vantagem do algoritmo KBANN sobre o algoritmo “Backpropagation” é que tipicamente generaliza com maior exactidão do que o “Backpropagation” quando é dada uma teoria do domínio aproximadamente correcta, especialmente quando os dados de treino são escassos.

As regras extraídas da rede final possibilitam uma teoria do domínio que melhor ajusta os dados observados. Apesar de por vezes ser possível a partir dos pesos aprendidos da rede mapear um conjunto de regras, geralmente esta tarefa é problemática pois algumas características dos pesos não têm uma regra análoga obtida directamente. [Graven-94; Craven-96] descrevem métodos alternativos para extrair regras simbólicas a partir de redes aprendidas.

A pesquisa do espaço de hipóteses do algoritmo KBANN e do algoritmo “Backpropagation” é puramente indutivo e é ilustrado esquematicamente na figura 4.1.

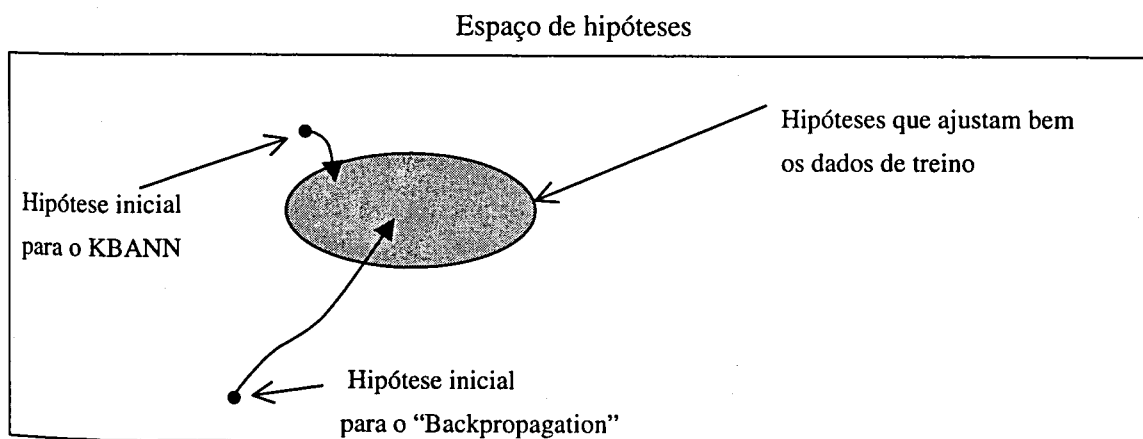


Figura 4.1 – Pesquisa do Espaço de Hipóteses no KBANN

A diferença chave reside na hipótese inicial à qual é aplicado o ajuste dos pesos. No caso de poderem ser encontradas muitas hipóteses (vectores de pesos) para ajustar os dados (uma condição que acontece especialmente quando os dados de treino são escassos), o KBANN converge para a hipótese que generaliza para além dos dados, da forma mais parecida com as predições da teoria do domínio.

Uma limitação do KBANN reside no facto de só poder acomodar teorias de domínio proposicionais e poder ser induzido em erro quando é dada uma teoria do domínio muito deficiente. Contudo, foi demonstrado que o KBANN é útil em muitos problemas práticos.

O algoritmo KBANN ilustra uma abordagem de inicialização da hipótese que combina a aprendizagem analítica e indutiva. Existem outras abordagens que variam na técnica de construção da rede inicial, na aplicação do “Backpropagation” para ajustar os pesos e nos métodos para extrair descrições simbólicas da rede refinada.

4.4 Uso do Conhecimento Prévio para Alterar o Objectivo da Pesquisa

Uma forma alternativa de usar o conhecimento prévio consiste em incorporá-lo no critério de erro minimizado pela descida da encosta.

4.4.1 Algoritmo TangentProp

O conhecimento prévio pode ser considerado na forma de derivadas da função alvo.

O algoritmo TangentProp [Simard-92] acomoda o conhecimento do domínio expresso por derivadas da função alvo relativamente a transformações dos seus "inputs".

Considere-se uma tarefa de aprendizagem envolvendo um espaço de instâncias X , uma função alvo f , e cada exemplo de treino na forma $(x_i, f(x_i), \frac{\partial f(x)}{\partial x} \Big|_{x_i})$, em que x_i é uma instância, $f(x_i)$ o

valor de treino e $\frac{\partial f(x)}{\partial x} \Big|_{x_i}$ a derivada da função alvo f relativamente a x , avaliada no ponto

$x=x_i$.

Como ilustração considere-se uma tarefa simples de aprendizagem da função f a partir de três exemplos de treino $(x_1, f(x_1))$, $(x_2, f(x_2))$ e $(x_3, f(x_3))$ como mostra o gráfico da figura 4.2 [Mitchell-97].

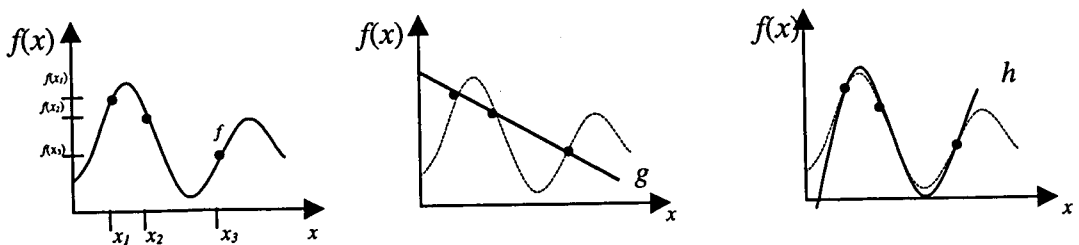


Figura 4.2 – Ajuste de valores e derivadas com o Algoritmo TangentProp

Dados estes três exemplos, o algoritmo "Backpropagation" poderá gerar uma hipótese como uma função plana, tal como a função g ilustrada no gráfico central da figura 4.2. O gráfico da direita da figura mostra o efeito possibilitado pelas derivadas de treino, ou declives, como informação adicional para cada exemplo de treino (por exemplo, $(x_i, f(x_i), \frac{\partial f(x)}{\partial x} \Big|_{x_i})$).

Ajustando-se tanto os valores de treino $f(x_i)$ como as derivadas de treino $\frac{\partial f(x)}{\partial x} \Big|_{x_i}$, obtêm-se

uma forma melhor de generalizar correctamente a partir dos dados de treino.

Com a inclusão das derivadas de treino ultrapassa-se a limitação indutiva do "Backpropagation" de favorecer uma interpolação entre os pontos, substituindo-a pela

informação das derivadas de treino. A hipótese resultante h do gráfico da direita da figura 4.2 possibilita uma estimativa mais exacta da verdadeira função alvo f .

O algoritmo TangentProp além de poder considerar este tipo de derivadas da função alvo pode aceitar derivadas de treino relativamente a várias transformações do “input” x .

Considere-se, por exemplo, a tarefa de aprendizagem para reconhecer caracteres escritos à mão. Assuma-se que o “input” x corresponde a uma imagem contendo um carácter escrito à mão, e a tarefa de aprendizagem consiste em classificar correctamente o carácter. Nesta tarefa, a função alvo é invariável a pequenas rotações do carácter dentro da imagem. Para isso, define-se a transformação $s(\alpha, x)$, que roda a imagem x , α graus. Pode-se então expressar uma asserção relativamente à invariância na rotação, determinando que para cada instância de treino x_i , a derivada da função alvo relativamente a esta transformação é zero, isto é, a rotação da imagem de “input” não altera o valor da função alvo. Assim, pode-se considerar a seguinte derivada de treino para todas as instâncias de treino x_i

$$\frac{\partial f(s(\alpha, x_i))}{\partial \alpha} = 0$$

em que f é a função alvo e $s(\alpha, x)$ é a imagem resultante da aplicação da transformação s à imagem x_i .

No algoritmo TangentProp estas derivadas de treino são incorporadas na função erro que é minimizada pela descida da encosta.

O algoritmo “Backpropagation” realiza a descida da encosta para minimizar a soma do quadrado dos erros

$$E = \sum_i (f(x_i) - \hat{f}(x_i))^2$$

em que x_i é a i^a instância de treino, f é a função alvo e $\hat{f}$ é a função representada pela rede neuronal aprendida.

No algoritmo TangentProp é incluído um termo adicional na função erro para penalizar discrepâncias entre as derivadas de treino e as derivadas actuais da função $\hat{f}$ da rede neuronal aprendida. Para cada transformação $s_j(\alpha, x)$, o TangentProp considera o quadrado do erro entre a derivada de treino e a derivada actual da rede neuronal aprendida. A função erro modificada é

$$E = \sum_i \left[(f(x_i) - \hat{f}(x_i))^2 + \mu \sum_j \left(\frac{\partial f(s_j(\alpha, x_i))}{\partial \alpha} - \frac{\partial \hat{f}(s_j(\alpha, x_i))}{\partial \alpha} \right)^2 \right]_{\alpha=0} \quad (4.1)$$

em que μ é uma constante fornecida pelo utilizador para determinar a importância relativa de ajustar os valores de treino versus as derivadas de treino.

4.4.2 Um exemplo ilustrativo

[Simard-92] apresenta resultados comparativos da exactidão da generalização do TangentProp e do “Backpropagation” puro para o problema de reconhecimento de caracteres escritos à mão. A tarefa consiste em classificar imagens contendo um dígito entre 0 e 9. Na experiência tanto

o TangentProp como o “Backpropagation” usaram um conjunto de dados de treino de tamanhos diferentes e depois foram avaliados a partir do seu desempenho sobre um conjunto de teste separado de 160 exemplos.

O conhecimento prévio fornecido ao TangentProp foi que a classificação do dígito é invariável à translação vertical e horizontal da imagem. Os resultados ilustrados na tabela 4.2, demonstram a capacidade do TangentProp usar o conhecimento prévio para generalizar com mais exactidão do que o “Backpropagation” puro [Mitchell-97].

Tabela 4.2 – Comparação de exactidão entre o Algoritmo TangentProp e o Algoritmo “Backpropagation”

Tamanho do conjunto de exemplos de treino	Percentagem de erro no conjunto de teste	
	TangentProp	“Backpropagation”
10	34	48
20	17	33
40	7	18
80	4	10
160	0	3
320	0	0

Apesar do algoritmo TangentProp combinar o conhecimento prévio com os dados de treino para conduzir a aprendizagem de redes neuronais, não é robusto a erros no conhecimento prévio. Quando o conhecimento prévio é incorrecto, isto é, quando as derivadas de treino não reflectem correctamente as derivadas da verdadeira função alvo, o algoritmo ajusta estas derivadas incorrectas, generalizando menos correctamente do que se tivesse ignorado esse conhecimento prévio e usasse o “Backpropagation” puro. Se for sabido em avanço o grau de erro nas derivadas de treino, pode-se usar esta informação para escolher a constante μ que determina a importância relativa do ajuste dos valores de treino e do ajuste das derivadas de treino. O algoritmo TangentProp incorpora conhecimento prévio para influenciar a pesquisa da hipótese alterando assim a função objectivo. Isto corresponde a alterar o objectivo da pesquisa da hipótese, como ilustra a figura 4.3.

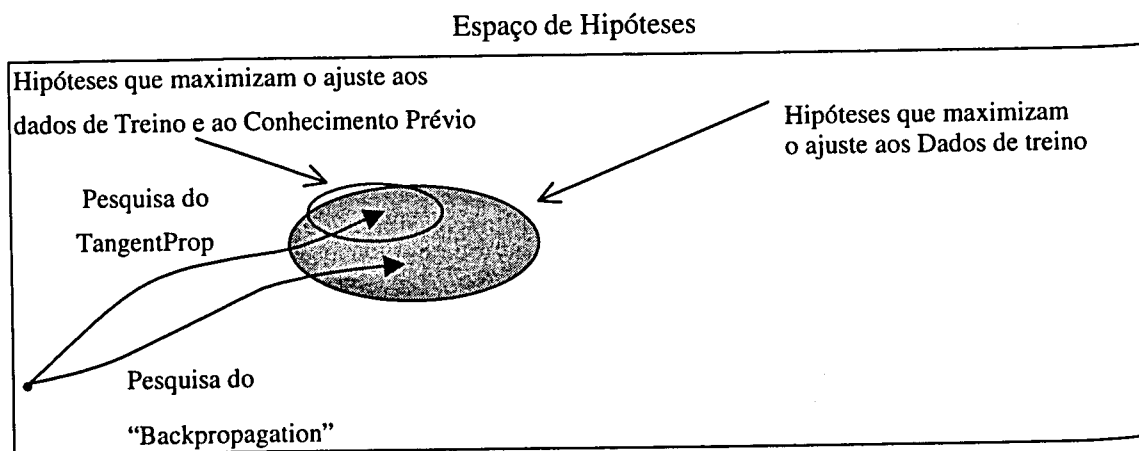


Figura 4.3 – Pesquisa do Espaço de Hipóteses no TangenteProp

Tal como no “Backpropagation” (mas não no KBANN), o TangentProp começa a pesquisa com uma rede inicial de pesos aleatórios pequenos. Contudo, a regra de treino da descida da encosta produz actualizações nos pesos diferentes das do “Backpropagation”, resultando numa hipótese final diferente.

Como mostra a figura, o conjunto de hipóteses que minimizam o objectivo do TangentProp deve diferir do conjunto que minimiza o objectivo do “Backpropagation”. Se os exemplos de treino e o conhecimento prévio são ambos correctos, e a função alvo pode ser representada com exactidão por uma rede neuronal, então o conjunto de vectores de pesos que satisfazem o objectivo do TangentProp será um subconjunto dos que satisfazem o objectivo do “Backpropagation”. A diferença entre estes dois conjuntos de hipóteses finais é o conjunto de hipóteses incorrectas que são consideradas pelo “Backpropagation” mas que são eliminadas pelo TangentProp devido ao seu conhecimento prévio.

4.5 Uso do Conhecimento Prévio para Aumentar os Operadores de Pesquisa

O conhecimento prévio pode ser utilizado para alterar a pesquisa do espaço de hipóteses mais concretamente pela alteração do conjunto de operadores que definem as etapas legais na pesquisa pelo espaço de hipóteses.

Esta abordagem é seguida por sistemas como o FOCL [Pazzani-91; Pazzani-92] e ML-SMART [Bergadeno-90].

4.5.1 Algoritmo FOCL

O FOCL é uma extensão do sistema FOIL, puramente indutivo. Tanto o FOIL como o FOCL aprendem um conjunto de regras para cobrir os exemplos de treino observados. Ambos os sistemas empregam um algoritmo de cobertura sequencial que aprende uma regra, remove os exemplos positivos cobertos por esta nova regra, e depois itera este procedimento sobre os restantes exemplos de treino. Em ambos os sistemas, cada nova regra é criada realizando uma pesquisa geral-para-específica, começando com a regra mais geral possível (a regra sem pré-condições). De seguida são geradas muitas especializações da regra corrente e a especialização com maior ganho de informação relativa aos exemplos de treino é escolhida. Este processo é iterado, gerando novas especializações candidatas e seleccionando a melhor, até uma regra com desempenho satisfatório ser obtida.

A diferença entre o FOIL e o FOCL reside na forma em como as especializações candidatas são geradas durante a pesquisa geral-para-específica por uma regra. O FOIL gera cada especialização candidata pela adição de um novo literal às pré-condições da regra. O FOCL usa este mesmo método para produzir especializações candidatas, mas também gera especializações adicionais baseadas na teoria do domínio.

Tanto o FOCL como o FOIL aprendem regras de primeira ordem.

Os literais (condições) operacionais são os literais da teoria do domínio que são usados para descrever os exemplos de treino. Por outro lado, os literais que representam só características intermédias na teoria do domínio são considerados literais inoperacionais. Em cada ponto na sua pesquisa geral-para-específica, o FOCL expande a sua hipótese corrente h usando os seguintes dois operadores:

1. Para cada literal operacional que não faz parte de h , criar uma especialização de h , pela

adição desse literal às pré-condições. Este também é o método usado por FOIL para gerar sucessores candidatos.

2. Criar uma condição lógica operacional para o conceito alvo de acordo com a teoria do domínio. Adicionar esse conjunto de literais às pré-condições de h . Finalmente, eliminar os literais das pré-condições que de acordo com os dados de treino não são necessários.

Relativamente ao segundo operador, o FOCL primeiro selecciona uma das regras da teoria do domínio cuja cabeça corresponda ao conceito alvo.

Se houverem várias regras, selecciona a regra cujas pré-condições tenham o maior ganho de informação relativo aos exemplos de treino do conceito alvo.

Cada literal inoperacional destas pré-condições é então substituído usando novamente a teoria do domínio, mais concretamente, pela substituição de pós-condições por pré-condições. Este processo de desdobramento da teoria do domínio repete-se até que as condições tenham sido especificadas em termos de literais operacionais. Se houverem várias regras alternativas na teoria do domínio que produzam resultados diferentes, é seleccionada a que tiver maior ganho de informação em cada etapa do processo de desdobramento.

Como etapa final na geração de especializações candidatas, cada literal na expressão é removido verificando-se de seguida se esta remoção leva a uma maior exactidão na classificação dos exemplos de treino. Se isso acontecer então esse literal é retirado da regra. Esta etapa é incluída para corrigir uma possível sobrespecialização no caso de se estar perante uma teoria do domínio imperfeita e que inclua literais irrelevantes.

Sumariando, o FOCL aprende regras da forma $c \leftarrow o_i \wedge o_b \wedge o_f$ em que c é o conceito alvo, o_i é uma conjunção inicial de literais operacionais adicionados um de cada vez pelo primeiro operador, o_b é uma conjunção de literais operacionais adicionados numa única etapa a partir da teoria do domínio, e o_f é uma conjunção final de literais operacionais adicionados um de cada vez pelo primeiro operador. Qualquer um destes conjuntos de literais pode ser vazio.

O FOCL usa a teoria do domínio para aumentar as especializações candidatas consideradas em cada etapa da pesquisa de uma regra. Comparando a pesquisa do espaço de hipóteses realizado pelo FOCL e pelo FOIL, verifica-se que o FOCL realiza especializações que correspondem a macro-etapas na pesquisa do FOIL, em que muitos literais são adicionados numa única etapa. Este processo pode ser visto como a escolha de uma hipótese que deveria ser considerada mais tarde na pesquisa e que é considerada de imediato. Se a teoria do domínio estiver correcta, os dados de treino, evidenciam a superioridade de uma especialização sobre as outras e então será seleccionada. Se a teoria do domínio é incorrecta, a avaliação empírica de todas as especializações candidatas direccionarão a pesquisa por um caminho alternativo.

O algoritmo escolhe entre as especializações candidatas baseado só nos dados de treino. Assim, a teoria do domínio é usada de uma forma que limita o algoritmo, mas deixa que as escolhas finais da pesquisa sejam feitas baseadas nos desempenhos sobre os dados de treino. A limitação introduzida pela teoria do domínio consiste na preferência por regras operacionais vinculadas pela teoria do domínio. Esta limitação é combinada com a limitação do FOIL, que reside na preferência por hipóteses mais pequenas.

O FOCL tem sido usado em domínios de aplicação em que a teoria do domínio disponível é imperfeita e verifica-se que generaliza com maior exactidão do que o algoritmo FOIL puramente indutivo. Por exemplo, [Pazzani-92] explora a aprendizagem do conceito "posições legais no xadrez". Dados 60 exemplos de treino descrevendo 30 posições legais e 30 posições ilegais, o FOIL obteve uma exactidão de 86% sobre um conjunto independente de exemplos de teste. Ao FOCL foram fornecidos os mesmos 60 exemplos de treino, com uma teoria do domínio com 76% de exactidão e obteve-se uma exactidão na generalização de 94% (menos do que metade do rácio de erro do FOIL). Resultados similares foram obtidos noutros domínios.

As abordagens analisadas de combinação da aprendizagem indutiva e analítica realizam aprendizagem puramente indutiva nos domínios seleccionados. Estes métodos não foram ainda completamente testados e provados numa grande variedade de domínios de problemas e continuam por isso a ser uma área activa de investigação.

Capítulo 5

Avaliação de Hipóteses

5.1 Introdução

A avaliação da exactidão das hipóteses aprendidas é de grande importância. De facto, é necessário saber se a hipótese aprendida a partir de uma amostra limitada de dados tem um desempenho aceitável sobre exemplos de treino adicionais. Por outro lado, a avaliação de hipóteses é um componente integral de muitos métodos de aprendizagem. Por exemplo, no “post-pruning” das árvores de decisão para evitar o sobreajustamento, é necessário avaliar o impacto dos passos de “pruning” na exactidão da árvore de decisão resultante.

A estimativa da exactidão de uma hipótese é relativamente linear quando existem dados suficientes. Contudo, quando é necessário aprender uma hipótese e estimar a sua futura exactidão a partir de um conjunto de dados limitado, surgem duas dificuldades chave:

- Limitações na estimativa. A exactidão da hipótese aprendida sobre os exemplos de treino é frequentemente um estimador pobre da sua exactidão sobre futuros exemplos. Para obter uma estimativa da exactidão futura, normalmente utiliza-se um conjunto de exemplos de validação independentes dos exemplos de treino e da hipótese, para testar a exactidão da hipótese aprendida
- Variância na estimativa. Mesmo que a exactidão da hipótese seja medida sobre um conjunto de exemplos independentes dos exemplos de treino, essa medida pode variar relativamente à verdadeira exactidão, dependendo do conjunto de exemplos de validação. Quanto mais pequeno for o conjunto de exemplos de validação, maior será a variância esperada em relação à verdadeira exactidão da hipótese.

5.2 Estimativa da Exactidão de Uma Hipótese

Quando se avalia uma hipótese aprendida é necessário estimar a exactidão com que esta classifica instâncias futuras. Simultaneamente é conveniente saber a probabilidade de erro nessa estimativa da exactidão.

Considere-se D a probabilidade de encontrar cada instância x em X (conjunto de todas as instâncias), o número de exemplos n e a função booleana alvo $f: x \rightarrow \{0,1\}$. O erro da amostra de uma hipótese h ($\text{erro}_A(h)$) relativamente à função alvo f e à amostra de dados A é

$$\text{erro}_A(h) = \frac{1}{n} \sum_{x \in A} \delta(f(x), h(x)) \text{ em que } \delta(f(x), h(x)) \text{ é } 1 \text{ se } f(x) \neq h(x) \text{ e } 0 \text{ caso contrário}$$

O erro verdadeiro de uma hipótese h ($erro_D(h)$) relativamente à função alvo f e à distribuição D é a probabilidade de h classificar incorrectamente uma instância aleatória de acordo com D

$$erro_D(h) = \Pr_{x \in D} [f(x) \neq h(x)]$$

em que $\Pr$ é a probabilidade sobre a distribuição de instâncias D .

O que normalmente se pretende saber é o erro verdadeiro das hipóteses, pois é esse erro que esperamos quando aplicamos a hipótese a exemplos futuros. Contudo, o que se pode medir é o erro da amostra, sendo possível estimar também o erro verdadeiro a partir do erro da amostra.

Pode-se estimar o erro verdadeiro da hipótese discreta h , baseando-se num erro da amostra A desde que :

- A amostra contenha n exemplos independentes uns dos outros, e independentes de h de acordo com a distribuição de probabilidades D
- Os dados futuros usam também a distribuição de probabilidades D
- $n \geq 30$
- A hipótese h comete r erros sobre os n exemplos ($erro_D(h) = r/n$)

Sob estas condições, a estatística permite-nos fazer as seguintes asserções:

1. Sem mais informação, o valor mais provável de $erro_D(h)$ é o $erro_A(h)$
2. Com aproximadamente 95% de probabilidade, o erro verdadeiro $erro_D(h)$ situa-se no intervalo

$$erro_A(h) \pm 1.96 \cdot \sqrt{\frac{erro_A(h)(1 - erro_A(h))}{n}}$$

A constante 1.96 é usada se pretendermos um intervalo de confiança de 95%, mas pode ser utilizada uma constante diferente Z_N para um determinado intervalo de confiança de $N\%$. A expressão geral para aproximar o $erro_D(h)$ com intervalo de confiança de $N\%$ é

$$erro_A(h) \pm Z_N \cdot \sqrt{\frac{erro_A(h)(1 - erro_A(h))}{n}}$$

em que a constante Z_N é escolhida dependendo do nível do intervalo de confiança desejado (usando os valores Z_N para intervalos de confiança com dois limites dados na tabela 5.1

[Mitchell-97]) e $\sqrt{\frac{erro_A(h)(1 - erro_A(h))}{n}}$ é a variância do erro esperado ($\sigma erro_A(h)$).

Tabela 5.1- Valores de Z_N para diferentes Intervalos de Confiança

Nível de confiança $N\%$:	50%	68%	80%	90%	95%	98%	99%
Constante Z_N :	0.67	1.00	1.28	1.64	1.96	2.33	2.58

5.3 Diferença no Erro de Duas Hipóteses

Considere-se o caso em que existem duas hipóteses h_1 e h_2 para uma função alvo discreta. A hipótese h_1 foi testada numa amostra A_1 contendo n_1 exemplos aleatórios, e a hipótese h_2 foi testada numa amostra independente A_2 contendo n_2 exemplos aleatórios da mesma distribuição. Considere-se que se pretende estimar a diferença d entre os erros verdadeiros destas duas hipóteses

$$d = \text{erro}_D(h_1) - \text{erro}_D(h_2)$$

em que d é o parâmetro a ser estimado e o estimador escolhido é a diferença entre os erros da amostra

$$\hat{d} = \text{erro}_{A_1}(h_1) - \text{erro}_{A_2}(h_2)$$

Para um grande valor de n_1 e n_2 (≥ 30), ambos os erros $\text{erro}_{A_1}(h_1)$ e $\text{erro}_{A_2}(h_2)$ seguem distribuições que são aproximadamente normais. Pelo facto da diferença de duas distribuições normais ser também uma distribuição normal, o estimador (a diferença entre os erros da amostra) segue também uma distribuição que é aproximadamente normal, com média d . Pode também ser provado que a variância desta distribuição é a soma das variâncias dos $\text{erro}_{A_1}(h_1)$ e $\text{erro}_{A_2}(h_2)$. Daí que

$$\sigma_d^2 \approx \frac{\text{erro}_{A_1}(h_1)(1 - \text{erro}_{A_1}(h_1))}{n_1} + \frac{\text{erro}_{A_2}(h_2)(1 - \text{erro}_{A_2}(h_2))}{n_2} \quad (5.1)$$

Para uma variável aleatória $\hat{d}$ com distribuição normal, média d e variância σ^2 , o intervalo de confiança de $n\%$ estimado para d é

$$\hat{d} \pm Z_N \sigma$$

Daí que, o intervalo de confiança de $N\%$ estimado para d é

$$\hat{d} \pm Z_N \cdot \sqrt{\frac{\text{erro}_{A_1}(h_1)(1 - \text{erro}_{A_1}(h_1))}{n_1} + \frac{\text{erro}_{A_2}(h_2)(1 - \text{erro}_{A_2}(h_2))}{n_2}}$$

onde Z_N é uma constante da tabela 5.1. Esta expressão fornece-nos um intervalo de confiança de $N\%$ com dois limites para estimar a diferença entre erros de duas hipóteses. Em algumas situações podemos estar interessados em ter só um limite no intervalo de confiança. Por exemplo, saber qual a probabilidade do erro verdadeiro da hipótese ser pelo menos um determinado valor. Assumindo que $N\% = 100(1-\alpha)\%$ e que α é a probabilidade do valor se situar fora de um intervalo de confiança com dois limites, então para um intervalo de confiança com só um limite $N\% = 100(1-\alpha/2)$.

Apesar de se considerar que h_1 e h_2 são testados em amostras de dados independentes é frequente aceitar o uso de intervalos de confiança em situações em que h_1 e h_2 são testados

numa única amostra A (onde A continua a ser independente de h_1 e h_2). Neste caso redefine-se

$$\hat{d} = \text{erro}_A(h_1) - \text{erro}_A(h_2)$$

A variância neste novo $\hat{d}$ normalmente é mais pequena do que a variância considerando os conjuntos A_1 e A_2 separados. Isto deve-se ao facto de que o uso de uma amostra única A elimina a variância resultante das diferenças aleatórias na composição de A_1 e A_2 .

5.3.1 Teste de Hipóteses

Considere-se os erros das amostras de duas hipóteses h_1 e h_2 (com 100 exemplos cada uma) $\text{erro}_{A_1}(h_1) = 0.30$ e $\text{erro}_{A_2}(h_2) = 0.20$ e a diferença dos erros de 0.10. Pretende-se saber qual a probabilidade de que o $\text{erro}_D(h_1) > \text{erro}_D(h_2)$, isto é, de $d > 0$ dada a diferença nos erros das amostras. A probabilidade de $d > 0$ é igual à probabilidade da diferença entre os erros da amostra ($\hat{d}$) não sobreajustar d em mais do que 0.10, isto é, é a probabilidade da diferença entre os erros da amostra situar-se num intervalo com um limite

$$\hat{d} < d + 0.10$$

Uma vez que d é o valor esperado da distribuição que governa $\hat{d}$, pode-se considerar

$$\hat{d} < \mu_{\hat{d}} + 0.10$$

Expressando este intervalo em termos dos desvios padrões em relação ao valor esperado e usando a equação (5.1) que permite calcular $\sigma_{\hat{d}} = 0.061$ pode-se rescrever o intervalo como

$$\hat{d} < \mu_{\hat{d}} + 1.64 \cdot \sigma_{\hat{d}}$$

Consultando a tabela 5.1, verifica-se que 1.64 corresponde a um intervalo de confiança com dois limites de 90%. O intervalo de confiança só com um limite é de 95%.

Assim, dado que $\hat{d} = 0.10$, a probabilidade de $\text{erro}_D(h_1) > \text{erro}_D(h_2)$ é de aproximadamente 95%.

5.4 Comparação de Algoritmos de Aprendizagem

Frequentemente existe interesse em comparar o desempenho de dois algoritmos de aprendizagem, e determinar se uma diferença verificada entre eles é estatisticamente significativa.

Inicialmente, é necessário especificar o parâmetro a estimar. Considere que se pretende determinar qual “em média” é o melhor algoritmo de aprendizagem para aprender uma função alvo f . Uma forma razoável de definir “em média”, consiste em considerar a média do desempenho relativo desses algoritmos relativamente a todos os conjuntos de treino de

tamanho n que podem ocorrer sobre a distribuição de instâncias D . Pretende-se no fundo, estimar o valor esperado da diferença nos erros dos algoritmos de aprendizagem

$$E_{A \subset D} [\text{erro}_D(L_1(A)) - \text{erro}_D(L_2(A))] \quad (5.2)$$

em que $L(A)$ é o “output” do método de aprendizagem L dada a amostra A de exemplos de treino e $A \subset D$ indica que o valor esperado é calculado sobre amostras A de acordo com a distribuição de instâncias D .

Evidentemente que, na prática, só existe uma amostra de dados limitada D_0 , quando se comparam algoritmos de aprendizagem. Uma abordagem seguida consiste em dividir D_0 num conjunto de treino A_0 e um conjunto de teste disjunto T_0 . Os dados de treino podem ser usados para treinar ambos os algoritmos L_1 e L_2 , e os dados de teste podem ser usados para comparar a exactidão das duas hipóteses aprendidas. Mede-se portanto o valor

$$\text{erro}_{T_0}(L_1(A_0)) - \text{erro}_{T_0}(L_2(A_0))$$

Uma forma de melhorar este estimador consiste em particionar repetidamente os dados de D_0 num conjunto de treino e num conjunto de teste disjunto e calcular a média dos erros de teste para estas diferentes experiências. Isto leva ao algoritmo da tabela 5.2 [Mitchell-97].

Tabela 5.2 – Algoritmo para Estimar a Diferença de Erro entre dois Métodos de Aprendizagem (L_1 e L_2)

-
1. Particionar os dados disponíveis D_0 em k subconjuntos disjuntos $T_1, T_2, \dots, T_k$ de igual tamanho, em que o tamanho é pelo menos 30
 2. Para i de 1 a k , Fazer
 - Usar T_i para o conjunto de teste, e os dados restantes para o conjunto de treino A_i
 - $A_i \leftarrow \{D_0 - T_i\}$
 - $h_1 \leftarrow L_1(A_i)$
 - $h_2 \leftarrow L_2(A_i)$
 - $\delta_i \leftarrow \text{erro}_{T_i}(h_1) - \text{erro}_{T_i}(h_2)$
 3. Retornar o valor $\bar{\delta}$, em que $\bar{\delta} = \frac{1}{k} \sum_{i=1}^k \delta_i$
-

Este valor $\bar{\delta}$ retornado pelo algoritmo da tabela 5.2 pode ser tomado como uma estimativa do valor desejado na equação (5.2). Mais concretamente, pode ser visto como uma estimativa do valor

$$E_{A \subset D_0} [\text{erro}_D(L_1(A)) - \text{erro}_D(L_2(A))] \quad (5.3)$$

em que A representa uma amostra aleatória de D_0 de tamanho $\frac{k-1}{k} |D_0|$.

A aproximação do intervalo de confiança $N\%$ para estimar a quantidade da equação (5.3) é dada por

$$\bar{\delta} \pm t_{N,k-1} A_{\bar{\delta}}$$

em que $t_{N,k-1}$ é uma constante que tem um papel análogo ao Z_N já referido anteriormente, e $A_{\bar{\delta}}$ é uma estimativa do desvio padrão da distribuição que governa $\bar{\delta}$

$$A_{\bar{\delta}} = \sqrt{\frac{1}{k(k-1)} \sum_{i=1}^k (\delta_i - \bar{\delta})^2}$$

Os valores do parâmetro t para intervalos de confiança com dois limites são dados na tabela 5.3 [Mitchell-97]. À medida que $k \rightarrow \infty$, o valor de $t_{N,k-1}$ aproxima-se da constante Z_N . O valor ν na tabela (graus de liberdade) é relativo ao número de eventos aleatórios independentes que ocorrem para produzir o valor para a variável aleatória $\bar{\delta}$. No exemplo, o número de graus de liberdade é $k-1$.

Tabela 5.3 – Níveis de Confiança de $t_{N,\nu}$ para intervalos de confiança com dois limites

	Nível de confiança N			
	90%	95%	98%	99%
$\nu=2$	2.92	4.30	6.96	9.92
$\nu=5$	2.02	2.57	3.36	4.03
$\nu=10$	1.81	2.23	2.76	3.17
$\nu=20$	1.72	2.09	2.53	2.84
$\nu=30$	1.70	2.04	2.46	2.75
$\nu=120$	1.66	1.98	2.36	2.62
$\nu=\infty$	1.64	1.96	2.33	2.58

Quando só uma amostra limitada D_0 está disponível, podem ser utilizados muitos métodos para dividir essa amostra. Um dos métodos é descrito na tabela 5.2 em que D_0 é particionado em k subconjuntos disjuntos de igual tamanho. Neste método, cada exemplo de D_0 é usado exactamente uma vez num conjunto de teste, e $K-1$ vezes no conjunto de treino. Outro método popular consiste em escolher aleatoriamente de D_0 um conjunto de teste de pelo menos 30 exemplos, usar os restantes exemplos para treino, e repetir este processo tantas vezes quanto desejado. Este método tem a vantagem de poder ser repetido um número de vezes infinito, enquanto o anterior, de validação cruzada, é limitado pelo número total de exemplos, pelo uso de cada exemplo só uma vez no conjunto de teste e pelo uso de amostras com pelo menos 30 exemplos. Contudo, o segundo método tem a desvantagem dos seus conjuntos de exemplos de teste não serem independentes relativamente à distribuição de instâncias em D . Em contrapartida, os conjuntos de teste gerados pelo método de validação cruzada, são independentes pois cada instância é incluída num só conjunto de teste.

Podemos concluir que nenhum procedimento para comparar algoritmos de aprendizagem a partir de um número de dados limitados satisfaz todas as restrições que desejamos. No entanto, os modelos estatísticos possibilitam aproximar intervalos de confiança que podem revelar-se muito úteis nas comparações dos algoritmos de aprendizagem.

Capítulo 6

TEMPUS – Uma ferramenta de Aprendizagem Automática

6.1 Aplicação da Aprendizagem Automática aos Sistemas Eléctricos de Energia

6.1.1 Introdução

Os métodos de Aprendizagem Automática têm sido usados nos Sistemas de Energia, nomeadamente, para extrair e sintetizar informação relevante de bases de dados e para reformular esta informação numa forma apropriada, isto é, transformar as bases de dados em bases de conhecimento.

A qualidade da informação das bases de dados é fundamental. Se as bases de dados não forem representativas ou muito pequenas, a informação extraída pela Aprendizagem Automática será provavelmente inútil.

Uma outra utilidade dos algoritmos de Aprendizagem Automática consiste no uso da informação existente na base de conhecimento, para a ajuda à tomada de decisões em tempo real, ou ainda, em “off-line”, possibilitar obter novo conhecimento que permita o desenvolvimento de melhores sistemas e melhores estratégias de planeamento e de exploração dos Sistemas de Energia.

6.1.2 Aplicações por Área

Actualmente, a principal área de aplicabilidade da Aprendizagem Automática em Sistemas de Energia é a da avaliação da sua segurança, a que tem que ser realizada no âmbito das seguintes tarefas:

- Planeamento.
 - Sistemas de produção e transporte.
 - Sistemas de protecção.
 - Sistemas de Controlo.
- Exploração.
 - Calendarização das acções de manutenção.
 - Escalonamento da produção (“unit commitment”).
 - Regulação das protecções.

- Controlo de emergência.
- Treino de operadores.

No entanto, existem outras classes de problemas nos Sistemas de Energia passíveis da aplicação dos métodos de Aprendizagem Automática:

- Localização de dispositivos, seu dimensionamento e ajuste de alguns dos seus parâmetros.
- Modelização de sistemas físicos inerentes às redes eléctricas.
- Previsão do desempenho (resultado) de sistemas que são intrinsecamente incertos (nomeadamente, previsão de carga, nível de reservatórios, condições meteorológicas).
- Monitorização (desde simples instalações até de sistemas sofisticados)

6.1.3 Métodos da Aprendizagem Automática aplicados aos Sistemas Eléctricos de Energia

Os métodos de Aprendizagem Automática que têm sido mais utilizados em Sistemas de Energia são: os métodos baseados em árvores de decisão [Wehenkel-97; Wehenkel-98], as redes neuronais [Togami-95; Rayudu-97; Wehenkel-97; Wehenkel-98], os métodos estatísticos [Wehenkel-97; Wehenkel-98] (por exemplo, o algoritmo dos K -vizinhos mais próximos que é bastante utilizado) e ainda alguns métodos híbridos que combinam as funcionalidades e vantagens de outros métodos [Wehenkel-97; Wehenkel-98].

Uma característica geral de muitos dos problemas dos Sistemas Eléctricos de Energia é o grande número de atributos candidatos existentes a tratar pelos algoritmos de Aprendizagem Automática. Por este facto, é necessário resolver estes problemas de dimensão.

Actualmente, de todos os métodos de aprendizagem, o método baseado em árvores de decisão, é o que mais eficientemente trata os problemas de dimensão, enquanto as redes neuronais se tornam excessivamente lentas e o método dos K -vizinhos mais próximos perde exactidão apresentando uma maior percentagem de erros.

Os métodos baseados em árvores de decisão são os que apresentam melhor interpretabilidade, à semelhança do método dos K -vizinhos mais próximos, enquanto que as redes neuronais apresentam uma baixa interpretabilidade.

De facto, os métodos baseados em árvores de decisão permitem a exploração de grandes bases de dados, extraíndo os atributos mais relevantes e permitindo a criação de regras de fácil interpretação.

As redes neuronais não seleccionam explicitamente os atributos chave dos problemas. Isto resulta numa aplicação tipo caixa negra, que faz com que se torne difícil aos peritos avaliar e aceitar este tipo de aplicações.

Em termos de exactidão, não existe unanimidade. De facto, embora na generalidade, cada método tenha o seu campo de competência, em problemas de avaliação de segurança nos Sistemas de Energia, os métodos que exploram tolerâncias (margens) em vez de classes, obtêm geralmente uma melhor exactidão e uma avaliação mais refinada da segurança.

Todavia, o método baseado em árvores de decisão para problemas simples apresenta uma boa exactidão, o mesmo não acontecendo para problemas mais complexos.

A redução do número de erros obtidos, pelos métodos baseados em árvores de decisão, pode ser feita com a utilização posterior de outros métodos de Aprendizagem Automática, nomeadamente, as redes neuronais (método híbrido de árvores de decisão - redes neuronais). A utilização deste método híbrido possibilita, por vezes, reduzir para metade o rácio de erro obtido pelo método baseado em árvores de decisão.

As redes neuronais apresentam uma boa exactidão, tanto para problemas simples como para os mais complexos e apresentam possibilidade de tratar tolerâncias.

Como já referido, o método dos *K*-vizinhos mais próximos para problemas complexos não permite bons resultados em termos de exactidão.

Em termos de tempo de CPU, as variações são grandes. O tempo de CPU para a aprendizagem de redes neuronais é da ordem das horas, enquanto que para a construção de árvores de decisão é da ordem dos minutos. Por exemplo, a construção de uma árvore de decisão, para um dado problema, pode ser mais de 1000 vezes mais rápida do que a optimização dos pesos de uma rede neuronal.

Relativamente ao algoritmo dos *K*-vizinhos mais próximos, o tempo de CPU é bastante elevado.

Assim, o método baseado nas árvores de decisão deve ser usado de uma forma interactiva em problemas menos complexos, dado o seu bom desempenho em termos de exactidão e tempo de CPU. Em problemas mais complexos (como é o caso de muitos problemas dos Sistemas de Energia), os seus resultados em termos de exactidão não são os melhores. Nestes casos, de forma a obter-se melhores resultados em termos de exactidão, podem ser usadas as redes neuronais como um bom método alternativo ou mesmo complementar das árvores de decisão.

De facto, parte das publicações recentes na área da Aprendizagem Automática debruçam-se sobre métodos híbridos, também chamados de multi-estratégia. Estas abordagens combinam métodos elementares da Aprendizagem Automática de forma a extraírem o máximo de informação possível das bases de dados. Uma destas abordagens tem sido aplicada no contexto de problemas de Sistemas de Energia e utiliza uma árvore de decisão e uma rede neuronal no seu algoritmo.

Outras abordagens híbridas foram também desenvolvidas conjuntamente com árvores de decisão, nomeadamente, com o método dos *K*-vizinhos mais próximos e com conjuntos difusos ("fuzzy sets"). No entanto, o desempenho conseguido por estes métodos híbridos não atingiram o obtido pelo método híbrido árvores de decisão - redes neuronais.

6.1.3.1 Método Híbrido Árvore de Decisão - Rede Neuronal

A abordagem híbrida Árvore de Decisão - Rede Neuronal combina as vantagens de ambos os métodos e contorna as limitações destes, utilizados em separado.

A Árvore de Decisão é usada para produzir um modelo interpretável das relações entre os atributos e a informação de "output". O poder de mapeamento não linear das redes neuronais é depois explorado para ampliar a classificação discreta realizada pela árvore.

Esta abordagem segue os seguintes passos:

- Construção da árvore de decisão, a partir de estados pré-classificados (exemplos) do sistema de energia.
- Reformulação (transformação) da informação proveniente da árvore de decisão para uma rede neuronal de quatro níveis.
- Ajuste dos pesos na rede neuronal com base no conjunto de aprendizagem e na informação de tolerâncias de segurança, de forma a aumentar a segurança da classificação e a transformar a classificação discreta da árvore numa classificação contínua, com tolerâncias de segurança.

Entre as possibilidades de reformulação da árvore de decisão numa rede neuronal, existe uma abordagem [Set90] que constrói uma rede neuronal com quatro níveis:

- Nível de "input" contém uma unidade por atributo seleccionado e testado pela árvore de

decisão. Os seus níveis de activação correspondem aos valores dos atributos do estado presente.

- Nível de Teste que contém uma unidade por cada nó de teste da árvore de decisão. Cada unidade do nível de teste é ligada à unidade do nível de “input” correspondente ao atributo testado.
- Nível “E Lógico” (ANDing Layer) que contém uma unidade por cada nó terminal da árvore de decisão. Cada unidade neste nível é ligada às unidades do nível de teste correspondentes aos nós de teste localizados no caminho desde o nó raiz (topo) até ao nó terminal da árvore.
- Nível “OU Lógico” (ORing Layer) que contém uma unidade por cada classe da árvore de decisão, ligada às unidades do nível “E Lógico” correspondentes aos nós terminais da árvore em que a classe em questão é a classe maioritária (com maior percentagem).

Comparando este método com as redes neuronais “puras”, verifica-se que permite reduzir significativamente o tempo necessário para a etapa de treino, já que a estrutura da rede neuronal é directamente derivada da árvore de decisão. Por outro lado, a tarefa tediosa da optimização da estrutura da rede é evitada. Assim, os tempos de treino da rede neuronal reduzem-se significativamente (cerca de 10 vezes mais), melhorando também a exactidão da árvore de decisão.

6.1.4 Resultados obtidos pelos Métodos de Aprendizagem Automática num problema de Sistemas Eléctricos de Energia

A título de exemplo, esta secção inclui uma análise aos resultados obtidos num problema do sistema de energia da Hidro-Québec [Wehenkel-98], pela aplicação dos métodos mais eficientes de Aprendizagem Automática no estudo de problemas dos Sistemas Eléctricos de Energia.

