FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Integration of Heterogeneous Hypotheses in Multiagent Learning

Ivo José Pinto de Macedo Timóteo

Mestrado Integrado em Engenharia Informática e Computação

Supervisors: Eugénio Oliveira (Full Professor), University of Porto
Michael Rovatsos (Senior Lecturer), University of Edinburgh

18^{th} June, 2012

Integration of Heterogeneous Hypotheses in Multiagent Learning

Ivo José Pinto de Macedo Timóteo

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Doctor Rosaldo J. F. Rossetti, University of Porto
External Examiner: Doctor Luis M. M. Nunes, ISCTE-IUL
Supervisor: Professor Eugénio da Costa Oliveira, University of Porto
Supervisor: Doctor Michael Rovatsos, University of Edinburgh

18^{th} June, 2012

Abstract

Multiagent systems are mainly characterized by the interaction between agents. This interaction raises several interesting questions concerning the representation of knowledge, the construction of communication structures and the study of social interaction mechanisms. When the agents are capable of learning, higher levels of expressiveness in the communication can be devised.

A framework for multiagent learning, MALEF (Tožička *et al*, 2008), proposes the exchange of meta-level descriptions of the learning process of each agent that contain, among other elements, the hypothesis (i.e., the reasoning mechanism of the agent). This thesis focuses on how can an agent integrate an external hypothesis (i.e., the reasoning mechanism of another agent) into its own hypothesis. We propose a formal definition of **hypothesis** and **integration of hypotheses** compatible with the formalization of MALEF. Based on those definitions, a method for the integration of heterogeneous hypotheses is derived.

The performance of the method is verified in several experiments of multiagent learning scenarios where agents using our method perform consistently better than agents keeping their initial hypothesis unaltered. The scenarios vary in terms of the difficulty of the dataset and of the main objectives. The first and second case studies focus on the study of the behaviour of the method under different conditions and parameter configurations. The final case study is a problem of financial prediction using real data from eight European countries spanning ten years.

Resumo

Os sistemas multi-agente são caraterizados principalmente pela interação entre os seus agentes. Esta interação levanta problemas interessantes no que diz respeito à representação do conhecimento, construção de estruturas de comunicação e ao estudo dos mecanismos de interação social. Caso os agentes tenham capacidades de aprendizagem, é interessante considerar novos níveis de expressividade na comunicação.

Tožička *et al*, em 2008, apresentaram uma plataforma para sistemas de aprendizagem multi-agente, MALEF, que propõe a troca de descrições do processo de aprendizagem de cada agente que contêm, entre outros, a hipótese (i.e., o mecanismo de raciocínio do agente). O trabalho desta tese foca-se na questão de como poderá um agente integrar, na sua hipótese, uma hipótese externa (i.e., o mecanismo de raciocínio de um outro agente). Nesta tese, propomos uma definição formal dos conceitos de **hipótese** e de **integração de hipóteses**, compatíveis com a formalização de base da MALEF. Com base nessas definições, um método para a integração de hipóteses foi construído.

A qualidade do método é verificada em diversas experiências em cenários de aprendizagem multi-agente onde os agentes que usaram o nosso método demonstraram ser melhores, de forma consistente, do que os agentes que mantiveram a sua hipótese inicial inalterada. Os cenários cobrem diversos níveis de dificuldade dos *datasets* e diversos objectivos. Os dois primeiros casos de estudo focam-se no comportamento do método em diferentes condições do ambiente e diferentes configurações dos seus parâmetros. O último caso de estudo é um problema de previsão financeira que usa dados reais de oito países europeus durante um intervalo de dez anos.

Acknowledgements

I would like to show my gratitude to my supervisors, Professor Oliveira and Dr Rovatsos, for their guidance and thoughtful contributions.

I also would like to thank Dr Rossetti for his role in the first contact with Dr Rovatsos.

Finally, I would like to thank my parents for supporting me in my academic pursuits.

Ivo José Pinto de Macedo Timóteo

To JuZé

"As far as the laws of mathematics refer to reality, they are not certain, as far as they are certain, they do not refer to reality."

Albert Einstein, 1921

Contents

List of Figures

LIST OF FIGURES

List of Tables

Glossary

Heterogeneous Agents Refers to agents that have different algorithms as basis for their reasoning mechanism. Other kinds of heterogeneity will be explicitly stated (e.g. heterogeneity in the perceived environment).

Hypothesis When referring to an agent's hypothesis, hypothesis is understood as the reasoning mechanism of the agent. This includes the algorithm, parameters and any other aspect of the agent's model. It is called hypothesis since the agent is formulating that model as an hypothesis that approximates, as well as possible, the real phenomenon considering the agent's capabilities and performance measure.
A formal definition is provided in Chapter 3.

Integration of hypotheses Refers to the merging of different hypotheses in order to form a new hypothesis which, in the optimum integration, contains the knowledge of all the integrated hypotheses. The formal definition used in this thesis is presented in the beginning of Chapter 3.

Off-the-shelf algorithm An algorithm that is used as provided by the implementation without alteration or additional tuning of the parameters.

Single-agent learning Refers to the field of machine learning in non-distributed systems.

Mathematical Notation

In some occasions mathematical notation is used to enhance expressibility and facilitate the presentation of formal definitions. The meaning of some of the notation is presented here even though the majority is standard.

- Important sets are named using calligraphic lettering. E.g. $\mathscr{A}, \mathscr{D}, \mathscr{P}$.
- $\wp(A)$ represents the power set of A, where A is a set.
- $\cup$ and $\cap$ refer to set union and intersection and $\vee$ and $\wedge$ refer to logical disjunction and conjunction. Also, $A \times B$ refers to the Cartesian product and $A \setminus B$ refers to the set difference, i.e., the elements in A that are not in B, where A and B are sets.
- The definition of functions is generally presented in the form $R : X \times Y \to Z$ where R is the function, X and Y are the domains of the arguments and Z is the range. This can be rewritten as $z = R(x,y), z \in Z, x \in X$ and $y \in Y$. R itself can be seen as a set of mappings from $X \times Y$ to Z, where each mapping is represented by $(x,y) \mapsto z$. This notation may be generalized for any number of arguments.
- Propositions, corollaries, lemmas and definitions are numbered independently and the numeration is absolute throughout the thesis.
- Equations are numbered with respect to the chapter in which they are presented and are referred between parenthesis. E.g. (3.4) in the text would refer to Equation 3.4 which would be the forth labelled equation in Chapter 3.

Chapter 1

Introduction

1.1 General background

Learning in highly complex domains has proven to be a challenging problem with very interesting applications in the huge networks of information available today. There is information about every airship and vessel in transit, large networks of weather stations, detailed economical and social statistics and incredible amounts of information being added to the internet by users of social networks, for example. Successful learning applications in such complex domains would not only allow intelligent computational interaction but also provide new insights into the underlying structure of the phenomena.

Centralized approaches tend to have difficulties not only with the sheer size of the data to be processed but also with the task of designing a single coherent model that could accommodate all the complexity present in the domain. This has fostered the use of decentralized systems in which the problems are tackled using distributed data mining techniques where a group of learning agents could work on different subsets and share the obtained results. Distributed data mining techniques, however, tend to consider very homogeneous learning agents, even all with the same algorithm, whose motivation for their use is the parallelization of the problem.

This homogeneity and implicit assumption of cooperation falls short of the more general definition of agent and multiagent systems and loses most of its expressive power. An agent is an entity with autonomy which will take actions according to its own reasoning mechanism, measure of utility and its perception of the external environment; and a multiagent system is a computational system where various agents act and interact to achieve, or try to achieve, their goals[1]. We believe that heterogeneity, proper autonomy and communication are essential for the expressiveness and ability to accommodate complex domains of the multiagent system metaphor as they allow the system to evolve and adapt to different conditions that may be offered by the domain.

The development of multiagent systems, however, raises interesting new questions not usually considered in the development of centralized solutions: the study of social behaviours and social mechanisms; the definition of communication protocols and languages for the description of the

[1] Classical definitions and further information on *agents* and *multiagent systems* can be found in [RN10, SLB09].

environment and actions; notions of trust and reputation among computational agents; and the understanding of what constitutes knowledge, how the agents understand knowledge and how the collective knowledge of the multiagent system can be characterized.

These questions have added relevance when the agents in the system are capable of learning due to the necessity of a deeper and shared understanding of the environment as well as the possibility of the integration of information received from the other agents, either explicitly shared or perceived through the observation of the other agents' behaviour.

1.2 Motivation

Human language and communication mechanisms are complex enough to be usable for teaching concepts but their higher expressiveness comes from their lack of formality and from the strong inference capabilities of the human brain. It is very hard to develop ontologies, languages and protocols expressive enough to enable one computational agent to transmit a completely new concept to another agent. However, computational agents have the advantage that their learning process and reasoning mechanisms can, themselves, be described in a structured language.

In fact, there is a very interesting proposal for a multiagent learning conceptual framework, MALEF [TRPU08], which suggests the exchange of meta-level descriptions of individual learning processes among heterogeneous agents. They start by presenting a formal definition of the learning problem from which they develop a description language for each step of the learning process of each individual agent. The description includes the data used in the step, the hypothesis space, the update function, the performance measure and the currently best hypothesis, where a **hypothesis is the reasoning model of the agent**.

An interesting feature of the exchange of learning process descriptions is the possibility of sharing hypotheses. In complex domains with large amounts of data, a hypothesis as an exchangeable object can be very useful: on the one hand, the hypothesis may contain the knowledge extracted from a very large dataset which could not be shared itself, either by its sheer size or due to policy constraints; on the other hand, the receiving agent could have a hypothesis whose expressive power would not be enough to correctly characterize the information in the dataset[2]. In both scenarios, the exchange of the hypothesis is clearly the most efficient means for propagating knowledge.