Para a análise deste problema, foram usados o método baseado em árvores de decisão, as redes neuronais e o algoritmo dos *K*-vizinhos mais próximos.

O primeiro estudo consiste no problema de avaliação da estabilidade transitória (“preventive transient stability assessment”) do sistema Hidro-Québec.

Por uma questão de simplicidade, foi analisada apenas uma parte de todo o sistema.

Uma base de dados, composta por 12497 estados normais de exploração foi gerada aleatoriamente. Para cada estado, foi realizada a correspondente classificação em estável/instável, obtendo-se 3938 estados estáveis e 8559 estados instáveis, entre os quais 393 eram marginalmente instáveis e 8166 eram completamente instáveis.

Para descrever os estados de exploração e com o intuito de caracterizar a sua estabilidade, foram consideradas os seguintes atributos:

- Trânsitos de potências nas linhas importantes.
- Energia produzida nas centrais.
- Dispositivos utilizados para correcção da tensão.
- Informação topológica, indicando, para cada linha, se está ou não em funcionamento no instante anterior à ocorrência do defeito.

Este conjunto, composto por 67 atributos foi determinado com a ajuda de um perito em estabilidade transitória da Hidro-Québec.

No estudo realizado, apoiado em métodos de Aprendizagem Automática, foram considerados 10.000 estados da base de dados e 87 atributos candidatos (pela inclusão de quatro combinações lineares de atributos e alguns atributos criados pela combinação de outros atributos).

O método baseado em árvores de decisão, construiu uma árvore de decisão com 57 nós de teste e 58 nós terminais. Dos 87 atributos em estudo, foram identificados 24 atributos considerados relevantes, a partir do ganho de informação por eles obtido. Verificou-se que 51.8% do ganho de informação total da árvore, foi possibilitado pelo atributo usado no nó raiz da árvore (um dos atributos formados pela combinação linear de atributos), e cerca de 40% por 4 atributos que envolvem a topologia e dois atributos relativos a transições de potências.

Para avaliar a capacidade de generalização da árvore, esta foi testada com 2500 estados da base de dados não usados, obtendo-se um rácio de erro de 4.3%. Dos 1622 estados completamente instáveis, só 30 foram classificados como estáveis, atingindo-se 1.85% de erros “perigosos”. Por outro lado, 23 dos estados marginalmente instáveis foram classificados como estáveis e 52 estados estáveis foram classificados como instáveis.

A mesma base de dados foi explorada por uma rede neuronal com um nível “escondido” de 20 unidades, a partir dos mesmos 10000 estados. A rede neuronal obteve um rácio de erro de 2.4%.

Em termos de recursos computacionais, são de referir os seguintes tempos: uma hora para a construção da árvore de decisão e um segundo para testar os 2500 estados de teste; 60 horas para aprender os pesos da rede neuronal e 10 segundos para 2500 estados de teste.

A tabela 6.1 mostra os resultados de exactidão obtidos com o método dos *K*-vizinhos mais próximos para duas classes diferentes [Wehenkel-98]. A primeira linha de resultados, corresponde ao uso de todos os atributos (67). Os resultados são muito piores do que os obtidos com o método baseado na árvore de decisão e com a rede neuronal. O resultado para $k=7$ possibilita o melhor resultado. A segunda linha de resultados, corresponde ao uso dos 24 atributos com melhor ganho de informação obtidos pela árvore de decisão. Neste caso, verifica-se uma redução significativa do rácio de erro, mas fica-se bastante aquém dos resultados obtidos pelos outros dois métodos. Neste exemplo, é notória a grande sensibilidade do algoritmo dos *K*-vizinhos mais próximos relativamente aos atributos usados no cálculo da distância entre as instâncias (estados normais de exploração).

Tabela 6.1 – Resultados da exactidão obtida pelo Método dos K-Vizinhos mais próximos

K	1	3	5	7	9
67 atributos candidatos					
% de erro	12.58	11.33	10.53	10.21	10.25
24 atributos candidatos					
% de erro	6.93	6.73	6.13	6.13	6.61

Neste exemplo, verifica-se que as redes neuronais obtêm o menor rácio de erro entre os três métodos de aprendizagem utilizados, no entanto, o tempo de CPU gasto para aprender os pesos da rede neuronal é muito superior (60 vezes) ao tempo necessário pelo método baseado em árvores de decisão para construir a árvore.

Os resultados obtidos neste problema, apesar de não serem os mesmos obtidos noutros problemas de Sistemas de Energia, são representativos de grande parte deles.

6.2 SPARSE

6.2.1 Introdução

As economias modernas estão cada vez mais dependentes do uso da energia eléctrica, verificando-se a nível económico uma tendência constante no sentido de reduzir os custos de exploração das redes eléctricas ao mesmo tempo que os níveis de exigência relativamente à qualidade da energia eléctrica são cada vez maiores.

O grande volume de informação em jogo, as apertadas limitações temporais impostas e o enorme impacto do bem produzido pela indústria em causa requerem o uso de técnicas muito elaboradas para o uso de um sistema pericial num meio industrial real.

Na década de 80, começaram a aparecer aplicações baseadas em Inteligência Artificial, especialmente Sistemas Periciais, para apoiar os operadores dos C/C nestas situações [Wollenberg-86; Talukdar-86; Kirschen-89; Kirschen-92; CIGRE-93].

O SPARSE¹ [Vale- 99] é um Sistema Baseado em Conhecimento para apoiar os operadores dos Centros de Controlo e Condução da Rede Nacional de Transporte de Energia Eléctrica na análise de incidentes e na reposição de serviço. Este sistema encontra-se em funcionamento no Centro de Condução de Vermoim, permitindo o processamento, em tempo real, da informação proveniente do sistema SCADA².

A exploração da rede eléctrica na qual o SPARSE está integrado (à semelhança de outras redes) tem uma grande complexidade devido não só aos respectivos equipamentos e configurações como também ao contexto social e económico em que as companhias de electricidade actualmente exercem a sua actividade. De facto, por razões de optimização económica, as redes eléctricas são actualmente exploradas muito próximo dos seus limites, requerendo um controlo muito preciso e eficiente.

Por outro lado, a exploração da rede baseia-se na existência de Centros de Controlo e Condução (CC), onde é disponibilizada informação, em tempo real. Esta informação provém, essencialmente, de um sistema SCADA, o qual permite, através do uso de sistemas de comunicação eficientes, o conhecimento, à distância, do que se passa na rede.

A exploração da rede eléctrica é, pois, feita de modo remoto, sem a presença, na maior parte das respectivas instalações, de meios humanos. As redes associadas a cada CC cobrem geralmente uma área geográfica bastante grande, sendo as ordens relativas à respectiva exploração emitidas pelos operadores dos CC, através de telecomando.

Esta exploração requer a utilização de equipamentos sofisticados de telecomunicação, quase sempre privativos das próprias empresas de electricidade, e também uma elevada capacidade, a nível informático, para processar e gerir toda a informação disponível.

Apesar dos avanços ocorridos na automatização e informatização das tarefas de controlo e exploração das redes, os operadores humanos continuam a desempenhar um papel crucial na exploração das redes eléctricas actuais. A estes operadores cabe tarefa de tomar decisões que condicionam a exploração da rede, nomeadamente a limitação das consequências dos incidentes ocorridos.

As decisões dos operadores dos CC baseiam-se na informação recebida e no seu conhecimento teórico e empírico acerca da rede e dos equipamentos nela envolvidos. Em situações de exploração normais, a informação disponível é geralmente adequada para que os operadores desempenhem as suas funções em boas condições. No entanto, em situações de incidente, o

¹ SPARSE – Sistema Pericial de Apoio à análise de avarias e à Reposição do Serviço

² SCADA - Supervisory Control AND Data Acquisition

volume de informação disponível pode dificultar a tomada de decisões de forma rápida e eficiente. Apesar de geralmente existirem um grande número de aplicações informáticas nos CC, os operadores não dispõem de ferramentas de apoio à decisão adequadas. As decisões têm que ser tomadas rapidamente, de modo a limitar, tanto quanto possível, as consequências dos incidentes, limitando as zonas afectadas.

Numa rede de dimensões médias, em caso de ocorrência de um incidente, o sistema SCADA pode gerar várias centenas de alarmes por segundo, provocando uma avalanche de informação. Por maior que seja a experiência dum operador, dificilmente ele poderá actuar de forma adequada numa situação dessas.

Verificou-se, em consequências destas constatações, a conveniência de passar a atribuir prioridades aos alarmes, de filtrar alarmes redundantes e de melhorar a forma como os alarmes eram apresentados. As técnicas tradicionais revelaram-se insuficientes para resolver estes problemas, por razões que têm a ver com a complexidade deste tipo de processamento.

Dada a natureza essencialmente simbólica do raciocínio utilizado pelos operadores nestas situações e a grande importância que tem o conhecimento empírico, desenvolveram-se aplicações baseadas em Inteligência Artificial, especialmente Sistemas Periciais, para apoiar os operadores dos CC nestas situações.

A utilização prática destas novas ferramentas informáticas, contudo, não se revelou fácil devido às exigências que lhes são colocadas. O processamento de alarmes é uma área com problemas específicos, que não era muito comum serem tratados por aplicações de Inteligência Artificial. De facto, é necessário tratar um grande volume de informação em tempo real, actualizar dinamicamente a base de conhecimento, e recorrer a raciocínio temporal complexo e raciocínio não-monotónico.

Paralelamente, a actividade das empresas eléctricas exige sistemas fiáveis, testados, suficientemente robustos para dar indicações mesmo nas situações mais adversas.

De referir que, tem sido necessário um grande esforço, por parte das equipas envolvidas neste tipo de projectos, para encontrar soluções adequadas que possam efectivamente proporcionar um avanço significativo na exploração das redes eléctricas.

6.2.2 Arquitectura do SPARSE

A arquitectura do SPARSE é constituída por módulos que geralmente integram um sistema pericial, tal como a Base de Conhecimento, o Motor de Inferência, a Interface com o Utilizador e o Módulo de Explicações [Vale-99a]. O SPARSE inclui ainda um Módulo de Pré-Processamento que permite promover a interface entre o sistema pericial e as aplicações SCADA, um Módulo Tutor, um Módulo de Verificação e Validação, um Assistente de Aquisição de Conhecimento e outro de Actualização do Conhecimento. Esta arquitectura é ilustrada na figura 6.1.

O SPARSE está instalado no Centro de Condução de Vermoim numa estação de trabalho DECstation³ 5000/240, com sistema operativo Ultrix. Esta DECstation está ligada através de uma rede local (LAN) de configuração duplicada com dois μ VaxII. Enquanto uma das máquinas μ VaxII está em funcionamento efectivo (on-line) suportando as funções SCADA e de controlo e condução da rede, a outra está pronta a entrar em funcionamento, em caso de falha da primeira ("hot standby").

Estas duas máquinas μ VaxII estão em funcionamento desde 1989, apresentando alguns problemas de sobrecarga de utilização, pelo que se decidiu instalar o sistema pericial noutra máquina. Isto permitiu causar uma interferência mínima nas actividades do CC, aquando da instalação do sistema pericial e dos respectivos testes de validação.

³ DECstation é uma marca registada da Digital Equipments Corporation

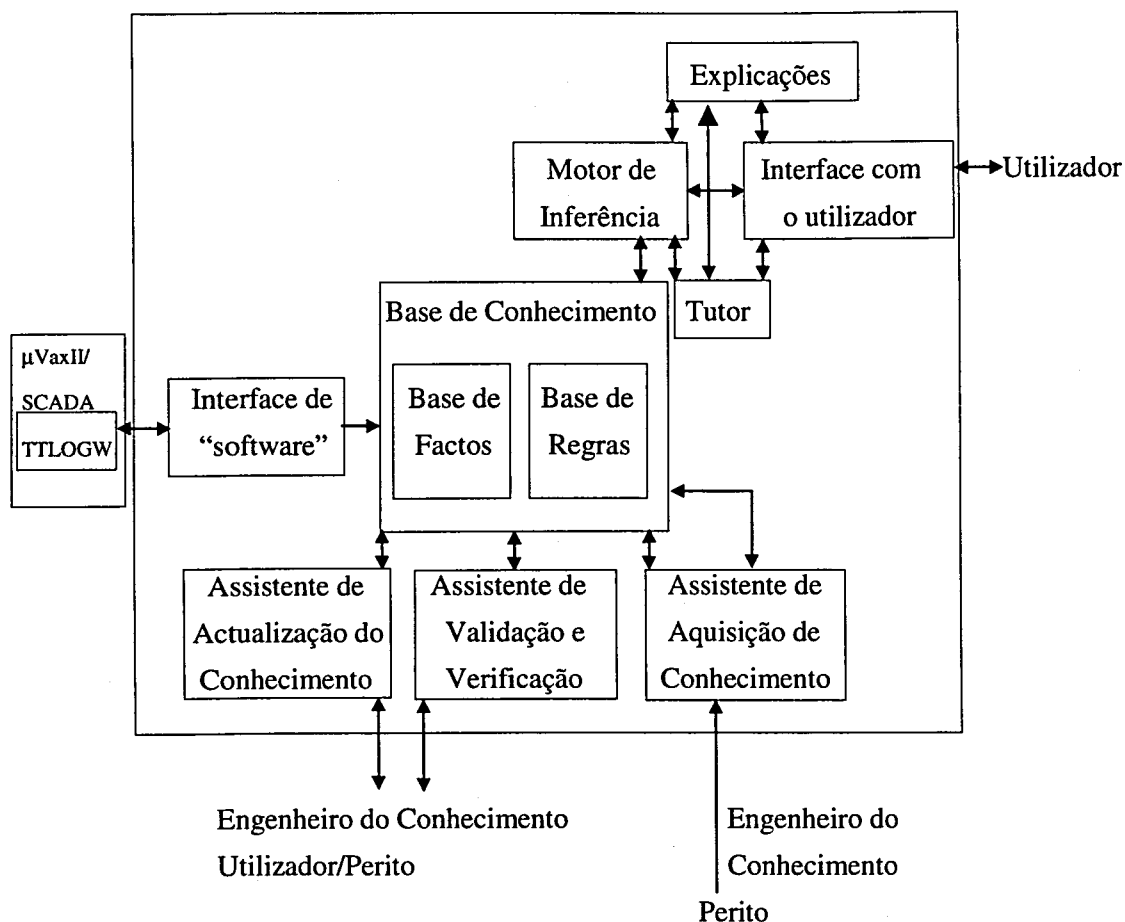


Figura 6.1 - Arquitectura do SPARSE

A aplicação TTLOGW foi desenvolvida pela REN (Rede Eléctrica Nacional, S.A.) e instalada nos microVAX II, para aquisição de informação (mensagens) em tempo real do SCADA e envio para o SPARSE dessas mensagens [Rosado-93].

O SPARSE foi essencialmente desenvolvido em PROLOG, existindo alguns módulos desenvolvidos em linguagem C, relacionados com tarefas de pré-processamento de mensagens e interligação com o sistema gráfico. A opção pela linguagem C para estes módulos deve-se ao facto de estes não estarem tipicamente ligados à Inteligência Artificial e ao bom desempenho que pode ser atingida com programas escritos nessa linguagem.

6.2.3 Objectivos do SPARSE

Os objectivos principais do SPARSE são auxiliar os operadores no centro de condução a interpretar, rápida e correctamente, as mensagens geradas pelo sistema SCADA, em situações de incidente e dar conselhos que levem à rápida reposição do serviço na rede, de forma a minimizar os custos do não fornecimento de energia eléctrica [Vale-99c].

De forma a poder atingir estes objectivos, o SPARSE tem as seguintes potencialidades em tempo real:

- Tem conhecimento, em tempo real, da topologia da rede, incluindo os estados (aberto

ou fechado) dos aparelhos de manobra e o modo de comando de cada instalação (manual ou automático, local ou remoto).

- Informa os operadores do início de um incidente.
- Disponibiliza conclusões relativas a um incidente, por ordem cronológica.
- Disponibiliza uma representação gráfica da rede, salientando a informação importante sobre as instalações e os componentes envolvidos no incidente.
- Dá conselhos sobre a reposição do serviço, indicando quais os casos que serão tratados por automatismos (Operadores Automáticos – OPA), que acções devem ser tomadas pelos operadores e que acções devem ser evitadas.
- Disponibiliza as acções tomadas, indicando quais os agentes que as executaram (OPA, operadores do CC ou operadores nas subestações).
- Informa os operadores sobre o fim da situação de incidente.

6.2.4 Base de Conhecimento

O conhecimento dos peritos necessário à interpretação das mensagens SCADA é de difícil obtenção. As dificuldades em compilar esse conhecimento são as dificuldades típicas da recolha do conhecimento dos peritos para um sistema pericial.

Como alternativa a este processo empírico existe a possibilidade de utilização de modelos. Este método permite o diagnóstico de defeitos nunca antes ocorridos a partir do conhecimento contido no modelo sobre as relações entre os vários componentes do sistema. Contudo, quando o número de elementos do sistema a modelizar é elevado, o tempo gasto no diagnóstico pode tornar o processo impraticável.

Os sistemas que se baseiam em regras traduzem a experiência acumulada de diagnóstico, apesar de apenas poderem tratar situações que já se conhecem, permitem diagnosticar defeitos em sistemas complexos, numa pequena fracção do tempo usado pelos sistemas baseados em modelos. Daí o processamento de alarmes se basear principalmente em sistemas usando o conhecimento empírico dos peritos do domínio.

De notar que no processamento de alarmes é necessário considerar a ordem de ocorrência dos eventos, o tempo que medeia entre eventos consecutivos e a ocorrência de eventos de um dado tipo num dado intervalo de tempo.

O SPARSE detém conhecimento de vários tipos, nomeadamente [Vale-94]:

- Conhecimento sobre os componentes (por exemplo, linhas de transporte e subestações) e a topologia da rede.
- Conhecimento sobre os alarmes que são gerados (listas de alarmes).
- Conhecimento sobre o sistema de transmissão da informação.
- Conhecimento relativo à interpretação dos alarmes.

O conhecimento dos dois primeiros tipos indicados está incluído na Base de Factos, sob a forma de factos PROLOG.

As mensagens SCADA são convertidas por um pré-processador desenvolvido em linguagem C para factos PROLOG que, por sua vez, são enviados para o SPARSE, que os processa de seguida.

O restante conhecimento necessário ao funcionamento do SPARSE (conhecimento sobre o sistema de transmissão de informação e de interpretação de alarmes) está incluído na Base de Regras. Este conhecimento inclui conhecimento de natureza técnica mas também conhecimento de natureza empírica.

O conhecimento técnico permite modelizar o funcionamento da rede eléctrica e do sistema de transmissão de informação. O conhecimento empírico foi obtido em entrevistas e reuniões com os peritos da REN.

A construção da Base de Regras de SPARSE baseou-se na utilização de uma linguagem quase natural. O formato estabelecido para as regras teve em conta as necessidades do sistema, relativamente à consideração do raciocínio temporal e não monótono. Pretendeu-se ainda implementar um formato de regras que facilite a manutenção e a verificação do conhecimento do sistema e que seja apropriado ao fornecimento de explicações.

Uma regra pode-se dividir em LHS e RHS [Vale-99c]. O RHS é constituído por uma identificação da regra e um bloco (lista) ou a disjunção de vários blocos (listas). Dentro de cada bloco as premissas estão ligadas por conjunções. O RHS é uma lista de acções ou conclusões intermédias. Estas acções podem ser dos seguintes tipos:

- Geração de factos (conclusões) que são inseridos na Base de Factos.
- “Eliminação” de factos que não tenham mais importância para o processo de inferência.
- Interacção com a interface do utilizador.

O formato normalizado de uma regra do SPARSE é do tipo considerado no seguinte exemplo:

REGRA X : 'Descrição da regra X' :

```
[
  [ P1 at T1 e P2 e P3 at T3 e C1(T1,T3) ]
  ou
  [ P1 at T1 e P2 e P4 at T4 e C2(T1,T4) ]
  ou
  ...
]
```

==>

```
[
  cria_facto( Nf1,Tnf1),
  retira_facto( P2, T2 ,Ttrp2),
  escreve_mens( MensUtil )
].
```

onde:

P_n : com $n \in \{1,2,3,4\}$. Mensagem SCADA ou facto de nível superior com tempo igual a T_n

$C1$ e $C2$: condições temporais entre $T1$ e $T3$ e entre $T1$ e $T4$ respectivamente

cria_facto(Nf1,Tnf1):

criação de um novo facto $Nf1$ com tempo $Tnf1$ na Base de Factos. Este novo facto $Nf1$ é, geralmente, dependente de alguma(s) premissa(s) da regra de produção. É gerada uma conclusão intermédia.

retira_facto(P2, T2 ,Ttrp2):

alteração do estado do facto $P2$ com tempo igual a $T2$ e tempo de alteração $Ttrp2$; esta alteração dita que $P2$ deve ser guardado de outra forma na Base de Conhecimento, tornando-o inactivo para processo de inferência

escreve_mens(MensUtil):

disponibiliza ao utilizador a mensagem *MensUtil*; esta mensagem é uma linha de texto formada com elementos constantes e/ou qualquer informação contida na regra.

6.2.5 Motor de Inferência

O motor de inferência do SPARSE foi desenvolvido integralmente em linguagem Prolog com o objectivo de aproveitar as características inerentes desta linguagem, de forma a solucionar os seguintes requisitos:

- Manipulação de grandes quantidades de informação.
- Processamento em tempo real.
- Raciocínio temporal complexo.
- Suporte de inferência com mecanismos não monotónicos.

O motor de inferência utiliza uma estratégia de encadeamento directo orientado pelos dados (“data driven”), que, no caso do SPARSE, são os alarmes. Não havendo informação a tratar, o SPARSE fica inactivo.

Quando é recebida informação relativa aos alarmes, esta é guardada, sendo de seguida usada juntamente com toda a informação de que o sistema já dispõe, no sentido de gerar nova informação. Esta nova informação sofrerá o mesmo tratamento da anterior, até que toda a informação que seja possível extrair seja obtida.

De forma a melhorar o desempenho e resolver as questões relacionadas com o raciocínio temporal, é utilizado um mecanismo de selecção das regras a disparar mediante a chegada de alarmes ou de conclusões intermédias produzidas pelo próprio sistema. O motor de inferência para fazer a selecção das regras a disparar utiliza então meta-conhecimento existente na Base de Conhecimento.

No SPARSE [Vale-99c], as mensagens recebidas são convertidas em factos PROLOG e guardadas numa Base de Factos. Cada novo facto origina o escalonamento para disparo das regras de produção que lhe estão atribuídas pelo meta-conhecimento.

O facto: *disparar(NF,NR,T1,T2)*.

indica que a regra número *NR* deve ser disparada a partir do instante *T1* e até ao instante *T2*, devido à chegada do facto número *NF*.

O meta-conhecimento utilizado especifica, para cada tipo de facto recebido (alarme ou conclusão intermédia), quais as regras que devem ser utilizadas. Além disso, especifica também um intervalo de tempo para a regra poder ser disparada. Assim, a introdução destes dois instantes associados a cada regra permite modelizar as seguintes situações:

- Aguardar, durante um certo tempo, a chegada de outros factos que sejam premissas da regra a disparar. Neste caso, poderão existir duas situações:
 - o atraso introduzido corresponde a uma tolerância permitida à chegada das mensagens (podem-se considerar, por exemplo, atrasos de transmissão),
 - o atraso introduzido corresponde ao tempo após o qual deverão chegar outras mensagens para poderem ser consideradas relacionadas com o mesmo evento.
- Aguardar um certo tempo durante o qual não devem chegar outros factos que aparecem negados nas premissas da regra a disparar

O formato genérico dos factos relativos ao meta-conhecimento sobre escalonamento para disparo de regras é apresentado de seguida:

facto_dispara_regras(*F*, [(*R1*, *TiR1*, *TfR1*), (*R2*, *TiR2*, *TfR2*), ...]).

em que :

F: Facto que provoca o escalonamento

R1: 1ª regra a ser escalonada para disparo

TiR1: Tempo (relativo ao instante actual) a partir do qual deve ser tentado o disparo da regra *R1*

TfR1: Tempo (relativo ao instante actual) a partir do qual deixa de fazer sentido tentar o disparo da regra *R1*

R2: 2ª regra a ser escalonada para disparo

TiR2: Tempo (relativo ao instante actual) a partir do qual deve ser tentado o disparo da regra *R2*

TfR2: Tempo (relativo ao instante actual) a partir do qual deixa de fazer sentido tentar o disparo da regra *R2*.

Esta informação é guardada numa estrutura que suporta o escalonamento de um número arbitrário de qualquer tipo de regra. Considere-se *R* o conjunto de regras contidas na estrutura referida. Quando existem regras para disparo em *R*, o motor de inferência vai tentar dispará-las.

As conclusões geradas pelas regras de produção serão tratadas como nova informação (novos factos) do sistema, podendo provocar novo escalonamento de regras, reiniciando o processo.

Quando se obtém uma conclusão com relevância para o utilizador do sistema, esta é enviada para o interface do utilizador, para ser visualizada.

De seguida, o sistema verifica se existem em *R* regras cujo instante final de tentativa de disparo é excedido pelo instante actual do sistema. As regras nestas condições são removidas da estrutura.

O SPARSE, à semelhança de outros Sistemas Baseados em Conhecimento na área dos Sistemas de Energia tem um comportamento dinâmico. A sua Base de Conhecimento e o seu Motor de Inferência foram desenvolvidos de forma a permitirem um raciocínio não monotónico. Por isso, os alarmes processados permitem inferir conhecimento que mais tarde pode ser retirado, caso exista informação que anule uma determinada conclusão obtida anteriormente. Estes factos que já não são válidos para o sistema não são apagados sendo armazenados como “factos antigos”. Este armazenamento serve para mais tarde poder ser utilizado no módulo de explicações e de Aprendizagem Automática.

Depois de todas as regras que podiam ser disparadas terem sido consideradas, o sistema irá processar os factos que não deram origem à selecção de nenhuma regra.

Esta metodologia para tratamento das questões temporais, por introdução de atrasos no disparo de certas regras, permite introduzir esses atrasos, sem implicar a paragem do processo de inferência.

Os atrasos introduzidos são inerentes ao disparo das regras, podendo o Motor de Inferência continuar a disparar as regras que tem em lista de espera, garantindo o processamento em tempo real de toda a informação que chega ao sistema.

6.2.5.1 Raciocínio Temporal

As questões temporais revelam-se da maior importância para o processamento inteligente de mensagens de alarme nos Sistemas Eléctricos de Energia [Vale-99c]. O Sistema Pericial tem que ser capaz de compreender a evolução temporal dos eventos, já que dessa evolução depende a interpretação correcta desses mesmos eventos. O sistema deve ser capaz de reconstituir os eventos, a partir dos eventos que lhe chegam, até em situações em que a ordem destes não esteja cronologicamente correcta. Daí que o processamento inteligente de alarmes envolva o tratamento de questões temporais de natureza diversa.

Para ajudar a resolver esta questão, às mensagens é associada uma data e uma hora, que constituem elementos fundamentais no respectivo processamento.

Como resultado da interpretação das listas de mensagens pretende-se obter uma lista cronológica dos acontecimentos que lhe deram origem. Esta interpretação depende não só das mensagens individuais que constituem a lista mas também da sequência dessas mensagens e da sua distribuição ao longo do tempo. As conclusões podem depender não só das mensagens anteriores mas também das mensagens que poderão ser geradas por acontecimentos posteriores. Assim, para se chegar a uma determinada conclusão, por vezes, é necessário esperar um certo tempo, para verificar se entretanto chega alguma mensagem que leve a uma conclusão diferente. Não é de desprezar também o facto de que a hora que acompanha cada mensagem possa conter erros.

Atendendo a tudo o que foi referido, facilmente se pode concluir que a tarefa de interpretar as listas de mensagens obriga a desenvolver um raciocínio fortemente relacionado com questões temporais, ou seja, um raciocínio temporal.

De notar que a abordagem do SPARSE no processamento de alarmes corresponde a processamento em tempo real. Daí que o raciocínio efectuado não deverá ser demasiado complexo nem profundo de modo a permitir a rapidez necessária ao processamento.

O conhecimento do sistema de geração de mensagens, e do correspondente sistema de transmissão de informação, revela-se de importância fundamental para a interpretação das mensagens recebidas nos centros de controlo e condução. No entanto, o desenvolvimento de um sistema inteligente vem levantar novas questões. O desenvolvimento de um sistema deste tipo funciona realmente como pretexto para aprofundar o conhecimento existente, para aglutinar conhecimento disperso e para abrir caminho a modificações necessárias e desejáveis.

Para explicar a forma como a informação temporal que está incluída nas mensagens SCADA é tratada, considere-se a mensagem seguinte:

25-MAR-2000 01:37:04:980 SVC 610 VISO -DJ DISJUNTOR 1 0

O pré-processor converte esta mensagem no facto PROLOG:

facto(404,mensagem(25-MAR-00,01:37:04:980,[SVC,610,[VISO,0,DISJ]],DISJUNTOR,10),t(204946624,980))

Este facto é uma instanciação da forma geral:

facto(N,F,T)

usada pelo SPARSE em que,

- *N* é o número de série atribuído ao facto.

- F corresponde a uma mensagem recebida ou a uma conclusão obtida pelo SPARSE.
- T é o instante absoluto correspondente ao $facto(t(204946624,980))$.

No caso de mensagens, F tem a seguinte forma:

mensagem(Data,Hora,[Instalação,Painel,Descrição],Evento,Estado)

em que *Descrição* é constituída por uma lista que pode ter vários elementos (neste caso, inclui a designação da linha – VISO, o numero da linha – 0, e informação complementar DISJ).

O instante correspondente ao $facto$ é representado de duas formas distintas:

- A data (25-MAR-00) e a hora (01:37:04:980) na sua forma habitual, usada para a interface com o utilizador.
- Uma estrutura $(t(204946624,980))$, correspondente a esse instante, calculado como o número de segundos decorridos desde 1 de Janeiro de 1992 e a correspondente fracção de milésimas de segundo, sendo esta estrutura usada na comparação temporal de factos e nos cálculos relativos às questões temporais, de modo a melhorar a eficiência do raciocínio.

O sistema considera o instante de tempo como o elemento temporal básico.

O motor de inferência do sistema prevê um atraso de n centésimos de segundos no disparo das regras. A representação deste atraso é conseguido com a introdução no LHS da regra de um instante de tempo associado aos factos e pela introdução de condições temporais.

O SPARSE, para analisar as condições temporais dispõe de um conjunto de predicados que definem relações temporais.

6.2.5.2 Raciocínio Não Monotónico

Nos sistemas de processamento de mensagens de alarme, como estas dizem respeito a eventos no sistema eléctrico, a veracidade dos factos evolui de forma dinâmica [Vale-99c]. Por isso, um facto que é verdadeiro num dado instante poderá deixar de o ser no instante seguinte. Por exemplo, um novo facto relativo ao estado de um órgão de manobra (disjuntores e seccionadores) pode anular um existente anteriormente. Este facto pode vir alterar as conclusões anteriores do sistema.

Estes sistemas de processamento de mensagens de alarme necessitam, por isso, de usar raciocínio não monotónico, fazendo evoluir a respectiva Base de Conhecimento de forma dinâmica.

Os factos existentes na Base de Factos que são, em consequência do sucesso de uma regra, considerados falsos são retirados desta. No entanto, estes são guardados com o formato "*facto_antigo*", podendo ser usados posteriormente, por exemplo, no fornecimento de explicações e na Aprendizagem Automática.

Considerando, por exemplo, a seguinte mensagem relativa à abertura de um disjuntor, em consequência de um disparo:

25-FEV-1999 22:34:15.781 SPR 636 ALFAR.1/TAV -DJ DISJUNTOR 1 0

o sistema cria o seguinte facto:

facto(7334,disj(25-FEV-99,22:34:15.781, SPR,636,_,ds,aberto),t(173226855,781))

e retira o *facto* existente na Base de Conhecimento sobre o último fecho do mesmo disjuntor:

facto(1497,disj(25-FEV-99,19:08:57, SPR,636,_,prot,fechado),t(173214537,0))

e cria, ainda, o seguinte "*facto_antigo*":

facto_antigo(1497,disj(25-FEV-99,19:08:57,SPR,636,_,prot,fechado),t(173214537,0),t(173226855,781)).

O último argumento de *facto_antigo* (*t(173226855,781)*) refere-se ao instante em que este *facto* deixa de ser verdadeiro.

6.2.5.3 Manutenção do Conhecimento

O uso eficiente de um Sistema Pericial num ambiente industrial põe alguns problemas que devem ser resolvidos de forma a poder continuar a usar o Sistema Pericial a longo prazo. Estes problemas incluem a manutenção do conhecimento. No caso do SPARSE [Vale-99a], isto levou ao desenvolvimento de um Assistente de Manutenção do Conhecimento. A actualização da Base de Conhecimento no SPARSE é realizada de uma forma automática sempre que possível. Toda a informação que é comum a outras aplicações é automaticamente actualizada. A manutenção da informação que é específica ao SPARSE é assistida por um conjunto de ferramentas especialmente desenvolvidas para esse propósito (Módulo de Actualização da Base de Conhecimento).

Quando uma alteração na rede é detectada (por exemplo, a introdução de uma nova linha) este módulo cria e envia um "ícon" para o écran indicando aos operadores que existem alterações que têm que ser realizadas. Este módulo possibilita o máximo de assistência possível aos operadores requerendo destes o mínimo de informação possível e disponibilizando uma ferramenta de desenho com um ambiente amigável para a introdução de alguns tipos de informação.

O SPARSE tem, pois, um Assistente de Manutenção do Conhecimento que permite aos operadores a resolução dos problemas de actualização da Base de Conhecimento. Muitos dos problemas dos Sistemas Baseados em Conhecimento para os Centros de Controlo e Condução devem-se à falta de ferramentas de assistência aos operadores na actualização da Base de Conhecimento.

6.2.6 Interface com o Utilizador

A interface com o utilizador é um constituinte vital das aplicações que funcionam em tempo real, especialmente em aplicações críticas, como é o caso do SPARSE [Vale-98].

De facto, uma interface desadequada pode comprometer o funcionamento de qualquer sistema deste tipo. Contrariamente, uma interface de boa qualidade facilita a interacção dos utilizadores com o sistema.

As necessidades específicas dos utilizadores da interface gráfica foram tomadas em linha de conta no desenvolvimento desta. Por exemplo, algumas possibilidades, consideradas geralmente bastante interessantes, tais como o uso de várias janelas sobrepostas, a abertura e fecho de janelas pelo utilizador e o redimensionamento dessas janelas, nem sempre é bem aceite pelos operadores dos CC.

O esforço requerido a estes operadores, em situações de emergência, para manipulação da interface deve ser mínimo e a informação mais urgente e mais relevante deve-lhes ser apresentada automaticamente. Em situações de funcionamento normal da rede eléctrica, a interface com o utilizador deve ser mais flexível e permitir que os operadores tenham um amplo leque de escolhas.

A interface do SPARSE tem características adaptativas que permitem a respectiva adaptação ao tipo de situação da rede e ainda ao tipo de utilizador.

Esta interface inclui três níveis de detalhe distintos:

- Nível 1: diagrama da rede, excluindo informação menos relevante
- Nível 2: diagrama da rede, incluindo informação adicional relativamente ao nível 1
- Nível 3: esquemas das subestações da rede.

No entanto, os operadores podem ainda configurar o tipo de informação apresentada nos vários níveis existentes.

Em caso de ocorrência de um evento, a interface pretende apresentar a informação mais relevante de modo a facilitar a tomada de decisões e diminuir o tempo necessário à reposição do serviço. A interface do SPARSE apresenta automaticamente a zona afectada pelo incidente, aparecendo os componentes e as instalações envolvidas no incidente assinaladas (geralmente em modo intermitente).

Se o incidente envolver vários componentes da rede, o SPARSE agrupa-os em tantas áreas quantas as necessárias. A imagem que é automaticamente apresentada diz respeito ao incidente mais recente, podendo os operadores seleccionar outras das áreas de incidente para serem apresentadas.

6.2.7 Explicações

Nos Sistemas Periciais de tempo real, como é o caso do SPARSE, a introdução de funcionalidades de explicação impõe várias restrições importantes ao seu funcionamento, nomeadamente restrições temporais, que devem ser tomadas em consideração [Malheiro-99].

As capacidades de explicação ainda não foram grandemente exploradas em aplicações de Sistemas de Energia eléctrica, apesar do contributo que poderiam dar para um uso mais efectivo e para uma melhor capacidade de aceitação dos Sistemas Baseados em Conhecimento.

No caso do SPARSE, foi desenvolvido um módulo para a produção de explicações em tempo real. Dado que a produção de explicações poderia comprometer o funcionamento em tempo real do sistema, foi desenvolvido e implementado um mecanismo especial.

No SPARSE existe uma separação do sistema pericial e da interface gráfica com o utilizador, em dois processos comunicantes.

O SPARSE tem um ambiente multiagente, em que mais do que um processo pode necessitar da disponibilização de um conjunto de funcionalidades. Daí a utilização da filosofia de Cliente-Servidor. Assim, o SPARSE, como Servidor, vai poder fornecer explicações a qualquer processo integrante do sistema (Cliente do SPARSE) que o interroge.

Esta filosofia é especialmente utilizada para que as explicações não influenciem o desempenho do sistema pericial numa situação de incidente, garantindo o seu funcionamento em tempo real.

Com a possibilidade de justificação das conclusões obtidas o uso do SPARSE torna-se muito mais efectivo, aumentando o grau de confiança dos utilizadores.

O servidor de explicações pode ser utilizado, nomeadamente, pelos operadores através do sistema em funcionamento “on-line” e pelo módulo de tutor inteligente.

6.2.8 Módulos Adicionais

Para que um sistema pericial como o SPARSE possa ser utilizado eficientemente, deve dispor de alguns módulos adicionais que viabilizem esse funcionamento. O SPARSE tem dois módulos adicionais: o módulo de Verificação e Validação e o módulo Tutor. Estes módulos podem ser considerados fundamentais para o sucesso de um projecto como o SPARSE.

6.2.8.1 Verificação e Validação

Como se sabe, os custos de correcção de “software” crescem de forma não linear [Boehm-88]. A correcção de erros tardiamente implica a afectação a essas tarefas de muito mais recursos (máquinas, pessoas) do que aqueles que seriam necessários se essa correcção fosse feita em fases anteriores.

Por isso, a equipa de desenvolvimento do SPARSE teve a preocupação de realizar esses testes o mais cedo possível [Vale-97].

A validação de um Sistema Baseado em Conhecimento (SBC) não difere da validação convencional, correspondendo à certificação de que o programa não tem erros e satisfaz as necessidades e requisitos do utilizador.

Contudo, existem diferenças entre os SBC e o “software” convencional que condicionam a utilização das ferramentas existentes no mercado. De entre estas diferenças, salientam-se as seguintes:

- Dificil definição de objectivos, dificultando a avaliação do desempenho do sistema a partir dos respectivos resultados.
- Necessidade de representação de incerteza no conhecimento, requerendo que sejam avaliados os mecanismos criados para manipulação das estruturas utilizadas para este fim.
- Dificuldade de execução de testes em laboratório, pois a avaliação não é feita simplesmente a partir do modelo físico subjacente como é feito no “software” convencional mas requer a opinião de um ou mais peritos.

Para a validação do “software” existem diversas técnicas, no entanto nenhuma assegura a completa correcção dos programas. Quando se pretendem níveis de confiança elevados, é comum recorrer a diferentes técnicas por forma a que estas se complementem.

Apesar de, por vezes, a Validação e a Verificação serem um pouco confundidas, têm funções diferentes. Enquanto que a Verificação assegura que o sistema foi construído correctamente e não contém erros técnicos, a Validação assegura que o sistema construído fornece soluções com um grau de confiança semelhante às soluções apresentadas por um perito.

Num sistema destinado a situações críticas, a V&V é de importância vital para a aceitação deste. No caso do SPARSE, a sua introdução em serviço experimental nos Centros de Condução da REN requereu importantes esforços de validação.

6.2.8.1.1 Validação do SPARSE

Durante o desenvolvimento do SPARSE foram utilizados vários métodos de validação [Vale-97].

Na primeira fase em que o sistema não estava ainda integrado no Centro de Condução e Controlo, como não era possível receber informação em tempo real, foram desenvolvidas e instaladas naquele Centro, aplicações que permitiam registar em ficheiros, informação relativa à rede eléctrica. Estes ficheiros eram então processados pelo SPARSE e os respectivos resultados eram analisados por peritos do sistema.

De forma a que o tratamento destes ficheiros fosse efectuado de forma idêntica ao tratamento da informação em tempo real, foi desenvolvido um simulador que fornecesse essa informação ao SPARSE a um ritmo adequado.

Numa fase posterior, o SPARSE foi instalado num Centro de Condução, permitindo-se assim que a informação fosse recebida do SCADA em tempo real e que se utilizassem outros métodos de validação. De forma a se reproduzir o melhor possível as situações reais de incidente foram usados dois métodos.

O primeiro método utilizava uma linha da rede eléctrica, colocada fora de serviço, onde eram simulados os incidentes. Este método além de requerer a disponibilização da linha, requeria também a coordenação de pessoas em pelo menos duas instalações distintas da rede (nós extremos da linha utilizada). Este método apesar de permitir efectuar testes em ambiente idêntico ao ambiente real, acarretava custos elevados para a empresa.

O segundo método utilizava um Gerador de Impulsos Programável (GIP) que era ligado a uma Unidade Remota Terminal. Este método necessita que o GIP seja operado a partir de uma instalação. A programação desse gerador permitia simular uma grande quantidade de incidentes, de modo a realizar uma validação tão exaustiva quanto possível. Este método tem a vantagem de necessitar de menos recursos humanos e de permitir escolher com maior facilidade a altura em que a validação era feita.

Contudo, a Base de Conhecimento do SPARSE evolui à medida que a rede eléctrica também evolui, requerendo que seja feita a V&V sempre que haja alterações. Os métodos anteriores requerem custos demasiado elevados para poderem ser disponibilizados sempre que estas alterações ocorram e não permitem analisar de uma forma exaustiva a Base de Conhecimento de modo a detectar possíveis anomalias. A fase de Validação não é capaz, por si só, de assegurar a correcção e consistência do conhecimento representado na Base de Conhecimento. Daí, surgir a necessidade de desenvolver uma ferramenta de Verificação da Base de Conhecimento.

6.2.8.1.2 Verificação do SPARSE

O grande número de formas utilizadas na representação do conhecimento condiciona fortemente a utilização de ferramentas genéricas de V&V. O SPARSE apresenta algumas características que exigem tratamento específico no campo da V&V, nomeadamente as seguintes [Vale-97]:

- A base de conhecimento tem características dinâmicas, obrigando à representação dos conceitos de data e hora associados às regras e aos factos.
- Cada regra pode dar origem a várias acções, o que não é geralmente considerado nos formalismos de V&V.
- Utilização de variáveis nas regras, exigindo um tratamento mais complexo.
- Utilização de disjunções no LHS das regras.

Para fazer a verificação do SPARSE, foi desenvolvida uma ferramenta designada por VERITAS. Esta ferramenta usa a detecção de anomalias como método de identificação de erros na base de conhecimento.

A detecção de anomalias nos SBC parte do pressuposto de que uma anomalia poderá implicar a existência de erros na concepção e/ou implementação da base de conhecimento.

Na verificação da Base de Conhecimento é utilizado o método CTMS (Contradiction Tolerant Truth Maintenance System) [Zlatareva-94]. Porém, a aplicação do método não é feita de uma forma directa pois a Base de Conhecimento tem características próprias que obrigaram a um processamento suplementar. Assim, foram criados filtros que convertem a sintaxe da base de conhecimento para uma sintaxe interna que o método reconhece.

6.2.8.2 Tutor Inteligente

O Tutor Inteligente destina-se a fornecer aos operadores dos Centros de Controlo e Condução da REN um ambiente de treino que lhes permita adquirir as competências necessárias à tarefa de interpretar as listas de mensagens recebidas em caso de incidente, de modo a poder rapidamente detectar a origem da perturbação ocorrida, sem o que não será possível tomar as medidas correctivas adequadas.

Esta tarefa é particularmente exigente, sobretudo em caso de incidentes que envolvam várias instalações, devido ao grande número de mensagens de alarme que podem ser recebidas num curto espaço de tempo. Nesta situação, a tarefa complica-se mais, se houver necessidade de diagnosticar quais são realmente as mensagens críticas e se os operadores estiverem sob condições de “stress” provocadas pelas próprias tomadas de atitude a tomar.

Apesar do grande número de aplicações informáticas existentes nos Centros de Controlo e Condução, os operadores são ainda o factor determinante na tomada de decisões e, muitas vezes, não recebem das ferramentas de apoio à decisão a assistência necessária.

Um número significativo de problemas de operação e controlo das redes eléctricas são atribuíveis a erros dos operadores, mesmo dos mais experimentados, devido à complexidade das tarefas e ao facto de, muitas vezes, os incidentes mais graves resultarem de situações que os operadores nunca enfrentaram.