Humans need to communicate explicitly in order to exchange concepts and knowledge but computational agents might be capable of sharing their reasoning mechanism directly; this would be as if one human could copy part of his brain and share it with other human. The main problem, and main focus of this thesis, is how the receiving agent can integrate an external reasoning mechanism, without any guarantees concerning its heterogeneity, into its own current hypothesis.

[2] In the same sense as a single-layer perceptron is incapable of correctly characterizing non linearly separable datasets.

1.3 Contributions

We start by proposing a formal definition of *hypothesis* and of *integration of hypotheses* compatible with the formalization and *learning process description* proposed in MALEF. From those definitions, we derive a method for the integration of a set of heterogeneous hypotheses which is guaranteed to terminate if the set of hypotheses is finite. The method depends heavily on a tree structure for which a pruning algorithm, guaranteeing the integrity of the resulting hypothesis, is also presented.

The method stands on the assumption of complete knowledge of the environment which, despite being required for the definition of the optimum solution, is unrealistic in most real problems. For scenarios where this assumption is not met, an heuristic approach is proposed which, despite not guaranteeing a perfect construction of the resulting hypothesis, is expected to lead to a good approximation.

The heuristic approach is shown to perform well in various experiments where there is a clear difference in the quality of the agents using our method to integrate external hypotheses and the agents that maintain their initial hypotheses unaltered. This difference is especially relevant in the scenarios that motivated our method for the integration of hypotheses exchanged among agents, i.e., scenarios characterized by heterogeneity both on the initial hypotheses and on the perception of the environment.

1.4 Thesis structure

The thesis is structured as follows:

Chapter 2 follows this introduction and presents a deeper overview on multiagent learning, the MALEF framework and the integration of hypotheses.

Chapter 3 starts by presenting our formal definition of the problem of the integration of hypotheses based on which an abstract method is developed. After this, an analysis of the characteristics of the domains for which the optimal solution is not applicable and the different heuristic possibilities for the extension of the method are also presented.

Chapter 4 describes the main abstractions and architectural solutions of the proposed implementation of the method.

Chapter 5 presents the evaluation methodology and the results of the various case studies.

Finally, Chapter 6 presents the conclusions and future work.

Chapter 2

Related Work

Most concepts and references are presented as needed throughout the thesis. Nevertheless, this chapter focuses on topics that are considered foundational to the understanding of the objectives, motivation and position of this thesis in relation to other work in the field.

This chapter starts by presenting a brief background on multiagent learning and distributed machine learning. Afterwards, a more detailed description of MALEF is presented focusing on its relation to the work presented in this thesis. Finally, we present related work on the integration of hypotheses and relate it to the method we propose.

2.1 Multiagent learning

Multiagent learning appeared as machine learning techniques started being used in conjunction with multiagent systems in an attempt to extend single-agent learning[1] to distributed scenarios ([SV00] presents a survey of multiagent systems from a machine learning perspective and [PL05] presents a state-of-the-art review of cooperative multiagent learning). However, this extension can have many different flavours depending on the problem being tackled, the objectives of the final solution and the motivation of the approach.

Shoham *et al* propose five "distinct coherent goals" [SPG07] in multiagent learning according to their motivation and success criteria: **computational** where learning algorithms are treated as iterative procedures aiming at the computation of the properties of the environment; **descriptive** that asks how agents learn in multiagent scenarios and the focus is on the investigation of models describing the learning behaviour of natural agents (e.g. people, organizations, animals, natural phenomena, etc); **normative** which tries to devise which sets of learning rules are in equilibrium with each other; **prescriptive, cooperative** that asks how the agents should learn in multiagent cooperative scenarios; and, finally, **prescriptive, non-cooperative** which asks how the agents should learn in non-cooperative scenarios.

Most of these goals are unrelated to our work but are useful to demonstrate the great diversity of the work in the field of multiagent learning. The work of this thesis is directly related to a

[1] By single-agent learning we refer to traditional machine learning.

prescriptive approach, both cooperative and non-cooperative which are closer, in terms of objectives and methodology, to the unification of machine learning and multiagent systems mentioned previously. This area could be described as focusing mainly on the development of learning mechanisms and intelligent solutions for scenarios where multiple agents learn **from** other agents, **with** other agents and/or **about** other agents [Rov08].

2.2 Distributed machine learning

Multiagent learning with a prescriptive goal and focusing on learning with other agents can also be called distributed machine learning. Distributed machine learning shares the learning goal of single-agent learning, i.e., the construction of a hypothesis that can characterize a phenomenon or environment, as well as possible, given some measure of performance. This similarity of goals was responsible for the attempt of generalizing the algorithms already developed in the field of machine learning but now in a distributed scenario.

One common motivation for the use of distributed machine learning is performance when dealing with large datasets. The main idea is that the distribution of problem throughout several different agents may improve the learning performance and the processing time of the dataset. Several techniques have been proposed distinguishing between the algorithms used, argumentation mechanisms and assumptions made, e.g. [DC04, BGST99, SFC97, TJP06, WMJ03, Nun06].

However, these systems tend to assume strict cooperation and homogeneity of the hypotheses of the different agents as well as proposing solutions fitted for a particular learning algorithm. MALEF [TRPU08], to the best of our knowledge, was the first generic abstract framework for distributed machine learning that can accommodate independent and heterogeneous learning agents. It's under these conditions that our method was developed and evaluated in the case studies.

The main reason for the choice of developing a method for the integration of heterogeneous hypotheses is the belief that distributed machine learning could benefit from that same heterogeneity. Different machine learning algorithms perform best in different scenarios and highly complex systems may be diverse enough to have different sub problems that could be tackled best by different algorithms. If a population of heterogeneous agents were to tackle such a complex system, it could eventually evolve into a population of agents expert in different subsets of the environment whose integration could, perhaps, display better learning performance than a equivalent homogeneous population.

2.3 MALEF framework

The MALEF framework [TRPU08] is a conceptual framework that suggests the exchange of meta-level descriptions of the individual learning process among the agents in order to enable new interaction mechanisms. They start by suggesting a formal definition of the learning problem:

> "Given data $D \subseteq \mathscr{D}$ taken from an instance space $\mathscr{D}$, a hypothesis space $\mathscr{H}$ and an (unknown) target function $c \in \mathscr{H}$[1], derive a function $h \in \mathscr{H}$ that approximates c as

well as possible according to some performance measure $g : \mathscr{H} \rightarrow \mathscr{Q}$ where $\mathscr{Q}$ is a set of possible levels of learning performance."

Based on this definition, they develop a notion of learning step of each individual agent of the form "$l = \langle D, H, f, g, h \rangle$" where D is the data used in the step, H is the hypothesis space, f is the update function, g is the performance measure and h the currently best hypothesis. A learning process is then defined as a finite, non-empty sequence of learning steps. Finally, they present a representation for the learning processes which they call *Learning Process Descriptions* which are used to pass information concerning the learning processes among the agents. The authors further describe the concept of *learning agent* in their framework and make general suggestions on how the information on the *Learning Process Descriptions* should be integrated by the receiving agent. This short presentation of the framework is included to provide a general idea of the MALEF to the reader. For further reading on the subject refer to [TRPU08].

We find the idea behind MALEF worth developing and, as stated before, it motivated the work on the integration of heterogeneous hypotheses. The abstraction is simple, in the sense that there are no obvious redundant objects, and flexible enough to accommodate most scenarios associated with the prescriptive goals of multiagent learning without prescribing a particular implementation. The learning problem definition itself could be used as basis for the majority of single-agent learning goals.

Furthermore, it also gives us enough liberty to consider the definition of **hypothesis** and **integration of hypotheses** presented in the proposed solution chapter from which our method is derived. However, despite being an important element on which the development of our method is based, the case studies presented in this thesis do not exchange complete *Learning Process Descriptions* between the agents considering, instead, only the exchange of the current best hypothesis. This is to guarantee that the improvements observed in the agents using our method are due to the good integration of the external hypotheses rather than by exchange of information through any other means (e.g., raw data).

2.4 Integration of hypotheses

The integration of hypotheses, viewed as the construction of a new classifier from a set of classifiers, is a common problem to which various techniques have been suggested with diverse motivations and applicabilities. Some of the most common techniques are inherited from model averaging algorithms of classic machine learning such as ensemble methods (e.g. voting, bagging, boosting among others) [BK99, Cha01, Die00] and Bayesian approaches [DC04, Bis06]. Other techniques are based on meta-learning [BGST99, SFC97], argumentation and market-based mechanisms [TJP06, WMJ03], and more. However, these techniques are usually applied in scenarios of strict cooperation where agent autonomy is disregarded or depend on conditions of homogeneity of the hypotheses or narrow heterogeneity (i.e., algorithms using similar structures or approaches; e.g. the use of clustering algorithms only).

Concerning integration of hypothesis in the context of MALEF, in [TRPU08], a case study accompanies the presentation of the framework that uses the exchange of learning process descriptions in a ship surveillance scenario and, for the integration of the external hypotheses, the case study uses three operations specific to the scenario. Another work using MALEF is [RPR09, RPRMPLA09][2] where the integration of hypotheses, considering only tree-based algorithms, is done by copying selected branches of the external hypothesis and then adding them to the original hypothesis. At a higher level, voting techniques were also used.