Com o propósito de possibilitar um ambiente de treino aos operadores foi desenvolvido um Tutor Inteligente (SPARSE-IT) para ser instalado nos Centros de Controlo e Condução da REN.

A interacção entre o Tutor e o SPARSE estabelece-se a dois níveis [Vale-99b]:

- Numa primeira fase, são fornecidas ao SPARSE as listas de mensagens que irão ser apresentadas aos operadores, para que aquele as processe, gerando as conclusões sobre os incidentes por elas descritos. Estas conclusões vão ser utilizadas pelo Tutor para as confrontar com as respostas dadas pelos alunos. O Tutor pode então verificar se o conhecimento manifestado pelo operador se aproxima ou não do conhecimento do perito.
- Numa segunda fase, o SPARSE vai ser consultado pelo Tutor, no sentido de fornecer as explicações sobre qual o raciocínio que seguiu para chegar a um dado conjunto de conclusões, isto é, quais os factos em que se baseou e quais as regras que foram disparadas pelo sistema pericial ao processar a lista de mensagens em causa.

6.3 TEMPUS

6.3.1 Introdução

O Sistema Pericial SPARSE foi construído com base no conhecimento extraído de peritos no Controlo e Condução de Redes de Transporte de Energia Eléctrica. Dada a complexidade deste tipo de aplicação, que envolve questões como o Raciocínio Temporal e não Monotónico, bem como a dificuldade de efectuar sessões de aquisição de conhecimento demoradas, visto que aos peritos cabem muitas tarefas dentro das empresas de Transporte de Energia Eléctrica, tornou-se adequado o desenvolvimento de um Sistema que fosse capaz de extrair automaticamente conhecimento a partir de análises efectuadas sobre os ficheiros de incidentes usados nos Centros de Condução da Rede Eléctrica Nacional.

O objectivo do trabalho no âmbito da presente Tese de Mestrado foi pois, o desenvolvimento de um Sistema de Extração Automática de Conhecimento que fosse capaz de confirmar ou sugerir modificações ao conhecimento já existente no SPARSE.

A manutenção do conhecimento é um dos factores chave para o sucesso de um Sistema Baseado no Conhecimento (SBC). As dificuldades encontradas na manutenção do conhecimento são a razão principal para o abandono do uso de muitos SBC.

Os SBSs devem permanecer eficientes, mesmo quando existe necessidade de alterações na rede eléctrica e nos seus métodos de operação. Estas mudanças requerem a alteração do SBC, e devem ser transparentes aos operadores dos Centros de Controlo.

No caso do SPARSE, tentou-se que sempre que possível o conhecimento fosse actualizado automaticamente. Isto é feito mediante a informação obtida pelo sistema SCADA. Contudo, isto não é possível para todo o conhecimento envolvido. No caso da representação gráfica, a necessidade de actualização é automaticamente detectada pelo SBC e os operadores são informados. Quando o operador pretende executar as alterações requeridas, o SBC disponibiliza um assistente que torna este processo de actualização muito simples.

Contudo, algumas alterações requerem conhecimento adicional que deve ser disponibilizado por operadores ou peritos. Nestes casos, o processo deve ser feito de uma forma tão fácil e rápida quanto possível utilizando ferramentas de assistência apropriados.

As operações mais difíceis de manutenção são frequentemente as que envolvem alterações na Base de regras. De facto, apesar de ser aparentemente fácil alterar a Base de Regras, essas alterações devem ser verificadas e validadas de forma a garantir que o SBC continua a operar sem problemas.

A verificação da Base de Regras deve ser considerada essencial para o desenvolvimento de um SBC. A maior parte dos SBC são só validados, usando incidentes reais e simulados, mas não passam por um processo de verificação. No SPARSE, como já foi referido, foi desenvolvida uma ferramenta de Verificação.

Quando a Base de Regras é alterada, é verificada usando essa ferramenta, que possibilita informação sobre possíveis erros. Estes resultados são analisados e as correcções são feitas, quando tal for considerado necessário.

Por outro lado, a Base de Regras também sofre um processo de validação. Para isso, desenvolveu-se um simulador que permite enviar as listas de mensagens guardadas em ficheiros, simulando o desempenho em tempo real do SBC.

Essas listas de alarme incluem:

- Incidentes reais, considerados relevantes para o processo de validação.
- Incidentes simulados que foram usados para a validação do SPARSE, obtidos:

- pelo uso de um Gerador de Impulsos Programável ligado a unidade terminal remota,
 - por operação do equipamento por operadores humanos.
- Incidentes simulados que foram obtidos pela geração de listas de mensagens.

A manutenção do conhecimento é de especial importância em SBSs para aplicações críticas, como é o caso dos SBS's que processam situações de incidente em CC.

Mesmo em situações em que não se verifiquem alterações na rede eléctrica, pode haver necessidade de fazer alterações no SBC. De facto, à medida que o SBC lida com mais situações e o conhecimento existente pode ter que ser alterado para reflectir essas situações.

Este processo requer o uso de técnicas de Aprendizagem Automática cujos resultados devem ser na maior parte dos casos, validados por humanos.

A Aprendizagem Automática é uma área de investigação bastante interessante que visa a aquisição automática de novo conhecimento ou um aperfeiçoamento do conhecimento já existente.

A aplicação das técnicas de Aprendizagem Automática, se for bem sucedida, podem superar os problemas de aquisição e actualização de conhecimento.

6.3.2 Algoritmos de Aprendizagem Automática Versus TEMPUS

Os métodos de Aprendizagem Indutiva analisados na presente tese, tais como os métodos de Aprendizagem de Conceitos, a Aprendizagem baseada em Árvores de Decisão, as Redes Neurais, a Aprendizagem Bayesiana, a Aprendizagem baseada em Instâncias, os Algoritmos genéticos e a Aprendizagem de Conjuntos de Regras generalizam a partir de exemplos de treino observados e pela identificação de características que distinguem os exemplos positivos dos negativos. Estes métodos, para adquirirem um dado nível de exactidão na generalização, necessitam de um certo número de exemplos de treino. O desempenho destes métodos não é adequado quando não estão disponíveis dados suficientes.

Por outro lado, a Aprendizagem Analítica usa o conhecimento prévio (Teoria do Domínio) e o raciocínio dedutivo para aumentar a informação obtida a partir dos exemplos de treino, não estando sujeita às mesmas limitações que os métodos anteriores no que diz respeito ao número de exemplos disponíveis. O conhecimento prévio, na Aprendizagem Analítica, é aceite explicitamente como “input” adicional aos dados de treino.

Na Aprendizagem Indutiva, o “output” deve ser uma hipótese que é consistente com os exemplos de treino, enquanto que na Aprendizagem Analítica, o “output” deve ser uma hipótese que é consistente com os exemplos de treino e o conhecimento prévio.

O sistema pericial SPARSE é um sistema baseado em conhecimento para apoiar os operadores dos centros de controlo e condução da rede de transporte de energia eléctrica portuguesa, na análise de incidentes e na reposição de serviço.

No processamento de alarmes, a utilização de uma abordagem puramente baseada em modelos é impraticável. De facto, o número de elementos de uma rede é muito elevado e, além da respectiva modelização se revelar praticamente impossível para os fins em vista, o tempo necessário ao processamento seria demasiado longo. Por outro lado, embora o número total de componentes seja muito elevado, eles pertencem a um número relativamente reduzido de categorias (linhas, subestações,...), pelo que o uso de sistemas baseados em regras se torna muito atractivo. Uma regra, desenvolvida para um componente de um determinado tipo, aplica-se a todos os componentes do mesmo tipo. Por outro lado, o uso de conhecimento técnico e empírico recolhido dos peritos e operadores experientes, codificado sob a forma de regras, permite um processamento rápido e eficiente.

Os testes feitos ao SPARSE demonstraram que, apesar da estrutura geral das regras estar correcta, os parâmetros temporais usados nas regras deveriam ser ajustados pela consideração de incidentes reais.

O principal objectivo do presente trabalho foi a análise do comportamento das premissas temporais das regras que constituem a Base de Regras do SPARSE.

Como se referiu, tanto os métodos de Aprendizagem Indutiva como Analítica necessitam de exemplos de treino para a obtenção das melhores hipóteses para os problemas em estudo.

No caso do SPARSE, existe uma base de dados disponibilizada pela REN, com informação dos incidentes ocorridos na rede eléctrica portuguesa. Essa base de dados é constituída por informação estruturada de uma forma completamente diferente da existente nas regras do SPARSE, não se conseguindo por isso fazer o mapeamento directo da informação contida nessa base de dados para as regras do SPARSE.

Pelo facto de não existirem exemplos de treino adequados para as regras do SPARSE em número suficiente, e dado ser imprescindível a sua existência para aplicação dos métodos de Aprendizagem Indutiva e Analítica estudados, não foi possível aplicar nenhum destes métodos.

Por este motivo, foi necessário conceber e implementar um novo algoritmo de Aprendizagem Automática capaz de dar resposta aos propósitos inicialmente definidos.

O algoritmo de Aprendizagem Automática desenvolvido para o SPARSE é designado por TEMPUS.

6.3.3 Objectivos do TEMPUS

O processo de aquisição de conhecimento para o SPARSE foi realizado através de um grande número de reuniões com os peritos. Os testes feitos ao SPARSE demonstraram que apesar da estrutura geral das regras estar correcta, os parâmetros temporais usados nas regras deveriam ser ajustados pela consideração de incidentes reais.

De facto, os peritos tendem a atribuir a estes parâmetros valores que resultam do seu conhecimento acerca da operação e controlo da rede e do equipamento utilizado para esse efeito. Frequentemente, só quando são confrontados com incidentes reais admitem que se podem verificar valores diferentes devido a vários factores, tais como a aquisição e comunicação de dados e particularidades de algumas instalações e equipamentos.

O desempenho em tempo real do SPARSE mostrou que a maior parte das modificações na Base de Regras se devia à necessidade de ajuste nos tempos que eram usados para o Raciocínio Temporal. Este ajuste requer a análise de um grande conjunto de casos.

Como esta análise beneficia da consideração de um grande conjunto de incidentes, o uso de técnicas de Aprendizagem Automática revela-se adequado. Baseado neste princípio, desenvolveu-se em Prolog um módulo de Aprendizagem Automática (TEMPUS) que faz sugestões sobre alterações na Base de Regras do SPARSE.

Os objectivos do algoritmo de Aprendizagem Automática TEMPUS são os seguintes:

- Análise do comportamento das premissas temporais das regras constituintes da Base de Regras do SPARSE. Para cada regra do SPARSE, a análise é realizada para cada instalação e, dentro de cada instalação, para cada painel.
- Análise específica de algumas regras e instalações ou uma análise geral incluindo todas as regras e todas as instalações existentes.
- Produção de um relatório com as conclusões obtidas relativamente às regras e instalações escolhidas pelo utilizador para a análise e eventual utilização

pelo editor de regras, caso o perito pretenda fazer alterações à Base de Regras do SPARSE.

6.3.4 Arquitectura do TEMPUS

O TEMPUS é uma ferramenta que possibilita o ajuste dos tempos considerados nas premissas temporais das regras do SPARSE. Este ajuste requer a análise de um grande conjunto de listas de mensagens e envolve vários tipos de tempos, tais como:

- tempo mínimo que deve ser verificado entre dois eventos,
- tempo máximo que deve ser verificado entre dois eventos,
- tempo máximo permitido para esperar por possíveis eventos relacionados relativos aos dois extremos da mesma linha,
- tempo máximo permitido para esperar por possíveis eventos relacionados relativos à mesma instalação,
- tempo máximo permitido para a chegada de informação do SCADA.

O TEMPUS usa listas de mensagens que incluem ficheiros mensais com todas as mensagens geradas durante o mês correspondente. Por outro lado, os incidentes mais interessantes ocorridos são classificados de acordo com o trabalho de um grupo da REN de análise de incidentes. Estes incidentes são classificados por tipos permitindo aos utilizadores o uso do TEMPUS para considerar tipos específicos de incidentes.

Este módulo de Aprendizagem Automática possibilita aos utilizadores:

- sugestões relativas a alterações dos tempos considerados na regras, baseando-se no conjunto de ficheiros de incidentes que foram usados,
- uma análise relativa a toda a Base de Regras ou só a uma parte dela (pode ser considerada uma única regra),
- uma análise relativa a todas as instalações ou só a uma parte delas (pode ser considerada uma única instalação); esta análise torna-se muito adequada em instalações com características específicas (por exemplo, instalações mais antigas, em que o equipamento apresente um comportamento particular),
- sugestões relativas à introdução de novas regras, específicas de instalações que têm um comportamento diferente do resto das instalações.

Depois do utilizador escolher quais as regras e instalações a analisar, o sistema pericial SPARSE analisa automaticamente as mensagens incluídas nos ficheiros a tratar. Este processo não requer nenhuma interacção com o utilizador, pois quando a análise cobre um grande período de tempo requer a disponibilização de bastante tempo para o processamento.

As listas de mensagens são fornecidas ao SPARSE por um simulador que permite que as mensagens sejam processadas como se estivessem a chegar em tempo real. Este simulador não é específico do TEMPUS, sendo usado para manter uma versão “off-line” do SPARSE sem necessidade de alterar a sua arquitectura. Isto permite o desenvolvimento e a manutenção do sistema com uma interferência mínima na versão em tempo real, que está a correr no Centro de Controlo.

Nos Sistemas Baseados em Regras que utilizam inferência com encadeamento directo, como o SPARSE, parte-se dos dados de entrada por forma a chegar a conclusões. Cada uma das regras é verificada para se determinar se os dados disponíveis permitem satisfazer as suas premissas.

Se o conjunto de premissas de uma regra é satisfeito, esta regra é executada, permitindo derivar novos factos, os quais podem ser usados por outras regras para derivar outros factos, e assim sucessivamente.

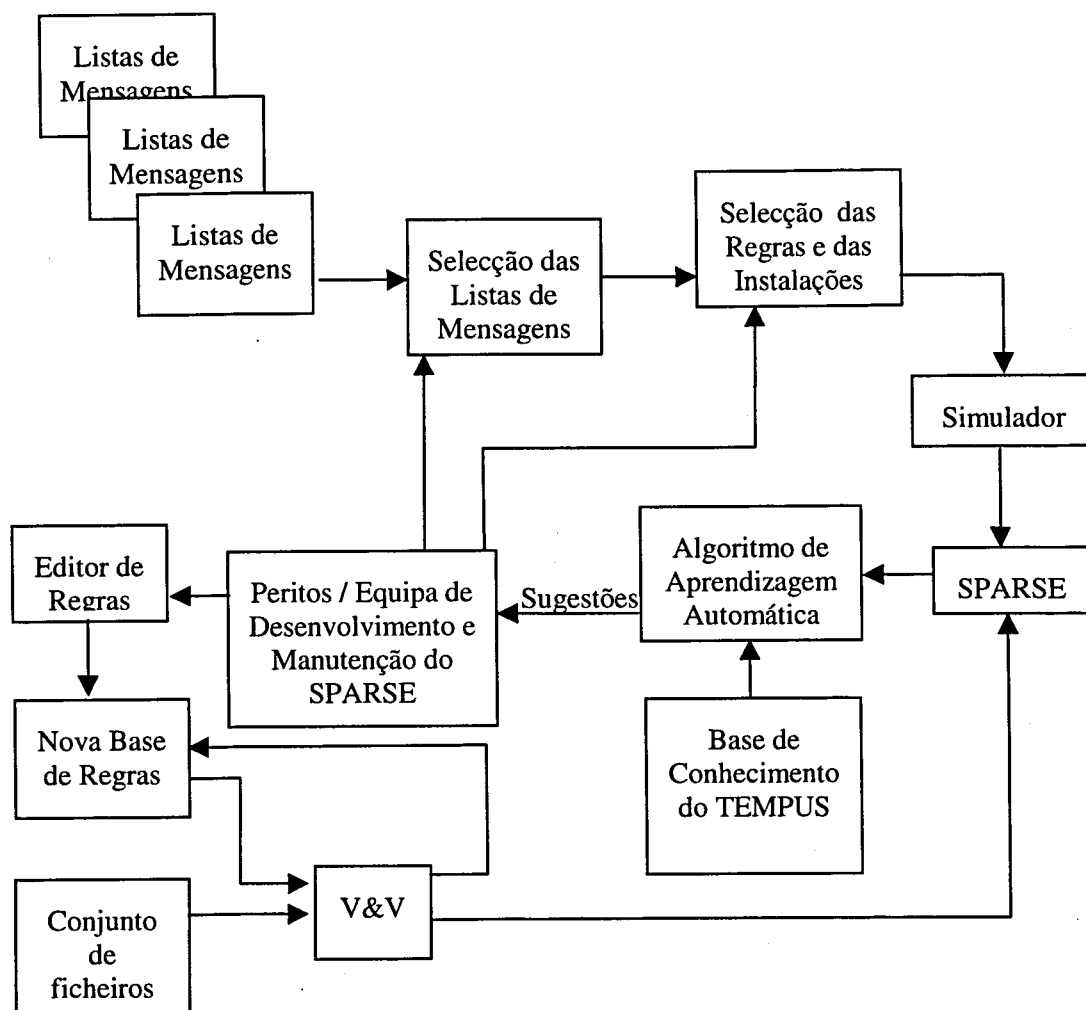


Figura 6.2 - Arquitectura do TEMPUS

O algoritmo de Aprendizagem Automática (TEMPUS) analisa o sucesso das regras não só em termos de “sim” ou “não”, mas também em termos da avaliação da “distância ao sucesso”.

Do ponto de vista do aperfeiçoamento do conhecimento, é importante para cada regra bem sucedida, saber a que distância esteve de não ter sido bem sucedida, e para cada regra não sucedida, saber a que distância esteve de ter sido bem sucedida.

Algumas regras da Base de Conhecimento do SPARSE, no LHS têm uma ou mais premissas temporais, a serem satisfeitas para o disparo da regra.

A “distância ao sucesso” (DtS) pode ser considerada tanto positiva como negativa. A DtS é considerada positiva quando a premissa ou premissas temporais em causa são satisfeitas, e é considerada negativa quando a regra não dispara pelo incumprimento de uma premissa temporal.

Esta filosofia tem o mérito de ser muito semelhante ao processo que os humanos utilizam frequentemente na análise de regras. Por outro lado, pode fazer uso de conhecimento de alto nível para definir a forma de avaliação da DtS e como fazer sugestões para alterações na Base de Regras.

O TEMPUS usa uma Base de Conhecimento que inclui as ocorrências consideradas nas regras, e que descrevem a sequência de eventos.

Os parâmetros temporais considerados na Base de Regras foram definidos tomando em linha de conta o equipamento envolvido, o sistema de transmissão de dados e algumas tolerâncias.

O uso deste conhecimento permite ao TEMPUS um desempenho mais flexível do que os algoritmos de Aprendizagem Indutiva. De facto, este conhecimento considera situações muito específicas e é caracterizado por uma natureza de alto nível que permite ao TEMPUS apresentar características mais inteligíveis.

Depois da análise do conjunto de mensagens de alarme escolhidas, realizada pelo SPARSE, o TEMPUS obtém as DtS de cada regra a analisar, apresentando de seguida as conclusões a que chegou e as sugestões de alterações da Base de Regras.

Se o utilizador decidir fazer alguma alteração na Base de Regras, é disponibilizado um editor de regras, o qual facilita o processo de alteração da Base de Regras.

Caso se tenha realizado alguma alteração na Base de Regras, é necessário fazer a verificação e validação da nova Base de Regras. Esta tarefa é realizada por um módulo de verificação e validação (VERITAS) desenvolvido anteriormente. O VERITAS baseia-se em métodos formais de verificação e usa um conjunto de ficheiros de mensagens escolhidos para esse propósito.

A figura 6.2 apresenta a arquitectura do módulo de Aprendizagem Automática do SPARSE (TEMPUS) e a sua interacção com os utilizadores.

6.3.5 Alterações ao Motor de Inferência do SPARSE

O Motor de Inferência do SPARSE, como já foi referido, utiliza uma estratégia de encadeamento directo do raciocínio. Quando chega um novo facto ao sistema, as regras apropriadas são disparadas de modo a obter conclusões.

O módulo seleccionador de regras usa o meta-conhecimento que detém, sempre que um novo facto chega ao SPARSE, de forma a saber quais devem ser as regras a disparar e em que instante devem ser disparadas.

O Motor de Inferência do SPARSE sempre que ocorre um disparo de uma regra, acrescenta à Base de Conhecimento um facto do seguinte tipo:

regra_nfatos(NR, NB, LFactoS, LA, LR)

em que :

- *NR* é o número da regra disparada.
- *NB* é o número do bloco da regra responsável pelo disparo.
- *LFactoS* é uma lista que pode conter:
 - O número de série do facto Prolog que satisfaz a respectiva premissa da regra. Estas premissas podem ser do tipo:
 - *X at Instante*
 - *X*
 - O número 0 caso tenha sido satisfeita uma premissa da regra do tipo :
 - *nao X at Instante*
 - *nao X*
- *LA* é uma lista com o número de série dos factos Prolog que foram acrescentados à base de factos.
- *LR* é uma lista com o número de série dos factos Prolog (da forma *facto(N, F, T)*). De notar que, simultaneamente à eliminação de um facto deste tipo é

acrescentado à Base de Conhecimento, um facto da forma *facto_antigo(N, F, T, TI)* em que *TI* é o instante em que o facto *facto(N, F, T)* deixa de ser verdadeiro.

Desta forma, as premissas temporais incluídas nas regras do SPARSE não são registadas no predicado *regra_nfatos(NR, NB, LFactosS, LA, LR)*.

O módulo de Aprendizagem Automática do SPARSE (TEMPUS) tem como um dos seus principais objectivos a análise do comportamento das premissas temporais das regras do SPARSE. O algoritmo analisa o disparo das regras em termos de “sim” ou “não” e em termos da “distância ao sucesso”. Para que essa análise possa ser feita, é necessário a disponibilização, por parte do SPARSE, de informação sobre os tempos associados às premissas temporais das regras.

Com o intuito de disponibilizar essa informação, foi necessário alterar o Motor de Inferência.

Dado que o TEMPUS é uma ferramenta para ser utilizada em “off-line”, e tem necessidades específicas relacionadas com os tempos verificados nas premissas temporais, a alteração a ser realizada ao Motor de Inferência é necessária só na versão a utilizar pelo TEMPUS.

As alterações ao Motor de Inferência incluem:

- A inclusão de mais um argumento (*LFactoNS*) no facto *regra_nfatos* ficando

regra_nfatos(NR, NB, LFactosS, LFactoNS, LA, LR)

em que *LFactoNS* é uma lista que contém informação sobre uma premissa temporal de uma regra (no caso da regra não ter disparado pelo incumprimento de uma premissa temporal) ou uma lista vazia no caso da regra ter sido bem sucedida.

- A inclusão nas listas *LFactosS* e *LFactoNS* de informação relativa às premissas temporais das regras e informação relativa às premissas *não X at Instante* e *não X*.
- A inclusão na Base de Conhecimento do facto *regra_nfatos*, nas situações em que não se verificou o disparo da regra pelo incumprimento de uma premissa.
- O armazenamento em disco (ficheiro) dos factos seguintes:
 - *regra_nfatos(NR, NB, LFactosS, LFactoNS)*
 - *facto_antigo(N, F, T, TI)*

De facto, são excluídas as listas *LA* e *LR* no facto *regra_nfatos*, pois esta informação não tem interesse no processamento realizado pelo TEMPUS.

6.3.6 Algoritmo de Aprendizagem Automática – TEMPUS

As instalações eléctricas relativas à produção e ao transporte de energia eléctrica distribuídas por todo o País podem ser de dois tipos:

- Centrais, que por sua vez podem ser :
 - Térmicas.
 - Hidráulicas.
- Subestações.

Cada uma destas instalações é constituída por um número variável de painéis.

No SPARSE, cada instalação é identificada por três letras. A primeira letra, “C” ou “S” identifica, respectivamente, uma central ou uma subestação. As duas letras restantes identificam a designação da central ou a designação da subestação.

No anexo A, inclui-se uma lista com todas as instalações existentes e os respectivos painéis. De notar que essa lista se refere a uma determinada data, e está sujeita a alterações, de acordo com as alterações efectuadas na rede eléctrica.

O formato normalizado de uma regra de produção do SPARSE, como já foi referido é do seguinte tipo:

REGRA X : 'Descrição da regra X' :

```
[
  [ P1 at T1 e P2 e P3 at T3 e C1(T1,T3) ]
  ou
  [ P1 at T1 e P2 e P4 at T4 e C2(T1,T4) ]
  ou
  ...
]
==>
[
  cria_facto( Nf1,Tnf1),
  retira_facto( P2, T2 ,Ttrp2),
  escreve_mens( MensUtil )
].
```

O LHS da regra é constituído por uma identificação e uma descrição da regra, e por um bloco ou uma disjunção de blocos. Cada bloco é constituído por uma conjunção de premissas, algumas delas temporais.

Para analisar as premissas temporais, o SPARSE dispõe de um conjunto de predicados que definem relações temporais.

Alguns desses predicados são os seguintes:

- a) *dif_tempos_maior*(I1,I2,T)
- b) *dif_tempos_menor*(I1,I2,T)
- c) *dif_tempos_maior_ou_igual*(I1,I2,T)
- d) *dif_tempos_menor_ou_igual*(I1,I2,T)
- e) *mod_dif_tempos_maior*(I1,I2,T)
- f) *mod_dif_tempos_menor*(I1,I2,T)
- g) *mod_dif_tempos_maior_ou_igual*(I1,I2,T)
- h) *mod_dif_tempos_menor_ou_igual*(I1,I2,T)

Nestes predicados, I1 e I2 são instantes de tempo e T é um número real, com significado diferente conforme o predicado. Os predicados a) a d) têm sucesso, respectivamente, se a diferença entre I1 e I2 for maior, menor, maior ou igual, menor ou igual a um dado valor T. Os predicados e) a h) são equivalentes aos predicados a) a d) mas consideram o módulo da diferença.

O algoritmo de Aprendizagem Automática do SPARSE, começa por permitir ao utilizador a escolha de quais as regras e instalações a analisar.

O utilizador tem, assim, a possibilidade de fazer uma análise mais específica a determinadas regras e/ou instalações, ou ainda, uma análise mais abrangente de todas as regras com premissas temporais e de todas as instalações.

Após a escolha, realizada pelo utilizador, das regras e instalações a analisar, o TEMPUS começa por fazer uma pesquisa na Base de Conhecimento de forma a obter todos os blocos a analisar, para cada uma das regras escolhidas.

A partir dos factos *regra_nfatos(NR, NB, LFactoS, LFactoNS, LA, LR)* existentes na Base de Conhecimento e por cada combinação regra/bloco (*NR/NB*) obtida no passo anterior, o TEMPUS, acrescenta à sua Base de Conhecimento um facto *regra_bloco(NR, NB, L)*, em que:

- *NR* é o número da regra.
- *NB* é o número do bloco.
- *L* é uma lista com elementos do tipo *LFactoS-LFactoNS*, relativos a todos os factos *regra_nfatos(NR, NB, LFactoS, LFactoNS)* existentes na Base de Conhecimento para a combinação *NR/NB* a tratar.

De notar que a lista *LFactoNS* pode conter ou não elementos. De facto, a lista *L* poderá conter tanto elementos relativos às situações de disparo da regra *NR* no bloco *NB* (*LFactoS-[]*), como também, conter elementos relativos às situações de não disparo da regra *NR* no bloco *NB* (*LFactoS-LFactoNS*), quando esse não disparo da regra se dever ao incumprimento de uma premissa temporal. Enquanto que os elementos do tipo *LFactoS-[]* contêm a segunda lista sem elementos (não têm premissas temporais não sucedidas), os elementos do tipo *LFactoS-LFactoNS* contêm na lista *LFactoNS* informação sobre a premissa temporal não satisfeita.

Na fase seguinte, o algoritmo, por cada regra escolhida a analisar, verifica a existência de factos *regra_bloco(NR, NB, L)* relativos a cada regra.

Se não existirem factos *regra_bloco(NR, NB, L)* relativos a uma determinada regra, é incluída informação no relatório alusiva a esse facto, com a mensagem “Regra sem situações de disparo sucedidos e sem situações de não disparo da regra devido ao incumprimento de premissas temporais”. Caso esta situação se verifique, de seguida, é analisada outra regra escolhida pelo utilizador. Se já todas as regras escolhidas tiverem sido analisadas, o processamento do algoritmo termina.

No caso de existirem factos *regra_bloco(NR, NB, L)* para a referida regra, a lista *L* do primeiro facto *regra_bloco(NR, NB, L)* é separada em duas listas, uma com as situações de disparo da regra (*LS*) e outra que contempla as situações de não disparo da regra (*LNS*).

Se a lista *LS* não tiver elementos, conclui-se que não se verificaram situações de disparo da regra. Perante esta situação, é incluída no relatório a mensagem “Regra sem disparos sucedidos”, e é analisada de seguida a Lista *LNS*.

Se a lista *LS* tiver elementos, e se o utilizador pretende que todas as instalações sejam analisadas, existe a necessidade de obter todas as instalações que existem na lista *LS*, armazenando-as na lista *LInst*. No caso do utilizador ter escolhido algumas instalações específicas, *LInst* é uma lista com essas instalações.

O algoritmo TEMPUS, para a primeira instalação de *LInst*, obtém todos os elementos da lista de situações de disparo (*LS*) que incluam essa instalação, armazenando-os na lista *LSI*.

Se a lista *LSI* não tiver elementos, é incluída no relatório a mensagem “Regra sem disparos nesta instalação” e é verificada a existência de mais instalações na lista *LInst*. Se existirem mais instalações em *LInst* a analisar, o processo de análise da instalação anterior repete-se para as restantes instalações existentes em *LInst*, caso contrário é analisada de seguida a lista *LNS*.

Se a lista *LSI* tiver elementos, é necessário obter os painéis existentes em *LSI* e armazená-los na lista *LPain*. De facto, é necessário fazer uma análise em cada instalação por cada painel nela existente.

Para cada painel existente em *LPain*, é necessário obter os elementos de *LS1* que incluam esse painel dando origem a uma nova lista (*LS2*). Cada elemento da lista *LS2* só contém elementos relativos a uma determinada instalação e a um determinado painel. Cada elemento da lista *LS2* é do tipo *LFactoS-[]*, em que *LFactoS* é a lista com informação associada às premissas satisfeitas para se ter verificado o disparo da regra.

Considere-se uma premissa temporal com a seguinte sintaxe:

$$Func(I1, I2, T)$$

em que *I1* e *I2* são instantes de tempo associados a dois eventos e *T* é um número atribuído por um perito.

Estes valores atribuídos pelos peritos podem eventualmente ter necessidade de ser ajustados em algumas situações devido a factores como a aquisição e comunicação de dados e particularidades de algumas instalações e equipamentos.

Com o intuito de analisar esses tempos e sugerir a sua eventual rectificação, é necessário, para cada elemento de *LS2*, calcular em cada uma das suas premissas temporais satisfeitas, a diferença ou o módulo da diferença de tempos entre *I1* e *I2*. No caso do functor da premissa temporal ser *dif_tempos_maior*, *dif_tempos_menor*, *dif_tempos_maior_ou_igual* ou *dif_tempos_menor_ou_igual* é calculada a diferença de tempos entre *I1* e *I2*. Se por outro lado, o functor da premissa temporal for *mod_dif_tempos_maior*, *mod_dif_tempos_menor*, *mod_dif_tempos_maior_ou_igual* ou *mod_dif_tempos_menor_ou_igual* é calculado o módulo da diferença de tempos entre *I1* e *I2*.

Por cada premissa temporal satisfeita, o algoritmo acrescenta à Base de Conhecimento um facto do seguinte tipo:

$$cond(NR, NB, Inst, Pain, Func, Tempo)$$

em que:

- *NR* é o número da regra.
- *NB* é o número do bloco.
- *Inst* é uma abreviatura da instalação associada à conclusão obtida.
- *Pain* é o painel da instalação anterior associado à conclusão obtida.
- *Func* é o functor da premissa temporal, por exemplo *dif_tempos_menor_ou_igual*.
- *Tempo* é a diferença ou o módulo da diferença de tempos entre *I1* e *I2* da premissa temporal com sintaxe *Func(I1, I2, T)*.

Depois de serem processados todos os elementos da lista *LS2*, é necessário obter por cada functor das premissas temporais, o maior ou o menor valor do argumento *Tempo*, de todos os factos do tipo *cond(NR, NB, Inst, Pain, Func, Tempo)*.

No caso das premissas temporais *dif_tempos_menor*, *dif_tempos_menor_ou_igual*, *mod_dif_tempos_menor* ou *mod_dif_tempos_menor_ou_igual*, obtém-se o maior valor do argumento *Tempo*, enquanto que com as premissas temporais *dif_tempos_maior*, *dif_tempos_maior_ou_igual*, *mod_dif_tempos_maior* ou *mod_dif_tempos_maior_ou_igual* obtém-se o menor valor do argumento *Tempo*.

Após este cálculo, por cada functor das premissas temporais, é incluída o relatório a produzir pelo TEMPUS uma conclusão, com uma mensagem indicando qual o novo valor que o argumento *T* da premissa temporal poderia assumir.

Caso o perito ache conveniente, pode substituir nessa premissa temporal da regra o valor T pelo valor obtido. Essa alteração à Base de Regras é possível no editor de regras (EDIREGRA).

Terminada a análise desse elemento da lista $LS2$, o passo seguinte consiste na verificação da existência de mais elementos nessa lista. Caso existam mais elementos em $LS2$, cada um desses elementos sofre o mesmo tratamento já referido para o anterior elemento de $LS2$. Caso contrário, termina a análise deste painel, sendo verificada, de seguida, a existência de mais painéis na lista $LPain$.

Caso existam mais painéis na lista $LPain$, toda a análise efectuada para o painel anterior é repetida para os restantes painéis existentes em $LPain$.

Se não existirem mais painéis em $LPain$, é verificada a existência de mais instalações na lista $LInst$ a serem analisadas.

Caso existam mais instalações em $LInst$, todo o processo descrito, de análise da instalação anterior, é repetido para todas as instalações existentes em $LInst$. Caso contrário, será analisada a lista LNS .

O primeiro passo na análise da lista LNS é a verificação da existência de elementos na lista. Se a lista LNS estiver vazia, poder-se-á concluir que, relativamente ao bloco NB da regra NR , não houve situações de não disparo da regra por não ter sido satisfeita alguma premissa temporal.

Assim, é incluída no relatório a mensagem “Regra sem disparos não sucedidos devido ao incumprimento de premissas temporais” e é verificada a existência de mais factos $regra_bloco(NR, NB, L)$ relativos à regra NR .

Se existirem mais factos $regra_bloco(NR, NB, L)$ para essa regra, todos esses factos são sujeitos ao mesmo processamento já descrito para o facto $regra_bloco(NR, NB, L)$ anterior. Caso contrário, é verificada a existência de mais regras na lista LR a serem analisadas.

Caso não existam mais regras para analisar, o algoritmo termina o seu processamento, senão todos os factos $regra_bloco(NR, NB, L)$ a tratar, relativos à nova regra, caso existam, sofrem a mesma análise já descrita para o facto $regra_bloco(NR, NB, L)$.

No caso da lista LNS conter elementos, e se o utilizador pretende que todas as instalações sejam analisadas, é necessário obter todos as instalações que existem na lista LNS , armazenando-as noutra lista, a lista $LNSInt$. Se, por outro lado, o utilizador pretender só a análise de algumas instalações específicas, $LNSInt$ é uma lista com essas instalações.

De seguida, o algoritmo TEMPUS, para a primeira instalação de $LNSInt$, obtém todos os elementos da lista de situações de não disparo da regra (LNS) que incluam essa instalação, armazenando esses elementos na lista $LNSI$.

Se a lista não tiver elementos, é incluída no relatório a mensagem “Regra sem disparos não sucedidos nesta instalação” e é verificada a existência de mais instalações em $LNIInst$ a serem analisadas.

Se existirem mais instalações a analisar na lista $LNIInst$, o processo de análise das instalações anterior é repetido para todas as instalações existentes em $LNSInt$. Caso contrário, de seguida, é verificada a existência de mais factos $regra_bloco(NR, NB, L)$ relativos à regra em análise. Caso existam ainda factos $regra_bloco(NR, NB, L)$ para essas regras, esses factos são sujeitos ao mesmo processamento já descrito para os factos do tipo $regra_bloco(NR, NB, L)$, caso contrário é verificada a existência de mais regras na lista LR a serem analisadas.

Se não existirem mais regras em LR , o algoritmo termina o seu processamento, senão todos os factos $regra_bloco(NR, NB, L)$ a tratar relativos à nova regra, caso existam, sofrem a mesma análise já descrita para os factos do tipo $regra_bloco(NR, NB, L)$.

Se a lista $LNSI$ tiver elementos, o algoritmo, obtém todos os painéis existentes em $LNSI$ e armazena-os na lista $LNSPain$.

Para cada painel existente em $LNSPain$, é necessário obter os elementos de $LNSI$ que incluam esse painel dando origem a uma nova lista, a lista $LNS2$.

Na fase seguinte, o algoritmo trata os elementos da lista *LNS2*, isto é, os elementos de *LNS1* que contêm o painel que está a ser tratado.

Cada elemento da lista *LNS2* é do tipo *LFactoS-LFactoNS*, em que *LFactoS* é uma lista com informação associada às premissas satisfeitas da regra e *LFactoNS* é uma lista com um só elemento com informação sobre a premissa temporal não satisfeita.

Se o algoritmo estiver em presença de uma regra com duas premissas temporais os elementos da lista *LNS2* podem ser de dois tipos, isto é, podem conter:

- Na lista *LFactoS*, premissas não temporais sucedidas e uma premissa temporal satisfeita e na lista *LFactoNS* uma premissa temporal não sucedida.
- Na lista *LFactoS*, só premissas não temporais sucedidas e na lista *LFactoNS* uma premissa temporal não sucedida.

Por sua vez, se o algoritmo estiver em presença de uma das restantes regras, os elementos da lista *LNS2* só podem ser de um tipo. Estes elementos são constituídos pela lista *LFactoS* que contém só premissas não temporais sucedidas e pela lista *LFactoNS* que contém uma premissa temporal não sucedida.

Dadas as anteriores possibilidades, o algoritmo verifica se o elemento de *LNS2* é um elemento com uma premissa temporal sucedida. No caso de isso se verificar, e considerando a premissa temporal na forma *Func(I1, I2, T)*, é necessário calcular a diferença ou o módulo da diferença de tempos (dependendo do functor da premissa temporal) entre *I1* e *I2*.

O algoritmo acrescenta ao relatório uma conclusão com o valor obtido no cálculo da diferença ou o módulo da diferença de tempos entre *I1* e *I2*, constituindo este valor um novo valor que essa premissa temporal poderá ter no argumento *T*.

De seguida, é necessário tratar a lista *LFactoNS* que contém a premissa temporal não satisfeita. Para isso, é calculada a diferença ou o módulo da diferença de tempos entre *I1* e *I2*. Este novo valor obtido (*Tempo*) para o argumento *T* da premissa temporal seria o valor que teria possibilitado a satisfação da premissa temporal e eventualmente o disparo da regra, podendo eventualmente ser aceite para a alteração da regra, caso o perito, chegue à conclusão que a alteração tem fundamento.

O perito, mediante a análise das conclusões obtidas pelo TEMPUS, tem no editor de regras (EDIREGRA), uma ferramenta que possibilita aceitar as sugestões de alteração dos argumentos *T* das premissas temporais das regras.

A conclusão obtida é incluída no relatório com uma mensagem indicativa de qual seria o valor a atribuir ao argumento *T* da premissa temporal de forma a possibilitar a satisfação dessa premissa.

Terminada a análise deste elemento da lista *LNS2*, de seguida, é verificada a existência de mais elementos nessa lista. Caso existam mais elementos em *LNS2*, cada um desses elementos passa pelos mesmos passos já referidos para o elemento anterior de *LNS2*, caso contrário, termina a análise deste painel, sendo o próximo passo a verificação da existência de mais painéis na lista *LNSPain*.

Caso existam mais painéis na lista *LNSPain*, toda a análise efectuada para o anterior painel repete-se para os restantes painéis existentes em *LNSPain*.

Se não existirem mais painéis em *LNSPain*, é verificada a existência de mais instalações na lista *LNSInt* a serem analisadas.

No caso de existirem mais instalações em *LNSInt*, todos os passos pelos quais a instalação anterior passou são repetidos para todas as instalações existentes em *LNSInt*, caso contrário, todas as instalações de *LNSInt* foram já analisadas, havendo necessidade, de seguida, de verificar a existência de mais factos *regra_bloco(NR, NB, L)* para a regra em análise.

Se existirem mais factos *regra_bloco(NR, NB, L)* relativos à regra em análise, todo o processamento já descrito de análise de factos como este, é repetido para todos os factos *regra_bloco(NR, NB, L)* existentes na base de conhecimento.

Se a nova regra não tiver a ela associada factos *regra_bloco*(*NR*, *NB*, *L*), é incluída no relatório a mensagem “Regra sem situações de disparo sucedidos e sem situações de não disparo da regra devido ao incumprimento de premissas temporais” e é verificada a existência de mais regras a analisar na lista *LR*, sendo o processamento das restantes regras, o mesmo já descrito para a regra anterior.

No anexo B, inclui-se um exemplo com :

- Os factos produzidos pelo SPARSE mediante o tratamento de um ficheiro de incidentes:
 - *regra_nfatos*(*NR*, *NB*, *LFatosS*, *LFatoNS*)
 - *facto_antigo*(*N*, *F*, *T*, *T1*)
- O relatório produzido pelo TEMPUS, resultante do processamento efectuado aos factos anteriores e mediante a escolha por parte do utilizador da análise específica das regras d1 e d21 e das instalações SPN e SVG.

O algoritmo do TEMPUS está ilustrado na tabela 6.2 e o respectivo fluxograma nas figuras 6.3 (a) e 6.3 (b).

Tabela 6.2- Algoritmo de Aprendizagem Automática TEMPUS

Algoritmo de Aprendizagem Automática TEMPUS (*Factos_antigo*, *Regras_nfatos*, *Regras_SPARSE*)

Factos_antigo : Conjunto de factos do tipo *facto_antigo*(*N*, *F*, *T*, *T1*) gerados pelo SPARSE

Regras_nfatos: Conjunto de factos *regra_nfatos*(*NR*, *NB*, *LFatosS*, *LFatoNS*) gerados pelo SPARSE

Regras_SPARSE : Base de Regras do SPARSE

- Escolha das regras (lista *LR*) e instalações (lista *LI*) a analisar

A partir dos factos *regra_nfatos*(*NR*, *NB*, *LFatosS*, *LFatoNS*) obter os blocos das regras da lista *LR* a analisar

- Para cada combinação regra/bloco (*NR/NB*)
 - Criar lista *L* com as situações de disparo e de não disparo da regra
 - Acrescentar à Base de Conhecimento do TEMPUS o facto *regra_bloco*(*NR*, *NB*, *L*)
- Enquanto existirem elementos na lista *LR*, Fazer
 - *NR* ← Primeiro elemento da lista *LR*
 - Se não existem factos *regra_bloco*(*NR*, *NB*, *L*) para a regra *NR* Então
 - Incluir mensagem no relatório : “Regra sem situações de disparo sucedidos e sem situações de não disparo da regra devido ao incumprimento de premissas temporais”

Senão

- Enquanto existirem factos *regra_bloco*(*NR*, *NB*, *L*) relativos à regra *NR*, Fazer
 - Separar a lista *L* do facto *regra_bloco*(*NR*, *NB*, *L*) em situações de disparo (lista *LS*) e situações de não disparo (lista *LNS*) da regra *NR*
 - Se a lista *LS* não tem elementos Então
 - Incluir mensagem no relatório : “Regra sem disparos sucedidos”

Senão

- Se a análise é para todas as instalações existentes

- Lista $LInst \leftarrow LI$

Senão

- Lista $LInst \leftarrow$ Todas as instalações dos elementos da lista LS

Fim Se

- Enquanto existirem elementos na lista $LInst$, Fazer

- $LInst1 \leftarrow$ Primeiro elemento da lista $LInst$
- Lista $LS1 \leftarrow$ Elementos da lista LS com a instalação $LInst1$
- Se a lista $LS1$ não tem elementos Então

- Incluir mensagem no relatório : “Regra sem disparos nesta instalação”

Senão

- Lista $LPain \leftarrow$ Todos os painéis dos elementos da lista $LS1$

- Enquanto existirem elementos na lista $LPain$, Fazer

- $LPain1 \leftarrow$ Primeiro elemento da lista $LPain$
- Lista $LS2 \leftarrow$ Elementos da lista $LS1$ com o painel $LPain1$

- Enquanto existirem elementos na lista $LS2$, Fazer

- $LS2_1 \leftarrow$ Primeiro elemento da lista $LS2$
- Enquanto houverem premissas em $LS2_1$
 - $Prem \leftarrow$ Primeira premissa de $LS2_1$
 - Se $Prem$ é uma premissa temporal Então
 - $Tempo \leftarrow$ Diferença ou módulo da diferença de tempos entre $I1$ e $I2$ da premissa temporal com a sintaxe $Func(I1,I2,T)$
 - Acrescentar à Base de Conhecimento do TEMPUS o facto $cond(NR,NB,Inst,Pain,Func,Tempo)$

Fim Se

- Retirar $Prem$ da lista $LS2_1$

Fim Enquanto

- Lista $LPre \leftarrow$ Functor(es) ($Func$) das premissas temporais existente(s) no facto $cond(NR,NB,Inst,Pain,Func,Tempo)$

- Enquanto existirem elementos na lista $LPre$, Fazer

- $LPre1 \leftarrow$ Primeiro elemento da lista $LPre$
- $Res \leftarrow$ Maior ou menor valor do argumento $Tempo$ dos factos $cond(NR,NB,Inst,Pain,Func,Tempo)$ que tenham o functor $LPre1$.

- Incluir no relatório, conclusão em relação ao valor obtido (*Res*) para o argumento *T* da premissa temporal
- Retirar *LPre1* da lista *LPre*

Fim Enquanto

- Retirar *LS2_1* da lista *LS2*

Fim Enquanto

- Retirar *LPain1* da lista *LPain*

Fim Enquanto

Fim Se

- Retirar *LInst1* da lista *LInst*

Fim Enquanto

Fim Se

- Se a lista *LNS* não tem elementos Então
 - Incluir mensagem no relatório : “Regra sem disparos não sucedidos devido ao incumprimento de premissas temporais”

Senão

- Se a análise é para todas as instalações existentes
 - Lista *LNSInt* $\leftarrow$ *LI*

Senão

- Lista *LNSInt* $\leftarrow$ Todas as instalações dos elementos da lista *LS*

Fim Se

- Enquanto existirem elementos na lista *LNSInt*, Fazer
 - *LNSInt1* $\leftarrow$ Primeiro elemento da lista *LNSInt*
 - Lista *LNSI* $\leftarrow$ Elementos da lista *LNS* com a instalação *LNSInt1*
 - Se a lista *LNSI* não tem elementos Então
 - Incluir mensagem no relatório : “Regra sem disparos não sucedidos nesta instalação”

Senão

- Lista *LNSPain* $\leftarrow$ Todos os painéis dos elementos da lista *LNSI*
- Enquanto existirem elementos na lista *LNSPain*, Fazer
 - *LNSPain1* $\leftarrow$ Primeiro elemento da lista *LNSPain*
 - Lista *LNS2* $\leftarrow$ Elementos da lista *LNSI* com o painel *LNSPain1*
 - Enquanto existirem elementos na lista *LNS2*, Fazer
 - *LNS2_1* $\leftarrow$ Primeiro elemento da lista *LNS2*
 - Se *LS2_1* tem premissa temporal Então
 - *Tempo* $\leftarrow$ Diferença ou módulo da

diferença de tempos entre *I1* e *I2* da premissa temporal com a sintaxe *Func(I1,I2,T)*

- Incluir no relatório, conclusão em relação ao valor obtido (*Tempo*) para o argumento *T* da premissa temporal satisfeita

Fim Se

- $Tempo \leftarrow$ Diferença ou módulo da diferença de tempos entre *I1* e *I2* da premissa temporal não satisfeita com a sintaxe *Func(I1,I2,T)*
- Incluir no relatório, conclusão em relação ao valor obtido (*Tempo*) para o argumento *T* da premissa temporal não satisfeita
- Retirar *LNS2_1* da lista *LNS2*

Fim Enquanto

- Retirar *LNSPain1* da lista *LNSPain*

Fim Enquanto

Fim Se

- Retirar *LNSInt1* da lista *LNSInt*

Fim Enquanto

Fim Se

Fim enquanto

Fim Se

Fim Enquanto



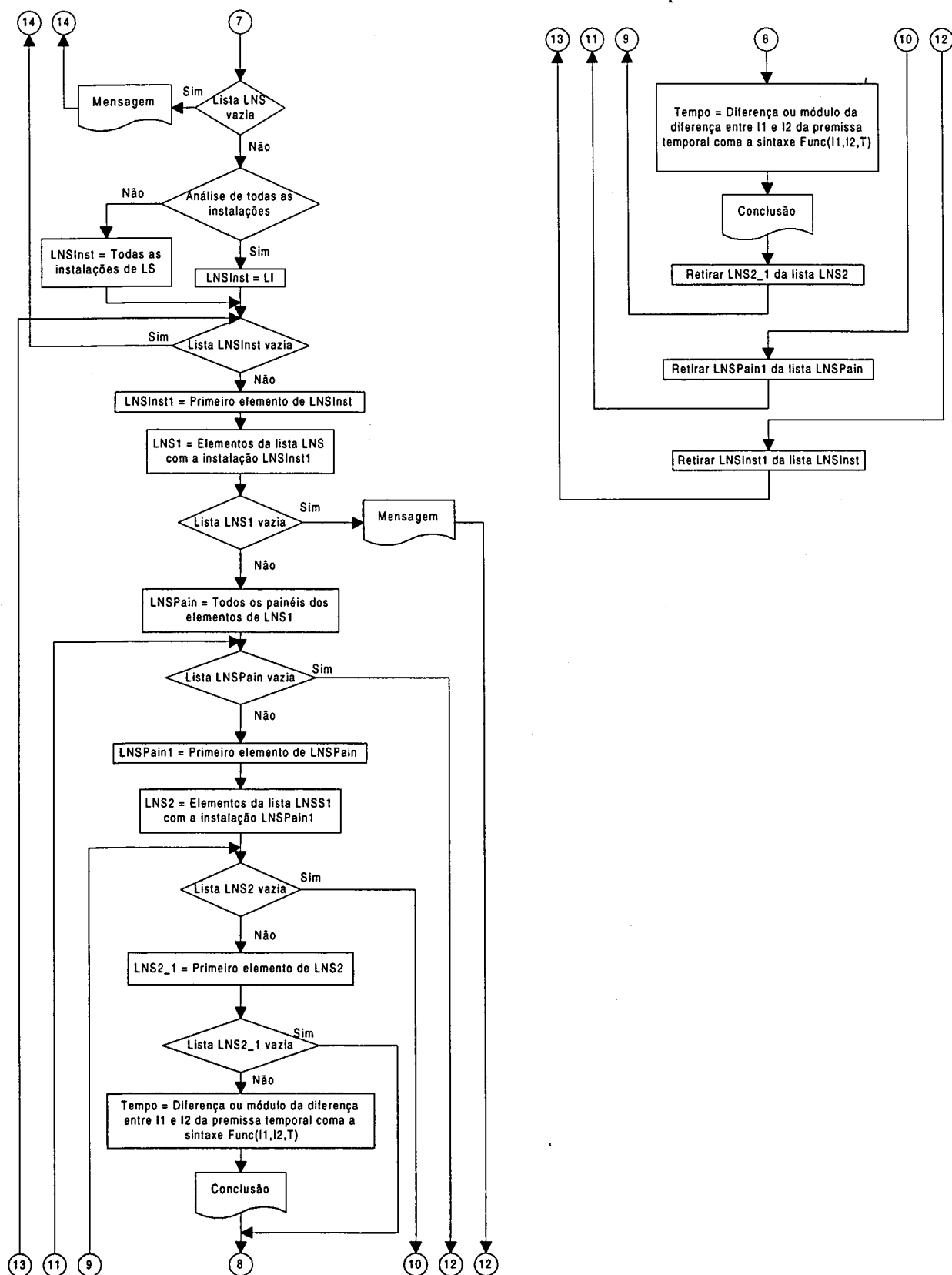


Figura 6.3 (b) – Fluxograma do TEMPUS

6.3.7 Base de Conhecimento do TEMPUS

O algoritmo de Aprendizagem Automática TEMPUS utiliza conhecimento produzido pelo sistema pericial SPARSE e conhecimento que o próprio algoritmo adquire durante o seu processamento.

A Base de Conhecimento do TEMPUS é constituída por:

- Base de Regras do SPARSE.
- Factos produzidos pelo SPARSE:
 - *regra_nfatos*(NR, NB, LFactoS, LFactoNS)
 - *facto_antigo*(N, F, T, T1)
- Factos produzidos pelo TEMPUS:
 - *regra_bloco*(NR, NB, L)
 - *cond*(NR, NB, Inst, Pain, Func, Tempo)

Os constituintes da Base da Base de Conhecimento do TEMPUS foram explicados já anteriormente, sempre que foi oportuno.

6.3.8 Estrutura do Relatório produzido pelo TEMPUS

Com o intuito de ilustrar a estrutura do relatório produzido pelo TEMPUS, considere-se o seguinte exemplo, em que as premissas temporais são identificadas por *Func*(I1, I2, T). Consideremos que o utilizador escolheu para serem analisadas as regras r1, r2 e r3 e as instalações Y e Z. Com este exemplo pretende-se fazer alusão a todas as mensagens possíveis produzidas pelo TEMPUS.

```
#####
REGRA r1:DESCRIÇÃO DA REGRA
[
[
P1 at T1
e
P2 at T2
e
Func(I1, I2, 300)
e
P3 at T3
e
Func(I1, I3, 300)
]
]
==>
[
cria_facto( Nf1,Tnf1),
retira_facto( P2, T2 ,Ttrp2),
cria_facto( Nf2,Tnf2),
retira_facto( P1, T1 ,Ttrp1),
retira_facto( P3, T3 ,Ttrp3),
escreve_mens( MensUtil )
].
```

Bloco 1

SUCEDIDAS

INSTALAÇÃO Y

PAINEL A

Mensagem1.
Mensagem2.

PAINEL 306

Mensagem3.
Mensagem4.

INSTALAÇÃO Z

Regra sem disparos nesta instalação

NÃO SUCEDIDAS

INSTALAÇÃO Y

PAINEL A

Mensagem5.

+++++

Mensagem6.

+++++

Mensagem7.
Mensagem8.

+++++

Mensagem9.
Mensagem10.

PAINEL B

Mensagem11.

+++++

Mensagem12.

+++++

Mensagem13.
Mensagem14.

+++++

Mensagem15.
Mensagem16.

INSTALAÇÃO Z

PAINEL B

Mensagem17.

+++++

Mensagem18.

+++++

Mensagem19.
Mensagem20.

```
#####
REGRA r2:DESCRIÇÃO DA REGRA
[
[
P1 at T1
e
P2 at T2
e
Func(I1, I2, 500)
]
]
==>
[
cria_facto( Nf1,Tnf1),
retira_facto( P2, T2 ,Ttrp2),
cria_facto( Nf2,Tnf2),
retira_facto( P1, T1 ,Ttrp1),
escreve_mens( MensUtil )
].
```

Regra sem situações de disparo sucedidos e sem situações de não disparo da regra devido ao incumprimento de premissas temporais

```
#####
REGRA r3:DESCRIÇÃO DA REGRA
-----
[
[
P1 at T1
e
P2 at T2
e
Func(I1, I2, 1000)
]
ou
[
P1 at T1
e
P3 at T3
e
Func(I1, I3, 1000)
]
]
==>
[
cria_facto( Nf1,Tnf1),
retira_facto( P2, T2 ,Ttrp2),
cria_facto( Nf2,Tnf2),
retira_facto( P1, T1 ,Ttrp1),
escreve_mens( MensUtil )
].
```

Bloco 1

SUCEDIDAS

Regra sem disparos sucedidos

NÃO SUCEDIDAS

INSTALAÇÃO Y

PAINEL A

Mensagem21.

Mensagem22.

Mensagem23.

PAINEL B

Mensagem24.

+++++

Mensagem25.

+++++

Mensagem26.

INSTALAÇÃO Z

Regra sem disparos não sucedidos nesta instalação

Bloco 2

SUCEDIDAS

INSTALAÇÃO Y

PAINEL B

Mensagem27.

INSTALAÇÃO Z

Regra sem disparos nesta instalação.

NÃO SUCEDIDAS

Regra sem disparos não sucedidos devido ao incumprimento de premissas temporais.

#####

O relatório produzido pelo TEMPUS, inclui o LHS de cada regra, no início das conclusões obtidas para cada regra.

No exemplo a regra r1 é constituída por um bloco e duas premissas temporais.

As situações de disparo das regras, estão documentadas no relatório por "SUCEDIDAS...".

No painel A da instalação Y existem duas mensagens. A *Mensagem1* contém uma sugestão para alteração do argumento T da primeira premissa temporal da regra enquanto a *Mensagem2* contém uma sugestão para alteração do argumento T da segunda premissa temporal da regra.

No painel B da mesma instalação, as mensagens *Mensagem3* e *Mensagem4* têm a mesma função das referidas anteriormente.

Na instalação Z não ocorreram disparos da regra e isso deveu-se ao facto do incumprimento de premissas não temporais.

As situações de não disparo da regra, estão documentadas por "NAO SUCEDIDAS...".

Na regra r1 existem oito situações de não disparo da regra. As situações de não disparo desta regra podem ser de dois tipos pois a regra é constituída por duas condições temporais.

A primeira possibilidade consiste no não cumprimento da primeira premissa temporal, dando origem a uma só conclusão. Esta conclusão tem a indicação de qual deveria ser o novo valor do argumento T da primeira premissa temporal para que essa premissa tivesse sucesso. No exemplo as mensagens *Mensagem5*, *Mensagem6*, *Mensagem11*, *Mensagem12*, *Mensagem17* e *Mensagem18* são conclusões desse tipo.

Outra possibilidade consiste em se ter cumprido a primeira premissa temporal e não se ter cumprido a segunda premissa temporal da regra. Estas situações dão origem a duas conclusões, a primeira com uma sugestão de qual poderia ser o novo valor do argumento T da primeira premissa temporal, e a segunda com uma sugestão de qual deveria ser o valor do argumento T da segunda premissa temporal, para que essa premissa tivesse sucesso.

No exemplo, as mensagens (*Mensagem7, Mensagem8*), (*Mensagem9, Mensagem10*), (*Mensagem13, Mensagem14*), (*Mensagem15, Mensagem16*) e (*Mensagem19, Mensagem20*) são conclusões desse tipo.

A regra r2, não teve situações de disparo sucedidos, nem situações de não disparo da regra por não se ter cumprido a premissa temporal.

A regra r3 é uma regra com dois blocos em que cada bloco contém só uma premissa temporal. No bloco 1 desta regra, não existiram disparos sucedidos.

Relativamente às situações de não disparo no bloco 1 da regra r3, verifica-se que no painel A da instalação Y foram incluídas três conclusões (*Mensagem21, Mensagem22, Mensagem23*) enquanto que no painel B da mesma instalação foram incluídas três conclusões (*Mensagem24, Mensagem25, Mensagem26*).

Estas conclusões são sugestões de alteração do argumento T da premissa temporal da regra para que essa premissa pudesse ter sucesso.

Quanto à instalação Z do bloco 1, verifica-se que não existiram disparos não sucedidos, pois não houveram premissas temporais por cumprir.

Relativamente às situações de disparo da regra r3, verifica-se que na instalação Y do bloco 2 existe uma conclusão com uma sugestão de alteração do argumento T da premissa temporal da regra (*Mensagem27*). Na instalação Z desse mesmo bloco não ocorreram situações de disparo da regra.

A regra r3 no bloco 2 não teve situações de não disparo da regra, pois não houve premissas temporais por cumprir.

6.4 EDIREGRA- Editor de Regras do SPARSE

O EDIREGRA é um Editor de Regras construído em Visual Basic, que possibilita a alteração dos tempos associados a às premissas temporais das regras do SPARSE e a criação de novas regras específicas para algumas instalações e/ou painéis.

Mediante as conclusões obtidas para cada regra pelo algoritmo de Aprendizagem Automática (TEMPUS) afiguram-se duas situações:

- Situações em que, dada a uniformidade dos tempos obtidos nas várias instalações e painéis, não se justifica a criação de mais regras para além das existentes, mas antes a alteração das regras existentes.
- Situações em que se obtêm tempos (conclusões) relativos às premissas temporais muito diferentes em algumas instalações ou painéis e que justificam a criação de uma nova regra para tratar essas instalações ou painéis.

O editor de regras prevendo estas duas situações, possibilita:

- A alteração da regra, repercutindo essa alteração para todas as tentativas de disparo da regra em todas as instalações da REN.
- A inserção de regras na Base de Regras do SPARSE que espelhem o comportamento temporal de determinadas instalações ou painéis.

A janela principal do editor de regras (Figura 6.3) é constituída por duas partes:

- A regra propriamente dita, que inclui as premissas a serem cumpridas para o seu disparo (RHS) e o que deve ser feito caso a regra dispare (LHS).
- As conclusões obtidas pelo algoritmo de Aprendizagem Automática (TEMPUS) para essa regra.

As conclusões foram retiradas do relatório produzido pelo TEMPUS e incluem os disparos (Sucedidas) e os não disparos (Não sucedidas) da regra. Por cada bloco da regra são incluídas as instalações, os painéis e as conclusões obtidas relativamente a cada premissa temporal da regra.

Na figura 6.3, existem botões com funções específicas à parte da figura onde se encontram.

Os dois botões iniciais ao lado da regra permitem a visualização de outras regras da Base de Regras do SPARSE e as respectivas conclusões obtidas.

As janelas do tipo das figuras 6.4 e 6.5, têm um botão que permite aceitar os novos valores introduzidos, um botão que permite repor os valores originais e um botão que permite voltar à Figura 6.3

Ao premir o botão “Aceitar Novos Valores”, é enviada uma mensagem de que esses valores foram aceites, mas é de notar, que a actualização destes valores só se verifica em memória e não ainda em disco. Esta opção visa eventuais possíveis alterações a esses valores realizadas por parte do perito, antes de abandonar o editor de regras. Em contrapartida, a Figura 6.3 tem o botão “Gravar” que permite guardar em disco todas as alterações efectuadas às regras da Base de Regras, bem como, as novas regras entretanto criadas.

O exemplo da figura 6.4, ilustra a situação em que não se justifica a criação de uma nova regra, mas antes, a alteração da regra d1, com a atribuição do tempo 114 à primeira premissa temporal e do tempo 74 à segunda premissa temporal da regra.

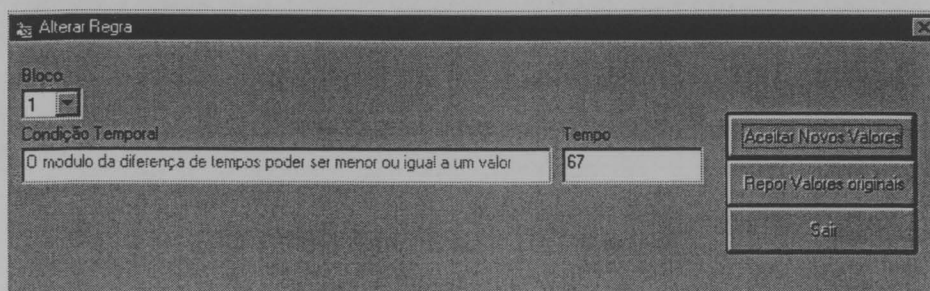


Figura 6.5 – Janela Alterar Regra do EDIREGRA (para regras com uma premissa temporal)

O exemplo da figura 6.5, ilustra a situação em que também não se justifica a criação de uma nova regra, mas antes a alteração da regra d3, com a atribuição do tempo 67 à premissa temporal.

Na Figura 6.3 existe o botão “Ver Alterações”, que permite visualizar as alterações já efectuadas às regras da Base de Regras do SPARSE. Como exemplo, a figura 6.6 (“Ver Alterações”), permite visualizar as alterações efectuadas à base de Regras do SPARSE. No exemplo, pode-se constatar as alterações às regras já efectuadas. A regra d1 sofreu uma actualização nas suas duas premissas temporais (de 114 e 74, respectivamente) e a regra d3 uma actualização nos seus dois blocos. A premissa temporal no bloco 1 assumiu o novo valor de 67, enquanto que no bloco 2, o novo valor de 20 milésimas de segundo. Os separadores entre os campos existentes nesta janela são móveis, permitindo a visualização das condições temporais.

Nesta janela, existe ainda a possibilidade de rectificar qualquer alteração já efectuada a uma determinada regra. Para isso, é necessário escolher o registo a alterar e premir o botão “Alterar”. Este botão leva-nos a uma outra janela (Figura 6.7), que possibilita somente a alteração do(s) tempo(s) da regra em questão. O botão “Aceitar Novos Valores” efectiva essa alteração em memória e o botão “Sair” permite-nos voltar à janela “Ver alterações” (Figura 6.6).

Na Figura 6.6, o botão “Eliminar” permite eliminar uma alteração já realizada a qualquer regra, sendo necessário escolher o registo previamente. Esta eliminação só é efectuada mediante confirmação. O botão “Sair” permite voltar à janela principal do editor de regras (Figura 6.3).

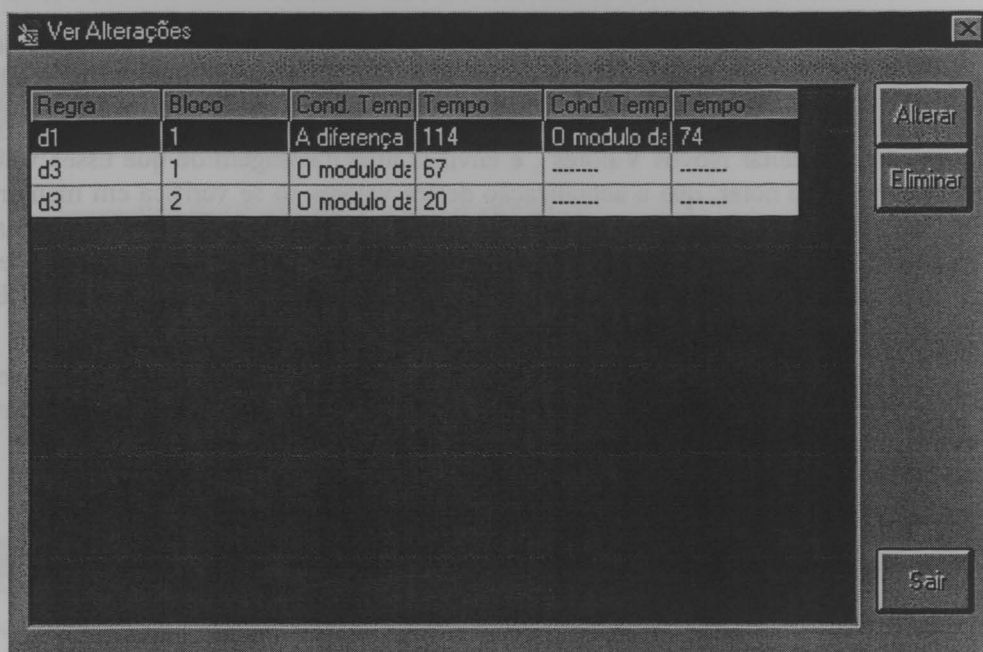


Figura 6.6 – Janela Ver Alterações do EDIREGRA

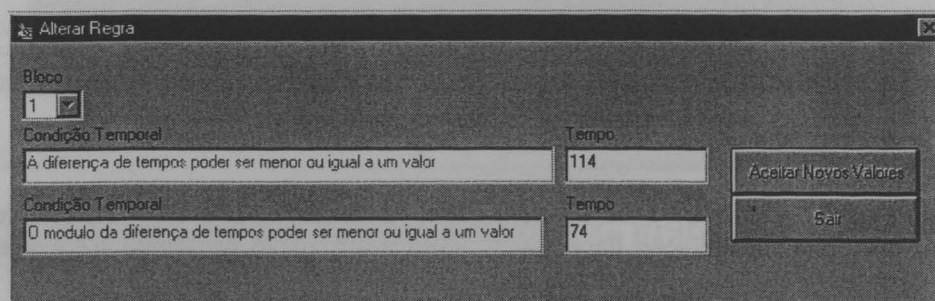


Figura 6.7 – Janela Alterar Regra do EDIREGRA

Na segunda parte da janela principal do editor de regras (Figura 6.3), existem dois botões associados a novas regras a acrescentar à Base de Regras.

O botão "Inserir" permite criar uma nova regra na Base de Conhecimento do SPARSE. Para tal, na janela "Nova Regra" (figura 6.8) é necessário introduzir a instalação, eventualmente o painel, e o(s) novo(s) tempo(s) associado(s) à(s) premissa(s) temporal(is) da regra. No caso da regra d3, como tem dois blocos é necessário escolher o bloco da regra.

Na regra d3, no caso de existir a necessidade de criar uma regra para uma determinada instalação ou painel associada a um só bloco da regra, a nova regra será constituída por um só bloco.

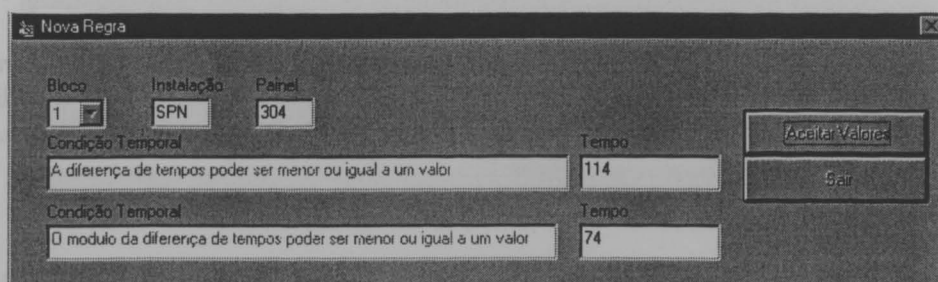


Figura 6.8 – Janela Nova Regra do EDIREGRA

A tentativa de inserção de uma regra com o mesmo bloco, instalação, painel e tempo(s) da(s) premissa(s) temporal(is) de uma regra já existente é validada. Caso isso aconteça é enviada uma mensagem ao perito de que já existe uma regra com essas características, não sendo portanto criada essa regra. Na Figura 6.8, o botão “Aceitar Valores”, efectiva a criação da regra em memória, enquanto que o botão “Sair” permite voltar à figura 6.3.

O botão “Ver Inserções” permite aceder à janela “Ver Inserções” para visualizar todas as regras entretanto criadas desde o início da utilização do editor de regras. No exemplo da figura 6.9, verifica-se que foram criadas duas novas regras, associadas ao bloco 1 da regra d1, a primeira para o painel 304 da instalação SPN enquanto que a segunda para o painel 306 da mesma instalação. Na figura 6.9, o botão “Sair” permite voltar à figura 6.3.

Na figura 6.9, o botão “Alterar” permite aceder à janela “Alterar Regra” (figura 6.10), que permite alterar o(s) tempo(s) da(s) premissa(s) temporal(is) de uma determinada regra já criada. Para tal, é necessário escolher primeiro a regra a alterar. O botão “Sair” no exemplo da figura 6.10, permite voltar à janela “Ver Inserções”.

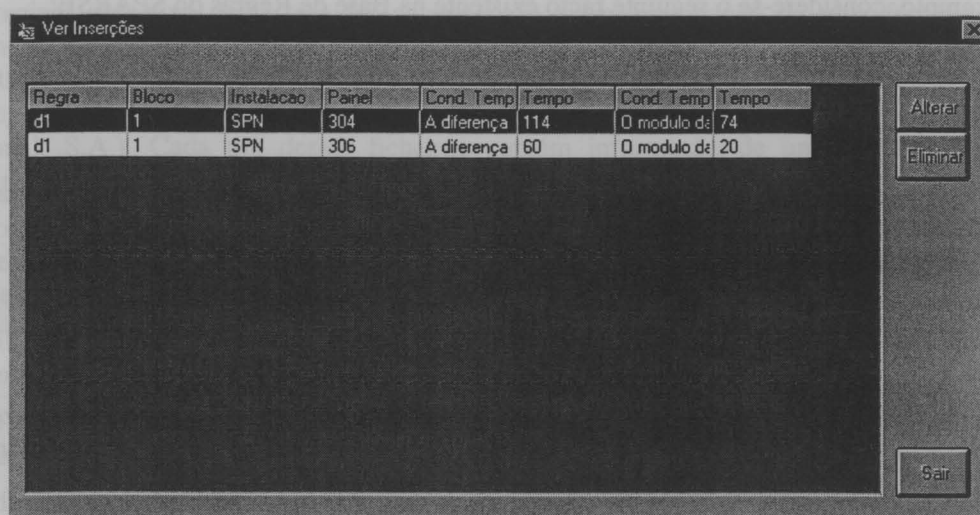


Figura 6.9 – Janela Ver Inserções do EDIREGRA

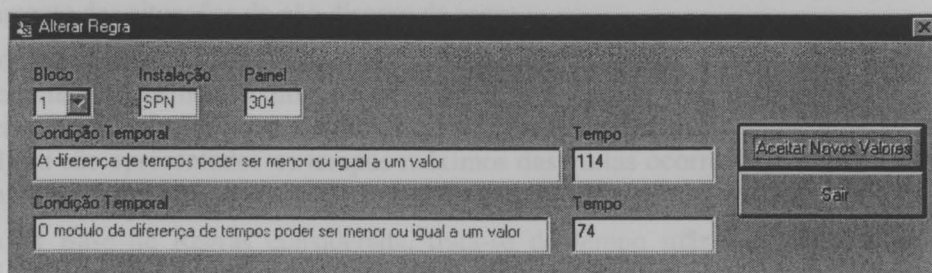


Figura 6.10 – Janela Alterar Regra do EDIREGRA

Quando o perito terminar todas as alterações e criações de regras, o botão “Gravar” permite gravar em disco (ficheiro) essas mesmas regras. De notar, que as regras que não foram alteradas permanecem intactas na nova Base de Regras.

Para cada regra acrescentada à Base de Regras do SPARSE, é necessário alterar o meta-conhecimento existente na Base de Regras, associado à regra que deu origem à nova regra.

Como já foi referido, o formato genérico dos factos relativos ao meta-conhecimento é:

`facto_dispara_regras( F, [(R1, TiR1, Tfr1), (R2, TiR2, Tfr2), ... ] ).`

Supondo que foi acrescentada à Base de Regras do SPARSE uma nova regra (regra R1_1) associada à regra R1, é necessário pesquisar todas as ocorrências da regra R1 em todos os factos `facto_dispara_regras` do meta-conhecimento existente. Para cada um dos factos `facto_dispara_regras` em que apareça a regra R1 na lista `[(R1, TiR1, Tfr1), (R2, TiR2, Tfr2), ... ]`, é necessário acrescentar antes do elemento `(R1, TiR1, Tfr1)`, o elemento `(R1_1, TiR1, Tfr1)`.

O novo elemento `(R1_1, TiR1, Tfr1)` tem que ser inserido na lista antes de `(R1, TiR1, Tfr1)`, pois o Sistema Pericial SPARSE, terá que tentar disparar a regra R1_1 antes da regra R1. De facto, como a regra R1 é uma regra genérica para todas as instalações, a tentativa do seu disparo antes de R1_1, que é uma regra específica a um painel ou instalação, poderia falsear a conclusão a tirar relativa ao disparo da regra.

Como exemplo, considere-se o seguinte facto existente na Base de Regras do SPARSE:

```
facto_dispara_regras(mensagem(_,_, [Inst1,Painel1,_], '>>>DISPARO', '01'), [(d1,30,50), (d2,31,51), (d3,52,52), (x,12000,12000)]).
```

Caso tivesse sido acrescentada a regra d1_1 à Base de Regras, era necessário alterar o `facto_dispara_regras` anterior e todos onde ocorresse a regra d1, de forma a ficar :

```
facto_dispara_regras(mensagem(_,_, [Inst1,Painel1,_], '>>>DISPARO', '01'), [(d1_1,30,50), (d1,30,50), (d2,31,51), (d3,52,52), (x,12000,12000)]).
```

No caso da Base de Regras do SPARSE ter sofrido alguma alteração, é necessário fazer a verificação e validação de toda a nova Base de Regras. Esta tarefa é realizada por um módulo de verificação e validação (VERITAS) já desenvolvido anteriormente.

Capítulo 7

Resultados obtidos pelo TEMPUS

7.1 Introdução

O principal objectivo desta tese foi a concepção e implementação de uma ferramenta de Aprendizagem Automática, com o intuito de analisar as premissas temporais das regras do SPARSE. Para a obtenção de resultados relativos ao comportamento das premissas temporais das regras, foram processados doze ficheiros mensais de incidentes da REN (Rede Eléctrica Nacional, S.A.). Cada um destes ficheiros contém informação de um determinado mês, permitindo, assim, uma análise de todos os incidentes verificados durante um ano.

Este período de tempo foi considerado suficiente, pois é representativo do comportamento da rede eléctrica nacional durante um período de tempo relativamente longo, abrangendo uma grande quantidade de incidentes.

A análise destes incidentes possibilitou chegar a determinadas conclusões e sugerir a alteração dos tempos assumidos inicialmente nas regras da Base de Regras do SPARSE.

Como já foi referido, para cada regra do SPARSE, foi realizada uma análise para cada instalação e, dentro de cada instalação, para cada painel.

De referir que, dada a fiabilidade do funcionamento da rede da REN, algumas instalações não registaram incidentes. Assim, os resultados aqui obtidos referem apenas parte das instalações e painéis, constituindo um subconjunto do universo de instalações e painéis existentes.

Os resultados obtidos pela análise efectuada às regras do SPARSE encontram-se no anexo C, sendo apresentados por regra. Em cada regra, as situações de disparo da regra são apresentadas separadamente das situações de não disparo da regra.

Se num determinado mês ocorrer mais do que um disparo de uma determinada regra (relativos a um determinado painel de uma instalação), é apresentado o tempo máximo das várias ocorrências nesse mês na premissa temporal da regra (caso a regra tenha uma só premissa temporal) ou são apresentados os tempos máximos das várias ocorrências nesse mês nas duas premissas temporais da regra (caso a regra tenha duas premissas temporais).

No caso da Base de Regras considerada, trata-se do tempo máximo porque as premissas temporais existentes nas regras são do tipo `dif_tempos_meior_ou_igual` (diferença de tempos menor ou igual entre os tempos associados a duas premissas da regra) e `mod_dif_tempos_meior_ou_igual` (módulo da diferença de tempos menor ou igual entre os tempos associados a duas premissas da regra) .

Nas situações de não disparo da regra, apresentam-se os resultados de todas as situações ocorridas, existindo registos de várias ocorrências em determinados meses.

Para auxílio à interpretação dos resultados são apresentados alguns indicadores em relação a cada premissa temporal : máximo, mínimo, média, moda e desvio padrão.

De modo a facilitar a interpretação dos resultados, em cada premissa temporal das regras, tanto nas situações de disparo como nas situações de não disparo da regra, são definidos intervalos para a análise dos tempos obtidos, com a indicação do número de ocorrências e a percentagem de ocorrências em cada intervalo.

7.2 Resultados Obtidos

7.2.1 Regra d1 – Disparo Trifásico Com Disjuntor Em Movimento

A regra d1 tem a seguinte estrutura:

```
regra d1 : 'DISPARO TRIFÁSICO COM DISJUNTOR EM MOVIMENTO' :
[
[
mensagem(Data1,Hora1,[Inst1,Painell1,[Inst2,NL]], '>>>DISPARO',
'01') at T1
e
disj(Data2,Hora2,Inst1,Painell1,_,nao_identificado,aberto) at T2
e
condicao(dif_tempos_menor_ou_igual(T2,T1,300))
e
mensagem(Data3,Hora3,[Inst1,Painell1,[Inst2,NL,_], 'DISJUNTOR',
'00') at T3
e
condicao(mod_dif_tempos_menor_ou_igual(T1,T3,300))
]
]
==>
[
cria_facto(disparo(Data1,Hora1,Inst1,Painell1,Inst2,NL,trifásico,
nao_identificado,T2),T1),
retira_facto(disj(_,_,Inst1,Painell1,_,nao_identificado,aberto),
_,T2),
cria_facto(disj(Data2,Hora2,Inst1,Painell1,_,disparo,aberto),T2),
retira_facto(mensagem(Data1,Hora1,[Inst1,Painell1,[Inst2,NL]],
'>>>DISPARO','01'),T1,T1),
retira_facto(mensagem(Data3,Hora3,[Inst1,Painell1,[Inst2,NL,_]],
'DISJUNTOR','00'),T3,T3),
retira_facto(disj_aberto_disparo(Inst1,Painell1),_,T1),
cria_facto(disj_aberto_disparo(Inst1,Painell1),T1)
].
```

A regra d1, corresponde a situações de disparo trifásico de um disjuntor, com a recepção de uma mensagem relativa a disjuntor em movimento.

A interpretação da regra d1 é a seguinte :

SE

Existe uma mensagem de disparo do disjuntor numa determinada subestação e painel

E

O mesmo disjuntor abriu por um motivo ainda por identificar

E

A diferença de tempos entre a abertura e o disparo é menor do que 300 milésimas de segundo

E

Existe uma mensagem de disjuntor em movimento para este disjuntor

E

O módulo da diferença de tempo entre a abertura e o movimento é menor ou igual a 300 milésimas de segundo

ENTÃO

Conclui-se que existiu um disparo trifásico desse disjuntor de tipo não identificado

E

Regista-se o motivo do disparo para a abertura do disjuntor

E

Retiram-se as mensagens SCADA e as conclusões anteriores que originaram a conclusão (pois tornaram-se obsoletas)

As referências a factos do tipo `disj_aberto_disparo/2` não são relevantes para a análise da regra, prendendo-se com a identificação de incidentes.

As restrições temporais da regra são directamente mapeáveis nos tempos médios do fenómeno físico que a regra pretende representar (disparo trifásico de um disjuntor).

A regra d1 é constituída por duas premissas temporais, cada uma delas com o tempo assumido de 300 milésimas de segundo.

A regra d1 apresenta situações de disparo e de não disparo da regra. As situações de não disparo da regra podem ser de dois tipos. A primeira possibilidade consiste no incumprimento da primeira premissa temporal da regra e a segunda possibilidade consiste no cumprimento da primeira premissa temporal da regra mas com incumprimento da segunda premissa temporal da regra.

Nas situações de disparo da regra, a primeira premissa temporal da regra registou tempos que se concentraram principalmente entre 0 e 240 milésimas de segundo (393 registos mensais correspondendo a 99.24% dos registos), existindo um registo no intervalo]240-270] e dois registos no intervalo]270-300].

Quanto à segunda premissa temporal da regra, os tempos concentraram-se principalmente entre 0 e 180 milésimas de segundo (385 registos mensais correspondendo a 97.23% dos registos), distribuindo-se os restantes 11 registos regularmente pelos quatro intervalos considerados entre 180 e 300 milésimas de segundo.

Nas situações de não disparo da regra devido ao incumprimento da primeira premissa temporal verifica-se que 36.66% dos registos (11 registos) se concentram entre 300 e 420 milésimas de segundo, 33.3% dos registos (10 registos) concentram-se entre 450 e 540 milésimas de segundo e os restantes 30% dos registos (9 registos) acima dos 600 milésimas de segundo.

Nas situações de não disparo da regra devido ao incumprimento da segunda premissa temporal verifica-se que 4.49% dos registos (4 registos) se concentram entre 300 e 360 milésimas de segundo, 11.22% dos registos (10 registos) concentram-se entre 420 e 600 milésimas de segundo e os restantes 84.27% dos registos (75 registos) acima das 600 milésimas de segundo.

Numa primeira análise, os resultados relativos à primeira premissa temporal levam à conclusão que provavelmente alguns casos que podem corresponder à situação descrita por esta regra foram classificados como situações de não disparo da regra. Este raciocínio baseia-se no facto de que o número de registos nos dois últimos intervalos de análise para as situações de disparo da regra (1 registo no intervalo]240-270] e 2 registos no intervalo]270-300]) são mais ou menos da mesma ordem de grandeza que o número de registos nos quatro primeiros intervalos de análise para as situações de não disparo da regra (3 registos no intervalo]300-330], 3 registos no intervalo]330-360], 1 registo no intervalo]360-390] e 4 registos no intervalo]390-420]). Por sua vez, o intervalo]420-450] milésimas de segundo não

registra nenhuma situação de não disparo, os intervalos compreendidos entre]450-540] milésimas de segundo registam 33.33% de situações de não disparo da regra (10 registos) e acima das 600 milésimas de segundo existem 30% de situações de não disparo da regra (9 registos). Neste último intervalo, os valores obtidos foram valores excessivamente elevados, à excepção de dois registos, um com um valor de 620 milésimas de segundo e outro com um valor de 868 milésimas de segundo.

Por outro lado, para todos os registos de situações de não disparo referidos (com excepção dos registos existentes no painel 220 da instalação SCG (868 milésimas de segundo), no painel 607 da instalação SCT (620 milésimas de segundo) e no painel 639 da instalação SPA (396 milésimas de segundo)) existem também situações de disparo da regra nessas mesmas instalações e painéis com tempos bastante baixos e regulares, o que pode levar a concluir que estas situações de não disparo que poderiam ser consideradas como situações de disparo da regra não devem na verdade ser consideradas como tal.

Perante a interpretação destes resultados, o perito vê-se obrigado a fazer uma análise muito detalhada a cada instalação e painel, verificando todas as suas particularidades, de forma a poder concluir qual o tempo a atribuir à primeira premissa temporal da regra d1 e, eventualmente, a seguir, a sugerir criar novas regras com tempos diferentes na primeira premissa temporal para determinadas instalações e painéis.

À semelhança da primeira premissa temporal da regra, numa primeira análise, os resultados relativos à segunda premissa temporal levam à conclusão que provavelmente alguns casos que podem corresponder à situação descrita por esta regra foram classificados como situações de não disparo da regra. Este raciocínio baseia-se no facto de que o número de registos nos dois últimos intervalos de análise para as situações de disparo da regra (3 registos no intervalo]240-270] e 2 registos no intervalo]270-300]) são da mesma ordem de grandeza que o número de registos nos dois primeiros intervalos de análise para as situações de não disparo da regra (3 registos no intervalo]300-330] e 1 registo no intervalo]330-360]. Por sua vez, os intervalos]420-450],]450-480],]480-510] e]420-450] milésimas de segundo apresentam uma situação de não disparo, o intervalo entre]510-540] milésimas de segundo apresenta 4 situações de não disparo da regra e acima das 600 milésimas de segundo existem 75 situações de não disparo da regra (84.27%). Neste último intervalo, os valores obtidos foram valores excessivamente elevados, à excepção de sete registos, que variam entre 620 e 858 milésimas de segundo.

Por outro lado, para todos os registos de situações de não disparo referidos (com excepção do registo existente no painel 626 da instalação SFA com 320 milésimas de segundo) existem também situações de disparo da regra nessas mesmas instalações e painéis com tempos bastante baixos, o que pode levar a concluir que estas situações de não disparo que poderiam ser consideradas como situações de disparo da regra não devem, na verdade, ser consideradas como tal.

No painel 626 da instalação SFA, existe um registo de uma situação de não disparo com o tempo de 320 milésimas de segundo, que constitui um valor muito próximo do tempo obtido numa situação considerada como disparo da regra d1. Essa situação de disparo tem o tempo limite de 300 milésimas de segundo. O perito poderá, eventualmente, considerar a criação de uma nova regra para este painel desta instalação caso chegue à conclusão que o valor 320 milésimas de segundo é um valor tolerável.

Mediante este quadro, o perito deverá fazer uma análise muito detalhada a cada instalação e painel, verificando todas as suas particularidades, de forma a poder concluir qual o tempo a atribuir à segunda premissa temporal da regra d1 e, eventualmente, a sugerir criar novas regras para determinadas instalações e painéis com tempos diferentes na segunda premissa temporal.

Na regra d1 existem registos de situações de disparo em 34 instalações. Pode-se constatar que 24 destas instalações apresentam, em ambas as premissas temporais da regra, valores inferiores a 200 milésimas de segundo. A excepção são as instalações SAM, SED, SFA, SPM,

SPN, SRM, SRR, SSB, SVG e SVM que, pontualmente, atingem valores entre 200 e 300 milésimas de segundo.

Nas 24 instalações referidas, existem mesmo instalações para as quais, dado o grande volume de registos obtidos de situações de disparo e não apresentando situações de não disparo, faz sentido criar novas regras que atendam às particularidades de cada uma dessas instalações.

De notar que, mesmo em algumas instalações que registaram valores superiores a 200 milésimas de segundo, essas situações ocorreram só pontualmente em alguns painéis. As instalações SFA e SPN são um bom exemplo destas situações. Como nestas duas instalações existem para outros painéis bastantes registos, todos eles bastante regulares e a não ultrapassar as 200 milésimas de segundo, provavelmente faz sentido criar apenas novas regras para os painéis em que se verificaram valores acima das 200 milésimas de segundo.

7.2.2 Regra d2 – Disparo Trifásico Sem Disjuntor Em Movimento

A regra d2 tem a seguinte estrutura:

```
regra d2 : 'DISPARO TRIFÁSICO SEM DISJUNTOR EM MOVIMENTO' :
[
[
mensagem(Data1,Hora1,[Inst1,Painel1,[Inst2,NL]], '>>>DISPARO',
'01') at T1
e
disj(Data2,Hora2,Inst1,Painel1,_,nao_identificado,aberto) at T2
e
condicao(dif_tempos_menor_ou_igual(T2,T1,300))
]
]
==>
[
cria_facto(disparo(Data1,Hora1,Inst1,Painel1,Inst2,NL,trifásico,
nao_identificado,T2),T1),
retira_facto(disj(,_,Inst1,Painel1,_,nao_identificado,aberto),
_,T2),
cria_facto(disj(Data2,Hora2,Inst1,Painel1,_,disparo,aberto),T2),
retira_facto(mensagem(Data1,Hora1,[Inst1,Painel1,[Inst2,NL]],
'>>>DISPARO','01'),T1,T1),
retira_facto(disj_aberto_disparo(Inst1,Painel1),_,T1),
cria_facto(disj_aberto_disparo(Inst1,Painel1),T1)
].
```

A regra d2 tenta identificar um disparo de um disjuntor nas suas três fases, sem a chegada de uma mensagem de disjuntor em movimento.