The method proposed in this thesis is constructed on a proposed formal definition of **integration of hypotheses** and **hypothesis**. The proposed definition of hypothesis does not conflict with the formal foundation of MALEF ensuring that the method can be used with the framework.

The main novelty of our method is that it is constructed from a formal definition of the hypothesis which encloses a complete independence from the actual reasoning mechanism. This ensures, contrary to most of previous proposals, the tractability of any level of heterogeneity when integrating hypotheses, which means that even the hypothesis resulting from the integration of a set of hypotheses may be integrated again with any type of hypothesis.

[2]Both publications are strongly related.

Chapter 3

Proposed Solution

In this chapter we present the proposed solution for the integration of heterogeneous hypotheses problem. We start by establishing the important definitions of **hypothesis** and **integration of hypotheses** and, based on these formal definitions, the problem is stated and the abstract solution is constructed. After this, we present the resulting method in the form of an algorithm as well as some considerations on its applicability in diverse scenarios. Finally, we propose an extension to the method using heuristic approaches where we try to find a balance between theory and engineering in order to expand the applicability of the method to more realistic scenarios.

3.1 Problem definition

The fundamental question that led to this thesis is the problem of integrating heterogeneous hypotheses initially crafted by different agents. This question may be posed loosely as follows: *how can an agent integrate the knowledge contained in another agents' hypotheses into its own hypothesis?* However, many semantical issues may arise given the open nature of this question and this motivates a more rigorous definition of the problem.

Therefore, in order to establish the meaning of **integration of hypotheses**, a formal definition of the concept is needed in conjunction with some necessary assumptions. In every future reference and discussion of the concept it should be understood strictly based on the formal definition presented on this chapter, even though an intuitive understanding should be enough to follow and understand the work presented in the following chapters.

The development of the solution for the integration of heterogeneous hypotheses is based on the sole assumption that every agent can understand, construct and use descriptors of both the data perceived by the agent, serving as input to the hypothesis, and of the output of the hypothesis. We call the former **perception** and the latter **prediction**[1]. We also assume that every agent can use a hypothesis by feeding perceptions and retrieving predictions without the need of extra information concerning the internal nature of the hypothesis.

[1] It should be noted that the terms chosen do not restrict the application of the method by their strict semantic interpretation – e.g. the hypothesis might output a plan and not a prediction and, nevertheless, it will be called a prediction in this thesis for lack of a better, more general, term.

Consider a hypothesis space $\mathscr{H}$ such that $\forall_{h\in\mathscr{H}} \quad h : \mathscr{D} \to \mathscr{P}$, where h is a hypothesis, $\mathscr{D}$ is the perception space and $\mathscr{P}$ is the prediction space. Let g be the perfect hypothesis, i.e. the hypothesis that correctly maps all the perceptions to predictions. For now, we will assume that there are no hidden variables and that the learning problem is well posed which implies, therefore, that each perception has one and only one corresponding prediction.

It should be noted that this definition of hypothesis does not depend on the internal representation ensuring that all theoretical work developed on top of this definition also does not depend on the internal representation of the hypothesis. In particular, the method for the construction of the integration of hypotheses presented in this chapter does not depend on the internal representation of the hypotheses and, therefore, can be used for the integration of heterogeneous hypotheses.

Definition 1. *Let $\mathscr{C} : \mathscr{H} \to \wp(\mathscr{D})$, be defined as the mapping of a hypothesis to the set of perceptions it correctly classifies. That is,*

$$\mathscr{C}(h) = \{d \in \mathscr{D} : h(d) = g(d)\}, \;\; h \in \mathscr{H}$$

Definition 2. *Let the equality between two hypotheses be defined as follows:*

$$\forall_{h_1,h_2\in\mathscr{H}} \quad h_1 = h_2 \Leftrightarrow \mathscr{C}(h_1) = \mathscr{C}(h_2)$$

Definition 3. *Let $\xi : \wp(\mathscr{H}) \to \mathscr{H}$, be called* **integration of hypotheses** *and defined such that*

$$\forall_{S\subseteq\mathscr{H}}, \quad \mathscr{C}(\xi(S)) = \bigcup_{h\in S} \mathscr{C}(h)$$

I.e., the hypothesis resulting from the application of the integration of hypotheses to a set of hypotheses should correctly classify exactly the same perceptions as the union of the sets of correctly classified perceptions by the hypotheses in that set.

In a less formal language, the hypothesis resulting from the application of the integration of hypotheses should contain all the knowledge contained in the hypotheses of the set being integrated. As an example, consider two agents, Alice and Bob, whose hypotheses can be described as follows: $h_{Alice} = \{yellow \mapsto banana, green \mapsto apple\}$ and $h_{Bob} = \{green \mapsto apple, red \mapsto tomato\}$. The integration of the hypotheses of Alice and Bob would result in $h_{Alice\&Bob} = \{yellow \mapsto banana, green \mapsto apple, red \mapsto tomato\}$.

The problem of the integration of heterogeneous hypotheses can now be stated as the construction of $\xi(S)$, for an arbitrary set $S \subseteq \mathscr{H}$. I.e., given a set of hypotheses, construct a hypothesis that is equal to the integration of the hypotheses in that set.

3.2 Abstract solution

Proposition 1. *The integration of hypotheses $\xi(S)$ can be constructed for any finite set $S \in \mathscr{H}$ as a finite composition of piecewise functions whose form, for an arbitrary set of cardinality $n > 1$, is:*

$$\forall_{d \in \mathscr{D}} \;\; \xi(S_n)(d) = \begin{cases} h_n(d) & \text{if } \; d \in \mathscr{C}(h_n) \\ \xi(S_{n-1})(d) & \text{if } \; d \notin \mathscr{C}(h_n) \end{cases}$$

where $S_i = \{h_1, h_2, ..., h_{i-1}, h_i\}, \;\; S_i \subseteq \mathscr{H}$.
And whose base case, for a set of cardinality one, is: $\xi(S_1) = h_1$

The proof for Proposition 1 can be found in Appendix A.

It should be noted that the enumeration of the hypotheses in the set S_n does not affect the generality of the proposition and is used only to simplify the presentation of the result. Recalling Definition 3 and given that the union of sets is a commutative operator, one can see that any possible enumeration or ordering of the hypotheses in the set S_n leads to the same resulting hypothesis and, therefore, does not affect the validity of Proposition 1.

Corollary 1. *The integration of hypotheses $\xi(S)$ can be constructed, for any finite set $S \in \mathscr{H}$, using only the hypotheses in S.*

From Corollary 1 we know that the hypothesis resulting from the integration of hypotheses can be constructed using only the hypotheses in the set, and Proposition 1 states that the integration of hypotheses can be seen as a finite composition of piecewise functions, for any finite set of hypotheses, which itself can be presented as a piecewise function. The piecewise function is described by the hypotheses and the subsets where they should be applied. We already have guarantees on the hypotheses used but we still need to prove that the needed partitions of the domain can be made.

Definition 4. *Let each perception $d \in \mathscr{D}$ be characterized by a finite set of features $\mathscr{F}$. For each feature $f \in \mathscr{F}$, let $f(d)$ be the value of that feature in the perception d.*

Definition 5. *Let the equality between two perceptions be defined such that two perceptions are said to be equal if all their features have the same value:*

$$\forall_{d_1, d_2 \in \mathscr{D}} \quad (d_1 = d_2 \Leftrightarrow \forall_{f \in \mathscr{F}} \quad f(d_1) = f(d_2))$$

Definition 6. *Let $P_f : \Delta(f) \times \wp(\mathscr{D}) \to \wp(\mathscr{D})$ be called a binary feature partition over feature f such that*

$$\forall_{\substack{K \subseteq \Delta(f) \\ D \subseteq \mathscr{D}}} \quad P_f(K, D) = \{d \in D : f(d) \in K\}$$

where $\Delta(f)$ is the domain of values a feature f can assume for every $d \in \mathscr{D}$.
$P_f(K, D)$ is called a binary feature partition because it selects a subset from a set (D) based on the value of a specific feature f (belonging or not to K) and, therefore, we can say that the set was

divided in two subsets: $P_f(K,D)$ *and* $\overline{P_f(K,D)} = D - P_f(K,D)$.

Proposition 2. *Any* $D \subseteq \mathscr{D}$ *can be described by a finite composition of binary feature partitions.*

The proof for Proposition 2 is presented in Appendix A.

From Proposition 2 we know that it is possible to describe any subset of $\mathscr{D}$ as a finite composition of binary feature partitions and from Proposition 1 we know that the hypothesis resulting from the integration of a set of hypotheses can be described as a finite composition of piecewise functions. As supported by the results of Corollary 1, the hypotheses needed to construct the integration of hypotheses are known by the agent *a priori*[2] and the sets where each hypothesis should be applied can all be described by a finite composition of binary feature partitions. Since the integration of hypotheses can be described as a finite composition of piecewise functions using finite compositions of binary feature partitions for each set, we know that it is possible to create a finite description of the hypothesis resulting from the integration of hypotheses for any finite set of hypotheses. Moreover, the structure of a possible method by which this can be achieved is clear after the previous results.

3.3 Abstract method for the integration of hypotheses

A method respecting all the definitions presented above for the construction of the integration of the hypotheses in a set $S \subseteq \mathscr{H}$ can be easily designed if we take into consideration the composition of the piecewise functions proposed by Proposition 1.