A regra d2 é constituída pelas três primeiras premissas da regra d1, isto é, são excluídas as duas últimas premissas da regra d1, relacionadas com a chegada ao sistema de uma mensagem de disjuntor em movimento.

A interpretação da regra é a mesma da anterior, com a exceção de informação relacionada com a mensagem de disjuntor em movimento.

A regra d2 tem o tempo assumido de 300 milésimas de segundo na sua premissa temporal e nos resultados obtidos, apresenta situações de disparo e situações de não disparo da regra.

Nas situações de disparo da regra, registaram-se tempos que se concentraram principalmente entre 0 e 240 milésimas de segundo (233 registos mensais correspondendo a 98.31% dos registos), existindo quatro registos no intervalo]270-300].

Dado que a regra d2 tem a mesma constituição do que a regra d1 até à sua primeira premissa temporal, os resultados obtidos nas situações de não disparo da regra d2 são os mesmos obtidos nas situações de não disparo da regra d1 para a sua primeira premissa temporal. Concretamente, verifica-se que 36.66% dos registos (11 registos) se concentram entre 300 e 420 milésimas de segundo, 33.3% dos registos (10 registos) entre 450 e 540 milésimas de segundo e os restantes 30% dos registos (9 registos) acima das 600 milésimas de segundo.

À semelhança do que aconteceu para a regra d1, numa primeira análise, os resultados relativos à primeira premissa temporal levam à conclusão que provavelmente alguns casos que podem corresponder à situação descrita por esta regra foram classificadas como situações de não disparo da regra.

Mas, por outro lado, para todos os registos de situações de não disparo referidos (com excepção do registo no painel 220 da instalação SCG com 868 milésimas de segundo), existem também situações de disparo da regra nessas mesmas instalações e painéis com tempos bastante regulares e baixos, o que pode levar a concluir que estas situações de não disparo que poderiam ser consideradas como situações de disparo da regra não devem na verdade ser consideradas como tal.

Como na regra anterior, perante estes resultados, o perito vê-se obrigado a fazer uma análise muito detalhada a cada instalação e painel, verificando todas as suas particularidades, de forma a poder concluir qual o tempo a atribuir à premissa temporal da regra d2 e eventualmente, a sugerir criar novas regras para determinadas instalações e painéis com tempos diferentes na premissa temporal.

Na regra d2 existem registos de situações de disparo em 39 instalações. Pode-se constatar que 33 destas instalações apresentam valores relativos à premissa temporal da regra inferiores a 200 milésimas de segundo. A excepção são as instalações SBL, SED, SPA, SPN, SVM e SZR que, pontualmente, atingem valores entre 200 e 300 milésimas de segundo.

Nas 33 instalações referidas, existem mesmo instalações para as quais dado o grande volume de registos obtidos de situações de disparo e não apresentando situações de não disparo, faz sentido criar novas regras que atendam às particularidades de cada uma dessas instalações.

De referir que, mesmo em algumas instalações que registaram valores superiores a 200 milésimas de segundo, essas situações ocorreram só pontualmente em alguns painéis. A instalação SPN é um bom exemplo destas situações. Como nesta instalação existem para outros painéis bastantes registos, todos eles bastante regulares e a não ultrapassar as 200 milésimas de segundo, provavelmente faz sentido criar apenas novas regras para os painéis em que se verificaram valores acima das 200 milésimas de segundo.

7.2.3 Regra d3 – Disparo Monofásico

A regra d3 tem a seguinte estrutura:

```
regra d3 : 'DISPARO MONOFÁSICO' :
[
[
mensagem(Data1,Hora1,[Inst1,Painel1,[Inst2,NL]], '>>>DISPARO',
'01') at T1
e
mensagem(Data2,Hora2,[Inst1,Painel1,[Inst2,NL|_]], 'DISJUNTOR',
'00') at T2
e
condicao(mod_dif_tempos_menor_ou_igual(T2,T1,300))
]
ou
```

```
[
mensagem(Data1,Hora1,[Inst1,Painel1,[Inst2,NL]], '>>>DISPARO',
'01') at T1
e
mensagem(Data2,Hora2,[Inst1,Painel1,[Inst2,NL]], 'DISJUNTOR',
'00') at T2
e
condicao(mod_dif_tempos_menor_ou_igual(T2,T1,300))
]
==>
[
cria_facto(disparo(Data1,Hora1,Inst1,Painel1,Inst2,NL,
monofásico,nao_identificado,T2),T1),
retira_facto(mensagem(Data1,Hora1,[Inst1,Painel1,[Inst2,NL]],
'>>>DISPARO','01'),T1,T1),
retira_facto(mensagem(Data2,Hora2,[Inst1,Painel1,[Inst2,NL|_]],
'DISJUNTOR','00'),T2,T2),
retira_facto(disj_aberto_disparo(Inst1,Painel1),_,T1),
cria_facto(disj_aberto_disparo(Inst1,Painel1),T1)
].
```

A regra d3 tenta identificar um disparo monofásico, ou seja, um disparo em que o dispositivo não abriu totalmente.

A interpretação da regra d3 é a seguinte :

SE

Existe uma mensagem de disparo do disjuntor numa determinada subestação e painel

E

Existe uma mensagem de disjuntor em movimento para este disjuntor

E

O módulo da diferença de tempo entre o disparo e o movimento é menor ou igual a 300 milésimas de segundo

ENTÃO

Conclui-se que existiu um disparo monofásico desse disjuntor de tipo não identificado

E

Retiram-se as mensagens SCADA e as conclusões anteriores que originaram a conclusão (pois tornaram-se obsoletas)

A restrição temporal da regra é directamente mapeável no tempo médio do fenómeno físico que a regra pretende representar - disparo monofásico de um disjuntor.

A regra d3 tem uma premissa temporal com o tempo assumido de 300 milésimas de segundo e apresenta situações de disparo e situações de não disparo da regra.

Os resultados obtidos indicam que existem situações de disparo da regra em todos os intervalos considerados para a análise, verificando-se uma maior incidência nos primeiros intervalos, diminuindo sucessivamente a frequência de disparos à medida que se avança nos intervalos. O último intervalo ([270-300]) regista apenas 0.54% das ocorrências, que corresponde a apenas 2 dos 369 disparos verificados.

Nas situações de não disparo da regra verifica-se um número de ocorrências semelhante em todos os intervalos, à excepção do intervalo [420-450], em que não existem registos e no

último intervalo considerado, já que este abrange todos os tempos superiores a 600 milésimas de segundo.

Dado que nas situações de disparo da regra, o último intervalo de análise apresenta registos em 2 meses, e que em cada um dos 4 primeiros intervalos das situações de não disparo (300 a 420 milésimas de segundo) existem registos de 2 a 4 meses, o perito, poderá concluir (caso reconheça estas situações como disparos da regra) que é necessário alterar o tempo da premissa temporal da regra para um valor aproximado dos 420 milésimas de segundo, acrescido de alguma tolerância.

No entanto, existem 8 meses com registos de não disparos da regra, distribuídos entre 450 e 600 milésimas de segundo (3 destes registos no painel 416 da instalação SPM). Perante este facto, o perito terá que analisar bem cada uma destas situações para ver qual a necessidade de criar alguma nova regra para as instalações ou painéis em causa.

Em algumas instalações e painéis, existe um elevado número de meses com resultados obtidos, registando estes uma certa regularidade e valores máximos bastante abaixo das 300 milésimas de segundo. Eventualmente, o perito poderá mediante o conhecimento que tem dessas instalações e painéis, ver necessidade de se criar novas regras específicas para essas instalações e painéis.

Como exemplo de instalações em que existe esta regularidade e algumas com tempos bastante baixos, poder-se-ão referir as instalações SCG, SFA, SFN, SOQ, SRA, SRR, SSN e SZR. Como exemplo, para a eventual criação de uma nova regra para um painel específico de uma instalação, atenda-se ao painel 103 da instalação SSN, que teve disparos em 7 dos 12 meses considerados para a análise, e que regista todos os tempos inferiores a 61 milésimas de segundo.

7.2.4 Regra d4 – Religação Trifásica

A regra d4 tem a seguinte estrutura:

```
regra d4 : 'RELIGAÇÃO TRIFASICA' :
[
[
disparo(Data1,Hora1,Inst1,Painel1,Inst2,NL,trifásico,
nao_identificado,Tabr) at T1
e
disj(Data2,Hora2,Inst1,Painel1,_,nao_identificado,fechado) at T2
e
condicao(mod_dif_tempos_menor_ou_igual(T2,Tabr,500))
]
]
==>
[
cria_facto(disj_rel(Data2,Hora2,Inst1,Painel1,Inst2,NL,
rel_rap_trif,fechado),T2),
retira_facto(disj(_,_,Inst1,Painel1,_,nao_identificado,fechado),_,T2),
cria_facto(disj(Data2,Hora2,Inst1,Painel1,_,rel_rap_trif,
fechado),T2),
retira_facto(disparo(Data1,Hora1,Inst1,Painel1,Inst2,NL,
trifásico,nao_identificado,Tabr),_,T2),
cria_facto(disparo(Data2,Hora2,Inst1,Painel1,Inst2,NL,trifásico,
rel_rap_trif,Tabr),T2)
].
```

A regra d4 permite modelizar a reposição rápida em serviço de um disjuntor, após disparo trifásico. Sempre que um disparo ocorre, existe a hipótese de um automatismo estar

programado para religar rapidamente um disjuntor e repor em serviço elementos que foram desactivados por defeitos fugitivos.

A interpretação da regra d4 é a seguinte:

SE

Existe um disparo trifásico de disjuntor numa determinada subestação e painel

E

O mesmo disjuntor se encontra fechado

E

O módulo da diferença de tempo entre o disparo e o fecho é menor ou igual a 500 milésimas de segundo

ENTÃO

Conclui-se que existiu um disparo trifásico desse disjuntor seguido de religação rápida

E

Regista-se o motivo de religação rápida trifásica para o fecho do disjuntor

E

Retiram-se as conclusões anteriores que originaram a conclusão (pois tornaram-se obsoletas)

Neste caso, para que exista ligação entre o disparo e o fecho do disjuntor, é necessário que estes sucedam com uma separação temporal nunca superior a 500 milésimas de segundo.

A regra d4 tem uma premissa temporal com o tempo assumido de 500 milésimas de segundo.

Nesta regra, registaram-se situações de disparo e situações de não disparo. Apesar das situações de disparo se repartirem por todos os intervalos considerados na análise, estas tiveram uma maior incidência (86%) entre 250 e 500 milésimas de segundo. As situações de não disparo da regra concentraram-se principalmente entre 500 e 600 milésimas de segundo (84.62% correspondendo a 18 registos) e entre 650 e 700 milésimas de segundo (7.69% correspondendo 2 registos). De considerar, ainda, 7.69% de situações de não disparo acima de 1000 milésimas de segundo, que correspondem a um registo de 9600 e outro de 14499 milésimas de segundo. Dado o elevado valor, estes dois últimos registos são de desprezar, pois não representam certamente o evento associado a esta regra.

Perante os resultados obtidos relativamente às situações de disparo e não disparo da regra, verifica-se a necessidade de aumentar o tempo da premissa temporal da regra para abarcar situações que foram classificadas como não sucedidas mas que provavelmente correspondem à situação descrita por esta regra. O novo tempo a atribuir à premissa temporal poderá ser da ordem de grandeza do valor mais alto considerado aceitável das situações de não disparo da regra (700 milésimas de segundo), acrescido de uma tolerância (por exemplo, 800 milésimas de segundo).

Por outro lado, o perito, constatando que um grande número de não disparos da regra foram registados até às 600 milésimas de segundo (18 registos), enquanto que só 2 se registaram entre 650 e 750 milésimas de segundo, poderá chegar à conclusão que o tempo a atribuir ao tempo da premissa temporal da regra será um tempo da ordem de grandeza de 600 milésimas de segundo, acrescido de uma certa tolerância.

Paralelamente a esta alteração a efectuar na regra d4, caso o perito ache que as duas situações de não disparo da regra de 670 e 700 milésimas de segundo são tempos aceitáveis para as instalações em causa, há necessidade de criar duas novas regras, uma para a instalação SEJ e outra para a instalação SET com tempos apropriados.

É de referir, que nas situações de disparo da regra, as instalações SFA e SPN têm bastantes registos (em vários meses) muito regulares. Por este facto, o perito perante o conhecimento que tem dessas instalações, pode chegar à conclusão que é conveniente criar novas regras específicas para essas instalações, considerando um tempo inferior na premissa temporal dessas regras.

7.2.5 Regra d5 – Religação Monofásica

A regra d5 tem a seguinte estrutura:

```
regra d5 : 'RELIGAÇÃO MONOFASICA' :
[
[
disparo(Data1,Hora1,Inst1,Painel1,Inst2,NL,monofásico,
nao_identificado,Tabr) at T1
e
disj(Data2,Hora2,Inst1,Painel1,_,nao_identificado,fechado) at T2
e
condicao(mod_dif_tempos_menor_ou_igual(T2,Tabr,1000))
]
]
==>
[
[
cria_facto(disj_rel(Data2,Hora2,Inst1,Painel1,Inst2,NL,
rel_rap_mono,fechado),T2),
retira_facto(disj(_,_,Inst1,Painel1,_,nao_identificado,fechado),_,T2),
cria_facto(disj(Data2,Hora2,Inst1,Painel1,_,rel_rap_mono,
fechado),T2),
retira_facto(disparo(Data1,Hora1,Inst1,Painel1,Inst2,NL,
monofásico,nao_identificado,Tabr),_,T2),
cria_facto(disparo(Data2,Hora2,Inst1,Painel1,Inst2,NL,
monofásico,rel_rap_mono,Tabr),T2)
].
]
```

A regra d5 permite modelizar a reposição rápida em serviço de um disjuntor, após disparo monofásico.

A interpretação da regra d5 é a seguinte:

SE

Existe um disparo monofásico de disjuntor numa determinada subestação e painel

E

O mesmo disjuntor se encontra fechado

E

O módulo da diferença de tempo entre o disparo e o fecho é menor ou igual a 1000 milésimas de segundo

ENTÃO

Conclui-se que existiu um disparo monofásico desse disjuntor seguido de religação rápida

E

Regista-se o motivo de religação rápida monofásica para o fecho do disjuntor

E

Retiram-se as conclusões anteriores que originaram a conclusão (pois tornaram-se obsoletas)

Neste caso, para que exista ligação entre o disparo e o fecho do disjuntor, é necessário que estes sucedam com uma separação temporal nunca superior a 1 segundo.

Neste caso, para modelar o equipamento físico real, usa-se uma premissa temporal de 1 segundo entre o disparo e o fecho.

Esta regra teve situações de disparo e situações de não disparo da regra. Os resultados obtidos indicam que 88,78% dos disparos da regra tiveram valores entre 800 e 1000 milésimas de segundo e que 100% das situações de não disparo da regra registaram valores entre 1000 e 1200 milésimas de segundo.

Perante estes resultados, verifica-se a necessidade de alterar o tempo da premissa temporal da regra para um valor aproximado do máximo valor obtido nas situações de não disparo da regra (1160 milésimas de segundo) acrescido de alguma tolerância (por exemplo, 1300 milésimas de segundo).

Este raciocínio prende-se com o considerável número de não disparos da regra (44) entre 1000 e 1160 milésimas de segundo, com a inexistência de não disparos nos intervalos seguintes de análise e com a grande concentração das situações de disparo entre 800 e 1000 milésimas de segundo.

Pode-se constatar a ocorrência de algumas situações de disparo da regra d5 com tempos abaixo de 400 milésimas de segundo (6,53%), que poderão ter a ver com particularidades de alguns painéis. Dado o reduzido número de meses em que estas situações foram verificadas (muitas delas só num mês), não é razoável criar novas regras para essas situações particulares, a menos que o perito tenha conhecimento adicional que confirme estes resultados.

7.2.6 Regra d6 – Religação Trifásica Mal Sucedida

A regra d6 tem a seguinte estrutura:

```
regra d6 : 'RELIGAÇÃO TRIFASICA MAL SUCEDIDA' :
[
disparo(Data1,Hora1,Inst1,Painel1,Inst2,NL,trifásico,
rel_rap_trif,Tabr1) at T1
e
disparo(Data2,Hora2,Inst1,Painel1,Inst2,NL,trifásico,
nao_identificado,Tabr2) at T2
e
condicao(dif_tempos_menor_ou_igual(T2,T1,5000))
]
==>
[
retira_facto(disparo(Data1,Hora1,Inst1,Painel1,Inst2,NL,
trifásico,rel_rap_trif,Tabr1),_,T2),
```

```

retira_facto(disparo(Data2,Hora2,Inst1,Painel1,Inst2,NL,
trifásico,nao_identificado,Tabr2),_,T2),
cria_facto(disparo(Data2,Hora2,Inst1,Painel1,Inst2,NL,trifásico,
dtd,Tabr2),T2)
].

```

A regra d6 modeliza a situação de disparo definitivo trifásico ou religação trifásica mal sucedida. Neste caso, deve existir um disparo trifásico inicial, seguido de religação rápida, seguido ainda de um novo disparo que confirma que o defeito não é fugitivo.

A interpretação da regra d6 é a seguinte:

SE

Existe um disparo trifásico inicial, seguido de religação rápida numa determinada subestação e painel

E

Existe um disparo trifásico no mesmo disjuntor

E

O módulo da diferença de tempo entre os dois disparos é menor ou igual a 5000 milésimas de segundo

ENTÃO

Conclui-se que existiu um disparo definitivo trifásico desse disjuntor ou religação trifásica mal sucedida

E

Retiram-se as conclusões anteriores que originaram a conclusão (pois tornaram-se obsoletas)

Neste caso, para que exista ligação entre os dois disparos (sendo o primeiro um disparo trifásico com religação e o segundo um disparo simples), é necessário que estes ocorram com uma separação temporal nunca superior a 5 segundos.

A regra d6 tem uma premissa temporal com o tempo assumido de 5000 milésimas de segundo. Os resultados observados nesta regra mostram que não ocorreram situações de não disparo da regra e que uma percentagem significativa dos disparos regista valores inferiores a 2500 milésimas de segundo (86,66%). No entanto, 13,34% dos disparos situam-se entre 3500 e 4500 milésimas de segundo.

De uma forma geral, para cada instalação, esta regra não apresenta situações de disparo em muitos meses, tornando-se difícil tirar conclusões bem fundamentadas. Daí que, se possa considerar a alteração do tempo da premissa temporal da regra, para um valor aproximado do máximo registado, acrescido de alguma tolerância (por exemplo, 4200 milésimas de segundo).

7.2.7 Regra d7 – Religação Monofásica Mal Sucedida

A regra d7 tem a seguinte estrutura:

```

regra d7 : 'RELIGAÇÃO MONOFASICA MAL SUCEDIDA' :
[
[
disparo(Data1,Hora1,Inst1,Painel1,Inst2,NL,monofásico,
rel_rap_mono,Tabr1) at T1
e
disparo(Data2,Hora2,Inst1,Painel1,Inst2,NL,trifásico,
nao_identificado,Tabr2) at T2
e
condicao(dif_tempos_menor_ou_igual(T2,T1,5000))
]
]
==>
[
retira_facto(disparo(Data1,Hora1,Inst1,Painel1,Inst2,NL,
monofásico,rel_rap_mono,Tabr1),_,T2),
retira_facto(disparo(Data2,Hora2,Inst1,Painel1,Inst2,NL,
trifásico,nao_identificado,Tabr2),_,T2),
cria_facto(disparo(Data2,Hora2,Inst1,Painel1,Inst2,NL,
monofásico,dmd,Tabr2),T2)
].

```

A regra d7 modeliza a situação de disparo definitivo monofásico ou religação monofásica mal sucedida. Neste caso, deve existir um disparo monofásico inicial, seguido de religação rápida, seguido ainda de um novo disparo que confirma que o defeito não é fugitivo.

A interpretação da regra d6 é a seguinte:

SE

Existe um disparo monofásico inicial, seguido de religação rápida numa determinada subestação e painel

E

Existe um disparo trifásico no mesmo disjuntor

E

O módulo da diferença de tempo entre os dois disparos é menor ou igual a 5000 milésimas de segundo

ENTÃO

Conclui-se que existiu um disparo definitivo monofásico desse disjuntor ou religação monofásica mal sucedida

E

Retiram-se as conclusões anteriores que originaram a conclusão (pois tornaram-se obsoletas)

Neste caso, para que exista ligação entre os dois disparos, é necessário também que estes sucedam com uma separação temporal nunca superior a 5 segundos.

A regra d7 tem uma premissa temporal com o tempo assumido de 5000 milésimas de segundo. Os resultados obtidos para esta regra evidenciam que não ocorreram situações de não disparo da regra e que uma percentagem bastante significativa dos disparos tem valores inferiores a 2000 milésimas de segundo (93,33%). No entanto, 16,67% dos disparos situam-se entre 2500 e 3500 milésimas de segundo.

Tal como a regra anterior, para cada instalação, esta regra não apresenta situações de disparo em muitos meses, tornando-se difícil tirar conclusões bem fundamentadas. Daí que, se possa

ser considerar a alteração do tempo da premissa temporal da regra, para um valor aproximado do máximo registado, acrescido de alguma tolerância (por exemplo, 3500 milésimas de segundo).

Nas instalações SRR e SVG, dados os baixos e relativamente regulares valores obtidos, o perito pode eventualmente apoiado no conhecimento que tem desse equipamento, ver necessidade em criar novas regras para essas duas instalações com um tempo da premissa temporal inferior a 3500 milésimas de segundo.

7.2.8 Regra d21 – Disparo Nos Dois Extremos

A regra d21 tem a seguinte estrutura:

```
regra d21 : 'DISPARO NOS DOIS EXTREMOS' :
[
[
disparo(Data1,Hora1,Inst1,Painel1,Inst2,NL,_,TipoDisparo1,Tabr1) at T1
e
disparo(Data2,Hora2,Inst2,Painel2,Inst1,NL,_,TipoDisparo2,Tabr2) at T2
e
verificar( disparo_identificado( TipoDisparo1 ) )
e
verificar( disparo_identificado( TipoDisparo2 ) )
e
condicao(mod_dif_tempos_menor_ou_igual(T1,T2,3000))
e
verificar( data_e_hora_do_tempo_max( T1, T2, Tmax, Data1, Hora1, Data2, Hora2,
Data, Hora ) )
e
verificar(mensagem_utilizador( TipoDisparo1, TipoDisparo2,Inst1, Inst2,
Mensagem ))
]
]
==>
[
cria_facto(disparo_2extremos(Data,Hora,Inst1,Painel1,Inst2,
Painel2,NL,TipoDisparo1,TipoDisparo2),Tmax),
retira_facto(disparo(Data1,Hora1,Inst1,Painel1,Inst2,NL,_,TipoDisparo1,
Tabr1),T1,T1),
retira_facto(disparo(Data2,Hora2,Inst2,Painel2,Inst1,NL,_,
TipoDisparo2,Tabr2),T2,T2),
escreve_mens1(Data,Hora,(0,Inst1),(0,Painel1),['linha ',Inst1,
'- ',Inst2,' ',NL,Mensagem])
].
```

A regra d21 permite identificar um disparo numa linha, através dos disparos dos dois elementos afectos a essa linha nos dois extremos. Assim, verificando-se um disparo (de qualquer tipo) num disjuntor de um extremo de uma linha e outro disparo (de qualquer tipo) no disjuntor do outro extremo da mesma linha, se a diferença de tempo entre os dois disparos não exceder os 3 segundos podemos concluir que a linha está fora de serviço.

A interpretação da regra d21 é a seguinte:

SE

Existe um disparo numa determinada subestação x com ligação à subestação y

E

Existe um disparo numa determinada subestação y com ligação à subestação x

E

Identifica-se o tipo de disparo na subestação y

E

Identifica-se o tipo de disparo na subestação x

E

O módulo da diferença de tempo entre os dois disparos é menor ou igual a 3000 milésimas de segundo

E

Identifica-se a data e a hora do tempo do maior tempo dos dois disparos

E

Prepara-se mensagem a enviar ao utilizador em relação ao tipo de disparo de linha ocorrida

ENTÃO

Conclui-se que houve disparo nos dois extremos de uma linha

E

Retiram-se as conclusões anteriores que originaram a conclusão (pois tornaram-se obsoletas)

E

Informa-se o utilizador do tipo de disparo de linha ocorrido

A regra d21 tem uma premissa temporal com o tempo assumido de 3000 milésimas de segundo. Os resultados verificados nesta regra evidenciam que uma percentagem significativa dos disparos se situa no intervalo [0-300] milésimas de segundo (64.18%) e que 94.04% verificam tempos inferiores a 1500 milésimas de segundo.

Uma possibilidade de alteração da regra d21, por parte do perito, será a atribuição do valor 1500 milésimas de segundo ao tempo da premissa temporal (registando-se ainda assim uma tolerância de 131 milésimas de segundo em relação ao maior valor obtido no intervalo [0-1500] milésimas de segundo) e, paralelamente, a criação de novas regras para as instalações SFN, SRM, SRR e SVG que verificam comportamentos diferentes da maior parte das restantes instalações.

Se, por outro lado, o perito verificar que o tempo de 1500 milésimas de segundo é um tempo bastante baixo e que pode impossibilitar o disparo da regra em situações excepcionais, poderá considerar um tempo mais alto, da ordem de grandeza do máximo registado (2950 milésimas de segundo).

Para a instalação SPM existem resultados de 9 meses, permitindo assim, conclusões mais seguras. Os valores obtidos para esta instalação foram todos muito regulares e baixos, sendo o valor mais elevado registado de 377 milésimas de segundo. Daí que, faça sentido criar uma nova regra específica para esta instalação com um tempo bastante inferior ao tempo atribuído à regra d21, genérica para todas as outras instalações.

Esta regra teve duas situações de não disparo pelo incumprimento da premissa temporal da regra. Uma destas situações ocorreu para o painel 204 da instalação SPN, com um tempo de 8850 milésimas de segundo e a outra ocorreu para o painel 203 da instalação SVG com um tempo de 3965 milésimas de segundo.

A primeira situação envolve um tempo bastante elevado, podendo-se concluir que não reflecte o incidente associado à regra, sendo por isso de desprezar uma eventual criação de uma nova regra para o painel 204 da instalação SPN. Para reforçar esta conclusão, verifica-se a ocorrência de um disparo da regra nesse mesmo painel com um tempo bastante baixo (370 milésimas de segundo).

Quanto à outra situação de não disparo da regra, o perito terá que verificar se o tempo obtido de 3965 milésimas de segundo estará relacionado com uma qualquer particularidade do painel ou da instalação em causa. Caso chegue à conclusão que sim, deverá ser criada uma nova regra específica para o painel 203 da instalação SVG.

7.2.9 Regra j2 – Fecho Sobre Defeito

A regra j2 tem a seguinte estrutura:

```
regra j2 : 'FECHO SOBRE DEFEITO' :
[
  [
    testa_fecho_sobre_defeito(Data1,Hora1,Inst1,Painel1,_,Motivo,
    fechado) at T1
  e
    disparo(Data2,Hora2,Inst1,Painel1,Inst2,NL,trifásico,
    nao_identificado,Tabr) at T2
  e
    condicao(dif_tempos_menor_ou_igual(T2,T1,5000))
  e
    verificar( mensagem_utilizador( comando_disj, Motivo, Mensagem))
  ]
]
==>
[
  retira_facto(testa_fecho_sobre_defeito(Data1,Hora1,Inst1,
  Painel1,_,Motivo,fechado),_,T2),
  retira_facto(disparo(Data2,Hora2,Inst1,Painel1,Inst2,NL,
  trifásico,nao_identificado,Tabr),_,T2),
  cria_facto(disparo(Data2,Hora2,Inst1,Painel1,Inst2,NL,trifásico,
  fecho_sobre_defeito,Tabr),T2),
  escreve_mens1(Data1,Hora1,(1,Inst1),(1,Painel1),['disjuntor fechado sobre
  defeito', Mensagem ] )
].
```

A regra j2 modeliza o fecho sobre defeito de um disjuntor, ou seja, dada uma ordem de fecho ao disjuntor, as protecções detectam um defeito, dando ordem para o disjuntor abrir novamente. Nestes casos, o defeito pode levar algum tempo a manifestar-se, logo é permitida uma distância temporal de 5 segundos entre o fecho do disjuntor e a sua possível reabertura por disparo.

A interpretação da regra j2 é a seguinte:

SE

Existe uma hipótese de fecho sobre defeito ao disjuntor numa determinada subestação e painel

E

Existe um disparo trifásico no mesmo disjuntor

E

O módulo da diferença de tempo entre a ordem de fecho sobre defeito e o disparo do disjuntor é menor ou igual a 5000 milésimas de segundo

E

Prepara-se mensagem a enviar ao utilizador com o motivo do fecho sobre defeito

ENTÃO

Conclui-se que houve fecho sobre defeito desse disjuntor

E

Retiram-se as conclusões anteriores que originaram a conclusão (pois tornaram-se obsoletas)

E

Informa-se o utilizador do tipo de disparo de linha ocorrida

A regra j2 tem uma premissa temporal com o tempo assumido de 5000 milésimas de segundo. Esta regra não teve situações de não disparo, verificando-se que todos os disparos da regra registaram tempos inferiores a 4319 milésimas de segundo. Não existindo situações em que a regra não disparou devido ao incumprimento de premissas temporais, e dado o grande volume de incidentes tratados, o perito poderá considerar este comportamento, alterando na regra j2, o tempo assumido na premissa temporal para 4500 milésimas de segundo. Este valor é mais condizente com o que aconteceu durante o ano analisado, permitindo até alguma tolerância.

Mediante os resultados obtidos em vários meses para determinadas instalações e painéis, verifica-se também, a possibilidade de criação de novas regras específicas dessas instalações e/ou painéis. Na instalação SPN, dados os resultados, poder-se-á fazer uma distinção entre painéis. Os painéis 302, 303, 304 e 306 registaram valores bastante inferiores aos do painel 307. O valor mais elevado obtido nos primeiros foi de 2700 milésimas de segundo, enquanto no painel 307 se registaram valores bem mais elevados (3940 milésimas de segundo e 4319 milésimas de segundo). Isto sugere que se criem para os primeiros painéis, regras com tempos menores na premissa temporal, de acordo com os respectivos resultados de cada painel. Este mesmo raciocínio poderá ser considerado na instalação SZR, pois os três painéis com registo apresentam valores bastante regulares e inferiores (o maior valor é de 985 milésimas de segundo) a 4319 milésimas de segundo. Neste caso poder-se-á criar uma nova regra, específica para esta instalação.

Capítulo 8

Conclusões

8.1 Objectivos Alcançados

Os objectivos essenciais que se pretendiam atingir com o presente trabalho consistiam na apresentação de um subconjunto representativo dos métodos de Aprendizagem Automática existentes, na concepção e desenvolvimento de uma ferramenta de Extracção Automática de Conhecimento que fosse capaz de confirmar ou sugerir modificações nas premissas temporais das regras do SPARSE e na construção de um Editor de Regras que possibilitasse a alteração da Base de Regras do SPARSE.

Nos primeiros capítulos desta tese são incluídos os principais métodos de Aprendizagem Automática. Nos métodos de Aprendizagem Automática podemos identificar três tipos de aprendizagem: a Aprendizagem Indutiva, a Aprendizagem Analítica e a combinação da Aprendizagem Indutiva e Analítica.

Os métodos de Aprendizagem Indutiva têm a particularidade de generalizarem a partir de exemplos de treino observados e pela identificação de características que distinguem os exemplos positivos dos negativos. Estes métodos necessitam de um grande número de exemplos de treino para adquirirem um nível aceitável de exactidão na generalização.

Por outro lado, os métodos de Aprendizagem Analítica usam o conhecimento prévio e o raciocínio dedutivo para aumentar a informação possibilitada pelos exemplos de treino, não estando sujeitos à limitação no número de exemplos disponíveis. O conhecimento prévio é aceite explicitamente como “input” adicional aos dados de treino.

A combinação da Aprendizagem Indutiva e da Aprendizagem Analítica oferece a possibilidade de se obterem métodos de aprendizagem mais potentes. Os métodos de Aprendizagem puramente Analíticos oferecem a vantagem de generalizarem com maior exactidão a partir de menos dados, usando o conhecimento prévio para conduzir a aprendizagem. No entanto, quando o conhecimento prévio é insuficiente ou incorrecto, estes métodos podem ser enganosos. Os métodos de Aprendizagem puramente Indutivos oferecem a vantagem de não requererem conhecimento prévio explícito e aprenderem regularidades baseados somente nos dados de treino. Contudo, estes métodos podem falhar quando não existem dados de treino suficientes, e podem ser induzidos em erro pela limitação indutiva que devem adoptar para generalizarem para além dos dados observados.

Foram desenvolvidas algumas abordagens para combinar a teoria do domínio e os dados de treino, de forma a restringir a pesquisa de uma hipótese aceitável (combinação da Aprendizagem Indutiva e Analítica). A maior parte destas abordagens são extensões aos métodos indutivos. Estas abordagens usam o conhecimento prévio para alterar a pesquisa realizada pelos métodos puramente indutivos. O uso do conhecimento prévio pode ser feito para obter: uma hipótese inicial a partir da qual começa a pesquisa; para alterar o objectivo da pesquisa do espaço de hipóteses; para alterar as etapas de pesquisa disponíveis.

A ferramenta de Aprendizagem Automática concebida e desenvolvida, para dar resposta aos propósitos inicialmente definidos de Extracção Automática do Conhecimento, é designada por TEMPUS. Por sua vez, o Editor de Regras, construído para dar possibilidade de alteração da Base de Regras do SPARSE, é designado por EDIREGRA.

Apesar da ideia geral do TEMPUS ser baseada na Aprendizagem Indutiva, este difere significativamente das aplicações conhecidas desta metodologia. De facto, os algoritmos Indutivos usam um conjunto de exemplos classificados à partida para treinar o sistema.

O TEMPUS usa uma abordagem diferente. A partir de incidentes ocorridos faz uma análise às regras do SPARSE e obtém resultados que permitem sugerir alterações adequadas. Estas alterações dizem respeito aos tempos das premissas temporais das regras.

As aplicações de processamento de alarmes, como é o caso do SPARSE, têm uma grande complexidade, pois envolvem questões como o raciocínio temporal e não monotónico.

De facto, apesar de ser possível, a partir do conhecimento dos peritos no Controlo e Condução de Redes de Transporte de Energia Eléctrica, produzir regras de diagnóstico correctas, o estabelecimento dos tempos correctos nas premissas temporais é praticamente impossível sem uma análise de um grande conjunto de incidentes. Por outro lado, estes tempos podem depender das instalações e painéis envolvidos e do desempenho do sistema de transmissão de informação.

Antes da existência do TEMPUS, os tempos das premissas temporais das regras do SPARSE eram testados manualmente e iterativamente, a partir de muitos exemplos, o que tornava o processo de obtenção de conclusões muito demorado e complexo.

Com o TEMPUS, esta limitação foi ultrapassada, pois este permite obter resultados que possibilitam o aconselhamento do ajuste dos parâmetros temporais das regras do SPARSE. Contudo, para que as conclusões obtidas pelo TEMPUS possam ser consideradas aceitáveis, é necessário processar um conjunto de incidentes relativos a um longo período de tempo.

Para a obtenção de resultados relativos ao comportamento das premissas temporais das regras, foram processados doze ficheiros mensais de incidentes da REN (Rede Eléctrica Nacional, S.A.).

A análise desses resultados mostrou que, provavelmente, será necessário fazer alterações à Base de Regras do SPARSE. De facto, pode-se constatar, em algumas regras, a existência de situações classificadas como situações de não disparo das regras que provavelmente correspondem a situações de disparo da regra. Este tipo de situação acontece quando o tempo assumido nas premissas temporais das regras tem um valor subestimado.

Por outro lado, em muitas instalações e painéis, existe um grande volume de registos de situações de disparo das regras, com tempos bastante regulares e inferiores aos assumidos nas premissas temporais das regras. Isto poderá, eventualmente, levar a que o perito considere que esses tempos estão sobrestimados e se possam criar novas regras específicas para essas instalações e painéis.

Perante os resultados obtidos, pode-se concluir que o TEMPUS provou ser bastante útil no ajuste dos parâmetros temporais das regras do SPARSE.

A manutenção do conhecimento é um dos factores chave para o sucesso de um Sistema Baseado no Conhecimento (SBC). As dificuldades encontradas na manutenção do conhecimento são a razão principal para o abandono do uso de muitos SBC.

Apesar do TEMPUS ter sido especialmente desenvolvido para ser usado pelo SPARSE, a metodologia utilizada mostra-se adequada a outras aplicações em que existem parâmetros numéricos envolvidos. Um bom exemplo são as aplicações financeiras.

O Editor de Regras construído (EDIREGRA) é uma ferramenta cuja utilização é extremamente intuitiva, em ambiente de janelas, que possibilita :

- A alteração de qualquer tempo das premissas temporais das regras do SPARSE, repercutindo essa alteração para todas as tentativas de disparo dessas regras em todas as instalações da REN.
- A criação de novas regras na Base de Regras do SPARSE (com tempos das premissas temporais diferentes da regra original), específicas para determinadas instalações e painéis que têm um comportamento diferente da generalidade das instalações e painéis.

Em suma, pode-se concluir que o saldo do trabalho realizado é positivo, tendo sido realizado trabalho na área da Aprendizagem Automática com resultados efectivos no campo dos Sistemas de Energia, e tendo sido atingidos os objectivos inicialmente propostos.

8.2 Trabalho Futuro

Devido à limitação de tempo inerente ao desenvolvimento do trabalho de suporte desta dissertação de mestrado, existem alguns itens passíveis de trabalho futuro, dos quais se destacam:

- A realização de uma análise mais exaustiva (com mais incidentes), que possibilite mais resultados, possibilitando tirar, eventualmente, melhores conclusões.
- O processamento de um número suficiente de ficheiros de incidentes com a nova Base de Regras obtida, resultante das modificações realizadas à Base de Regras original, de forma a confirmar a utilidade das modificações realizadas.
- A confrontação das conclusões obtidas pelo TEMPUS com os resultados da análise realizada pelos peritos.
- A realização de uma análise ao meta-conhecimento existente na Base de Conhecimento do SPARSE, para se poder aferir da necessidade do seu ajuste.

De facto, o meta-conhecimento utilizado, especifica, para cada tipo de facto recebido (alarme ou conclusão intermédia), quais as regras que devem ser disparadas. Além disso, especifica também um intervalo de tempo para a regra poder ser disparada, modelando assim o comportamento temporal adequado à natureza do problema.

No entanto, os intervalos de tempo assumidos, associados à tentativa de disparo de cada regra, poderão não estar completamente correctos. De facto, estes intervalos poderão estar sub-dimensionados ou sobre-dimensionados, havendo eventualmente, necessidade de algum ajuste. Daí, a necessidade de uma análise ao meta-conhecimento existente na Base de Conhecimento do SPARSE.

- A construção de uma ferramenta que realize o mapeamento da informação contida na base de dados disponibilizada pela REN (com informação dos incidentes ocorridos na rede eléctrica portuguesa) para as regras do SPARSE. Essa base de dados é constituída por informação estruturada de uma forma completamente diferente da existente nas regras do SPARSE, não se conseguindo por isso fazer o mapeamento directo.

Pretende-se com este mapeamento, obter exemplos positivos para as regras do SPARSE, de forma a poder-se construir uma ferramenta que utilize os métodos de Aprendizagem Indutiva e/ou Analítica referidos na tese.

Referências

- [Baldwin-1896] J. M. Baldwin, "A new factor in evolution", *American Naturalist*, 3, pp. 441-451, pp 536-553, 1896
- [Bentley-75] J. L. Bentley, "Multidimensional binary search trees used for associative searching", *Communications of the ACM*, 18(9), pp. 509-517, 1975
- [Bergadano-90] F. Bergadano and A. Giordana, "Guiding induction with domain theories", R. Michalski et al. (Eds.), *Machine Learning: An artificial intelligence approach 3*, pp. 474-492, San Mateo, CA: Morgan Kaufmann, 1990
- [Carbonell-90] J. Carbonell, C. Knoblock and S. Minton, "PRODIGY: An integrated architecture for planning and learning", K. VanLehn (Ed.), *Architectures for Intelligence*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1990
- [Cestnik-91] B. Cestnik and I. Bratko, "On estimating probabilities in tree pruning", *Proceedings of the European Working Session on Machine Learning*, pp. 138-150, Porto, Portugal, February 1991
- [CIGRE-93] CIGRE - Working Group 38.06.03, "Expert Systems: Development Experience and User Requirements", *Electra*, 146, pp. 29-67, February 1993
- [Clark-89] P. Clark and R. Niblett, "The CN2 induction algorithm", *Machine Learning*, 3, pp. 261-284, 1989
- [Cooper-97] G. Cooper et al., "An evaluation of machine-learning methods for predicting pneumonia mortality", *Artificial Intelligence in Medicine*, 1997
- [Cottrell-90] G. W. Cottrell, "Extracting features from faces using compression networks: Face, identity, emotion and gender recognition using holons", D. Touretzky (Ed.), *Connection Models: Proceedings of the 1990 Summer School*, San Mateo, CA: Morgan Kaufmann, 1990
- [Craven-94] M. W. Craven and J. W. Shavlik, "Using sampling and queries to extract rules from trained neural networks", *Proceedings of the 11th International Conference on Machine Learning*, pp. 37-45, San Mateo, CA: Morgan Kaufmann, 1994
- [Craven-96] M. W. Craven, "Extracting comprehensible models from trained neural networks", PhD thesis, Department of Computer Sciences, University of Wisconsin-Madison, 1996
- [Fayyad-91] U. M. Fayyad, "On the induction of decision trees for multiple concept learning", Ph.D. dissertation, EECS Department, University of Michigan, 1991

- [Fayyad-93] U. M. Fayyad and K. B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning", Bajcsy (Ed.), Proceedings of the 13th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, pp. 1022-1027, 1993
- [Friedman-77] J. Friedman, J. Bentley and R. Finkel, "An algorithm for finding best matches in logarithmic expected time", ACM Transactions on Mathematical Software, 3(3), pp. 209-226, 1977
- [Fu-89] L. M. Fu, "Integration of neural heuristics into knowledge-based inference", Connection Science, 1(3), pp. 325-339, 1989
- [Fu-93] L. M. Fu, "Knowledge-based connectionism for revising domains theories", IEEE Transactions on Systems, Man and Cybernetics, 23(1), pp. 173-182, 1993
- [Hinton-87] G. E. Hinton and S. J. Nowlan, "How learning can guide evolution", Complex Systems, 1, pp. 495-502, 1987
- [Kedar-Cabelli-87] S. Kedar-Cabelli and T. McCarty, "Explanation-based generalization as resolution theorem proving", Proceedings of the Fourth International Workshop on Machine Learning, pp. 383-389, San Francisco: Morgan Kaufmann, 1987
- [Kirschen-89] D. S. Kirschen, B. F. Wollenberg, G. D. Irisarri, J. J. Bann and B. N. Miller, "Controlling Power systems During Emergencies: The Role of Expert Systems", IEEE Computer Applications in Power, pp. 41-45, 1989
- [Kirschen-92] D. S. Kirschen and B. F. Wollenberg, "Intelligent Alarm Processing in Power Systems", Proceedings of the IEEE, 80(5), pp. 663-672, 1992
- [koza-92] J. Koza, "Genetic programming: On the programming of computers by means of natural selection", Cambridge, MA: MIT Press, 1992
- [Lang-90] K. J. Lang, A. H. Waibel and G. E. Hinton Lee, "A time-delay neural network architecture for isolated word recognition", Neural Networks, 3, pp. 33-43, 1990
- [Lee-89] K. Lee, "Automatic speech recognition: The development of the Sphinx system", Boston: Kluwer Academic Publishers, 1989
- [Malheiro-99] Nuno Malheiro, Zita A. Vale, Carlos Ramos, António Silva and Albino Marques, "Providing Real-Time Explanations for control Center Operator assistance", Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing, Honolulu, HI, USA, August 9-12, 1999
- [Mehta-95] M. Mehta, J. Rissanen and R. Agrawal, "MDL-based decision tree pruning", Proceedings of the First International Conference on Knowledge Discovery and Data-Mining, Menlo Park, CA: AAAI Press, pp. 216-221, 1995
- [Minton-88] S. Minton, "Learning search control knowledge: An explanation-based approach, Boston, MA: Kluwer Academic Publishers, 1988
- [Michalski-69] R. S. Michalski, "On the quasi-minimal solution of the general covering problem", Proceedings of the First International Symposium on Information Processing, pp. 125-128, Bled, Yugoslavia, 1969

- [Michalski-86] R. S. Michalski, I. Mozetic, J. Hong, H. Lavrac, "The multi-purpose incremental learning system AQ15 and its testing application to three medical domains", Proceedings of the Fifty National Conference on AI, pp. 1041-1045, Philadelphia: Morgan Kaufmann, 1986
- [Mingers-89] J. Mingers, "An empirical comparasion of pruning methods for decision-tree induction", Machine Learning, 4(2), pp. 227-243, 1989
- [Mitchell-79] T. M. Mitchell, "Version Spaces : An approach to concept learning", Ph.D. dissertation, Electrical Engineering Dept., Stanford University, Stanford, CA, 1979
- [Mitchell-83] T. M. Mitchell, P. E. Utgoff and R. Banerji, "Learning by experimentation: Acquiring and modifying problem-solving heuristics", Michalski, Carbonell e Mitchell (Eds.), Machine Learning, Tioga Press, 1, pp. 163-190, 1983
- [Mitchell-97] T. M. Mitchell, "Machine Learning", McGRAW-HILL International Editions, Computer Science Series, 1997
- [Muggleton-88] S. Muggleton and W. Buntine, "Machine invention of first-order predicates by inverting resolution", Proceedings of the Fifth International Machine Learning Conference, pp. 339-352, Ann Arbor, Michigan: Morgan Kaufmann, 1988
- [Muggleton-90] S. Muggleton and C. Feng, "Efficient induction of logic programs", Proceedings of the First Conference on Algorithmic Learning Theory", Ohmsha, Tokyo, 1990
- [Opper-91] M. Opper and D. Haussler, "Generalization performance of Bayes optimal prediction algorithm for learning a perception", Physics Review Letters, 66, pp. 2677-2681, 1991
- [Pazzani-91] M. J. Pazzani, C. A. Brunk and G. Silverstein, "A knowledge-intensive approach to learning relational concepts", Proceedings of the Eighth International Workshop on Machine Learning, pp. 432-436, San Mateo, CA: Morgan Kaufmann, 1991
- [Pazzani-92] M. J. Pazzani and D. Kibler, "The utility of knowledge in inductive learning", Machine Learning, 9(1), pp. 57-94, 1992
- [Pomerleau-89] D. A. Pomerleau, "ALVINN: An autonomous land vehicle in a neural network", Technical Report CMU-CS, pp. 89-107, Pittsburgh, PA: Carnegie Mellon University, 1989
- [Pomerleau-93] D. A. Pomerleau, "Knowledge-based training of artificial neural networks for autonomous robot driving", J. Connell e S. Mahadevan (Eds), J. Connell e S. Mahadevan (Eds), Robot Learning, , pp. 19-43, Boston: Kluwer Academic Publishers 1993
- [Pratt-93a] L. Y. Pratt, "Transferring previously learned Backpropagation neural networks to new learning tasks, Ph. D. thesis, Department of Computer Science, Rutgers University, New Jersey, 1993
- [Pratt-93b] L. Y. Pratt, "Discriminability-based transfer among neural networks", J. E. Moody et al. (Eds.), Advances in Neural Information Processing Systems 5, San Mateo, CA: Morgan Kaufmann, 1993
- [Quinlan-86] J. R. Quinlan, "Induction of decision trees", Machine Learning, 1(1), pp. 81-106, 1986
- [Quinlan-87] J. R. Quinlan, "Rule induction with statistical data – a comparasion with multiple regression", Journal of the Operational Research Society, 38, pp. 347-352, 1987

- [Quinlan-89] J. R. Quinlan and R. Rivest, *Information and Computation*, 80, pp. 227-248, 1989
- [Quinlan-90] J. R. Quinlan, "Learning logical definitions from relations", *Machine Learning*, 5, pp. 239-266, 1990
- [Quinlan-93] J. R. Quinlan, "C4.5: Programs for Machine Learning", San Mateo, CA: Morgan Kaufmann, 1993
- [Rayudu-97] R. Rayudu, S. Samarasinghe, D. Kulasiri and J. Ypsilantis, "Model-Based Learning for Fault Diagnosis in Power Transmission Networks", *Engineering Intelligent Systems for Electrical Engineering and Communications*, 5(2), pp. 63-74, June 1997
- [Robinson-65] J. A. Robinson, "A machine-oriented logic based on the resolution principle", *Journal of the ACM*, 12(1), pp. 23-41, 1965
- [Rosado-93] C. Rosado, "Process TTLOGW", EDP Report RESP/SCDS 20/93, July 1993
- [Simard-92] P. S. Simard, B. Victorri, Y. LeCun and J. Denker, "Tangent prop-A formalism for specifying selected invariances in an adaptive network", J. Moody et al. (Eds.), *Advances in Neural Information Processing Systems 4*, San Mateo, CA: Morgan Kaufmann, 1992
- [Shavlik-89] J. Shavlik and G. Towell, "An approach to combining explanation-based and neural learning algorithms", *Connection Science*, 1(3), pp. 223-255, 1989
- [Talukdar-86] S. N. Talukdar, E. Cardozo and T. Perry, "The Operator's Assistant: an Expandable Program for Power System Trouble Analysis", *IEEE Transactions on Power systems*, 1(3), pp. 182-187, 1986
- [Tan-90] M. Tan and J. C. Schlimmer, "Two case studies in cost-sensitive concept acquisition", *Proceedings of the AAAI-90*, 1990
- [Tan-93] M. Tan, "Cost-sensitive learning of classification Knowledge and its applications in robotics", *Machine Learning*, 13(1), pp. 1-33, 1993
- [Tesauro-92] G. Tesauro, "Practical issues in temporal difference learning", *Machine Learning*, 8, pp. 257, 1992
- [Tesauro-95] G. Tesauro, "Temporal difference learning and TD-gammon", *Communications of the ACM*, 37(3), pp. 87-92, 1995
- [Togami-95] M. Togami et al., "On the Application of a Machine Learning Technique to Fault Diagnosis of Power Distribution Lines", *IEEE Transactions on Power Delivery*, 10(4), October 1995
- [Towell-94] G. Towell and J. Shavlik, "Knowledge-based artificial neural networks", *Artificial Intelligence*, 70(1-2), pp. 119-165, 1994
- [Vale-94] Zita A. Vale, A. M. Moura, M. Fernanda Fernandes, and Albino Marques, "SPARSE - An Expert System for Alarm Processing and Operator Assistance in Substations Control Centers", *Applied Computing Review*, 2(2), 1994
- [Vale-97] Zita A. Vale, Jorge Santos, Carlos Ramos, M. Fernanda Fernandes, Couto Rosado and Albino Marques, "SPARSE - A Prolog Based Application for the Portuguese Transmission Network: Verification and Validation", *The 5th International Conference on the Practical Application of Prolog - PAP 97*, pp. 291-310, London, UK, 22-24 April, 1997
- [Vale-98] Zita A. Vale, Carlos Ramos, Luiz Faria, Jorge Santos and António Silva, "Knowledge-Based Systems for Power System Control Centers", *Workshop: Intelligent Computation and Simulation in Planning and Operation of Power Systems*, Wroclaw, Poland, June 22, 1998

- [Vale-99a] Zita A. Vale, Carlos Ramos, Luiz Faria, Jorge Santos, Nuno Malheiro, António Silva and Albino Marques' "An Expert System for Intelligent Information Processing in Portuguese Power System Control Centers", International Conference on Enterprise Information Systems (ICEIS), Setúbal, Portugal, 1999
- [Vale-99b] Zita A. Vale, António Silva, Luiz Faria, Nuno Malheiro, Carlos Ramos and Albino Marques, "SPARSE-IT: An Intelligent Tutor for Control Center Operators", The 1999 International Conference on Intelligent Systems Application to Power Systems (ISAP)", Rio de Janeiro, Brazil, April 4-7, 1999
- [Vale-99c] Zita A. Vale, Carlos Ramos, Luiz Faria, Nuno Malheiro, Jorge Santos, Albino Marques e Couto Rosado, "SPARSE: Um Sistema Pericial para apoio aos Operadores dos Centros de Condução da Rede Eléctrica Nacional", 6^{as} Jornadas Luso-Espanholas de Engenharia Electrotécnica, Lisboa, Portugal, 2, pp. 263-270, 7 a 9 de Julho de 1999
- [Waibel-89] A. Waibel, T. Hanazawa, G. Hinton, H. Shikano e K. Lang, "Phoneme recognition using time-delay neural networks", IEEE Transactions on Acoustics, Speech and Signal Processing, 37(3), pp. 328-339, 1989
- [Waldinger-77] R. Waldinger, "Achieving several goals simultaneously", E. Elcock & D. Michie (Eds.), Machine Intelligence 8, London: Ellis Horwood Ltd, 1977
- [Wehenkel-97] Louis A. Wehenkel, "Machine-Learning Approaches to Power System Security Assessment", IEEE Expert, pp. 60-72, September/October 1997
- [Wehenkel-98] Louis A. Wehenkel, "Automatic Learning Techniques in Power Systems", IEEE Expert, Power Electronics and Power Systems Series, pp. 1-4, 1998
- [Wollenberg-86] B. F. Wollenberg, "Feasibility Study for an Energy Management System Intelligent Alarm Processor", IEEE Transactions on Power Systems, 1(2), pp. 241-247, 1986
- [Zlatareva-94] N. Zlatareva, "An effective Logical Framework for Knowledge-Based Systems Verification", International Journal of Expert Systems, 7(3), pp. 239-260, 1994

Anexo A

Instalações

Este anexo inclui uma lista com todas as instalações existentes e os respectivos painéis. De notar que essa lista refere-se a uma determinada data, e está sujeita a alterações, de acordo com as alterações efectuadas na rede eléctrica.

```
62
*****
CAG
9
202 SPR,1
204 SPR/MORTAGU.,0
210 GRUPO,3
212 GRUPO,2
214 GRUPO,1
BOM1 BOMBA,1
BOM2 BOMBA,2
BOM3 BOMBA,3
PLC TREG.GRUPO,3
*****
CAL
1
PLC TREG.GRUPO,2
*****
CBR
12
602 QUIMIPAR/SFF,0
604 LAVRADIO,1
610 GRUPO,2
612 S.AUX.,0
614 COINA/BARREI,0
616 GRUPO,1
620 INT.BAR,60
622 F.FERRO,0
624 LAVRADIO,2
626 FISIFE,2
628 FISIFE,1
SGER SGER,0
*****
CBT
1
204 SAA,0
*****
CCB
1
PLC TREG.GRUPO,3
*****
CCR
1
PLC TREG.GRUPO,2
*****
CPG
4
411 GRUPO,1
421 GRUPO,2
PLC1 TREG.GRUPO,1
PLC2 TREG.GRUPO,2
*****
CSB
6
401 GRUPO,1
402 GRUPO,2
403 GRUPO,3
404 GRUPO,4
PLC TREG.GRUPO,4
```

```

SGER SGER,0
*****
CSN
9
100 GRUPO,1
400 GRUPO,2
401 GRUPO,3
402 GRUPO,4
PLC1 TREG.GRUPO,1
PLC2 TREG.GRUPO,2
PLC3 TREG.GRUPO,3
PLC4 TREG.GRUPO,4
SGER SGER,0
*****
CTG
7
201 INT.BAR,220
202 SRR,0
203 SCN/SRR,3
204 SCN/SEJ,0
205 GRUPO,3
206 GRUPO,2
207 GRUPO,1
*****
CTO
11
602 GRUPO,1
604 SED,1
606 S.AUX.,1
608 SED,2
610 GRUPO,2
612 S.AUX.,2
614 SVM,0
616 GRUPO,3
618 ALFENA,0
620 CCM/LEVER,0
SGER SGER,0
*****
CTR
4
BOM1 BOMBA,1
BOM2 BOMBA,2
PLC TREG.GRUPO,2
RTU DUMMY,0,1
*****
DRP
3
AC AR_CONDICIONAD.,4
SGER SUB._MOSCAVIDE,0
TLDE VIGILANCIA_1,V,2
*****
SAL
15
411 SRA,1
412 MOD.41,0
413 MOD.41,0
422 SRA,2
423 MOD.42,0
431 MOD.43,0
432 MOD.43,0
441 GRUPO,1
442 GRUPO,2
443 SAL,0
BAR4 BARR,400
COM SEQ.DE_COMANDO,0
M41 CTLE/MOD.SRA,1
M42 MOD.RES./SRA2,0
SGER SGER,0
*****
SAM
44
205 TRANSF,6
206 STJ,0
207 TRANSF,5
208 SFN,2
209 TRANSF,4
210 SCH,0
211 TRANSF,3
212 SCG/SCH,0
213 INT.BAR,220

```

214 SCH/SFN,3
617 SABUGO,2
618 SABUGO,1
620 AMADORA,1
621 AMADORA,2
622 TRANSF.GRUPO,1
624 TRANSF,6
626 CARENQUE,5
627 CARENQUE,4
628 CARENQUE,3
629 TRANSF,5
630 INT.BAR,60
631 CARENQUE,2
632 CARENQUE,1
633 COMET/SCH,1
634 TRANSF,4
635 TRANSF.GRUPO,2
636 SCH,2
639 TRANSF,3
640 TRANSF.GRUPO,3
642 TRANSF.GRUPO,6
643 TRANSF.GRUPO,4
B61 BARR.1,60
B62 BARR.1,60
BAR2 BARR,220
BAR6 BARR,60
OPA OPA,0
RTU DUMMY,0,1
SGER SGER,0
TUR1 TURBINA,1
TUR2 TURBINA,2
TUR3 TURBINA,3
TUR4 TURBINA,4
TUR5 TURBINA,5
TUR6 TURBINA,6

SBL

25
236 SRM,2
237 TRANSF,3
238 SPB/SPR,2
239 TRANSF,4
240 INT.BAR,220
241 TRANSF,5
242 SPR,1
243 SRM,1
610 TRANSF,5
613 ENTRONCAM.,0
614 S.JORGE,0
615 TRANSF,4
616 AZOIA,0
617 INT.BAR,60
618 ANDRINOS,0
619 B.COND.,1
620 TRANSF,3
621 B.COND.,2
B61 BARR.1,60
B62 BARR.2,60
BAR2 BARR,220
BAR6 BARR,60
OPA OPA,0
RTU DUMMY,0,1
SGER SGER,0

SCF

16
210 GVA/SVC,1
211 SPN,1
212 GVA/SVC,2
213 SPN,2
214 IB/BYP,220
217 TRANSF,2
219 TRANSF,3
220 SOBRAL,1
221 SOBRAL,2
614 TRANSF,2
616 SCC,0
618 TRANSF,3
B61 BARR.1,60
COM SEQ.DE_COMANDO,0

```

OPA OPA,0
SGER S.GERAIS_(SCF2),0
*****
SCG
26
202 INT.BAR.2201,0
204 TRANSF.GRUPO,1
206 SRM,1
208 S.AUX.GB,0
210 SSV,0
212 TRANSF.GRUPO,2
214 BARR.2,220
216 TRANSF.GRUPO,3
218 SAM/SCH,0
220 S.AUX.GC,0
224 TRANSF.GRUPO,4
226 BARR.2,220
228 TRANSF.GRUPO,5
230 SFN,2
232 TRANSF,1
236 TRANSF.GRUPO,6
240 TRANSF,2
242 SRM,2
244 TRANSF,3
246 SRM,3
250 INT.BAR.2202,0
BAR2 BARR,220
PLC5 TREG.GRUPO,5
PLC6 TREG.GRUPO,6
RTU DUMMY,0,1
SGER SGER,0
*****
SCH
18
210 SAM/SCG,0
211 TRANSF,1
212 SAM/SFN,3
213 TRANSF,2
214 TRANSF,3
215 INT.BAR,220
216 SFN,0
217 TRANSF,4
218 ALTO_MIRA,0
609 TRANSF,4
610 TRANSF,3
611 INT.BAR,60
613 TRANSF,2
614 TRANSF,1
617 COMET/SAM,1
618 ALTO_MIRA,2
COM SEQ.DE_COMANDO,0
SGER SGER,0
*****
SCL
12
200 BARR.2,220
201 CCL,0
202 STR,0
203 SVG,3
204 SEJ,2
205 SVG,2
206 SMC,0
207 SVG,1
209 INT.BAR,220
210 SEJ,1
BAR2 BARR,220
SGER SGER,0
*****
SCN
30
211 IB/BYP,220
213 SRR,1
214 TRANSF,1
215 SRR,2
216 TRANSF,2
217 CTG/SEJ,0
218 TRANSF,3
219 CTG/SRR,3
221 TRANSF,4
611 TRANSF,1

```

```
613 PEDROSO,2
614 PEDROSO,1
615 CCM,0
616 TRANSF,2
617 B.COND.,1
618 IB/BYP,60
619 B.COND.,2
621 TRANSF,3
622 V.N.GAIA,0
624 V.PARAISO,0
626 NOG.DA_REGE,0
627 TRANSF,4
B61 BARR.1,60
B62 BARR.2,60
BAR2 BARR,220
BAR6 BARR,60
COM SEQ.DE_COMANDO,0
OPA OPA,0
RTU DUMMY,0,1
SGER SGER,0
*****
SCT
18
204 SRR,0
205 SVM,0
206 IB/BYP,220
210 TRANSF,1
211 TRANSF,2
602 INT.BAR,60
603 MATOSIN.,1
604 MATOSIN.,2
605 S.C.BISPO,0
607 B.COND.,1
608 TRANSF,1
609 B.COND.,2
611 TRANSF,2
616 SCT,1
617 SCT,2
COM SEQ.DE_COMANDO,0
OPA OPA,0
SGER SGER,0
*****
SCV
4
OPA OPA,0
SGER SGER,0
T TRANSF,0
TB TRANSF.B,0
*****
SED
38
101 SVM,1
102 SRA,3
103 SRA,2
104 TRANSF,1
105 SRA,1
106 TRANSF,2
107 INT.BAR,150
108 TRANSF,3
109 SVM,2
110 TRANSF,4
111 MAIA/SVM,3
112 TRANSF,5
113 SRA,4
619 B.COND.,1
620 TRANSF,1
621 B.COND.,2
622 ANTAS,3
623 AMIEIRA,0
624 TRANSF,2
625 ANTAS,1
626 INT.BAR,60
627 ANTAS,2
628 TRANSF,3
629 FREIXO,0
630 C.POR.COBRE,0
631 FANZERES,0
632 TRANSF,4
633 CTO,1
634 CTO,2
```

635 VAL./REBOR.,0
636 TRANSF,5
B61 BARR.1,60
B62 BARR.2,60
BAR1 BARR,150
BAR6 BARR,60
OPA OPA,0
RTU DUMMY,0,2
SGER SGER,0

SEJ
29
202 TRANSF,2
203 SCN/CTG,0
204 CCL,2
205 TRANSF,3
206 IB/BYP,220
207 TRANSF,4
208 CCL,1
603 DEV.VELHA,0
604 TRANSF,2
605 RIO_MEAO,0
606 AVANCA,2
607 B.COND.,1
608 B.COND.,2
609 AVANCA,1
610 DOW/OXINOR.,0
611 TRANSF,3
612 UNITECA,0
613 QUIMIGAL,0
614 TRANSF,4
615 SALREU,1
616 SALREU,2
617 ALBERGARIA,0
618 INT.BAR,60
B61 BARR.1,60
B62 BARR.2,60
COM SEQ.DE_COMANDO,0
OPA OPA,0
RTU DUMMY,0,2
SGER SGER,0

SER
11
101 IB/BYP,150
103 SPM/PEGOES,0
106 F.ALENTEJO,0
107 TRANSF,2
109 TRANSF,3
602 TRANSF,2
604 TRANSF,3
605 B.COND.,1
B62 BARR.2,60
OPA OPA,0
SGER SGER,0

SET
16
103 STN,0
104 TRANSF,2
106 SOQ,0
107 TRANSF,3
110 IB/BYP,150
601 TAVIRA,0
602 LOULE,0
606 INT.BAR,60
608 TRANSF,2
609 BRACIAIS,1
611 OLHAO,0
612 B.COND.,2
613 TRANSF,3
COM SEQ.DE_COMANDO,0
OPA OPA,0
SGER SGER,0

SFA
26
102 SOQ,0
103 TRANSF,4
104 SSN,0

```
105 TRANSF,3
106 RESERVA,0
107 TRANSF,2
108 INT.BAR,150
109 TRANSF,1
110 SER,0
615 TRANSF,1
617 BEJA,2
618 BEJA,1
619 TRANSF,2
621 INT.BAR,60
623 TRANSF,3
624 ALJUSTREL,0
625 SANTIAGO,0
626 VALE_DO_GAIO,0
627 TRANSF,4
B61 BARR.1,60
B62 BARR.2,60
BAR1 BARR,150
BAR6 BARR,60
OPA OPA,0
RTU DUMMY,0,1
SGER SGER,0
*****
SFF
26
103 SPM,1
104 SPM,2
105 QTA.DO_ANJO,0
106 SPM/SEIXAL,0
107 INT.BAR,150
108 TRANSF,1
110 TRANSF,2
111 FOGUETEIRO,2
112 TRANSF,3
615 B.COND.,1
617 TRANSF,1
618 C.BARREIRO,0
619 INT.BAR,60
620 QUIMIG./CBR,0
621 TRANSF,2
622 MATA,2
623 MATA,1
624 CAPARICA,0
625 TRANSF,3
B61 BARR.1,60
B62 BARR.2,60
BAR1 BARR,150
BAR6 BARR,60
OPA OPA,0
RTU DUMMY,0,1
SGER SGER,0
*****
SFN
31
101 SSV,0
205 IB/BYP,220
207 SCG,2
208 A.MIRA,2
209 TRANSF,5
211 SCH/SAM,3
212 SCH,0
213 SSV,2
214 TRANSF,6
215 INT.BAR,220
217 TRANSF,7
402 TRANSF,2
403 TRANSF,3
406 ALTO_MIRA,4
407 ALTO_MIRA,5
409 TRANSF,5
410 IB/BYP,400
412 SRM,0
413 TRANSF,6
414 SPM,0
415 INT.BAR,400
604 LTE,2
605 TRANSF,2
606 BARR.60_B,0
607 LTE,3
```

```
608 TRANSF,3
BAR2 BARR,220
BAR4 BARR,400
BAR6 BARR,60
OPA OPA,0
SGER SGER,0
*****
SFR
9
104 SZR,0
105 CFT/SPC,0
106 TRANSF,2
109 IB/BYP,150
605 SPC,0
606 TRANSF,2
COM SEQ.DE_COMANDO,0
OPA OPA,0
SGER SGER,0
*****
SGR
5
B61 BARR,60
OPA OPA,0
SGER SGER,0
T TRANSF,0
TB TRANSF,0
*****
SMC
16
221 SPR/SMR,0
222 TRANSF,1
223 BARR,220
224 CCL,0
225 TRANSF,2
611 ESGUEIRA,2
612 ESGUEIRA,1
614 TRANSF,1
615 BARRO/BUSTOS,0
617 IB/BYP,60
618 B.COND.,1
619 TRANSF,2
620 AGUEDA,0
B61 BARRB,60
OPA OPA,0
SGER SGER,0
*****
SMG
9
202 CPT,0
203 TRANSF,2
204 CVR,0
602 E.NORTE,1
603 BARR,60
604 TRANSF,2
605 E.NORTE,2
OPA OPA,0
SGER SGER,0
*****
SMR
6
B61 BARR,60
OPA OPA,0
RTU DUMMY,0,1
SGER SGER,0
T TRANSF,0
TB TRANSF,0
*****
SOQ
8
101 STN,0
105 SSN,0
106 F.ALENTEJO,0
108 NEVES_CORVO,0
109 IB/BYP,150
111 SET,0
OPA OPA,0
SGER SGER,0
*****
SOR
15
```

```
111 IB/BYP,150
112 SVI,0
114 TRANSF,1
116 CCD,0
117 TRANSF,2
118 SRA,0
612 INT.BAR,60
613 PENIDE,0
615 TRANSF,1
616 DUME,0
617 BRAGA_URB.,0
619 TRANSF,2
COM SEQ.DE_COMANDO,0
RTU DUMMY2_(SOR1),0
SGER SGER,0
*****
SPA
23
109 TRANSF,4
110 TRANSF,3
112 INT.BAR,150
114 SPM,2
115 SPM,1
116 SEIXAL,0
117 Q.GRANDE,0
118 SZR,2
119 SZR,1
122 SSV,0
123 SSV/SZR_2,0
634 TRANSF,3
635 INT.BAR,60
636 TRANSF,4
638 CARRASCAL,1
639 CARRASCAL,2
B61 BARR.1,60
B62 BARR.2,60
BAR1 BARR,150
BAR6 BARR,60
OPA OPA,0
RTU DUMMY,0,1
SGER SGER,0
*****
SPB
7
201 TRANSF,1
202 TRANSF,2
601 TRANSF,0
602 TRANSF,2
B61 BARR,60
OPA OPA,0
SGER SGER,0
*****
SPC
6
604 TRANSF.B,0
OPA OPA,0
RTU DUMMY,0,1
SGER SGER,0
T TRANSF,0
TB TRANSF.B,0
*****
SPG
11
411 MOD.41,0
412 MOD.41,0
413 MOD.41,0
421 MOD.42,0
422 SRM,0
423 MOD.42,0
BAR4 BARR,400
COM SEQ.DE_COMANDO,0
M41 CCIE/MOD.CPG,1
M42 SRM/MOD.CPG,2
SGER SGER,0
*****
SPM
31
106 SER/PEGOES,0
108 P.ALTO,2
109 TRANSF,2
```

110 P.ALTO,1
 115 SEIXAL/SFF,0
 116 QTA.DO_ANJO,0
 117 IB/BYP,150
 118 F.FERRO,2
 119 F.FERRO,1
 120 INT.BAR,150
 122 SSB,1
 123 SSB,2
 124 TRANSF,3
 125 SSB,3
 126 SSN,1
 127 RESERVA,0
 403 SFN,0
 407 SSN,3
 408 TRANSF,2
 413 IB/BYP,400
 414 SRM,0
 415 TRANSF,3
 416 SSN,2
 418 C.SETUBAL,1
 419 C.SETUBAL,2
 420 INT.BARB,400
 422 C.SETUBAL,3
 423 C.SETUBAL,4
 COM SIST.DE_COMANDO,0
 RTU DUMMY,0,1
 SGER SGER,0

SPN

27
 203 TRANSF,1
 204 SVG,1
 205 SVG,2
 206 CHAFARIZ1,0
 209 INT.BAR,220
 210 CPT,0
 211 CBT,0
 212 SAA,0
 213 CHAFARIZ,2
 214 SAUCELLE,0
 215 CPN,0
 301 TRANSF,3
 302 MONCORVO,0
 303 CPN,0
 304 FREIXO_NUMAO,0
 305 SA/TRANSF,4
 306 VILA_FLOR,0
 307 CARRAZEDA,0
 605 FOZ_COA,0
 606 INT.BAR,60
 607 PINH./MAC.,2
 608 MAC.,1
 609 TRANSF,1
 610 TRANSF,3
 B61 BARR.1,60
 B62 BARR.2,60
 SGER SGER,0

SPR

39
 119 TRANSF,1
 120 SZR,2
 121 INT.BAR,150
 122 SZR,1
 123 TRANSF,2
 126 TRANSF,3
 204 CAG/MORTAGU.,0
 205 CAG,1
 206 INT.BAR,220
 207 SVC,2
 209 SVC/MORTAGU.,0
 210 SMC/SMR,0
 211 TRANSF,6
 212 SBL,1
 213 TRANSF,1
 214 SPB/SBL,2
 215 TRANSF,5
 216 TRANSF,2
 634 ALFAR.,2

635 TRANSF,3
636 TAV/ALFAR.,1
637 TRANSF,5
638 S.CLARA,3
640 S.CLARA,2
641 INT.BAR,60
642 S.CLARA,1
643 TRANSF,6
644 B.COND.,3
645 A.S.JOAO,0
646 RAIVA/AGR,0
647 B.COND.,1/2
B61 BARR.1,60
B62 BARR.2,60
BAR1 BARR,150
BAR2 BARR,220
BAR6 BARR,60
OPA OPA,0
RTU DUMMY,0,2
SGER SGER,0

SQA

5
100 BARR.1,150
101 F.FERRO,0
102 SPM,0
RTU DUMMY,0,1
SGER SGER,0

SRA

46
117 OLEIROS,0
118 SED,4
119 TRANSF,2
121 BARR.1,150
122 BARR.2,150
123 TRANSF,1
124 TRANSF,6
125 TRANSF,5
126 SGR/CCD,2
127 CCD,1
128 IB/BYPB,150
129 VILA_NOVA,0
130 SED,1
131 SED,2
132 SED,3
133 SRU,2
413 TRANSF,4
415 TRANSF,3
416 SAL,1
417 SAL,2
418 IB/BYP,400
420 TRANSF,2
421 SRR,2
422 SRR,1
423 TRANSF,1
614 B.COND.,1
617 PEVIDEM,0
618 S.JOAO_PONTE,0
619 IB/BYPA,60
620 DUME,0
621 TRANSF,6
623 TRANSF,5
624 SRU,1
625 RESERVA,0
627 BARR.1,60
628 BARR.2,60
629 B.COND.,2
630 TRANSF,3
633 IB/BYPB,60
635 AREIAS,0
637 S.M.CAMPO,0
638 TRANSF,4
B61 BARR.1,60
B62 BARR.2,60
OPA OPA,0
SGER SGER,0

SRM

38

```

205 SBL,2
206 IB/BYPA,220
207 SBL,1
209 TRANSF,2
210 RESERVA,0
211 BARR.1,220
212 BARR.1,220
213 SCG,3
214 TRANSF,3
215 SCG,2
216 SCG,1
217 IB/BYPB,220
218 STJ,0
220 TRANSF,4
221 INT.BAR,220
223 TRANSF,5
408 TRANSF,2
411 SRR,1
412 SRR,2
416 PCPG,0
417 SPM,0
418 IB/BYP,400
421 SFN,0
422 TRANSF,3
424 INT.BAR,400
601 TRANSF,4
602 IB/BYP,60
604 SANCHEIRA,0
605 TURQUEL/RM,0
606 TRANSF,5
B61 BARR.1,60
B62 BARR.2,60
BAR2 BARR,220
BAR4 BARR,400
COM SIST.DE_COMANDO,0
OPA OPA,0
RTU DUMMY1_(SRM),0
SGER S.GERAIS_(SRM2),0
*****
SRR
24
201 SVG,1
202 TRANSF,1
203 SVG,2
204 IB/BYP,220
206 TRANSF,2
207 SCT,0
208 SVM,1
209 SVM,2
211 SCN,1
212 SCN,2
215 TURBOGAS,0
216 CTG/SCN,3
217 STR,0
402 SRA,1
403 TRANSF,1
404 SRA,2
406 TRANSF,2
410 SRM,2
411 IB/BYP,400
414 SRM,1
COM SEQ.DE_COMANDO,0
OPA OPA,0
RTU DUMMY1_(SRR5),0
SGER S.GERAIS_(SRR5),0
*****
SSB
28
101 SPM,1
102 TRANSF,1
103 SPM,2
104 TRANSF,2
105 INT.BAR,150
106 TRANSF,3
107 SPM,3
108 TRANSF,4
620 B.COND.,1/2
621 TRANSF,1
623 ALGERUZ,2
624 CSB,1

```

625 TRANSF,2
626 CSB,2
627 SADO,0
628 EUROMINAS,0
629 TRANSF,3
630 ALGERUZ,1
631 S.SEBAST.,2
632 S.SEBAST.,1
633 TRANSF,4
634 INT.BAR,60
B61 BARR.1,60
B62 BARR.2,60
BAR6 BARR,60
OPA OPA,0
RTU DUMMY,0,1
SGER SGER,0

SSN

35
103 SPM,1
104 F.ALENTEJO,0
105 TRANSF,1
106 SOQ,0
107 RESERVA,0
108 TRANSF,2
109 STN,1
110 STN,2
111 C.SINES,1
113 TRANSF,5
114 INT.BAR,150
115 TRANSF,6
402 INT.BAR,400
404 IB/BYP,400
405 SPM,2
406 SPM,3
409 TRANSF,5
410 C.SINES,2
412 C.SINES,3
413 C.SINES,4
605 BOREALIS,1
606 BOREALIS,2
607 M.FEIO,1
608 M.FEIO,2
609 TRANSF,2
610 INT.BAR,60
612 PETROGAL,0
614 C.SINES,0
615 V.N.MILFONT,0
616 SANTIAGO,0
B61 BARR.1,60
B62 BARR.2,60
BAR6 BARR,60
RTU DUMMY,0,1
SGER SGER,0

SSV

41
102 TRANSF,1
103 TRANSF,2
104 TRANSF,3
105 SFN,0
106 P.ALTO,0
107 TRANSF,4
108 SPA/SZR,2
109 TRANSF,6
110 TRANSF,5
111 TRANSF,7
113 INT.BAR,150
300 BARR.2/30,0
324 TRANSF,3
325 S.AUX.,1
326 MOSCAVIDE,2
327 TRANSF,2
328 MOSCAVIDE,1
329 OFICINAS,0
330 TRANSF,1
331 RESERVA_EQ.,0
332 EDINFOR,1
333 MOSCAVIDE,5
335 B.COND.,4

```

336 B.COND.,3
338 RESERVA_EQ.,0
339 EDINFOR,2
340 RESERVA_EQ.,0
341 INT.BARR.30,0
342 MOSCAVIDE,4
343 TRANSF,5
344 MOSCAVIDE,3
345 S.AUX.,2
346 TRANSF,4
615 TRANSF,6
B31A BARR.1/30A,0
B32B BARR.2/30BB,0
BAR1 BARR,150
BAR3 BARR.30,0
OPA OPA_(150KV),0
RTU DUMMY2_(150KV),0
SGER S.GERAIS_(150KV),0
*****
STJ
13
214 SRM,0
215 A.MIRA,0
216 TRANSF,2
217 IB/BYP,220
219 TRANSF,3
613 B.COND.,1
615 TRANSF,2
616 BARR.1,60
618 BARR.1,60
619 TRANSF,3
OPA OPA,0
RTU DUMMY,0,1
SGER SGER,0
*****
STN
35
100 TRANSF.GRUPO,3
101 TRANSF,6
102 SSN,2
103 TRANSF,4
104 SSN,1
105 TRANSF,2
106 SOQ,0
107 TRANSF,1
109 INT.BAR,150
614 B.COND.,1
617 LOULE,2
618 LOULE,1
619 TRANSF,1
620 INT.BAR,60
621 ALBUFEIRA,2
622 ALBUFEIRA,1
623 TRANSF,2
624 LAMEIRAS,0
625 TRANSF.GRUPO,1
626 B.COND.,2
627 TRANSF,4
628 PORTO_LAGOS,0
629 TRANSF.GRUPO,2
631 TRANSF,6
632 TRANSF.GRUPO,4
B61 BARR.1,60
B62 BARR.2,60
BAR1 BARR,150
BAR6 BARR,60
OPA OPA,0
SGER SGER,0
TUR1 TURBINA_A_GAS,1
TUR2 TURBINA_A_GAS,2
TUR3 TURBINA_A_GAS,3
TUR4 TURBINA_A_GAS,4
*****
STR
12
210 SRR,0
211 CTR,2
213 IB/BYP,220
215 TRANSF,2
216 CTR,1

```

```
217 CCL,0
615 IB/BYP,60
616 ENTRE_RIOS,0
617 TRANSF,2
618 BUSTELO,0
OPA OPA,0
SGER SGER,0
*****
SVC
28
213 IB/BYPA,220
214 GVA/SCF,1
215 SPR/MORTAGU.,0
216 TRANSF,1
217 TRANSF,2
218 TRANSF,3
219 TRANSF,4
220 GVA/SCF,2
221 SPR,2
222 IB/BYPB,220
601 B.COND.,2
602 TRANSF,4
603 SEIA,2
604 SEIA,1
605 TRANSF,3
606 B.COND.,1
607 INT.BAR,60
608 TRANSF,2
609 TONDELA,0
610 VISO,0
611 TRANSF,1
B61 BARR.1,60
B62 BARR.2,60
BAR2 BARR,220
BAR6 BARR,60
OPA OPA,0
RTU DUMMY,0,1
SGER SGER,0
*****
SVG
29
200 BARR.2,220
201 INT.BARA,220
202 SRR,2
203 SRR,1
204 SVM,0
206 CRG,0
207 TRANSF,2
208 TRANSF,1
209 CVR,2
210 CVR,1
211 SPN,1
212 SPN,2
213 CCL,3
214 CCL,2
215 CCL,1
216 INT.BARB,220
613 B.COND.,1
616 INT.BAR,60
617 TELHEIRA,0
618 TRANSF,2
622 VAROSA,1
623 TRANSF,1
624 VAROSA,2
B61 BARR.1,60
B62 BARR.2,60
BAR2 BARR,220
BAR6 BARR,60
OPA OPA,0
SGER SGER,0
*****
SVI
17
106 OLEIROS,0
107 TRANSF,2
108 CCD,2
109 CVF/CCD,1
110 TRANSF,3
603 B.COND.,2
607 TRANSF,2
```

```
608 IB/BYP,60
610 S.R.NEIVA,0
612 TRANSF,3
614 PORTUZELO,1
615 FEIT./PORT.,0
616 PORTUZELO,2
B61 BARR.1,60
B62 BARR.2,60
OPA OPA,0
SGER SGER,0
*****
SVM
53
100 BARR,150
141 MAIA/SED,3
142 TRANSF,0
143 SED,2
144 TRANSF,1
145 SED,1
147 TRANSF,2
202 TRANSF,7
204 TRANSF,6
205 SCT,0
206 TRANSF,5
207 SRR,1
208 TRANSF,4
209 INT.BAR,220
210 TRANSF,3
211 SVG,0
212 TRANSF,2
213 SRR,2
214 TRANSF,1
216 TRANSF,0
617 TRANSF,3
618 TRAVAGEM,1
619 TRAVAGEM,2
620 AMIEIRA,2
621 AMIEIRA,1
622 TRANSF,4
623 ALFENA,0
624 TAP.OUTEIRO,0
625 AMIEIRA,3
626 CIRC.,3
627 TRANSF,5
628 CIRC.,4
629 GUEIFAES,0
630 CIRC.,1
631 CIRC.,2
632 TRANSF,6
633 INT.BAR,60
634 MAIA,1
635 SCT/SCBISPO,0
636 MAIA,2
637 TRANSF,7
638 B.COND.,3
639 BEIRIZ,0
640 LOUSADO,0
655 B.COND.,2
B61 BARR.1,60
B62 BARR.2,60
BAR1 BARR,150
BAR2 BARR,220
BAR6 BARR,60
OPA OPA,0
RTU DUMMY,0,2
SGER SGER,0
*****
SXX
16
210 TRANSF,1
211 SCG,0
212 SFN,2
213 INT.BAR,220
214 TRANSF,2
610 TRANSF,1
611 MOSCAVIDE,1
612 MOSCAVIDE,2
613 INT.BAR,60
614 TRANSF,7
BAR2 BARR,220
```

```
BAR6 BARR,60
COM SEQ.DE_COMANDO,0
OPA OPA,0
RTU DUMMY2_(220KV),0
SGER S.GERAIS_(220KV),0
*****
SYN
3
104 GRUPO,0
112 TRANSF,0
RTU CONTROLO_INST.,0
*****
SZR
35
100 BARR.2,150
101 INT.BAR,150
102 SPR,1
103 SPR,2
104 TRANSF,1
105 CBC,1
106 TRANSF,2
107 CBC,2
109 C.BODE,1
110 TRANSF,3
111 C.BODE,2
112 SPA/SSV,2
113 C.BODE,3
114 P.ALTO,1
115 SPA/Q.GRANDE,0
116 SFG,0
620 B.COND.,1
621 VENDA_NOVA,0
622 TRANSF,1
623 ENTRONCAM.,0
624 OLHO_BOI,1
625 ABRANTE/CP,1
626 TRANSF,2
627 ABRANTE/CP,2
628 OLHO_BOI_2/3,0
629 FONTAINHAS,0
631 SANTA_CITA,0
632 TRANSF,3
633 INT.BAR,60
B61 BARR.1,60
B62 BARR.2,60
BAR1 BARR,150
BAR6 BARR,60
RTU DUMMY,0,2
SGER SGER,0
*****
TDR
4
104 GRUPO,0
112 TRANSF,0
PLC TREG.GRUPO,3
RTU CONTROLO_INST.,0
*****
```

Anexo B

Exemplo com Factos produzidos pelo SPARSE e Relatório produzido pelo TEMPUS

Este anexo, inclui :

- Os factos produzidos pelo SPARSE mediante o tratamento de um ficheiro de incidentes:
 - *regra_nfatos(NR, NB, LFactoS, LFactoNS)*
 - *facto_antigo(N, F, T, TI)*
- O relatório produzido pelo TEMPUS, resultante do processamento efectuado aos factos anteriores e mediante a escolha por parte do utilizador da análise específica das regras d1 e d21 e das instalações SPN e SVG.

Factos produzidos pelo SPARSE mediante o tratamento de um ficheiro de incidentes :

```
facto_antigo(1,mensagem('01-JUL-98','04:44:49.
53',['SVG','209',['CVR','2']], '>>>DISPARO','01'),t(204957889,53),t(204957889,53)).
facto_antigo(2,mensagem('01-JUL-98','04:44:49.
73',['SVG','209',['CVR','2','DISJ']], 'DISJUNTOR','00'),t(204957889,73),t(204957889,73))
.
regra_nfatos(d3,1,[1,2,condicao(mod_dif_tempos_menor_ou_igual(t(204957889,73),t(204957
889,53),30))],[[]]).
regra_nfatos(i1,1,[3,nao_inicio_de_incidente(_471,_472),nao
verifica_fim_incidente(_476,_477)],[[]]).
facto_antigo(-1,disj('01-JUL-98','04:44:48','SVG','209','
',prot,fechado),t(204957888,0),t(204957889,933)).
facto_antigo(6,mensagem('01-JUL-
98','04:44:49.933',['SVG','209',['CVR','2','DISJ']], 'DISJUNTOR','01'),t(204957889,933),
t(204957889,933)).
regra_nfatos(e2,1,[6,-1],[[]]).
facto_antigo(7,disj('01-JUL-
98','04:44:49.933','SVG','209',_430,nao_identificado,fechado),t(204957889,933),t(204957
889,933)).
facto_antigo(3,disparo('01-JUL-98','04:44:49.
53','SVG','209','CVR','2',monofásico,nao_identificado,t(204957889,73)),t(204957889,53),
t(204957889,933)).
regra_nfatos(d5,1,[3,7,condicao(mod_dif_tempos_menor_ou_igual(t(204957889,933),t(20495
7889,73),100))],[[]]).
facto_antigo(4,disj_aberto_disparo('SVG','209'),t(204957889,53),t(204957889,933)).
regra_nfatos(i3,1,[9,4],[[]]).
facto_antigo(10,disparo('01-JUL-
98','04:44:49.933','SVG','209','CVR','2',monofásico,rel_rap_mono,t(204957889,73)),t(204
957889,933),t(204957889,933)).
regra_nfatos(d9,1,[10],[[]]).
facto_antigo(-2,disj('01-JUL-98','04:44:48','SPN','304','
',prot,fechado),t(204957888,0),t(204957942,58)).
facto_antigo(15,mensagem('01-JUL-98','04:45:42.
58',['SPN','304',['FREIXO_NUMAO','0','DISJ']], 'DISJUNTOR','10'),t(204957942,58),t(20495
7942,58)).
regra_nfatos(e1,1,[15,-2],[[]]).
```

```

facto_antigo(16,disj('01-JUL-98','04:45:42.
58','SPN','304',_610,nao_identificado,aberto),t(204957942,58),t(204957942,58)).
facto_antigo(13,mensagem('01-JUL-
98','04:45:41.944',['SPN','304',['FREIXO_NUMAO','0']], '>>>DISPARO','01'),t(204957941,94
4),t(204957941,944)).
facto_antigo(14,mensagem('01-JUL-98','04:45:42.
18',['SPN','304',['FREIXO_NUMAO','0','DISJ']], 'DISJUNTOR','00'),t(204957942,18),t(20495
7942,18)).
regra_nfactos(d1,1,[13,16,condicao(dif_tempos_menor_ou_igual(t(204957942,58),t(20495794
1,944),30)),14,condicao(mod_dif_tempos_menor_ou_igual(t(204957941,944),t(204957942,18),
30))),[]).
facto_antigo(12,disparo('01-JUL-
98','04:44:49.933','SVG','209','CVR','2',monofásico,dmr,t(204957889,73)),t(204957889,93
3),t(204957889,933)).
regra_nfactos(d22,1,[12,verificar(disparo_identificado(dmr)),verificar(mensagem_utiliza
dor(dmr,['aberta por DmR']))],[]).
facto_antigo(17,disparo('01-JUL-
98','04:45:41.944','SPN','304','FREIXO_NUMAO','0',trifásico,nao_identificado,t(20495794
2,58)),t(204957941,944),_552).
regra_nfactos(d10,1,[17],[]).
facto_antigo(18,disj('01-JUL-98','04:45:42.
58','SPN','304',_754,disparo,aberto),t(204957942,58),t(204957956,178)).
facto_antigo(25,mensagem('01-JUL-
98','04:45:56.178',['SPN','304',['FREIXO_NUMAO','0','DISJ']], 'DISJUNTOR','01'),t(204957
956,178),t(204957956,178)).
regra_nfactos(e2,1,[25,18],[]).
facto_antigo(22,mensagem('01-JUL-
98','04:45:55.973',['SVG','212',['SPN','2']], '>>>DISPARO','01'),t(204957955,973),t(2049
57955,973)).
facto_antigo(23,mensagem('01-JUL-
98','04:45:55.993',['SVG','212',['SPN','2','DISJ']], 'DISJUNTOR','00'),t(204957955,993),
t(204957955,993)).
regra_nfactos(d3,1,[22,23,condicao(mod_dif_tempos_menor_ou_igual(t(204957955,993),t(204
957955,973),30))),[]).
facto_antigo(-3,disj('01-JUL-98','04:44:48','SVG','212','
',prot,fechado),t(204957888,0),t(204957956,853)).
facto_antigo(31,mensagem('01-JUL-
98','04:45:56.853',['SVG','212',['SPN','2','DISJ']], 'DISJUNTOR','01'),t(204957956,853),
t(204957956,853)).
regra_nfactos(e2,1,[31,-3],[]).
facto_antigo(26,mensagem('01-JUL-
98','04:45:56.365',['SPN','205',['SVG','2']], '>>>DISPARO','01'),t(204957956,365),t(2049
57956,365)).
facto_antigo(28,mensagem('01-JUL-
98','04:45:56.432',['SPN','205',['SVG','2','DISJ']], 'DISJUNTOR','00'),t(204957956,432),
t(204957956,432)).
regra_nfactos(d3,1,[26,28,condicao(mod_dif_tempos_menor_ou_igual(t(204957956,432),t(204
957956,365),30))),[]).
facto_antigo(27,disj('01-JUL-
98','04:45:56.178','SPN','304',_670,nao_identificado,fechado),t(204957956,178),t(204957
956,178)).
regra_nfactos(j1,3,[27,-6,verificar(motivo_de_alteracao_de_disjuntor(opa,opa))],[]).
facto_antigo(32,disj('01-JUL-
98','04:45:56.853','SVG','212',_605,nao_identificado,fechado),t(204957956,853),t(204957
956,853)).
facto_antigo(29,disparo('01-JUL-
98','04:45:55.973','SVG','212','SPN','2',monofásico,nao_identificado,t(204957955,993)),
t(204957955,973),t(204957956,853)).
regra_nfactos(d5,1,[29,32,condicao(mod_dif_tempos_menor_ou_igual(t(204957956,853),t(204
957955,993),100))),[]).
facto_antigo(19,disj_aberto_disparo('SPN','304'),t(204957941,944),t(204957956,178)).
facto_antigo(11,verifica_fim_incidente('01-JUL-
98','04:44:49.933'),t(204957889,933),t(204957956,178)).
regra_nfactos(i3,1,[35,19],[]).
facto_antigo(30,disj_aberto_disparo('SVG','212'),t(204957955,973),t(204957956,853)).
facto_antigo(40,verifica_fim_incidente('01-JUL-
98','04:45:56.178'),t(204957956,178),t(204957956,853)).
regra_nfactos(i3,1,[38,30],[]).
facto_antigo(-4,disj('01-JUL-98','04:44:48','SPN','205','
',prot,fechado),t(204957888,0),t(204957957,352)).
facto_antigo(42,mensagem('01-JUL-
98','04:45:57.352',['SPN','205',['SVG','2','DISJ']], 'DISJUNTOR','01'),t(204957957,352),
t(204957957,352)).
regra_nfactos(e2,1,[42,-4],[]).
facto_antigo(43,disj('01-JUL-
98','04:45:57.352','SPN','205',_487,nao_identificado,fechado),t(204957957,352),t(204957
957,352)).