Consider a set $S \subseteq \mathscr{H}$ with cardinality n. We can enumerate the hypotheses in set S and we will designate them as $S_n = \{h_1, h_2, ..., h_{n-1}, h_n\}$[3]. Following the description for the general step until the we arrive at the base case we can see that the resulting function follows an evident pattern.

$$\begin{aligned}
\xi(S_n)(d) &= \begin{cases} h_n(d) & ,d \in \mathscr{C}(h_n) \\ \xi(S_{n-1})(d) & ,d \notin \mathscr{C}(h_n) \end{cases} \\
\xi(S_{n-1})(d) &= \begin{cases} h_{n-1}(d) & ,d \in \mathscr{C}(h_{n-1}) \\ \xi(S_{n-2})(d) & ,d \notin \mathscr{C}(h_{n-1}) \end{cases} \\
&\vdots \\
\xi(S_1) &= h_1
\end{aligned} \tag{3.1}$$

[2]The agent could not set the integration of a set of hypotheses as a goal if it did not know the hypotheses in that set.
[3]As stated before, the enumeration of the hypotheses is irrelevant to the validity of Proposition 1.

Working on the composition we have

$$\begin{aligned}
\xi(S_n)(d) &= \begin{cases} h_n(d) & ,d \in \mathscr{C}(h_n) \\ h_{n-1}(d) & ,d \in \mathscr{C}(h_{n-1}) \wedge d \notin \mathscr{C}(h_n) \\ \xi(S_{n-2})(d) & ,d \notin \mathscr{C}(h_{n-1}) \wedge d \notin \mathscr{C}(h_n) \end{cases} \\
\xi(S_{n-2})(d) &= \begin{cases} h_{n-2}(d) & ,d \in \mathscr{C}(h_{n-2}) \\ \xi(S_{n-3})(d) & ,d \notin \mathscr{C}(h_{n-2}) \end{cases} \\
&\vdots \\
\xi(S_1) &= h_1
\end{aligned} \tag{3.2}$$

leading to

$$\xi(S_n)(d) = \begin{cases} h_n(d) & ,d \in \mathscr{C}(h_n) \\ h_{n-1}(d) & ,d \in \mathscr{C}(h_{n-1}) \wedge d \notin \mathscr{C}(h_n) \\ h_{n-2}(d) & ,d \in \mathscr{C}(h_{n-2}) \wedge d \notin \mathscr{C}(h_{n-1}) \wedge d \notin \mathscr{C}(h_n) \\ \vdots & \\ h_2(d) & ,d \in \mathscr{C}(h_2) \wedge d \notin \mathscr{C}(h_3) \wedge \ldots \wedge d \notin \mathscr{C}(h_{n-1}) \wedge d \notin \mathscr{C}(h_n) \\ h_1(d) & ,d \notin \bigcup_{h \in S_n, h \neq h_1} \mathscr{C}(h) \end{cases} \tag{3.3}$$

which covers all the domain, $\mathscr{D}$, and where the partition in each of the subsets for the application of each hypothesis can be interpreted as the successive application of binary partitions on $\mathscr{D}$ (Proposition 2). This suggests a binary tree structure for the representation of the integration of hypotheses where the leafs are hypotheses from the initial set S_n, the branches represent conditions of membership (or not) to a certain set and the navigation through the branches is equivalent to the conjunction of the conditions in these branches.

As can be seen from Fig. 3.1, the navigation from the root to any particular hypothesis results in the same conditions as those presented in the piecewise representation of the integration of hypotheses (3.3). The resulting structure is a binary decision tree which means that, as we navigate the tree, each query is always on the membership of a simple set instead of the membership of a conjunction of many sets. In fact, the conjunction of the sets is *hidden* in the structure of the tree and does not need to be considered explicitly.

The tree presented in Fig. 3.1, however, is the trivial construction of the integration of hypotheses by a direct application of the general solution (3.3) of the method derived from Proposition 1. In fact, the resulting tree is the worst case scenario in terms of using the tree structure as it is essentially the linear application of conditions. It could be suggested that the hypotheses would be ordered by the size of their domain of applicability and, therefore, we would have a greater probability of finding an appropriate hypothesis earlier in the tree but the worst case (i.e., when all hypotheses have equal domains of applicability) would still require, on average, the verification of

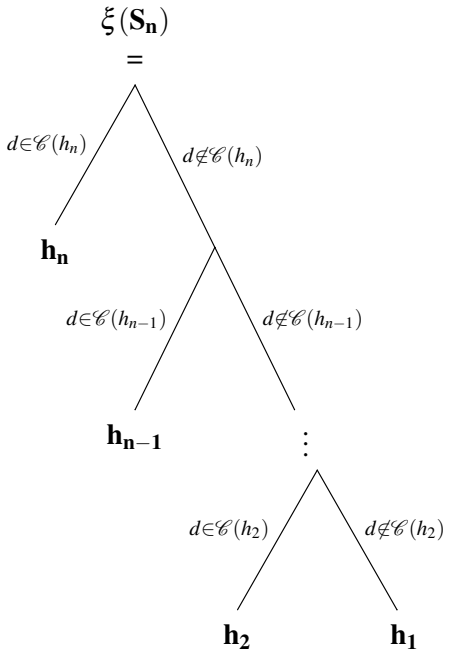


Figure 3.1: A tree representation of the integration of hypotheses

$\frac{n}{2}$ conditions, which is not very satisfying considering its tree structure.

However, we could consider branching by the membership of a particular feature value $f_i(d)$ to a subset of $\Delta(f_i)$ instead of the membership of d to some $\mathscr{C}(h_i)$. In fact, each branch in Fig. 3.1 is already a finite composition of binary feature partitions. This means that a tree where each branch of Fig. 3.1 is replaced by the corresponding sequence of branches corresponding to the different partitions of each composition of binary partitions can be built.

But we are not interested in a tree with similar structure to the tree in Fig. 3.1 as it is very inefficient. E.g., to reach h_1 we would query each feature $n-1$ times. What is interesting when using a tree structure is to have a fairly balanced tree and to make the structure small.

In order to obtain a better structure that still represents the same information but more efficiently, we should choose the *best* partition to use in every node of the tree to avoid redundancy and large unbalanced trees. Having a best partition for a specific node implies the necessity of having a measure of the quality of the partition. If this measure is appropriate we will obtain a small and efficient tree (e.g., Fig. 3.2). The idea of using this kind of measures is very widespread and almost every decision tree algorithm uses this approach [BFOS84, Qui93, Mit97, Loh08].

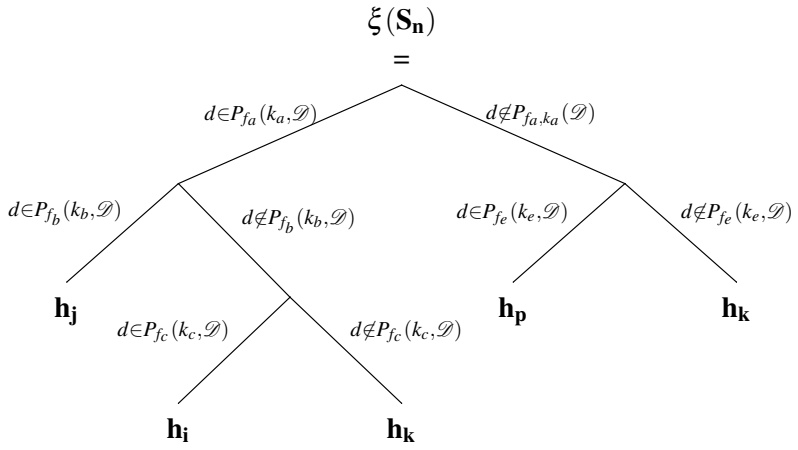


Figure 3.2: A possible tree with carefully chosen partitions.

f_i and k_i define an appropriate partition in this example.

3.4 Algorithm

From the abstract method it is possible to devise an algorithm to perform the integration of hypotheses. As stated in the problem definition, we shall assume that a hypothesis is some function that transforms perceptions into predictions and that every hypothesis uses the same representation for the said perceptions and predictions. The objective of the algorithm is the construction of a decision tree, similar to Fig.3.2, that would refer to which hypothesis should be used to classify a given perception that could, itself, be considered a hypothesis in future integrations.

3.4.1 Construction of the resulting hypothesis

Our idea is to create a decision tree such as the one described in the previous section with the information of which hypothesis can correctly classify each leaf of the tree. Therefore, given the dataset $\mathscr{A}$, a list of pairs $(Perception, Prediction)$ such that $Perception \in \mathscr{D}$ and $Prediction \in \mathscr{P}$, and two hypotheses h_1 and h_2 both in $\mathscr{H}$, the integration of hypotheses can be computed as follows:

1. Create a list $\mathscr{C}$ such that for each $(Perception, Prediction) \in \mathscr{A}$ yields $(Perception, X)$ where

$$X = \begin{cases} 0, & h_1(Perception) \neq Prediction \wedge h_2(Perception) \neq Prediction \\ 1, & h_1(Perception) = Prediction \wedge h_2(Perception) \neq Prediction \\ 2, & h_1(Perception) \neq Prediction \wedge h_2(Perception) = Prediction \\ 3, & h_1(Perception) = Prediction \wedge h_2(Perception) = Prediction \end{cases}$$

   I.e. for each *Perception*, we encode which hypothesis can correctly classify it so that later we can use X as the class attribute when building the decision tree.

2. Run a decision tree algorithm over the features of the perceptions in $\mathscr{C}$ using X as the class attribute. At this stage the decision tree algorithm is considered to run without any heuristics or pruning and only accepting leafs with confidence equal to one. I.e., providing an exact description of X in relation to the perceptions. The choice of the algorithm is irrelevant if these conditions are met.