```

Relatório produzido pelo TEMPUS

```

facto_antigo(33,disparo('01-JUL-
98','04:45:56.365','SPN','205','SVG','2',monofásico,nao_identificado,t(204957956,432)),
t(204957956,365),t(204957957,352)).
regra_nfactos(d5,1,[33,43,condicao(mod_dif_tempos_menor_ou_igual(t(204957957,352),t(204
957956,432),100))],[[]).
facto_antigo(34,disj_aberto_disparo('SPN','205'),t(204957956,365),t(204957957,352)).
facto_antigo(41,verifica_fim_incidente('01-JUL-
98','04:45:56.853'),t(204957956,853),t(204957957,352)).
regra_nfactos(i3,1,[45,34],[[]).
regra_nfactos(j3,1,[36,verificar(mensagem_utilizador(comando_disj,opa,[' pelo
opa']))],[[]).
facto_antigo(39,disparo('01-JUL-
98','04:45:56.853','SVG','212','SPN','2',monofásico,rel_rap_mono,t(204957955,993)),t(20
4957956,853),t(204957956,853)).
regra_nfactos(d9,1,[39],[[]).
facto_antigo(46,disparo('01-JUL-
98','04:45:57.352','SPN','205','SVG','2',monofásico,rel_rap_mono,t(204957956,432)),t(20
4957957,352),t(204957957,352)).
regra_nfactos(d9,1,[46],[[]).
facto_antigo(50,disparo('01-JUL-
98','04:45:57.352','SPN','205','SVG','2',monofásico,dmr,t(204957956,432)),t(204957957,3
52),t(204957957,352)).
facto_antigo(48,disparo('01-JUL-
98','04:45:56.853','SVG','212','SPN','2',monofásico,dmr,t(204957955,993)),t(204957956,8
53),t(204957956,853)).
regra_nfactos(d21,1,[50,48,verificar(disparo_identificado(dmr)),verificar(disparo_ident
ificado(dmr)),condicao(mod_dif_tempos_menor_ou_igual(t(204957957,352),t(204957956,853),
300)),verificar(data_e_hora_do_tempo_max(t(204957957,352),t(204957956,853),t(204957957,
352),'01-JUL-98','04:45:57.352','01-JUL-98','04:45:56.853','01-JUL-
98','04:45:57.352'))),verificar(mensagem_utilizador(dmr,dmr,'SPN','SVG',[' Dmr nos dois
extremos']))],[[]).
facto_antigo(-5,disj('01-JUL-98','04:44:48','SPN','306','
',prot,fechado),t(204957888,0),t(204957998,154)).
facto_antigo(54,mensagem('01-JUL-
98','04:46:38.154',['SPN','306',['VILA_FLOR','0','DISJ']], 'DISJUNTOR','10'),t(204957998
,154),t(204957998,154)).
regra_nfactos(e1,1,[54,-5],[[]).
facto_antigo(55,disj('01-JUL-
98','04:46:38.154','SPN','306',_452,nao_identificado,aberto),t(204957998,154),t(2049579
98,154)).
facto_antigo(52,mensagem('01-JUL-98','04:46:38.
94',['SPN','306',['VILA_FLOR','0']], '>>>DISPARO','01'),t(204957998,94),t(204957998,94))
.
facto_antigo(53,mensagem('01-JUL-
98','04:46:38.114',['SPN','306',['VILA_FLOR','0','DISJ']], 'DISJUNTOR','00'),t(204957998
,114),t(204957998,114)).
regra_nfactos(d1,1,[52,55,condicao(dif_tempos_menor_ou_igual(t(204957998,154),t(2049579
98,94),30)),53,condicao(mod_dif_tempos_menor_ou_igual(t(204957998,94),t(204957998,114),
30))],[[]).
facto_antigo(35,disj('01-JUL-
98','04:45:56.178','SPN','304',_482,opa,fechado),t(204957956,178),t(204957998,518)).
facto_antigo(60,mensagem('01-JUL-
98','04:46:38.518',['SPN','304',['FREIXO_NUMAO','0','DISJ']], 'DISJUNTOR','10'),t(204957
998,518),t(204957998,518)).
regra_nfactos(e1,1,[60,35],[[]).
regra_nfactos(d1,1,[49,61],[condicao(dif_tempos_menor_ou_igual(t(204957998,518),t(20495
7998,54),30))]).
regra_nfactos(d2,1,[49,61],[condicao(dif_tempos_menor_ou_igual(t(204957998,518),t(20495
7998,54),30))]).
regra_nfactos(d1,1,[49,61],[condicao(dif_tempos_menor_ou_igual(t(204957998,518),t(20495
7998,54),30))]).
regra_nfactos(d2,1,[49,61],[condicao(dif_tempos_menor_ou_igual(t(204957998,518),t(20495
7998,54),30))]).
regra_nfactos(d1,1,[49,61],[condicao(dif_tempos_menor_ou_igual(t(204957998,518),t(20495
7998,54),30))]).
regra_nfactos(d2,1,[49,61],[condicao(dif_tempos_menor_ou_igual(t(204957998,518),t(20495
7998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
facto_antigo(56,disparo('01-JUL-98','04:46:38.
94','SPN','306','VILA_FLOR','0',trifásico,nao_identificado,t(204957998,154)),t(20495799
8,94),_788).
regra_nfactos(d10,1,[56],[[]).
regra_nfactos(j5,3,[61,-
6,verificar(motivo_de_alteracao_de_disjuntor(opa,opa)),verificar(mensagem_utilizador(co
mando_disj_abert,automatico,['com subestacao em automatico sem disparo']))],[[]).

```

Relatório produzido pelo TEMPUS

```

facto_antigo(21,disparo('01-JUL-
98','04:45:41.944','SPN','304','FREIXO_NUMAO','0',trifásico,ds,t(204957942,58)),t(20495
7941,944),t(204957941,944)).
regra_nfactos(d22,1,[21,verificar(disparo_identificado(ds)),verificar(mensagem_utilizad
or(ds,['aberta por DS']))],[]).
regra_nfactos(d1,1,[49,61],[condicao(dif_tempos_menor_ou_igual(t(204957998,518),t(20495
7998,54),30))]).
regra_nfactos(d2,1,[49,61],[condicao(dif_tempos_menor_ou_igual(t(204957998,518),t(20495
7998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d1,1,[49,61],[condicao(dif_tempos_menor_ou_igual(t(204957998,518),t(20495
7998,54),30))]).
regra_nfactos(d2,1,[49,61],[condicao(dif_tempos_menor_ou_igual(t(204957998,518),t(20495
7998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d1,1,[49,61],[condicao(dif_tempos_menor_ou_igual(t(204957998,518),t(20495
7998,54),30))]).
regra_nfactos(d2,1,[49,61],[condicao(dif_tempos_menor_ou_igual(t(204957998,518),t(20495
7998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d1,1,[49,61],[condicao(dif_tempos_menor_ou_igual(t(204957998,518),t(20495
7998,54),30))]).
regra_nfactos(d2,1,[49,61],[condicao(dif_tempos_menor_ou_igual(t(204957998,518),t(20495
7998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d1,1,[49,61],[condicao(dif_tempos_menor_ou_igual(t(204957998,518),t(20495
7998,54),30))]).
regra_nfactos(d2,1,[49,61],[condicao(dif_tempos_menor_ou_igual(t(204957998,518),t(20495
7998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
facto_antigo(61,disj('01-JUL-
98','04:46:38.518','SPN','304',_463,nao_identificado,aberto),t(204957998,518),t(2049580
12,164)).
facto_antigo(66,mensagem('01-JUL-
98','04:46:52.164',['SPN','304',['FREIXO_NUMAO','0','DISJ']],DISJUNTOR,'01'),t(204958
012,164),t(204958012,164)).
regra_nfactos(e2,1,[66,61],[]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
facto_antigo(67,disj('01-JUL-
98','04:46:52.164','SPN','304',_605,nao_identificado,fechado),t(204958012,164),t(204958
012,164)).
regra_nfactos(j1,3,[67,-6,verificar(motivo_de_alteracao_de_disjuntor(opa,opa))],[]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(j3,1,[69,verificar(mensagem_utilizador(comando_disj,opa,[' pelo
opa']))],[]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
facto_antigo(64,disj('01-JUL-
98','04:46:38.518','SPN','304',_631,opa,aberto),t(204957998,518),t(204958024,64)).
facto_antigo(72,mensagem('01-JUL-98','04:47:04.
64',['SPN','304',['FREIXO_NUMAO','0','DISJ']],DISJUNTOR,'10'),t(204958024,64),t(20495
8024,64)).
regra_nfactos(e1,1,[72,68],[]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d1,1,[70,73,condicao(dif_tempos_menor_ou_igual(t(204958024,64),t(20495802
3,954),30)),24],[condicao(mod_dif_tempos_menor_ou_igual(t(204958023,954),t(204957956,11
8),30))]).

```

Relatório produzido pelo TEMPUS

```

facto_antigo(73,disj('01-JUL-98','04:47:04.
64','SPN','304',_818,nao_identificado,aberto),t(204958024,64),t(204958024,64)).
facto_antigo(70,mensagem('01-JUL-
98','04:47:03.954','SPN','304',['FREIXO_NUMAO','0'],'>>>DISPARO','01'),t(204958023,95
4),t(204958023,954)).
regra_nfactos(d2,1,[70,73,condicao(dif_tempos_menor_ou_igual(t(204958024,64),t(20495802
3,954),30))],[]).
facto_antigo(47,verifica_fim_incidente('01-JUL-
98','04:45:57.352'),t(204957957,352),t(204957957,352)).
regra_nfactos(i5,1,[47],[]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
facto_antigo(75,disparo('01-JUL-
98','04:47:03.954','SPN','304','FREIXO_NUMAO','0',trifásico,nao_identificado,t(20495802
4,64)),t(204958023,954),_636).
regra_nfactos(d10,1,[75],[]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
facto_antigo(68,disj('01-JUL-
98','04:46:52.164','SPN','304',_407,opa,fechado),t(204958012,164),t(204958038,4)).
facto_antigo(79,mensagem('01-JUL-98','04:47:18.
4',['SPN','304',['FREIXO_NUMAO','0'],'DISJ'],'DISJUNTOR','01'),t(204958038,4),t(2049580
38,4)).
regra_nfactos(e2,1,[79,68],[]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
facto_antigo(80,disj('01-JUL-98','04:47:18.
4','SPN','304',_549,nao_identificado,fechado),t(204958038,4),t(204958038,4)).
regra_nfactos(j1,3,[80,-6,verificar(motivo_de_alteracao_de_disjuntor(opa,opa))],[]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
facto_antigo(77,disj_aberto_disparo('SPN','304'),t(204958023,954),t(204958038,4)).
regra_nfactos(i3,1,[81,77],[]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(j3,1,[82,verificar(mensagem_utilizador(comando_disj,opa,[' pelo
opa']))],[]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
facto_antigo(63,disparo('01-JUL-98','04:46:38.
94','SPN','306','VILA_FLOR','0',trifásico,ds,t(204957998,154)),t(204957998,94),t(204957
998,94)).
regra_nfactos(d22,1,[63,verificar(disparo_identificado(ds)),verificar(mensagem_utilizad
or(ds,[' aberta por DS']))],[]).
facto_antigo(78,disparo('01-JUL-
98','04:47:03.954','SPN','304','FREIXO_NUMAO','0',trifásico,ds,t(204958024,64)),t(20495
8023,954),t(204958023,954)).
regra_nfactos(d22,1,[78,verificar(disparo_identificado(ds)),verificar(mensagem_utilizad
or(ds,[' aberta por DS']))],[]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30))]).

```

Relatório produzido pelo TEMPUS

```

regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30)))).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30)))).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30)))).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30)))).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30)))).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30)))).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30)))).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30)))).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30)))).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30)))).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30)))).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30)))).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30)))).
regra_nfactos(d3,1,[49,24],[condicao(mod_dif_tempos_menor_ou_igual(t(204957956,118),t(2
04957998,54),30)))).
facto_antigo(49,mensagem('01-JUL-98','04:46:38.
54',['SPN','304',['FREIXO_NUMAO','0']], '>>>DISPARO','01'),t(204957998,54),t(204957998,5
4)).
regra_nfactos(x,1,[49],[ ]).
facto_antigo(57,disj('01-JUL-
98','04:46:38.154',['SPN','306'],_379,disparo,aberto),t(204957998,154),t(204958129,494)).
facto_antigo(112,mensagem('01-JUL-
98','04:48:49.494',['SPN','306',['VILA_FLOR','0'],'DISJ']], 'DISJUNTOR','01'),t(204958129
,494),t(204958129,494)).
regra_nfactos(e2,1,[112,57],[ ]).
facto_antigo(113,disj('01-JUL-
98','04:48:49.494',['SPN','306'],_523,nao_identificado,fechado),t(204958129,494),t(204958
129,494)).
regra_nfactos(j1,4,[113,verificar(motivo_de_alteracao_de_disjuntor(outro,outro))],[ ]).
facto_antigo(58,disj_aberto_disparo('SPN','306'),t(204957998,94),t(204958129,494)).
facto_antigo(105,verifica_fim_incidente('01-JUL-
98','04:48:12.553'),t(204958092,553),t(204958129,494)).
regra_nfactos(i3,1,[114,58],[ ]).
regra_nfactos(j3,1,[115,verificar(mensagem_utilizador(comando_disj,outro,['por comando
desconhecido'])],[ ]).
facto_antigo(110>manual('01-JUL-
98','04:48:19.324',['SPN'],t(204958099,324),t(204958176,344)).
regra_nfactos(g2,1,[117,110],[ ]).
facto_antigo(76,disj('01-JUL-98','04:47:04.
64',['SPN','304'],_757,disparo,aberto),t(204958024,64),t(204958187,784)).
facto_antigo(121,mensagem('01-JUL-
98','04:49:47.784',['SPN','304',['FREIXO_NUMAO','0'],'DISJ']], 'DISJUNTOR','10'),t(204958
187,784),t(204958187,784)).
regra_nfactos(e1,1,[121,81],[ ]).
regra_nfactos(d1,1,[118,123,condicao(dif_tempos_menor_ou_igual(t(204958187,784),t(20495
8187,654),30)),24],[condicao(mod_dif_tempos_menor_ou_igual(t(204958187,654),t(204957956
,118),30)))).
facto_antigo(123,disj('01-JUL-
98','04:49:47.784',['SPN','304'],_878,nao_identificado,aberto),t(204958187,784),t(2049581
87,784)).
facto_antigo(118,mensagem('01-JUL-
98','04:49:47.654',['SPN','304',['FREIXO_NUMAO','0']], '>>>DISPARO','01'),t(204958187,65
4),t(204958187,654)).
regra_nfactos(d2,1,[118,123,condicao(dif_tempos_menor_ou_igual(t(204958187,784),t(20495
8187,654),30))],[ ]).
facto_antigo(116,verifica_fim_incidente('01-JUL-
98','04:48:49.494'),t(204958129,494),t(204958129,494)).
regra_nfactos(i5,1,[116],[ ]).
facto_antigo(124,disparo('01-JUL-
98','04:49:47.654',['SPN','304'],'FREIXO_NUMAO','0',trifásico,nao_identificado,t(20495818
7,784)),t(204958187,654),_608)).
regra_nfactos(d10,1,[124],[ ]).

```

Relatório produzido pelo TEMPUS

```

facto_antigo(81,disj('01-JUL-98','04:47:18.
4','SPN','304',_379,opa,fechado),t(204958038,4),t(204958201,544)).
facto_antigo(128,mensagem('01-JUL-
98','04:50:01.544',['SPN','304',['FREIXO_NUMAO','0','DISJ']], 'DISJUNTOR','01'),t(204958
201,544),t(204958201,544)).
regra_nfactos(e2,1,[128,81],[ ]).
facto_antigo(129,disj('01-JUL-
98','04:50:01.544','SPN','304',_523,nao_identificado,fechado),t(204958201,544),t(204958
201,544)).
regra_nfactos(j1,3,[129,119,verificar(motivo_de_alteracao_de_disjuntor(opa,opa))],[ ]).
facto_antigo(126,disj_aberto_disparo('SPN','304'),t(204958187,654),t(204958201,544)).
regra_nfactos(i3,1,[130,126],[ ]).
regra_nfactos(j3,1,[131,verificar(mensagem_utilizador(comando_disj,opa,[' pelo
opa'])),[ ]).
facto_antigo(127,disparo('01-JUL-
98','04:49:47.654','SPN','304','FREIXO_NUMAO','0',trifásico,ds,t(204958187,784)),t(2049
58187,654),t(204958187,654)).
regra_nfactos(d22,1,[127,verificar(disparo_identificado(ds)),verificar(mensagem_utiliza
dor(ds,[' aberta por DS'])),[ ]).
facto_antigo(132,verifica_fim_incidente('01-JUL-
98','04:50:01.544'),t(204958201,544),t(204958201,544)).
facto_antigo(5,inicio_de_incidente('01-JUL-98','04:44:49.
53'),t(204957889,53),t(204958201,544)).
regra_nfactos(i4,1,[132,nao_disj_aberto_disparo(_352,_353)],[ ]).
facto_antigo(114,disj('01-JUL-
98','04:48:49.494','SPN','306',_362,outro,fechado),t(204958129,494),t(204958310,394)).
facto_antigo(138,mensagem('01-JUL-
98','04:51:50.394',['SPN','306',['VILA_FLOR','0','DISJ']], 'DISJUNTOR','10'),t(204958310
,394),t(204958310,394)).
regra_nfactos(e1,1,[138,114],[ ]).
facto_antigo(125,disj('01-JUL-
98','04:49:47.784','SPN','304',_605,disparo,aberto),t(204958187,784),t(204958310,424)).
facto_antigo(140,mensagem('01-JUL-
98','04:51:50.424',['SPN','304',['FREIXO_NUMAO','0','DISJ']], 'DISJUNTOR','10'),t(204958
310,424),t(204958310,424)).
regra_nfactos(e1,1,[140,130],[ ]).
regra_nfactos(d1,1,[134,141,condicao(dif_tempos_menor_ou_igual(t(204958310,424),t(20495
8310,314),30)),24],[condicao(mod_dif_tempos_menor_ou_igual(t(204958310,314),t(204957956
,118),30))]).
facto_antigo(141,disj('01-JUL-
98','04:51:50.424','SPN','304',_677,nao_identificado,aberto),t(204958310,424),t(2049583
10,424)).
facto_antigo(134,mensagem('01-JUL-
98','04:51:50.314',['SPN','304',['FREIXO_NUMAO','0']], '>>>DISPARO','01'),t(204958310,31
4),t(204958310,314)).
regra_nfactos(d2,1,[134,141,condicao(dif_tempos_menor_ou_igual(t(204958310,424),t(20495
8310,314),30))],[ ]).
regra_nfactos(d1,1,[135,139,condicao(dif_tempos_menor_ou_igual(t(204958310,394),t(20495
8310,334),30)),111],[condicao(mod_dif_tempos_menor_ou_igual(t(204958310,334),t(20495812
9,454),30))]).
facto_antigo(139,disj('01-JUL-
98','04:51:50.394','SPN','306',_710,nao_identificado,aberto),t(204958310,394),t(2049583
10,394)).
facto_antigo(135,mensagem('01-JUL-
98','04:51:50.334',['SPN','306',['VILA_FLOR','0']], '>>>DISPARO','01'),t(204958310,334),
t(204958310,334)).
regra_nfactos(d2,1,[135,139,condicao(dif_tempos_menor_ou_igual(t(204958310,394),t(20495
8310,334),30))],[ ]).
regra_nfactos(i1,1,[143,nao_inicio_de_incidente(_590,_591),nao
verifica_fim_incidente(_595,_596)],[ ]).
facto_antigo(143,disparo('01-JUL-
98','04:51:50.314','SPN','304','FREIXO_NUMAO','0',trifásico,nao_identificado,t(20495831
0,424),t(204958310,314),_608)).
regra_nfactos(d10,1,[143],[ ]).
facto_antigo(146,disparo('01-JUL-
98','04:51:50.334','SPN','306','VILA_FLOR','0',trifásico,nao_identificado,t(204958310,3
94),t(204958310,334),_608)).
regra_nfactos(d10,1,[146],[ ]).
facto_antigo(130,disj('01-JUL-
98','04:50:01.544','SPN','304',_379,opa,fechado),t(204958201,544),t(204958324,284)).
facto_antigo(152,mensagem('01-JUL-
98','04:52:04.284',['SPN','304',['FREIXO_NUMAO','0','DISJ']], 'DISJUNTOR','01'),t(204958
324,284),t(204958324,284)).
regra_nfactos(e2,1,[152,130],[ ]).
facto_antigo(153,disj('01-JUL-
98','04:52:04.284','SPN','304',_523,nao_identificado,fechado),t(204958324,284),t(204958
324,284)).
regra_nfactos(j1,3,[153,119,verificar(motivo_de_alteracao_de_disjuntor(opa,opa))],[ ]).

```

```

facto_antigo(145,disj_aberto_disparo('SPN','304'),t(204958310,314),t(204958324,284)).
regra_nfactos(i3,1,[154,145],[[]]).
regra_nfactos(j3,1,[155,verificar(mensagem_utilizador(comando_disj,opa,[' pelo
opa'])))],[[]]).
facto_antigo(150,disparo('01-JUL-
98','04:51:50.314','SPN','304','FREIXO_NUMAO','0',trifásico,ds,t(204958310,424)),t(2049
58310,314),t(204958310,314)).
regra_nfactos(d22,1,[150,verificar(disparo_identificado(ds)),verificar(mensagem_utiliza
dor(ds,[' aberta por DS'])))],[[]]).
facto_antigo(151,disparo('01-JUL-
98','04:51:50.334','SPN','306','VILA_FLOR','0',trifásico,ds,t(204958310,394)),t(2049583
10,334),t(204958310,334)).
regra_nfactos(d22,1,[151,verificar(disparo_identificado(ds)),verificar(mensagem_utiliza
dor(ds,[' aberta por DS'])))],[[]]).
facto_antigo(156,verifica_fim_incidente('01-JUL-
98','04:52:04.284'),t(204958324,284),t(204958324,284)).
regra_nfactos(i5,1,[156],[[]]).
facto_antigo(119,automatico('01-JUL-
98','04:49:36.344','SPN'),t(204958176,344),t(204958454,884)).
regra_nfactos(g1,1,[159,119],[[]]).
facto_antigo(147,disj('01-JUL-
98','04:51:50.394','SPN','306',_559,disparo,aberto),t(204958310,394),t(204958471,635)).
facto_antigo(162,mensagem('01-JUL-
98','04:54:31.635','SPN','306',['VILA_FLOR','0','DISJ'],'DISJUNTOR','01'),t(204958471
,635),t(204958471,635)).
regra_nfactos(e2,1,[162,147],[[]]).
facto_antigo(164,disj('01-JUL-
98','04:54:31.635','SPN','306',_577,nao_identificado,fechado),t(204958471,635),t(204958
471,635)).
regra_nfactos(j1,4,[164,verificar(motivo_de_alteracao_de_disjuntor(outro,outro))],[[]]).
facto_antigo(148,disj_aberto_disparo('SPN','306'),t(204958310,334),t(204958471,635)).
regra_nfactos(i3,1,[165,148],[[]]).
regra_nfactos(j3,1,[166,verificar(mensagem_utilizador(comando_disj,outro,['por comando
desconhecido'])))],[[]]).
facto_antigo(161>manual('01-JUL-
98','04:54:14.884','SPN'),t(204958454,884),t(204958489,764)).
regra_nfactos(g2,1,[163,161],[[]]).
facto_antigo(167,verifica_fim_incidente('01-JUL-
98','04:54:31.635'),t(204958471,635),t(204958471,635)).
facto_antigo(149,inicio_de_incidente('01-JUL-
98','04:51:50.314'),t(204958310,314),t(204958471,635)).
regra_nfactos(i4,1,[167,nao_disj_aberto_disparo(_518,_519)],[[]]).
facto_antigo(24,mensagem('01-JUL-
98','04:45:56.118',['SPN','304',['FREIXO_NUMAO','0','DISJ'],'DISJUNTOR','00'),t(204957
956,118),t(204957956,118)).
regra_nfactos(x,1,[24],[[]]).
facto_antigo(59,mensagem('01-JUL-
98','04:46:38.458',['SPN','304',['FREIXO_NUMAO','0','DISJ'],'DISJUNTOR','00'),t(204957
998,458),t(204957998,458)).
regra_nfactos(x,1,[59],[[]]).
facto_antigo(62,mensagem('01-JUL-
98','04:46:52.104',['SPN','304',['FREIXO_NUMAO','0','DISJ'],'DISJUNTOR','00'),t(204958
012,104),t(204958012,104)).
regra_nfactos(x,1,[62],[[]]).
facto_antigo(71,mensagem('01-JUL-98','04:47:04.
24',['SPN','304',['FREIXO_NUMAO','0','DISJ'],'DISJUNTOR','00'),t(204958024,24),t(20495
8024,24)).
regra_nfactos(x,1,[71],[[]]).
facto_antigo(74,mensagem('01-JUL-
98','04:47:17.944',['SPN','304',['FREIXO_NUMAO','0','DISJ'],'DISJUNTOR','00'),t(204958
037,944),t(204958037,944)).
regra_nfactos(x,1,[74],[[]]).

```

Relatório produzido pelo TEMPUS

De seguida, encontra-se o relatório produzido pelo TEMPUS, resultante do processamento efectuado aos factos anteriores e mediante a escolha por parte do utilizador da análise específica das regras d1 e d21 e das instalações SPN e SVG

#####

REGRA d1:DISPARO TRIFÁSICO COM DISJUNTOR EM MOVIMENTO

```
[
  [
    mensagem(Data1,Hora1,[Inst1,Painel1,[Inst2,NL]], '>>>DISPARO','01') at T1
  e
    disj(Data2,Hora2,Inst1,Painel1,_,nao_identificado,aberto) at T2
  e
    condicao(dif_tempos_menor_ou_igual(T2,T1,300))
  e
    mensagem(Data3,Hora3,[Inst1,Painel1,[Inst2,NL,_]], 'DISJUNTOR','00') at T3
  e
    condicao(mod_dif_tempos_menor_ou_igual(T1,T3,300))
  ]
]

==>
[
  cria_facto(disparo(Data1,Hora1,Inst1,Painel1,Inst2,NL,trifásico,
    nao_identificado,T2),T1),
  retira_facto(disj(_,_,Inst1,Painel1,_,nao_identificado,aberto),_,T2),
  cria_facto(disj(Data2,Hora2,Inst1,Painel1,_,disparo,aberto),T2),
  retira_facto(mensagem(Data1,Hora1,[Inst1,Painel1,[Inst2,NL]], '>>>DISPARO',
    '01'),T1,T1),
  retira_facto(mensagem(Data3,Hora3,[Inst1,Painel1,[Inst2,NL,_]], 'DISJUNTOR',
    '00'),T3,T3),
  retira_facto(disj_aberto_disparo(Inst1,Painel1),_,T1),
  cria_facto(disj_aberto_disparo(Inst1,Painel1),T1)
].
```

Bloco 1

SUCEDIDAS

INSTALAÇÃO SPN

PAINEL 304

A diferença de tempos poderá ser menor ou igual a 114 milésimas de segundo.
O módulo da diferença de tempos poderá ser menor ou igual a 74 milésimas de segundo.

PAINEL 306

A diferença de tempos poderá ser menor ou igual a 60 milésimas de segundo.
O módulo da diferença de tempos poderá ser menor ou igual a 20 milésimas de segundo.

INSTALAÇÃO SVG

Regra sem disparos nesta instalação

NÃO SUCEDIDAS

INSTALAÇÃO SPN

PAINEL 304

A diferença de tempos terá que ser menor ou igual a 464 milésimas de segundo.

+++++

A diferença de tempos poderá ser menor ou igual a 110 milésimas de segundo.
O módulo da diferença de tempos terá que ser menor ou igual a 67836 milésimas de segundo.

Relatório produzido pelo TEMPUS

+++++

A diferença de tempos poderá ser menor ou igual a 130 milésimas de segundo.
O módulo da diferença de tempos terá que ser menor ou igual a 231536 milésimas de segundo.

+++++

A diferença de tempos poderá ser menor ou igual a 110 milésimas de segundo.
O módulo da diferença de tempos terá que ser menor ou igual a 354196 milésimas de segundo.

PAINEL 306

A diferença de tempos poderá ser menor ou igual a 60 milésimas de segundo.
O módulo da diferença de tempos terá que ser menor ou igual a 180880 milésimas de segundo.

INSTALAÇÃO SVG

Regra sem disparos não sucedidos nesta instalação

REGRA d21:DISPARO NOS DOIS EXTREMOS

```
[
[
disparo(Data1,Hora1,Inst1,Painel1,Inst2,NL,_,TipoDisparo1,Tabr1) at T1
e
disparo(Data2,Hora2,Inst2,Painel2,Inst1,NL,_,TipoDisparo2,Tabr2) at T2
e
verificar( disparo_identificado(TipoDisparo1))
e
verificar( disparo_identificado(TipoDisparo2))
e
condicao(mod_dif_tempos_menor_ou_igual(T1,T2,300))
e
verificar(data_e_hora_do_tempo_max(T1,T2,Tmax,Data1,Hora1,Data2,Hora2,Data,
Hora))
e
verificar(mensagem_utilizador(TipoDisparo1,TipoDisparo2,Inst1,Inst2,
Mensagem))
]
]
==>
[
cria_facto(disparo_2extremos(Data,Hora,Inst1,Painel1,Inst2,Painel2,NL,
TipoDisparo1,TipoDisparo2),Tmax),
retira_facto(disparo(Data1,Hora1,Inst1,Painel1,Inst2,NL,_,TipoDisparo1,
Tabr1),T1,T1),
retira_facto(disparo(Data2,Hora2,Inst2,Painel2,Inst1,NL,_,TipoDisparo2,
Tabr2),T2,T2),
escreve_mens1(Data,Hora,(0,Inst1),(0,Painel1),['linha ',Inst1,'- ',Inst2,
' ',NL,Mensagem])
].
```

Bloco 1

SUCEDIDAS

INSTALAÇÃO SPN

PAINEL 205

O módulo da diferença de tempos poderá ser menor ou igual a 499 milésimas de segundo.

INSTALAÇÃO SVG

PAINEL 212

O módulo da diferença de tempos poderá ser menor ou igual a 160 milésimas de segundo.

NÃO SUCECIDAS

Regra sem disparos não sucedidos devido ao incumprimento de premissas temporais.

Anexo C

Resultados

Anexo C

Resultados

Este anexo inclui os resultados obtidos pelo TEMPUS na análise efectuada a 12 ficheiros mensais de incidentes. Os resultados são apresentados por regra e as situações de disparo da regra são apresentadas separadamente das situações de não disparo da regra.

Regra d1	LOG	Instalação	Painel	Condição 1	Condição 2
				300	300
SUCEDIDAS					
	Mai-98	CTO	608	40	20
	Aug-98			60	20
	Mai-98		614	40	20
	Dez-98		618	40	20
	Mai-98			60	20
	Aug-98	SAM	208	80	60
	Aug-98		212	54	34
	Dez-98			56	36
	Set-98		214	112	49
	Aug-98			190	50
	Mai-98		617	138	58
	Mar-98			144	64
	Mai-98		618	118	82
	Mar-98			124	43
	Aug-98		620	104	24
	Jul-98		632	97	58
	Aug-98		635	238	138
	Aug-98		640	284	164
	Mai-98		643	140	60
	Aug-98			222	142
	Aug-98	SBL	238	40	20
	Mai-98		613	120	40
	Aug-98		616	121	99
	Mai-98			140	40
	Jun-98		618	180	80
	Dez-98			200	80
	Jul-98		619	80	20
	Aug-98	SCF	210	60	20
	Set-98		211	50	30

	Mai-98			150	10
	Aug-98			160	20
	Jul-98		212	20	0
	Aug-98			30	10
	Set-98			30	10
	Set-98		213	30	10
	Set-98		216	109	89
	Set-98		224	109	89
	Apr-98	SCG	204	99	79
	Mar-98			102	82
	Jan-99			105	85
	Dez-98			119	80
	Jul-98			120	83
	Aug-98			179	79
	Mar-98		212	80	60
	Aug-98			83	63
	Nov-98			89	69
	Jul-98			98	78
	Jan-99			99	79
	Nov-98		216	91	71
	Aug-98			92	72
	Jul-98			110	90
	Jan-99			119	99
	Dez-98			123	103
	Aug-98		218	32	12
	Jan-99			99	79
	Aug-98			106	86
	Dez-98			112	88
	Jul-98			120	83
	Nov-98			120	80
	Nov-98		236	173	93
	Mar-98			176	96
	Jan-99			179	99
	Aug-98			180	94
	Mai-98			181	101
	Jul-98			182	102
	Apr-98			197	97
	Aug-98	SCH	210	30	10
	Dez-98			30	10
	Mai-98	SCL	203	50	10
	Jul-98		205	40	10
	Jun-98		207	20	0
	Apr-98		210	20	10
	Aug-98			20	0
	Set-98	SCN	613	85	65
	Mai-98		614	100	80
	Aug-98		615	120	40
	Mai-98			120	40

	Dez-98		622	80	60
	Mai-98		626	100	40
	Set-98	SED	629	248	88
	Mai-98		630	227	81
	Mai-98		633	141	101
	Mai-98		634	121	41
	Aug-98			127	68
	Aug-98		635	143	23
	Set-98			167	47
	Apr-98	SEJ	208	30	10
	Aug-98			118	30
	Apr-98		603	40	10
	Jan-98		605	30	140
	Set-98			30	90
	Set-98		607	50	20
	Set-98		612	30	10
	Dez-98		613	30	0
	Jun-98		615	40	0
	Nov-98			60	30
	Nov-98		616	50	20
	Mai-98		617	30	0
	Set-98			30	0
	Feb-98	SFA	617	40	20
	Mai-98			60	20
	Feb-98		618	80	40
	Mar-98			80	40
	Jun-98			80	40
	Set-98			80	40
	Mai-98			100	40
	Dez-98			140	121
	Mai-98		624	60	40
	Jan-98			80	60
	Jul-98			80	60
	Nov-98			80	60
	Set-98			80	60
	Dez-98			80	60
	Apr-98		625	79	40
	Feb-98			100	40
	Mar-98			100	160
	Set-98			100	40
	Mai-98			120	80
	Jul-98		626	100	300
	Jun-98			100	80
	Dez-98			100	80
	Apr-98			120	80
	Mar-98			120	200
	Mai-98			120	80
	Nov-98			120	80

	Set-98	SFF	615	48	28
	Mai-98		618	89	30
	Mai-98		620	86	6
	Dez-98		622	145	55
	Dez-98		623	104	56
	Nov-98		624	118	62
	Aug-98	SFN	211	79	59
	Set-98			99	39
	Dez-98		212	60	40
	Aug-98			100	40
	Apr-98		213	60	20
	Mai-98			60	40
	Dez-98		412	40	20
	Dez-98		414	40	20
	Feb-98	SMC	611	0	20
	Mar-98			21	0
	Nov-98			21	0
	Set-98		612	100	20
	Dez-98			100	40
	Mai-98	SOQ	101	60	40
	Set-98			60	40
	Set-98		105	140	40
	Aug-98	SOR	112	0	20
	Mai-98			80	20
	Set-98		116	40	20
	Jan-99		616	40	0
	Jan-99	SPA	114	104	24
	Jun-98			106	26
	Mar-98			119	39
	Mar-98		116	124	44
	Nov-98		118	118	38
	Apr-98			122	42
	Jan-99			122	22
	Feb-98			123	43
	Jul-98			124	44
	Mai-98			124	44
	Dez-98			126	46
	Jan-98			127	47
	Jun-98			130	30
	Set-98		122	123	23
	Mar-98			142	42
	Jun-98		210	105	65
	Jun-98		302	153	93
	Jun-98		303	139	79
	Jun-98		304	115	75
	Jun-98		306	80	41
	Jun-98		307	80	41
	Jun-98		607	88	48

	Jun-98		608	111	12
	Jan-99	SPM	108	133	113
	Dez-98		110	158	138
	Dez-98		118	162	102
	Dez-98		119	121	82
	Aug-98		414	240	120
	Nov-98		416	180	140
	Aug-98		418	138	118
	Set-98			141	121
	Mar-98			143	103
	Jan-99		419	233	203
	Aug-98		422	180	140
	Aug-98	SPN	204	67	47
	Aug-98		205	127	67
	Jul-98			128	69
	Aug-98		210	59	20
	Mai-98			63	43
	Aug-98		211	58	18
	Mai-98		212	55	15
	Set-98		213	80	40
	Set-98		214	70	50
	Aug-98			80	60
	Dez-98		302	138	79
	Jan-99			139	79
	Jan-98			149	78
	Apr-98			151	79
	Mai-98			160	100
	Aug-98			236	176
	Jul-98			237	177
	Set-98			237	158
	Feb-98			239	179
	Jan-98		303	154	94
	Apr-98			157	96
	Set-98			171	111
	Aug-98			176	116
	Jul-98			177	117
	Nov-98			276	114
	Mar-98		304	137	77
	Mai-98			137	257
	Jan-99			138	182
	Apr-98			139	79
	Jan-98			140	79
	Dez-98			152	96
	Nov-98			154	97
	Set-98			158	150
	Aug-98			170	110
	Jul-98			176	116
	Jul-98			176	116

	Feb-98			179	119
	Jan-98		306	60	40
	Feb-98			60	40
	Jul-98			60	41
	Set-98			79	40
	Aug-98			80	41
	Mar-98			80	41
	Apr-98			81	60
	Mai-98			81	41
	Dez-98			81	41
	Jan-99			81	161
	Nov-98			220	201
	Jan-98		307	60	40
	Apr-98			79	41
	Jul-98			80	40
	Mai-98			80	40
	Jan-99			80	120
	Aug-98			180	161
	Nov-98			180	160
	Set-98			180	161
	Feb-98			200	159
	Aug-98		607	95	55
	Jan-98			98	58
	Dez-98			98	58
	Apr-98			99	51
	Mai-98			100	57
	Mai-98		608	149	70
	Aug-98	SPR	120	180	120
	Mai-98		205	80	40
	Mai-98		210	116	36
	Aug-98		214	117	57
	Aug-98		634	107	47
	Jul-98		636	134	60
	Jun-98			137	77
	Aug-98			138	78
	Mai-98		646	136	56
	Mai-98	SRA	129	60	0
	Dez-98			80	20
	Aug-98		416	80	40
	Aug-98		417	60	40
	Mai-98			60	20
	Aug-98		422	60	40
	Set-98		617	100	40
	Set-98		618	60	10
	Mai-98			61	0
	Mai-98		620	80	20
	Jun-98		629	140	100
	Aug-98		635	100	20
	Aug-98	SRM	411	138	213

	Mai-98		412	131	101
	Aug-98			156	125
	Apr-98			160	130
	Mar-98		417	162	141
	Aug-98			180	160
	Aug-98		421	200	111
	Apr-98			206	127
	Mai-98	SRR	201	50	30
	Jan-99			50	30
	Jul-98			60	30
	Aug-98			60	240
	Mai-98		203	50	30
	Jan-99			50	30
	Aug-98			60	30
	Aug-98		402	41	20
	Apr-98		410	40	10
	Aug-98			40	20
	Jan-98			50	40
	Aug-98		414	50	30
	Nov-98	SSB	628	135	75
	Aug-98		630	120	80
	Set-98		631	80	300
	Set-98		632	80	20
	Aug-98			100	39
	Mai-98	SSN	104	121	60
	Set-98		605	113	32
	Jun-98		607	116	36
	Dez-98		615	112	52
	Set-98			113	167
	Jan-98		616	100	80
	Set-98			120	100
	Set-98	SSV	220	64	44
	Apr-98		332	125	84
	Jul-98		336	117	77
	Jan-98			120	80
	Set-98		610	108	44
	Aug-98	STJ	613	40	20
	Set-98	STN	106	115	52
	Mai-98			121	48
	Dez-98		617	62	2
	Jul-98			68	28
	Set-98			91	31
	Dez-98		624	55	15
	Jan-98		628	99	59
	Jul-98			100	59
	Aug-98			100	60
	Jan-99	STR	206	60	20
	Jan-98		211	60	0

	Feb-98		216	60	0
	Mar-98			60	20
	Jul-98			80	60
	Mai-98		618	80	40
	Aug-98	SVC	220	69	49
	Jul-98		601	74	54
	Jun-98		609	86	46
	Nov-98			101	61
	Aug-98			107	67
	Mai-98			110	70
	Aug-98		610	106	43
	Jul-98			108	29
	Mai-98			113	53
	Jul-98	SVG	202	0	60
	Mai-98			40	0
	Jan-99			60	20
	Aug-98			140	60
	Mai-98		203	0	40
	Jul-98			60	0
	Jan-99			80	40
	Aug-98			240	40
	Aug-98		204	80	40
	Jan-98		211	80	41
	Aug-98			80	60
	Jul-98		212	80	20
	Aug-98			80	39
	Mai-98		213	80	40
	Jul-98		214	60	40
	Mai-98			60	40
	Jun-98		215	60	20
	Set-98		617	80	40
	Jan-98		622	120	20
	Dez-98			120	20
	Set-98	SVI	106	60	40
	Aug-98			61	40
	Mai-98		108	59	40
	Aug-98			60	20
	Set-98			60	20
	Set-98		109	20	0
	Mar-98			40	20
	Mai-98			40	20
	Aug-98		603	80	40
	Mai-98		610	100	40
	Aug-98			119	59
	Mai-98		614	83	43
	Mai-98		615	100	40
	Mai-98		616	100	40
	Aug-98	SVM	211	82	94