At the end of the execution, we have a decision tree that for each *Perception* returns X allowing us to infer which hypothesis, if any, may correctly classify the *Perception*. The extension to the integration of more than two hypotheses is done by sequential application of this algorithm as the resulting hypothesis can be used for further integration in future iterations.

3.4.2 Classification of new perceptions

In order to classify a new instance, d, using the hypothesis resulting from the integration of hypotheses we use the decision tree to obtain X and then act accordingly:

$X = 1$, we return $h_1(d)$

$X = 2$, we return $h_2(d)$

$X = 0 \vee X = 3$, it depends on the agent strategy when dealing with these situations. In all the case studies presented in this thesis we opted to give priority to the hypothesis previously owned by the agent performing the integration.

3.4.3 Pruning

The trees resulting from the construction of the integration of hypotheses according to the algorithm presented can be further simplified, without any change in the correctness of the resulting hypotheses, by pruning. Common decision tree algorithms also apply pruning, generally with the objective of simplifying the tree to avoid overfitting. However, they use heuristic approaches that are willing to misclassify some instances as a trade-off for better generalization capabilities. This is not our objective as we wish to be coherent with the theoretical definition presented above. Therefore, the pruning algorithm presented here guarantees the equality between the unpruned and pruned tree.

We start by presenting the branch configurations that can be safely pruned without changing the resulting hypotheses followed by the complete algorithm.

The examples shown in Fig. 3.3 are cases where there are only two leaves. However, in the descriptions that follow, the case of multiple child nodes is discussed.

a) The simplest pruning operation is when all leaves represent the same hypothesis. We can prune those leaves and consider a new leaf representing the same hypothesis. This is true because the last condition is irrelevant as the chosen hypothesis is always the same independently of the result of the aforementioned condition.

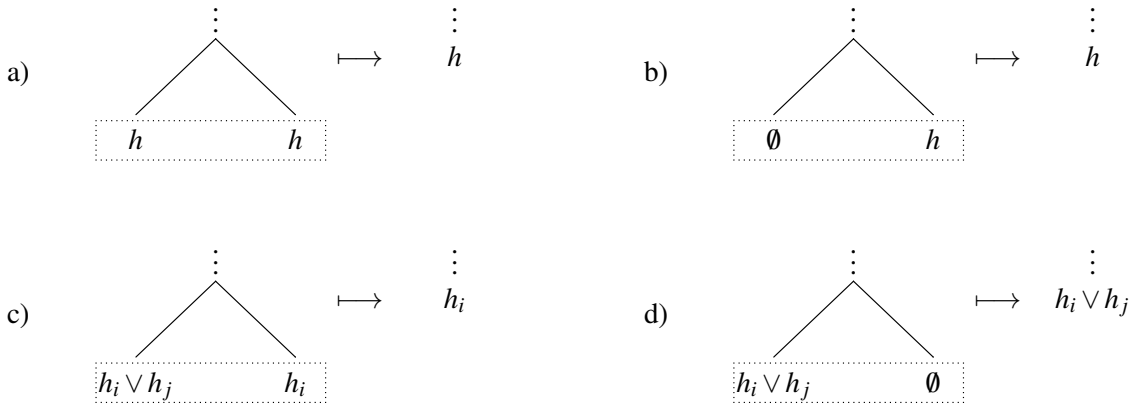


Figure 3.3: Pruning operations

b) If the leaves only represent one hypothesis h or the fact that no known hypothesis can correctly classify the perceptions leading to that leaf, then we can prune those leaves and consider a new leaf representing h since h is a valid substitute for the leaves without hypotheses[4] and a valid substitute for itself.

c) If all the leaves contain one hypothesis h_i, either isolated or in a disjunction, then all the leaves are correctly classified by h_i, from which we know that it is possible to prune those leaves an create a new leaf representing the hypothesis h_i. Additionally, some of the leaves can also mean that no known hypothesis can correctly classify the perceptions leading to that leaf that this pruning operation is still possible (equivalent to operation b) explained above). Also relevant for situations of iterative pruning is the special rule that when more than one hypothesis can be considered, i.e., when all leaves are $h_i \vee h_j$, the resulting leaf should represent $h_i \vee h_j$.

d) Similarly to operation b), when all leaves are either representative of $h_i \vee h_j$ or the fact that no known hypothesis can correctly classify the perceptions leading to that leaf, then we can prune those leaves and consider a new leaf representing $h_i \vee h_j$ instead of choosing one over the other.

From the rules above we can devise an algorithm that will continuously prune the tree until it is impossible to continue. This continuous pruning is what motivates the propagation of $h_i \vee h_j$ instead of choosing one of the hypotheses when executing the pruning operation, as this may allow further pruning afterwards leading to a more efficient representation of the resulting hypothesis. View Fig. 3.4 for an example scenario.

[4] h, such as all the other known hypotheses, cannot correctly classify perceptions leading to those leaves. Therefore, it is irrelevant which hypothesis is used in this situation.

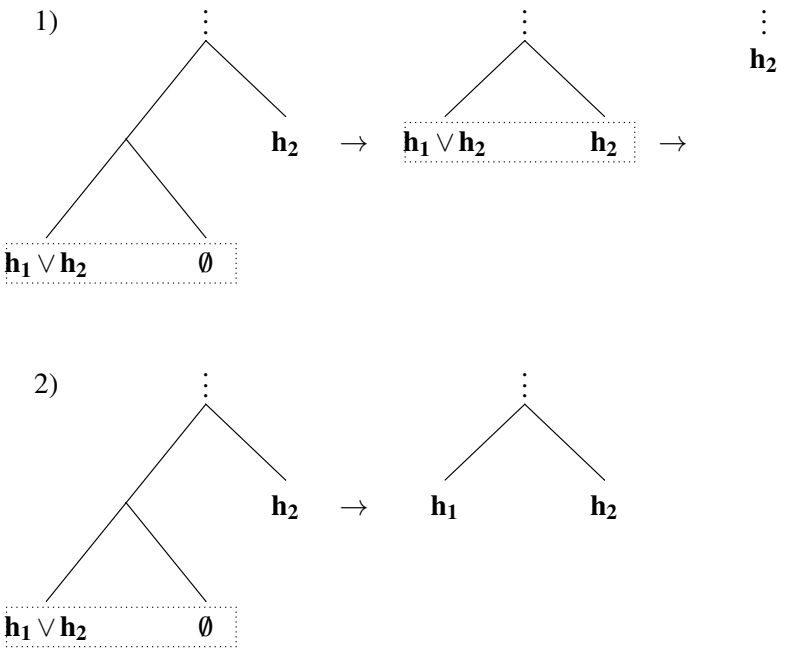


Figure 3.4: An example comparing strategies of propagation: 1) with the propagation of both hypotheses and 2) by choosing one hypothesis.

One possible implementation of the pruning algorithm is as a recursive procedure starting from the root of the tree. Let this procedure be called $pruning(node)$ and receive a tree node as parameter. The pruning procedure can be described as follows:

- if $node$ is a leaf then $pruning(node) = X$, where X is the same as the one defined is Section 3.4.1.
- if $node$ is not a leaf then $pruning(node) = \begin{cases} 1, & \text{if } \forall_{i \in L(node)} \ i \in \{0,1,3\} \\ 2, & \text{if } \forall_{i \in L(node)} \ i \in \{0,2,3\} \\ 3, & \text{if } \forall_{i \in L(node)} \ i \in \{0,3\} \\ \textbf{imp}, & otherwise \end{cases}$

  where $L(node) = \{pruning(n) : \forall n \in children(node)\}$ and $children(node)$ returns the children of the node $node$. $L(node)$, therefore, is a list of integers corresponding to the values returned by the application of the procedure $pruning(n_1)$ to all the child nodes of node $node$. When $pruning(node) = \textbf{imp}$, all the children of that node whose pruning value is not **imp** are transformed into leaves pruning the whole sub-tree.

3.5 Application of the algorithm in different scenarios

The algorithm previously described follows directly from the definition of the abstract method which, in turn, is bound by the strict adherence to the definition of integration of hypotheses presented in Section 3.1. This has led to the necessity of considering the possession of knowledge concerning the environment which will rarely be available in real scenarios (e.g., we would need to know g and all the possible perceptions in order to know which perceptions a hypothesis correctly classifies).

However, while the application of the algorithm, as it is defined, requires some unrealistic knowledge over the environment, its theoretical foundation provides a strong basis on which to evolve and adapt. In fact, a careful choice of alterations, where all the consequences are known and their interference with the basic guarantees of the algorithms is minimized, can be made in order to accommodate a greater range of scenarios. This Section focus on the application of the algorithm in more realistic scenarios, where a balance between theory and engineering must be achieved in order to tackle problems under less controlled conditions.

3.5.1 Data

The initial problem definition assumes we know $\mathscr{D}$ and g, i.e., all the possible perceptions and their respective correct predictions. It is obvious that in a scenario in which we aim at the construction of a predictor that will not be true. However, we can restate the problem to be the creation of the hypothesis resulting from the integration of hypotheses given all the known data, which would be $\mathscr{A}$ in the algorithm. This data can be provided in a training phase or even gathered as the agent interacts with its environment but we will still have the guarantee that the resulting integration of hypotheses will keep its optimality over the data known to the agent, $\mathscr{A}$, instead of $\mathscr{D}$. I believe that this is acceptable and common to all learning processes as it would be impossible to provide guarantees on the unknown.

A related problem that arises when receiving external hypotheses is the evaluation of their quality and if they should be integrated. A possible approach is to use only the known pairs of perceptions and predictions as $\mathscr{A}$ and integrate them based on this information. Another approach is to evaluate if the information gain obtained from the integration of the external hypothesis is worth of the higher complexity resulting from the integration. A possible approach would be to test the external hypothesis against pairs of perceptions and predictions that the current hypothesis is incapable of correctly classifying to see if the integration with the external hypothesis is promising or not. At first thought it may seem as if always integrating would definitely lead to better hypotheses but given that we do not possess complete information over $\mathscr{D}$ and g, increases in complexity can lead to a loss of the generality of the predictor resulting in worse results in the long term, specially in highly dynamic stochastic environments.

Finally, while the creation of test and validation sets is very important in the evaluation of predictors and a well established practice, different tactics might be used when integrating:

i) In the theoretical case study we do not perform any heuristic approaches and we have complete knowledge, therefore, we know that the hypothesis resulting from the integration will never perform worse than any of the hypotheses integrated. In this scenario we decide to always integrate.

ii) In the other case studies, since we are interested on the behaviour of our method under different conditions, we also always integrate.

iii) If we want to decide whether to integrate then we could use pre-integration tests, use validation sets, base our decisions on the reputation of the agent that provided the external hypothesis, and many other possibilities dependent on the specific scenario. As a conclusion, whether an agent integrates or not should be part of his strategy.

3.5.2 Decision tree algorithms

The proposed algorithm mentions the use of a decision tree algorithm for the construction of our tree structure. No particular algorithm was mentioned beside the fact that it should not perform heuristic pruning and that the leaves should not contain incorrectly classified instances. These restrictions make sense when we are trying to implement the abstract method as defined previously, for which the simplest version of a decision tree algorithm is enough, i.e., a measure to know which partition to use in a particular node and the continuous expansion of every node that contains instances from more than one class.

However, as stated in the previous subsection, the data available to the agent is limited which forces the agent to apply a less strict approach where using the known data to generalize is better than eventually overfitting the predictor to the known subset of data. While it could be seen as a deviation from the original abstract method, the use of decision tree algorithms which apply techniques to build a tree that attempts to generalize well is actually closer to the abstract method in scenarios where the available data is clearly limited.

There are various decision tree algorithms with different characteristics that make them better at handling specific scenarios. The choice of which algorithm to use and how to tune it depends on the characteristics of the environment such as whether there are continuous numerical values or we are interested in strong generalization, for example. The only restriction is that the chosen algorithm performs classification and not regression but, apart from this, every decision tree algorithm in the literature should perform well (e.g., [BFOS84, Qui93, Mit97, Loh08]).

3.5.3 Set partitions

When presenting the algorithm we assumed that the needed subsets, i.e. partitions, can easily be constructed by applying certain conditions over the features of the perceptions. However, most decision tree algorithms tend to oversimplify by considering that symbolic attributes have finite domains which are distinguished by simple enumeration or by considering ordering conditions for continuous numeric attributes, i.e., $x \leq K$ and $x > K$.

In terms of ordering conditions used to distinguish between continuous numerical attributes, problems could arise. In fact, there are various relevant partitions that cannot be described using a finite sequence of such conditions. As an example, consider the separation of even and odd numbers where we can see that only a infinite length sequence of ordering conditions could describe such a partition. In general, any partition where the subsets are scattered in terms of order poses problems.

Nevertheless, such partitions can usually be easily described by other conditions, e.g., in the case of the division in odd and even numbers the remainder of the integer division by two can be used to successfully describe the partition. The reason why common decision tree algorithms do not consider those families of conditions is because the options are infinite and they happen to use ordering conditions which behave well in most scenarios. However, it is easy to extend decision tree algorithms to additionally consider one new family of conditions that seems to part some feature well in a particular scenario.

This section discussed some general possibilities of expanding the algorithm for the treatment of more realistic scenarios under conditions of incomplete knowledge. The actual approach that should be taken depends on the characteristics of the scenario, access to data, questions of performance, to enumerate some. In the next chapter we present our implementation of the method and the main decisions that were taken relative to the transition from the theory to the engineering.

Chapter 4

Implementation

This chapter presents the implementation of the method and of the whole framework used to run the case studies. The main guideline for the implementation was that it should be completely independent from the scenario; from the kind of features that described both the perceptions and the predictions; and from the algorithms that the agents might use, in order to promote future reuse and an easy creation of the different case studies.

The language chosen for the implementation was Scala [Oa04, OSV10] due to its smooth integration of features from object-oriented and functional programming paradigms. This integration, if used carefully, can lead to an object-oriented approach useful to structure the solution, at a high level of abstraction, and a functional approach to write concise and highly maintainable code at a lower level of abstraction, i.e., inside the methods of each class or object[1]. Another advantage, besides being a modern language with many interesting new features, is the fact that it compiles to Java bytecode, which ensures that it runs on a Java virtual machine and that it can be directly integrated with pure Java libraries (e.g., the WEKA toolkit [HFH+09]).

A general view of the architecture of the system is presented in Fig. 4.1. The diagram does not focus on the implementation of a particular solution but only on the framework based on which every case study was built. New scenarios can be developed using this implementation and should only need to extend two classes: *Agent* and *Hypothesis*. Naturally, a new launcher should be created to initialize the solution as desired.

The architecture is made of three main conceptual parts: the integration of hypotheses, the multiagent system and the integration with the WEKA toolkit. Each of the following three sections focuses on the presentation of one of these parts.

[1] Scala allows the declaration of objects without a class that behave as singletons.

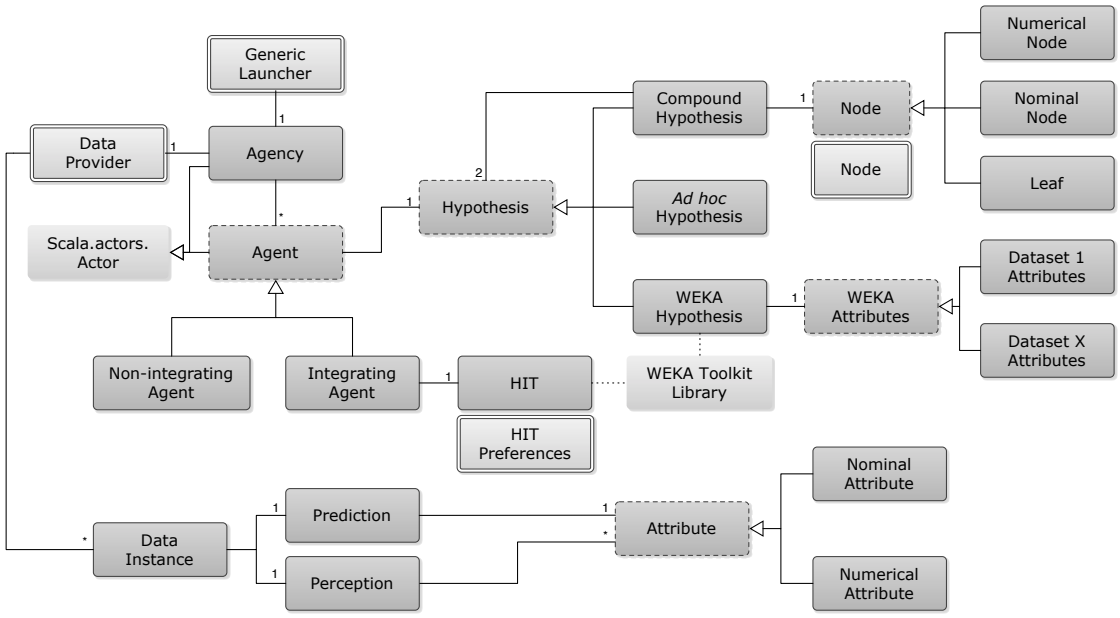


Figure 4.1: Architecture of the implementation

Grey with solid border: class; Grey with dashed border: abstract class; Light grey and double border: object (Singleton); Light grey without border: external libraries or classes.
Dashed connection: connection to an external library, various classes and objects may be referenced.

4.1 Implementing the method for the integration of hypotheses

The implementation of the method for the integration of hypotheses is strongly dependent on the creation of the necessary abstractions that were considered in the formal definition: the notions of perception, prediction, attribute[2] and hypothesis. Therefore, their implementation will be presented before the implementation of the method for the integration of hypotheses.

4.1.1 *Perception* and *Prediction*

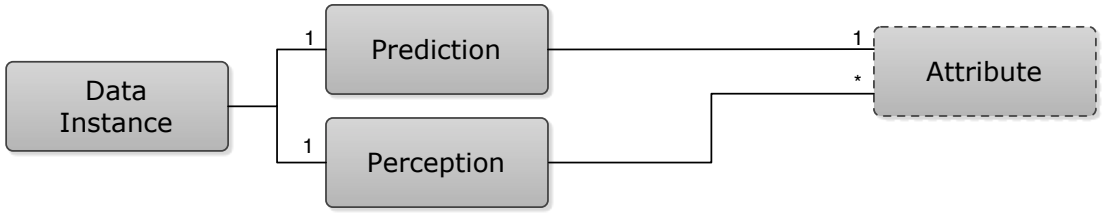


Figure 4.2: Perception and Prediction implementation

[2]In the implementation, the term *feature* was changed to *attribute* to be consisted with the terminology used by the WEKA toolkit.