	Aug-98		638	94	54
	Jun-98			112	268
	Mar-98		640	121	40
	Aug-98	SZR	103	114	94
	Set-98		112	103	82
	Aug-98			153	133
	Jul-98		114	100	92
	Jan-98		115	63	43
	Feb-98			71	51
	Mai-98			86	66
	Jul-98			90	70
	Nov-98			91	71
	Dez-98			91	71
	Jan-99			92	72
	Jun-98			95	74
	Apr-98			111	91
	Mai-98		116	72	52
	Mai-98		621	118	78
	Mai-98		623	79	40
	Jan-98		625	89	49
	Mai-98			94	46
	Jul-98			107	47
	Jul-98		627	70	50
	Set-98			74	54
	Nov-98			76	57
	Mai-98			92	72
	Set-98		628	110	50
	Mai-98			111	55
	Mai-98		631	99	267

Máximo	284	300
Mínimo	0	0
Média	102.24	62.29
Moda	80	40
Desvio Padrão	49.32	48.66

[0-30]	5.81%	25.51%
]30-60]	18.43%	36.87%
]60-90]	19.70%	17.93%
]90-120]	27.27%	10.10%
]120-150]	14.14%	3.54%
]150-180]	9.09%	3.28%
]180-210]	2.02%	1.01%
]210-240]	2.78%	0.51%
]240-270]	0.25%	0.76%
]270-300]	0.51%	0.51%

[0-30]	23	101
]30-60]	73	146
]60-90]	78	71

]90-120]	108	40
]120-150]	56	14
]150-180]	36	13
]180-210]	8	4
]210-240]	11	2
]240-270]	1	3
]270-300]	2	2

NÃO SUCEDIDAS				
1ª Cond. Temporal				
	Mai-98	SCG	220	868
	Jan-98	SCT	607	620
	Jun-98	SET	601	2220
	Jun-98			8320
	Jun-98			9320
	Dez-98	SFA	626	380
	Nov-98	SFF	619	1720
	Dez-98	SFN	410	6520
	Mar-98	SPA	114	337
	Set-98		117	6328
	Mar-98		122	484
	Mar-98			485
	Mar-98			488
	Mar-98			504
	Mar-98			504
	Mar-98			508
	Mar-98			508
	Mar-98			528
	Jun-98		639	396
	Jun-98			396
	Aug-98	SPM	416	360
	Dez-98			14660
	Mai-98	SPN	302	314
	Mai-98			314
	Mai-98			317
	Mai-98			396
	Mai-98			500
	July-98		304	464
	Feb-98		306	339
	Jul-98			400

Mínimo 314

]300-330]	10.00%	3
]330-360]	10.00%	3
]360-390]	3.33%	1
]390-420]	13.33%	4
]420-450]	0.00%	0

]450-480]	3.33%	1
]480-510]	26.67%	8
]510-540]	3.33%	1
]540-570]	0.00%	0
]570-600]	0.00%	0
Mais que 600	30.00%	9

2ª Cond. Temporal					
	Set-98	SBL	243	0	3880
	Set-98		618	300	3860
	Aug-98	SCF	210	20	1420
	Aug-98			40	7500
	Set-98		211	30	6790
	Jul-98	SCG	212	79	14899
	Dez-98		224	20	2240
	Mai-98		236	180	10420
	Set-98	SED	635	146	514
	Set-98			146	514
	Set-98	SEJ	605	50	5780
	Jun-98	SFA	618	99	2660
	Feb-98			100	1680
	Mai-98			100	2700
	Mar-98			140	4220
	Mai-98		624	60	340
	Nov-98			60	320
	Apr-98			61	320
	Feb-98		625	100	520
	Mar-98		626	100	320
	Mai-98			120	600
	Nov-98	SFF	624	95	2105
	Nov-98			98	442
	Jan-99	SFN	213	80	14660
	Mai-98	SOQ	101	60	460
	Aug-98	SOR	112	40	14460
	Jan-99	SPM	108	108	1972
	Jan-99			126	2133
	Jan-99			127	2153
	Jan-99			132	2148
	Aug-98		414	140	620
	Nov-98		416	80	2540
	Apr-98	SPN	302	129	1730
	Nov-98			129	1311
	Jan-98			130	5590
	Mai-98			131	489
	Apr-98			134	1726
	Jan-98			136	14944
	Jul-98			140	1400
	Mai-98			140	1240

	Dez-98			140	1200
	Jan-99			140	1300
	Aug-98			150	1710
	Dez-98			151	1209
	Aug-98			156	1364
	Aug-98			157	1403
	Jan-98		303	154	766
	Mai-98		304	111	669
	Mai-98			117	663
	Dez-98		306	59	8019
	Jan-98			60	5660
	Dez-98			60	1281
	Jan-99			60	1220
	Set-98			61	1840
	Jan-98			79	2080
	Jan-98			80	14360
	Dez-98			221	14199
	Jan-98		307	60	4439
	Jan-98			60	5781
	Jan-98			60	1239
	Jan-98			60	3980
	Set-98			60	2000
	Jan-99			60	820
	Jan-98			80	1100
	Jan-98			93	5567
	Aug-98		607	90	1310
	Dez-98	SRA	129	80	7200
	Mai-98	SRM	412	88	1841
	Aug-98	SRR	201	50	7160
	Aug-98			50	8680
	Aug-98			60	4090
	Aug-98		410	40	1850
	Nov-98	SSB	628	115	585
	Set-98		632	80	1100
	Jan-98	SSV	335	130	2770
	Dez-98	STN	617	62	858
	Jan-99	STR	216	80	1260
	Jan-99			80	1280
	Aug-98	SVC	609	80	7460
	Aug-98			87	3073
	Aug-98	SVI	106	40	2760
	Aug-98		108	60	12020
	Mai-98		603	80	5040
	Aug-98		610	120	7420
	Mai-98		614	84	2197
	Mai-98	SZR	623	60	541
	Mai-98		625	95	825
	Mai-98		627	75	1005
	Set-98		628	110	530

Mínimo	530
]300-330]	3.37% 3
]330-360]	1.12% 1
]360-390]	0.00% 0
]390-420]	0.00% 0
]420-450]	1.12% 1
]450-480]	1.12% 1
]480-510]	1.12% 1
]510-540]	4.49% 4
]540-570]	1.12% 1
]570-600]	2.25% 2
Mais que 600	84.27% 75

Regra d2	LOG	Instalação	Painel	Condição 1
				300
SUCEDIDAS				
	Apr-98	CTG	204	96
	Mai-98	CTO	614	40
	Mai-98		618	40
	Aug-98	SAL	411	40
	Set-98			100
	Jan-98	SAM	205	120
	Aug-98		631	92
	Jul-98		632	94
	Jun-98		641	47
	Aug-98		642	24
	Feb-98			161
	Set-98	SBL	237	0
	Set-98		239	0
	Set-98		243	0
	Mai-98		613	100
	Aug-98		616	120
	Set-98		618	300
	Aug-98	SCF	210	40
	Set-98		211	30
	Jul-98			110
	Mai-98	SCG	204	1
	Aug-98			79
	Jan-99			82
	Apr-98			83
	Mai-98		208	1
	Apr-98		212	79
	Jul-98			79
	Aug-98			79
	Mai-98			85

	Mar-98			87
	Jul-98		216	104
	Dez-98		218	73
	Jun-98		220	79
	Dez-98		224	20
	Jan-99			96
	Jul-98			108
	Set-98		236	160
	Mar-98			172
	Mai-98			180
	Aug-98			181
	Jul-98	SCL	205	40
	Aug-98	SCN	216	80
	Aug-98		218	80
	Aug-98		221	60
	Mai-98		626	100
	Jan-98	SCT	607	120
	Jan-99			120
	Feb-98		609	120
	Set-98	SED	629	228
	Mai-98		630	201
	Set-98		635	146
	Set-98	SEJ	605	50
	Set-98		617	30
	Jan-98	SET	106	120
	Jun-98		601	130
	Mai-98		609	30
	Jul-98		612	120
	Jul-98	SFA	103	40
	Nov-98		108	100
	Dez-98		618	80
	Jun-98			99
	Feb-98			100
	Mai-98			100
	Mar-98			140
	Mai-98		624	60
	Nov-98			60
	Apr-98			61
	Dez-98			80
	Nov-98		625	0
	Mai-98			80
	Feb-98			100
	Mar-98		626	100
	Mai-98			120
	Dez-98			280
	Dez-98	SFF	111	61
	Nov-98		619	0
	Dez-98		623	113

	Nov-98		624	98
	Dez-98	SFN	101	60
	Jan-99		207	100
	Jan-99		208	100
	Jan-99		211	59
	Jan-99		212	100
	Jan-99		213	100
	Set-98		217	80
	Apr-98		412	40
	Dez-98			80
	Mai-98		413	20
	Dez-98		414	80
	Dez-98		415	40
	Set-98	SFR	606	166
	Mai-98	SMC	611	0
	Mai-98	SOQ	101	60
	Set-98			60
	Aug-98	SOR	112	40
	Mai-98			40
	Set-98		114	60
	Jan-99		617	32
	Jun-98	SPA	114	125
	Mar-98			149
	Feb-98		115	143
	Jan-98			147
	Mar-98		116	124
	Feb-98		117	101
	Set-98			101
	Aug-98		118	126
	Mar-98			146
	Mar-98		119	120
	Mar-98		122	240
	Jun-98		639	100
	Aug-98	SPB	201	120
	Nov-98			140
	Jul-98	SPC	T	160
	Jan-99	SPM	108	132
	Set-98		126	168
	Jul-98		413	105
	Aug-98		414	140
	Dez-98		416	167
	Nov-98			185
	Aug-98	SPN	204	55
	Set-98		302	131
	Nov-98			133
	Apr-98			138
	Jan-98			140
	Jul-98			140

	Jan-99			140
	Dez-98			151
	Mai-98			153
	Aug-98			157
	Apr-98		303	134
	Jan-98			154
	Jul-98		304	130
	Dez-98			131
	Aug-98			137
	Mai-98			138
	Jan-99			138
	Jun-98			139
	Apr-98			151
	Nov-98			154
	Jul-98		306	60
	Aug-98			61
	Jan-98			80
	Jan-99			80
	Apr-98			81
	Mai-98			81
	Set-98			201
	Nov-98			220
	Dez-98			221
	Jul-98		307	60
	Jan-98			80
	Apr-98			80
	Mai-98			80
	Set-98			80
	Jan-99			80
	Aug-98			201
	Aug-98		607	90
	Jan-98			93
	Mai-98			97
	Jan-98	SPR	212	146
	Mai-98		646	127
	Dez-98	SRA	129	80
	Mai-98		618	60
	Aug-98	SRM	411	129
	Mai-98		412	88
	Aug-98			134
	Apr-98		421	189
	Aug-98	SRR	201	60
	Aug-98		202	70
	Aug-98		203	50
	Aug-98		410	40
	Feb-98	SSB	102	140
	Mar-98		104	160
	Jan-98		620	300

	Nov-98		628	115
	Set-98		632	80
	Aug-98	SSN	405	60
	Jan-98		410	40
	Set-98		615	112
	Set-98	SSV	102	119
	Aug-98			129
	Nov-98		105	102
	Set-98			124
	Nov-98		106	94
	Dez-98		110	105
	Set-98		328	129
	Set-98		333	133
	Jan-98		335	130
	Jan-99			150
	Jul-98	STN	105	151
	Set-98		106	115
	Dez-98		617	66
	Nov-98		622	0
	Jul-98		625	71
	Jul-98		629	77
	Set-98	STR	211	40
	Jan-99			60
	Jan-98		216	60
	Jan-99			80
	Mai-98		618	80
	Set-98	SVC	214	83
	Aug-98			88
	Apr-98		215	2
	Aug-98		609	87
	Mai-98			107
	Nov-98			110
	Jun-98		610	106
	Aug-98	SVG	204	101
	Aug-98	SVI	106	40
	Aug-98		108	60
	Mai-98		603	80
	Aug-98		610	120
	Mai-98		614	84
	Mai-98		615	80
	Jan-98	SVM	204	201
	Mar-98		210	106
	Aug-98		211	113
	Jul-98		214	94
	Feb-98			115
	Feb-98		638	31
	Mar-98		655	100
	Feb-98			101

	Dez-98	SZR	105	160
	Mai-98		107	96
	Aug-98		114	293
	Jul-98		115	66
	Dez-98			72
	Apr-98			172
	Mai-98		623	60
	Mai-98		624	73
	Jul-98		625	89
	Mai-98			115
	Mar-98		627	66
	Aug-98			70
	Mai-98			75
	Set-98		628	110
	Mai-98		631	79
	Set-98			100

Máximo	300
Mínimo	0
Média	101.08
Moda	80
Desvio Padrão	52.84

[0-30]	6.75%	16
]30-60]	15.61%	37
]60-90]	22.78%	54
]90-120]	25.32%	60
]120-150]	15.61%	37
]150-180]	7.59%	18
]180-210]	2.95%	7
]210-240]	1.69%	4
]240-270]	0.00%	0
]270-300]	1.69%	4

NÃO SUCEDIDAS				
	Mai-98	SCG	220	868
	Jan-98	SCT	607	620
	Jun-98	SET	601	2220
	Jun-98			8320
	Jun-98			9320
	Dez-98	SFA	626	380
	Nov-98	SFF	619	1720
	Dez-98	SFN	410	6520
	Mar-98	SPA	114	337
	Set-98		117	6328
	Mar-98		122	484
	Mar-98			485
	Mar-98			488
	Mar-98			504

	Mar-98			504
	Mar-98			508
	Mar-98			508
	Mar-98			528
	Jun-98		639	396
	Jun-98			396
	Aug-98	SPM	416	360
	Dez-98			14660
	Mai-98	SPN	302	314
	Mai-98			314
	Mai-98			317
	Mai-98			396
	Mai-98			500
	Jul-98		304	464
	Feb-98		306	339
	Jul-98			400

Mínimo 314

]300-330]	10.00%	3
]330-360]	10.00%	3
]360-390]	3.33%	1
]390-420]	13.33%	4
]420-450]	0.00%	0
]450-480]	3.33%	1
]480-510]	26.67%	8
]510-540]	3.33%	1
]540-570]	0.00%	0
]570-600]	0.00%	0
Mais que 600	30.00%	9

Regra d3	LOG	Instalação	Painel	Condição 1
				300
SUCEDIDAS				
	Aug-98	SAL	411	20
	Mai-98			40
	Mai-98		422	20
	Aug-98			40
	Mai-98	CBR	602	60
	Apr-98		614	16
	Mai-98		622	118
	Mai-98	CTG	203	133
	Jul-98	SAM	206	29
	Aug-98		212	58
	Mar-98		642	172
	Aug-98	SBL	236	41
	Nov-98		238	40

	Dez-98			40
	Jan-98		242	65
	Mai-98		613	20
	Set-98	SCF	210	10
	Aug-98			20
	Aug-98		212	10
	Mai-98		614	10
	Mai-98		618	10
	Jul-98	SCG	204	79
	Set-98			79
	Nov-98			88
	Mai-98		206	1
	Jun-98			2
	Aug-98			30
	Mai-98		208	1
	Mai-98		210	9
	Jun-98			10
	Mai-98		216	9
	Mai-98		218	9
	Dez-98			12
	Aug-98			120
	Mai-98		220	9
	Mai-98		224	9
	Mai-98		228	9
	Mai-98		230	9
	Mai-98		232	8
	Mai-98		240	8
	Mai-98		242	8
	Mai-98		246	0
	Aug-98	SCH	210	30
	Aug-98		216	30
	Mai-98	SCL	203	110
	Mai-98		204	0
	Mai-98		205	20
	Jun-98			20
	Mai-98		206	20
	Set-98			100
	Mai-98		210	0
	Jan-99	SED	635	44
	Set-98	SEJ	605	90
	Jan-98	SER	106	40
	Apr-98		107	80
	Nov-98			80
	Apr-98		109	81
	Jan-98	SFA	102	20
	Apr-98			20
	Mai-98			20
	Dez-98			20

	Nov-98		104	20
	Jan-98			40
	Mai-98			40
	Aug-98			60
	Jun-98			60
	Jan-99			60
	Nov-98		105	80
	Nov-98		106	20
	Nov-98		107	60
	Nov-98		108	20
	Jan-98		110	20
	Nov-98			20
	Mai-98		617	20
	Nov-98			20
	Nov-98		618	20
	Nov-98		621	20
	Nov-98		624	80
	Mai-98		625	20
	Jul-98			40
	Mar-98			80
	Nov-98			80
	Nov-98		626	80
	Nov-98	SFF	103	214
	Nov-98		104	214
	Nov-98		105	214
	Mai-98		106	10
	Nov-98			214
	Nov-98		111	159
	Nov-98		112	20
	Dez-98			39
	Nov-98		619	260
	Nov-98		622	156
	Nov-98		623	156
	Nov-98		624	156
	Dez-98	SFN	101	80
	Aug-98		212	40
	Mai-98		213	40
	Set-98		217	100
	Nov-98		409	20
	Apr-98		412	40
	Aug-98			40
	Dez-98			40
	Aug-98		414	20
	Jun-98			20
	Feb-98			39
	Mai-98			40
	Dez-98			40
	Jun-98	SGR	T	40

	Aug-98	SOQ	101	40
	Mai-98			40
	Set-98			40
	Jan-98		106	20
	Apr-98			40
	Mai-98			40
	Dez-98			40
	Feb-98		111	140
	Mai-98			140
	Mai-98		115	5
	Set-98	SOR	116	20
	Jan-98	SPA	114	1
	Jun-98			4
	Feb-98			19
	Mar-98			19
	Jan-99			20
	Apr-98			23
	Aug-98		115	9
	Mar-98			19
	Apr-98			20
	Jul-98			20
	Jan-99			21
	Feb-98			299
	Set-98		116	0
	Feb-98			19
	Aug-98			27
	Feb-98		117	16
	Feb-98		118	19
	Jan-98			27
	Apr-98			47
	Aug-98		119	20
	Feb-98			68
	Jul-98			126
	Jun-98			238
	Feb-98		122	19
	Mar-98			43
	Mar-98	SPC	T	10
	Jan-98	SPG	422	20
	Apr-98			20
	Aug-98			20
	Feb-98			40
	Feb-98	SPM	108	98
	Apr-98			102
	Jun-98			103
	Jan-98			107
	Jan-99			118
	Aug-98		110	179
	Jan-99			202

	Jul-98			207
	Mai-98			207
	Apr-98			208
	Feb-98			220
	Mai-98		115	143
	Jul-98		119	97
	Aug-98		122	125
	Feb-98		126	133
	Mai-98			140
	Dez-98			144
	Jan-99			165
	Jun-98			166
	Aug-98			176
	Jul-98			204
	Mai-98		403	142
	Jun-98			145
	Aug-98			165
	Feb-98			172
	Dez-98		407	220
	Set-98			260
	Jul-98		414	99
	Aug-98			122
	Feb-98			124
	Feb-98			124
	Aug-98		416	100
	Mai-98			104
	Jan-98			123
	Nov-98			147
	Set-98			166
	Aug-98		423	151
	Mai-98	SPN	204	21
	Aug-98			38
	Jul-98		205	67
	Mai-98			70
	Jan-98		210	19
	Aug-98			28
	Jun-98			44
	Aug-98		211	20
	Mai-98			79
	Aug-98		212	50
	Jul-98		302	4
	Feb-98			9
	Set-98			9
	Aug-98			11
	Mai-98			254
	Nov-98		304	74
	Set-98			80
	Jan-99			80

	Aug-98			191
	Jul-98		306	20
	Nov-98			21
	Mai-98			41
	Set-98		307	20
	Feb-98			77
	Aug-98			160
	Nov-98			161
	Jan-99	SPR	120	98
	Jul-98			100
	Aug-98			100
	Mai-98		210	37
	Dez-98			123
	Jan-98		212	46
	Mar-98		213	87
	Dez-98		214	48
	Nov-98			54
	Apr-98		215	112
	Mai-98	SRA	131	20
	Aug-98		205	188
	Aug-98		213	181
	Aug-98		216	141
	Aug-98		218	99
	Aug-98		411	132
	Aug-98		412	165
	Nov-98		415	0
	Mai-98		416	19
	Aug-98			40
	Aug-98			191
	Aug-98		417	20
	Mai-98			40
	Aug-98			181
	Aug-98		421	121
	Aug-98		422	40
	Nov-98		630	40
	Jan-98	SRM	42	166
	Mar-98		110	187
	Dez-98		411	119
	Apr-98			128
	Mar-98			129
	Apr-98		412	160
	Jul-98			161
	Jan-99			169
	Jun-98			193
	Apr-98		416	241
	Feb-98			250
	Jan-98			260
	Feb-98		417	167

	Jan-99	SRM	606	60
	Jan-99	SRR	201	10
	Jul-98			40
	Aug-98			150
	Jan-99		202	10
	Jan-99		203	10
	Jul-98			30
	Aug-98			30
	Jan-99		206	10
	Mai-98		216	20
	Jan-99		403	20
	Jan-99		406	10
	Jan-99		410	0
	Apr-98			10
	Jul-98			10
	Jun-98			10
	Aug-98			20
	Jan-98			40
	Apr-98		414	30
	Aug-98			30
	Mar-98			30
	Dez-98			30
	Dez-98	SSN	103	20
	Aug-98			39
	Jul-98			40
	Mai-98			60
	Jun-98			60
	Jan-99			60
	Feb-98			61
	Aug-98		104	0
	Jan-98			40
	Jan-99			61
	Jun-98			80
	Feb-98		110	20
	Jan-98			40
	Jan-98		405	60
	Set-98			60
	Mai-98			120
	Mai-98		406	60
	Set-98			60
	Dez-98			100
	Set-98		615	167
	Jan-98	SSV	335	87
	Aug-98	STJ	214	20
	Jul-98		215	40
	Nov-98	STN	101	0
	Feb-98		102	54
	Set-98		104	20

	Aug-98		106	13
	Set-98			21
	Mai-98			34
	Nov-98		215	85
	Nov-98		620	100
	Nov-98		621	100
	Nov-98		622	100
	Nov-98		625	220
	Nov-98		629	100
	Jul-98	STR	216	60
	Jul-98	SVC	214	62
	Aug-98			67
	Apr-98			88
	Set-98			208
	Mai-98		215	86
	Apr-98			88
	Jan-98			272
	Aug-98		220	9
	Jun-98		610	46
	Jul-98	SVG	202	20
	Aug-98			60
	Jul-98		203	20
	Aug-98			220
	Mai-98		204	20
	Aug-98			40
	Jul-98		209	20
	Mai-98		211	41
	Aug-98			60
	Jul-98		212	20
	Mai-98			40
	Mai-98		213	100
	Mai-98		214	20
	Jun-98			20
	Dez-98		624	20
	Mai-98	SVI	106	21
	Mai-98		108	0
	Set-98			20
	Aug-98		109	0
	Mar-98			0
	Mai-98			0
	Mai-98	SVM	211	92
	Aug-98			258
	Nov-98	SZR	102	140
	Nov-98		103	0
	Aug-98			91
	Jul-98			101
	Nov-98		105	40
	Nov-98		107	40

	Nov-98		110	52
	Nov-98		111	52
	Nov-98		112	11
	Jul-98			63
	Aug-98			68
	Set-98			72
	Jan-98			94
	Nov-98		113	11
	Nov-98		114	52
	Jun-98			62
	Aug-98			73
	Jan-99			75
	Feb-98			91
	Jul-98			168
	Jan-98		115	42
	Nov-98			52
	Apr-98			103
	Nov-98		116	52
	Nov-98		620	52
	Nov-98		621	52

Máximo	299
Mínimo	0
Média	70.79
Moda	20
Desvio Padrão	65.12

[0-30]	37.40%	138
]30-60]	21.95%	81
]60-90]	10.57%	39
]90-120]	8.94%	33
]120-150]	6.50%	24
]150-180]	6.23%	23
]180-210]	3.52%	13
]210-240]	2.44%	9
]240-270]	1.90%	7
]270-300]	0.54%	2

NÃO SUCEDIDAS				
	Feb-98	SAM	205	481
	Mai-98	SBL	613	2580
	Feb-98	SET	106	594
	Jan-98			694
	Apr-98		609	8366
	Nov-98	SFA	106	821
	Nov-98			8281
	Nov-98			8981
	Nov-98			9021

	Nov-98			9061
	Nov-98			14312
	Mar-98		625	1500
	Mar-98			6140
	Mar-98			7780
	Aug-98			8190
	Dez-98		626	360
	Nov-98	SFF	103	830
	Nov-98			930
	Nov-98		104	930
	Nov-98		105	830
	Nov-98			930
	Nov-98		106	930
	Nov-98		107	12414
	Nov-98			13057
	Nov-98			13394
	Nov-98		108	12414
	Nov-98			13057
	Nov-98			13394
	Nov-98		110	12414
	Nov-98			13057
	Nov-98			13394
	Nov-98		111	409
	Nov-98			12948
	Nov-98			13589
	Nov-98			13928
	Nov-98		615	12414
	Nov-98			13057
	Nov-98			13394
	Nov-98		618	12414
	Nov-98			13057
	Nov-98			13394
	Nov-98		619	4828
	Nov-98			5471
	Nov-98			5808
	Nov-98			9248
	Nov-98		620	891
	Nov-98			1532
	Nov-98			1871
	Nov-98			4252
	Nov-98			5231
	Nov-98		622	376
	Nov-98			1355
	Nov-98		623	376
	Nov-98			1355
	Nov-98		624	376
	Nov-98			1355
	Nov-98	SFN	402	10960

	Dez-98		410	9080
	Dez-98			19800
	Dez-98		414	7600
	Mai-98	SMG	204	462
	Mai-98	SMR	T	2400
	Mar-98	SPA	116	325
	Mar-98		119	318
	Jun-98			697
	Jul-98			1286
	Mar-98		122	324
	Set-98	SPM	126	14372
	Mai-98		407	400
	Dez-98		416	463
	Dez-98			487
	Nov-98			520
	Nov-98			2473
	Nov-98	SPN	302	526
	Aug-98			1332
	Aug-98			1382
	Aug-98		304	331
	Feb-98		306	319
	Set-98			1840
	Mai-98			6760
	Nov-98			13679
	Aug-98			13901
	Aug-98			14000
	Dez-98			14180
	Nov-98			14500
	Apr-98			14699
	Set-98		307	2000
	Nov-98			14100
	Feb-98			14260
	Set-98			1840
	Nov-98	SRM	214	1396
	Jun-98		220	1710
	Jan-99	SRR	403	11710
	Aug-98		410	1970
	Set-98	SSB	632	1100
	Jan-98	SSV	335	756
	Dez-98	STR	211	14820
	Aug-98	SVG	613	2900
	Mai-98	SVI	108	10600
	Dez-98		616	14200
	Aug-98	SVM	211	542
	Nov-98	SVM	655	2019
	Apr-98	SZR	115	393

]300-330]	3.88%	4
]330-360]	1.94%	2
]360-390]	2.91%	3
]390-420]	2.91%	3
]420-450]	0.00%	0
]450-480]	1.94%	2
]480-510]	1.94%	2
]510-540]	1.94%	2
]540-570]	0.97%	1
]570-600]	0.97%	1
Mais que 600	80.58%	83

.....				
Regra d4	LOG	Instalação	Painel	Condição 1
				500
SUCEDIDAS				
	Mai-98	SAM	618	300
	Mai-98	SBL	613	280
	Aug-98		616	360
	Jul-98	SCL	205	330
	Mai-98	SCN	614	279
	Mai-98		626	280
	Set-98	SED	635	200
	Apr-98	SEJ	603	290
	Set-98		605	280
	Set-98		612	320
	Dez-98		613	300
	Mai-98		617	440
	Jun-98	SET	601	400
	Mai-98		609	300
	Set-98	SFA	618	300
	Mai-98			320
	Dez-98			320
	Feb-98			360
	Dez-98		624	280
	Apr-98			299
	Jul-98			300
	Nov-98			300
	Set-98			300
	Jan-98			320
	Mai-98			320
	Mai-98		625	200
	Feb-98			220
	Set-98			220
	Mar-98			240
	Apr-98			261
	Mar-98		626	321

	Nov-98			360
	Apr-98			380
	Mai-98			380
	Dez-98			399
	Jul-98			400
	Jun-98			400
	Dez-98	SFF	623	80
	Nov-98		624	400
	Dez-98	SFN	101	260
	Dez-98		412	340
	Dez-98	SFN	414	320
	Set-98	SOQ	101	40
	Jan-99	SOR	616	320
	Mar-98	SPA	114	300
	Jan-98		115	280
	Feb-98			300
	Mar-98		116	320
	Mar-98		118	280
	Mar-98		119	300
	Mar-98		122	280
	Dez-98	SPM	110	300
	Set-98		126	279
	Dez-98		416	400
	Jun-98	SPN	302	420
	Apr-98			440
	Feb-98			500
	Aug-98			500
	Mai-98			500
	Set-98			500
	Apr-98		304	120
	Jun-98			140
	Jan-99			260
	Jan-98		306	400
	Feb-98			400
	Mar-98			400
	Dez-98			400
	Apr-98			401
	Set-98			419
	Jan-99			419
	Nov-98			420
	Mai-98			421
	Jun-98		307	460
	Jan-98			479
	Apr-98			480
	Jan-99			480
	Jul-98			481
	Nov-98			481
	Aug-98			500

	Mai-98			500
	Set-98			500
	Jun-98		607	240
	Jan-98			300
	Aug-98			300
	Mai-98			300
	Mai-98		608	240
	Jun-98			241
	Mai-98	SRA	620	260
	Aug-98	SRR	201	350
	Aug-98		203	330
	Jan-98	SSB	620	460
	Dez-98	STN	617	380
	Aug-98	SVG	203	280
	Jan-98		211	300
	Jul-98		214	260
	Mai-98	SVI	614	360
	Mai-98		615	320
	Feb-98	SVM	214	120
	Mar-98		640	279
	Apr-98	SZR	115	320

Máximo	500
Mínimo	40
Média	334.19
Moda	300
Desvio Padrão	95.75

[0-50]	1.00%	1
]50-100]	1.00%	1
]100-150]	3.00%	3
]150-200]	2.00%	2
]200-250[	6.00%	6
]250-300]	33.00%	33
]300-350]	15.00%	15
]350-400]	17.00%	17
]400-450]	8.00%	8
]450-500]	14.00%	14

NÃO SUCEDIDAS				
	Jan-98	SEJ	605	670
	Jun-98	SET	601	700
	Jun-98			9600
	Jun-98			14499
	Aug-98	SPN	302	519
	Jan-98			520
	Aug-98			520
	Aug-98			520
	Feb-98			520

	Mai-98			520
	Mai-98			520
	Mai-98			520
	Jan-99			520
	Jan-98			521
	Aug-98			521
	Mai-98			521
	Jan-98			540
	Jul-98			540
	Mai-98			540
	Mai-98			541
	Aug-98			560
	Mai-98			560
	Mai-98			560
	Dez-98			561
	Jan-99			580
	Aug-98	SSV	102	560

Mínimo 519

]500-550]	61.54%	16
]550-600]	23.08%	6
]600-650]	0.00%	0
]650-700]	7.69%	2
]700-750]	0.00%	0
]750-800]	0.00%	0
]800-850]	0.00%	0
]850-900]	0.00%	0
]900-950]	0.00%	0
]950-1000]	0.00%	0
Mais que 1000	7.69%	2

Regra d5	LOG	instalação	Painel	Condição 1
				1000
SUCEDIDAS				
	Apr-98	CBR	614	900
	Aug-98	SAL	411	920
	Aug-98		422	940
	Aug-98	SAM	212	900
	Aug-98	SBL	236	901
	Dez-98		238	860
	Jan-98		242	760
	Set-98	SCF	210	910
	Aug-98			950
	Aug-98		212	930
	Aug-98	SCG	206	899
	Aug-98		218	840

	Dez-98			840
	Aug-98	SCH	210	900
	Aug-98		216	940
	Mai-98	SCL	203	950
	Jun-98		205	980
	Mai-98		206	370
	Mai-98		210	180
	Jan-98	SER	106	980
	Mai-98	SFA	102	550
	Jan-98			880
	Apr-98			900
	Dez-98			900
	Aug-98		104	859
	Mai-98			880
	Jan-98			900
	Jun-98			920
	Jan-99			920
	Jan-98		110	1000
	Mai-98		617	260
	Jul-98		625	260
	Mai-98	SFF	106	960
	Aug-98	SFN	212	940
	Dez-98		412	880
	Apr-98			900
	Aug-98			900
	Mai-98		414	860
	Feb-98			861
	Aug-98			880
	Jun-98			880
	Dez-98			880
	Aug-98	SOQ	101	960
	Mai-98			960
	Set-98			961
	Apr-98		106	920
	Mai-98			920
	Jan-98			940
	Dez-98			940
	Feb-98		111	820
	Mai-98			820
	Feb-98	SPA	114	880
	Jan-99			880
	Mar-98			920
	Jan-98			940
	Jun-98			940
	Apr-98			1000
	Mai-98		115	920
	Mar-98			960
	Feb-98			980

	Apr-98			980
	Jul-98			980
	Jan-99			980
	Aug-98			1000
	Feb-98		116	940
	Set-98			960
	Aug-98			980
	Jan-98		118	340
	Jun-98		119	900
	Aug-98			920
	Feb-98			940
	Jul-98			940
	Mar-98		122	940
	Mar-98	SPC	T	160
	Jan-98	SPG	422	980
	Feb-98			980
	Aug-98			1000
	Jan-98	SPM	108	920
	Jan-99			940
	Apr-98			960
	Jun-98			960
	Feb-98			1000
	Feb-98		110	861
	Mar-98			861
	Mai-98			880
	Jan-99			880
	Apr-98			920
	Jul-98			920
	Aug-98			920
	Mai-98		115	960
	Jul-98		119	900
	Aug-98		122	960
	Jul-98		126	800
	Jun-98			800
	Dez-98			800
	Aug-98			840
	Feb-98			860
	Mai-98			860
	Jan-99			860
	Feb-98		403	840
	Mai-98			960
	Jun-98			960
	Aug-98			980
	Set-98		407	760
	Dez-98			820
	Feb-98		414	980
	Set-98		416	860
	Nov-98			940

	Jul-98	SPN	205	940
	Mai-98			940
	Jun-98		210	960
	Jan-98			980
	Aug-98			980
	Mai-98		211	940
	Aug-98			960
	Jul-98		302	80
	Mai-98			580
	Mai-98		306	40
	Aug-98		307	40
	Nov-98			60
	Jan-99	SPR	120	820
	Aug-98			940
	Jul-98			960
	Mai-98		210	960
	Aug-98	SRA	416	960
	Aug-98		417	940
	Aug-98		422	940
	Aug-98	SRM	201	970
	Aug-98		203	980
	Aug-98		205	881
	Aug-98		213	820
	Aug-98		216	860
	Aug-98		218	880
	Aug-98		410	930
	Dez-98		411	981
	Jan-98		412	801
	Apr-98			810
	Jan-99			811
	Jul-98			819
	Jun-98			820
	Aug-98			860
	Aug-98		414	890
	Jan-98		416	781
	Feb-98			790
	Apr-98			800
	Aug-98			871
	Aug-98		417	869
	Feb-98			890
	Jul-98	SRR	201	920
	Jan-98		410	870
	Jun-98			870
	Jan-99			870
	Apr-98			880
	Jul-98			880
	Apr-98		414	890
	Mar-98			890

	Dez-98			890
	Dez-98	SSN	103	940
	Aug-98			960
	Mai-98			960
	Jun-98			980
	Jul-98			981
	Feb-98			1000
	Jan-99			1000
	Aug-98		104	980
	Jan-98			1000
	Jun-98			1000
	Feb-98		110	920
	Jan-98			980
	Mai-98		405	860
	Jan-98			900
	Set-98			960
	Dez-98		406	940
	Set-98		615	80
	Jan-98	SSV	335	60
	Aug-98	STJ	214	960
	Jul-98		215	960
	Feb-98	STN	102	920
	Set-98		104	840
	Set-98		106	980
	Mai-98			1000
	Set-98	SVC	214	20
	Apr-98		215	20
	Aug-98		220	920
	Aug-98	SVG	202	920
	Jul-98		203	840
	Aug-98			840
	Mai-98		204	900
	Aug-98			920
	Jul-98		209	860
	Mai-98		211	919
	Aug-98			980
	Jul-98		212	880
	Mai-98			900
	Mai-98		213	939
	Mai-98		214	919
	Jun-98			920
	Mai-98	SVI	106	959
	Mai-98		108	1000
	Mar-98		109	960
	Mai-98			980
	Aug-98			1000
	Aug-98	SVM	211	980
	Jul-98	SZR	103	900

	Aug-98			1080
	Aug-98			1080
	Aug-98			1080
	Aug-98	SCG	216	1119
	Aug-98	SET	612	1080
	Mai-98	SFN	213	1020
	Mai-98			1020
	Mai-98			1020
	Aug-98	SPA	117	1109
	Jan-98			1160
	Jan-98			1160
	Jan-98			1160
	Jun-98	SPN	606	1060
	Jun-98			1060
	Jun-98			1060
	Mai-98	SPR	644	1080
	Set-98	SSN	610	1020
	Aug-98	STJ	613	1061
	Aug-98	STN	106	1040
	Jun-98		614	1019
	Set-98		619	1060
	Jun-98	SVG	202	1020
	Aug-98	SVM	633	1080
	Jul-98	SZR	101	1020

Mínimo 1019

]1000-1100]	88.64%	39
]1100-1200]	11.36%	5
]1200-1300]	0.00%	0
]1300-1400]	0.00%	0
]1400-1500]	0.00%	0
]1500-1600]	0.00%	0
]1600-1700]	0.00%	0
]1700-1800]	0.00%	0
]1800-1900]	0.00%	0
]1900-2000]	0.00%	0
Mais que 2000	0.00%	0

.....				
Regra d6	LOG	Instalação	Painel	Condição 1
				5000
SUCEDIDAS				
	Mai-98	SAM	618	2
	Mai-98	SET	609	1870
	Dez-98	SFA	618	40
	Feb-98			1600
	Feb-98		625	480

	Mar-98		626	240
	Mai-98			520
	Set-98	SPM	126	2012
	Mai-98	SPN	302	1160
	Mai-98		306	859
	Jan-98		307	3940
	Mar-98		607	1270
	Mar-98	SRR	201	4040
	Dez-98	STN	617	778
	Mai-98	SVI	614	2117

Máximo	4040
Mínimo	2
Média	1395.20
Moda	Não existe
Desvio Padrão	1256.27

[0-500]	26.67%	4
]500-1000]	20.00%	3
]1000-1500]	13.33%	2
]1500-2000]	13.33%	2
]2000-2500[	13.33%	2
]2500-3000]	0.00%	0
]3000-3500]	0.00%	0
]3500-4000]	6.67%	1
]4000-4500]	6.67%	1
]4500-5000]	0.00%	0

NÃO SUCEDIDAS				

Regra d7	LOG	Instalação	Painel	Condição 1
				5000
SUCEDIDAS				
	Aug-98	SAL	411	3340
	Aug-98	SAM	212	986
	Aug-98	SCF	210	160
	Aug-98		212	1040
	Aug-98	SCH	210	1020
	Mai-98	SFA	617	560
	Apr-98	SFN	412	1120
	Dez-98			1780
	Aug-98	SPN	210	2601
	Mai-98	SPR	210	2664
	Aug-98	SRA	416	3280
	Apr-98	SRM	412	199
	Aug-98			1655
	Aug-98		417	600

	Aug-98	SRR	201	220
	Jul-98			500
	Aug-98		203	480
	Apr-98		410	280
	Aug-98			1300
	Aug-98		414	50
	Aug-98	SVC	220	1032
	Aug-98	SVG	202	440
	Aug-98		203	400
	Jul-98			600
	Aug-98		204	40
	Aug-98		211	1840
	Mar-98	SVI	109	2800
	Aug-98	SVM	211	398
	Aug-98	SZR	112	152
	Jul-98		114	1152

Máximo	3340
Mínimo	40
Média	1089.63
Moda	600
Desvio Padrão	982.01

[0-500]	40.00%	12
]500-1000]	13.33%	4
]1000-1500]	20.00%	6
]1500-2000]	10.00%	3
]2000-2500[	0.00%	0
]2500-3000]	10.00%	3
]3000-3500]	6.67%	2
]3500-4000]	0.00%	0
]4000-4500]	0.00%	0
]4500-5000]	0.00%	0

NÃO SUCEDIDAS				

Regra d21	LOG	Instalação	Painel	Condição 1
				3000
SUCEDIDAS				
	Aug-98	SAL	411	92
	Aug-98	SBL	236	301
	Jan-98		242	134
	Aug-98	SCH	216	964
	Mai-98	SED	634	2
	Aug-98			448
	Aug-98	SFN	212	82
	Aug-98		412	972

	Apr-98			2037
	Jun-98		414	182
	Feb-98			269
	Aug-98			294
	Mai-98			385
	Set-98	SOQ	101	118
	Mai-98			515
	Dez-98	SPM	126	24
	Aug-98			40
	Jan-99			40
	Mai-98			46
	Jul-98			117
	Jun-98			126
	Feb-98			331
	Aug-98		403	377
	Set-98		416	73
	Aug-98	SPN	204	370
	Aug-98		205	118
	Mai-98			278
	Jul-98			499
	Jul-98	SPR	120	72
	Aug-98			403
	Aug-98	SRA	417	133
	Aug-98		422	1019
	Aug-98	SRM	205	21
	Aug-98		216	100
	Aug-98		218	28
	Jan-99		412	226
	Jun-98			280
	Apr-98			397
	Aug-98			2950
	Feb-98		417	207
	Aug-98			1369
	Jan-99	SRR	201	92
	Jul-98			158
	Mai-98			181
	Aug-98			775
	Jan-99		203	112
	Mai-98			171
	Aug-98		402	1937
	Apr-98		410	73
	Jan-98			76
	Jul-98			209
	Aug-98			356
	Jun-98			361
	Aug-98		414	42
	Dez-98			44
	Feb-98	SSN	110	95

	Dez-98		406	43
	Mai-98	STN	106	38
	Set-98			168
	Aug-98	SVG	202	195
	Aug-98		203	649
	Aug-98		204	77
	Aug-98		211	2818
	Jul-98		212	160
	Aug-98	SVM	211	451
	Jul-98	SZR	103	285
	Aug-98		103	589

Máximo	2950
Mínimo	2
Média	396.93
Moda	92
Desvio Padrão	592.92

[0-300]	64.18%	43
]300-600]	20.90%	14
]600-900]	2.99%	2
]900-1200]	4.48%	3
]1200-1500]	1.49%	1
]1500-1800]	0.00%	0
]1800-2100]	2.99%	2
]2100-2400]	0.00%	0
]2400-2700]	0.00%	0
]2700-3000]	2.99%	2

NÃO SUCEDIDAS				
	Aug-98	SPN	204	8850
	Aug-98	SVG	203	3965

Mínimo	3965	
]3000-3300]	0.00%	0
]3300-3600]	0.00%	0
]3600-3900]	0.00%	0
]3900-4200]	50.00%	1
]4200-4500]	0.00%	0
]4500-4800]	0.00%	0
]4800-5100]	0.00%	0
]5100-5400]	0.00%	0
]5400-5700]	0.00%	0
]5700-6000]	0.00%	0
Mais que 6000	50.00%	1

.....				
Regra j2	LOG	Instalação	Painel	Condição 1

				5000
SUCEDIDAS				
	Aug-98	SCF	210	3460
	Jan-98	SCT	607	2880
	Jun-98	SET	601	1680
	Jun-98	SFA	618	2580
	Mai-98			2620
	Mar-98			4140
	Nov-98	SFF	624	2045
	Jan-99	SFN	207	4060
	Jan-99		208	440
	Jan-99		212	400
	Jan-99		213	420
	Dez-98		414	1240
	Jun-98	SPA	114	994
	Aug-98		118	654
	Mai-98	SPN	302	689
	Nov-98			711
	Jan-99			720
	Jul-98			840
	Dez-98			1120
	Aug-98			1150
	Apr-98			1171
	Jan-98		303	706
	Jan-99		304	583
	Mai-98			603
	Jan-99		306	780
	Dez-98			1220
	Set-98			1400
	Jan-98			2020
	Aug-98			2740
	Jan-98		307	3940
	Jan-99			4319
	Aug-98	SRR	410	1800
	Aug-98	SSV	102	431
	Jan-98		335	2710
	Jan-99	STR	216	1220
	Aug-98	SVC	609	3013
	Aug-98	SVG	204	600
	Aug-98		212	3021
	Aug-98	SVI	106	2700
	Apr-98	SZR	115	777
	Mai-98		625	765
	Mai-98		627	985

Máximo	4319
Mínimo	400
Média	1674.93
Moda	1220

Desvio Padrão	1184.45	
[0-500]	9.52%	4
]500-1000]	33.33%	14
]1000-1500]	16.67%	7
]1500-2000]	4.76%	2
]2000-2500[	4.76%	2
]2500-3000]	14.29%	6
]3000-3500]	7.14%	3
]3500-4000]	2.38%	1
]4000-4500]	7.14%	3
]4500-5000]	0.00%	0

NÃO SUCEDIDAS				



FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

BIBLIOTECA



0000051686