In the formalization a perception is characterized by a set of attributes and the prediction is considered a value. In order to reuse an existing class we consider that the prediction is characterized by a single attribute without loss of the initial meaning. The *Data Instance* class is just a utility class to help structure the experiments by using pairs of *Perception*s and *Prediction*s, i.e., the perception and the correct prediction as an instance of the dataset. The code definition of the *Perception* class is presented as an example; both the *Prediction* and the *Data Instance* are similar but simpler.

```
1 class Perception(l: Attribute*) {
2   val dict = mutable.Map.empty[String, Attribute]
3   l.foreach(x => dict += (x.name -> x))
4
5   def getAtt(name: String): Attribute = dict(name)
6 }
```

A *Perception* object is created from a variable number of attributes and has a method that returns an attribute given its name.

4.1.2 *Attribute*

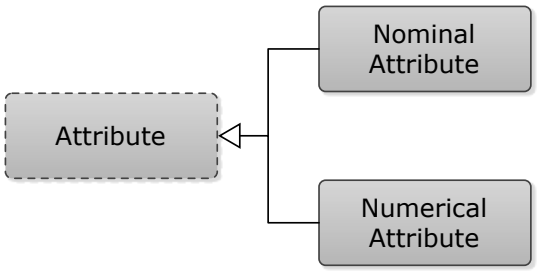


Figure 4.3: Attribute implementation

The formal definition of *Perception* considered that it was characterized by a set of features (i.e., attributes) which had values. However, it could be any type of value provided it was comparable (equality) with another value of the same type.

In order to obtain a class that could represent different types of values we made the class *Attribute* abstract and extended it with concrete classes according to our needs. Since in most datasets the values are real or symbolic those were the ones we considered but the extension to integers or complex numbers, for example, is reduced to the addition of a new class that extends *Attribute*.

To guarantee that they are comparable (equality), each concrete class should override the method **equals**, a requirement of the Scala language. Below is an example declaration of a numerical attribute class with the method **equals** being overridden.

```
class NumericalAttribute(n: String, v: Double) extends Attribute(n){
  def value = v

  override def equals(other: Any) = other match {
    case that: NumericalAttribute => this.value == that.value
    case _ => false
  }
}
```

4.1.3 *Hypothesis*

A hypothesis is a function that receives a perception as argument and returns a prediction independently of the reasoning mechanism used. The implementation of this concept is simple using an abstract class which defines a method that every concrete class inheriting from *Hypothesis* will have to implement. That method is the evaluation of a perception returning a prediction. Below is the definition of the abstract class *Hypothesis*:

```
abstract class Hypothesis {
  def evaluate(input: Perception): Prediction
}
```

4.1.4 The integration of hypotheses

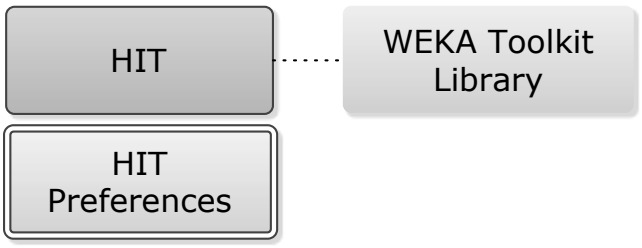


Figure 4.4: The HIT class

The integration of hypotheses is constructed by an object of the class *HIT* which has a single method called **integrate** that receives two hypotheses and a training set and returns the resulting hypothesis.

The reason why *HIT* is a class and not an object, despite having a single method and no variables, is due to the concurrent nature of the multiagent system. Being a class, every agent can have a different instance of the class *HIT* instead of competing for it as an unique centralized resource. Since the class does not have any information besides the definition of the method, its

footprint in the memory is very reduced and the creation of various objects of the class allows simultaneous integration by different agents.

The object represented in Fig. 4.4, *HIT Preferences*, holds the parameters that govern the integration to facilitate the development of launchers that execute multiple runs while experimenting different parameter configurations.

```
1 def integrate(h1: Hypothesis, h2: Hypothesis, data: List[DataInstance]): Hypothesis
2   for(datum <- data)
3     Generate WEKA instance and 'class' attribute identifying the hypotheses that
          correctly classify the instance (datum)
4   j48.buildClassifier(trainingSet)
5   val description = getTreeDescription
6   return new CompoundHypothesis(h1, h2, description)
7 }
```

The method **integrate**, presented in pseudo-code due to its size, is implemented as a direct transcription of the algorithm presented in Section 3.4 and uses WEKA's j48 algorithm to build the tree. The resulting hypothesis is then created using a description of this tree and both hypothesis. As presented before, the *Hypothesis* class is an abstract class and so there must be a concrete class that will be used to build the resulting hypothesis. That class is the *Compound Hypothesis*.

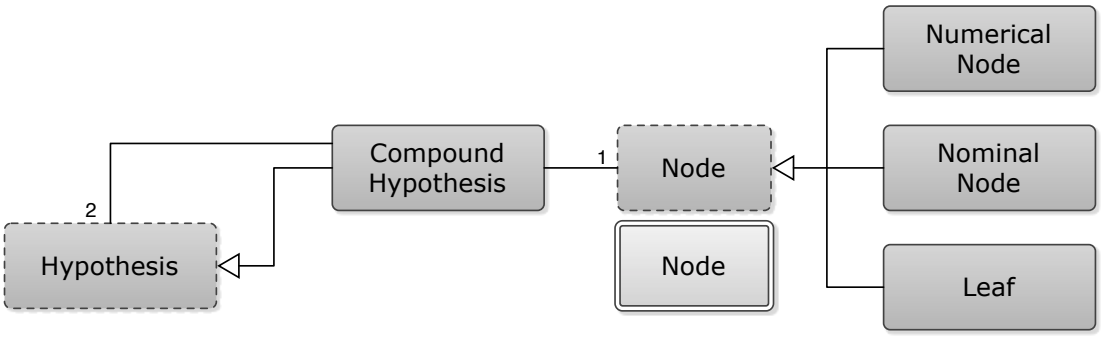


Figure 4.5: The Compound Hypothesis

The *Compound Hypothesis* extends *Hypothesis* and, as stated before, is created from the tree description and two hypotheses. However, it also has a fourth argument that is a boolean informing if pruning should be executed or not. The complete code definition of the *Compound Hypothesis* class is presented below.

```
1  class CompoundHypothesis(h1: Hypothesis, h2: Hypothesis, treeDescription: String,
       hitpruning: Boolean) extends Hypothesis {
2    var tree = Node("""[\s\n]*""".r.replaceAllIn(treeDescription, ""))
3    if(hitpruning)
4      pruneTree
5
6    def evaluate(input: Perception): Prediction = {
7      return if(tree.evaluate(input) == 2)
8                h2.evaluate(input)
9              else
10               h1.evaluate(input)
11   }
12
13   def pruneTree = {
14     val i = tree.prune
15     if(i != -1)
16       tree = new Leaf(i);
17   }
18 }
```

The tree description received as argument is used to build a tree structure that is used to evaluate new perceptions and to allow the implementation of the pruning algorithm (implemented as the recursive procedure proposed in the end of Section 3.3). From the definition of the method **evaluate** it is possible to observe that the first hypothesis is given priority over the second when their use is interchangeable. In this way, agents can decide on which hypothesis they are willing to prioritize by controlling the order of the arguments.

The code in line 2 of the *Compound Hypothesis* code definition is a call to a method of an object with the same name as the class *Node* that is also shown in Fig. 4.5. This object is called a companion object[3] in the Scala language and is commonly used for the creation of factory methods [GHJV95] such as **apply** in the code below[4].

The importance of the method **apply** is that it creates the appropriate concrete classes based on the tree description received, effectively hiding that complexity from the objects' creating nodes. (See below)

```
1  abstract class Node {
2    def evaluate(input: Perception): Int
3    def prune : Int
4  }
5
6  object Node {
7    val leaf = """\[[\w\/\(\)\.]+\]""".r
8    val leafExtract = """\[([-\d]+).*""".r
```

[3] Companion objects have special properties but those are not relevant to the understanding of this example.

[4] The method **apply** is special in the Scala language as object.apply() is equivalent to object().

```
9   val numeric = """\[(\w+):<=.*""".r
10  val nominal = """\[(\w+):=.*""".r
11
12  def apply(s: String) : Node = {
13    if(leaf.pattern.matcher(s).matches) { val leafExtract(id) = s; return new Leaf(
          id.toInt)}
14    else if(numeric.pattern.matcher(s).matches) return new NumericalNode(s)
15    else if(nominal.pattern.matcher(s).matches) return new NominalNode(s)
16    else return null}
17  }
18 }
```

The abstract class *Node* also defines two methods: **evaluate** and **prune**. The **evaluate** method, in each node, returns the result of the evaluate method in the correct subtree selected after testing the perception against a condition. If the node is a leaf, then it returns its integer value – X in the presentation of the pruning algorithm in Section 3.4. The **prune** method applies the pruning algorithm described in Section 3.3.

4.2 Using WEKA

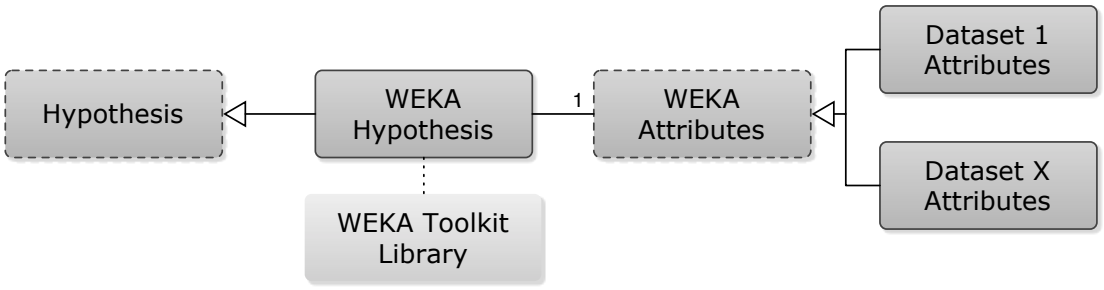


Figure 4.6: Integration with WEKA toolkit

The WEKA toolkit offers a great range of machine learning algorithms that are accessible through a Java API. This motivated the creation of a generic hypothesis that would have as a reasoning mechanism a WEKA algorithm. This generic hypothesis uses as a classifier any WEKA classifier that is passed as an argument to its constructor and was used extensively in the evaluation of this thesis. The *WEKA Hypothesis* implements **evaluate** as well as **train**:

```
1 class WekaHypothesis(cls: Classifier, wekaAttributes: WekaAttributes) extends
      Hypothesis {
2   def evaluate(input: Perception): Prediction
3   def train(data: List[DataInstance]): Unit
4 }
```

where **train** is used to train the Classifier (WEKA's abstract class from which all classifiers inherit) using a set of *Data Instance* and **evaluate** uses the Classifier to make its prediction.

Since the attributes that characterize a dataset are the same for every agent, the *WEKA Hypothesis* also receives as argument an object of a class inheriting from the abstract class *WEKA Attributes* that works as a provider of the data structures required by WEKA for a particular dataset. This avoids the repetition of the same procedure for every agent.

```
1 abstract class WekaAttributes {
2   def listAttributes : FastVector
3   def map : mutable.Map[String, Attribute]
4 }
```

4.3 The agents' infrastructure

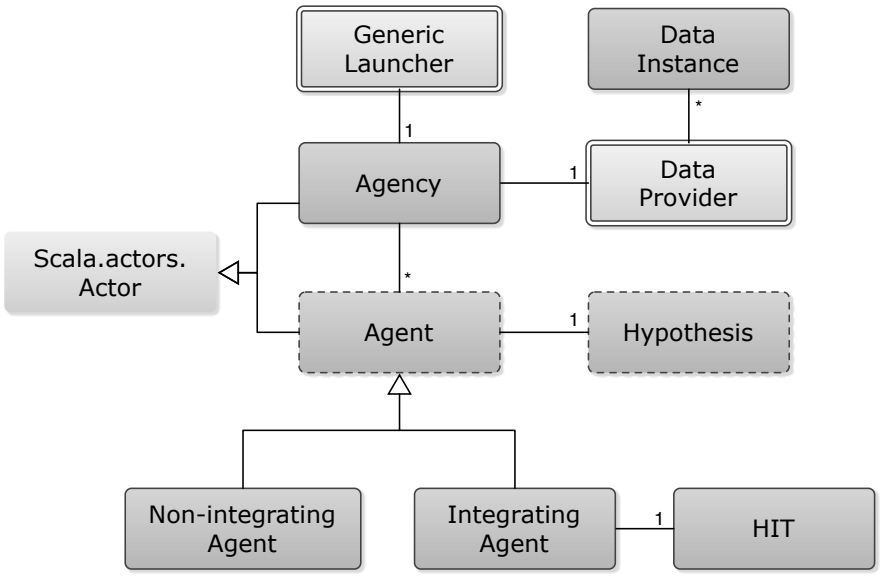


Figure 4.7: The multiagent system

The Scala language includes support for concurrent systems in the form of the actor model [HBS73]. Very briefly, and of direct interest to our implementation, every actor can be seen as a different concurrent entity that does not share memory with other actors. Even though centralized information can be shared, it would require explicit synchronization which removes value from the actor abstraction in which no explicit control of the concurrency is needed. Actors communicate by sending and receiving messages to and from other actors. Scala further provides another abstraction that is to "react" to messages which further simplifies the definition of the agent and has positive implications on performance [OSV10].

Both the *Agency* and the *Agent*s extend the Scala implementation of Actor and communicate with each other using asynchronous messages. A subset of the protocol of communication is represented for a simple *Agent*:

```
def act = {
  loop {
    react {
      case Train(data) => predictor.train(data)
                  sender ! Ready
      case Classify(datum) => sender ! Result(predictor.evaluate(datum))
      case Stop => exit
    }
  }
}
```

and the *Agency*:

```
def act = {
  agent ! Train(trainingData)
  receive{
    case Ready => println("train complete")
  }
  for(datum <- testData) {
    agent ! Classify(datum.input)
    receive {
       case Result(prediction) => println(prediction)
    }
  }
  agent ! Stop
}
```

In this simple example interaction, as well as in all of our case studies, the *Agency* has the initiative and the *Agent* merely reacts in the protocol. The *Agency* starts by sending a training message with the training set to the *Agent* which responds informing that his training has completed. Then, for each instance in the test set, the *Agency* asks the *Agent* to classify it and waits for the result. Finally, the *Agency* sends a termination message.

The abstraction over the concurrency control problems and the pattern matching capabilities of Scala motivated the use of the Scala.actors.Actor for the construction of the multiagent system. Pattern matching in Scala allows a very simple definition of protocols through the use of case classes that can be used almost as messages and that are the basis for the interaction protocol between the agency and the agents.

```
case class Train(l: List[DataInstance])
case class Ready(actor: Agent)
case class Classify(perception: Perception)
case class Result(actor: Agent, prediciton: Prediction)
case class Integrate(l: List[InteractiveWekaAgent])
case class Predictor(h: Hypothesis)
case class Stop
```

The protocol used is very simple and can be seen as a series of queries and responses where the agency has always the initiative and controls the interactions:

a) Agency: Train $\longrightarrow$ Agent: Ready

b) Agency: Classify $\longrightarrow$ Agent: Result

c) Agency: Integrate $\longrightarrow$ Agent: Ready

d) Agency: Stop

Additionally, Predictor is used by agents to share their predictor with other integrating agents (the Agency provides the list of the agents willing to integrate in the case class *Predictor*).

As previously mentioned, the *Learning Process Description* presented in MALEF proposes the exchange of more than just the hypothesis. However, the protocol only considers the exchange of hypotheses in order to be certain that the results obtained in the case studies are indeed due to the integration of external hypotheses and not the exchange of data, for example. Nevertheless, the extension to the protocol can be done by redefining or defining new case classes.

The *Data Provider* object in Fig. 4.7 has the role of retrieving the information concerning a dataset from files or databases and compile the dataset as a list of *Data Instance* to be used by the *Agency* to train and test the agents.

Chapter 5

Evaluation

The proposed solution for the integration of heterogeneous hypotheses resulted in a method whose theoretical foundations were presented in Chapter 3. However, the theoretical foundation is built on assumptions of complete knowledge of the environment which are impossible to assert in most multiagent learning realistic scenarios. In order to expand the applicability of the method, an algorithm was derived directly from the abstract method and some relaxation of the initial conditions was considered. The relaxation of the initial conditions is achieved mainly through the use of heuristic methods and by considering the integration of hypotheses using a limited training set instead of the entire set of possible perceptions in the environment.

However, this approach does not guarantee the construction of the hypothesis resulting from the integration of hypotheses as initially defined (Definition 3) creating, instead, a hypothesis that tries to approximate this theoretical solution. This motivates an empirical study to evaluate the quality of the new approximate hypothesis and how it behaves under different combinations of parameters and scenarios in comparison with the original hypotheses. To investigate this relationship between the quality of the hypotheses constructed through the integration of hypotheses and the original hypotheses, we propose the following evaluation methodology.

5.1 Evaluation methodology

In realistic scenarios, a complete description of the environment perceptions and corresponding predictions is not known. This renders the computation of the perfect theoretical solution impossible due to the lack of information about the environment[1]. This means that the approximate hypothesis resulting from the integration of hypotheses cannot be compared directly with the theoretical solution and, therefore, the quality of the approximate hypotheses must be measured through a comparison to the quality of the hypotheses of agents that do not perform integration.

The general experimental setting considers two types of agents: agents willing to integrate external hypotheses, using our method, and agents that do not. Each experiment is made up of

[1] In fact, if we had the complete knowledge, the learning algorithm would be meaningless.

several runs which are used to gain statistical confidence and explore different parameter configurations and conditions, both of the algorithm for the integration of hypotheses and of the scenario. A run in the experiment is composed by a training phase and a test phase. During the training phase a training set is provided to the agents in order to create an initial predictor. Afterwards, the agents which are willing to integrate external hypotheses may do so and, finally, the agents are assessed based on the performance obtained during a test phase where they are challenged to classify data contained in a test dataset.

All agents have off-the-shelf[2] algorithms from the WEKA toolkit [HFH+09] as initial hypotheses. In particular, three algorithms are used in the case studies: J48, a Java open-source implementation of the C4.5 algorithm by Quinlan [Qui93]; WEKA's implementation of the naive Bayes algorithm; and WEKA's implementation of a multilayer perceptron. The choice of the algorithms was motivated by their heterogeneity in terms of the basic approach each takes to the classification task and by the fact that their implementation can deal with both continuous and symbolic attributes.

No particular attention is given, in the case studies, to the processing times of the integration of hypotheses or the execution of the resulting hypotheses as they are strongly dependent on the choice of the initial algorithms of each hypothesis and the algorithm for the construction of the decision tree. The expected performance can be easily predicted as the evaluation of the set by both hypotheses and then the construction of the decision tree of the classified set (i.e. direct application of the algorithm).

Finally, the exchange of raw data among agents is not considered in any experiment despite being both simple and natural in multiagent systems and part of the learning process descriptions in MALEF. In fact, if we were to consider the exchange of data among the agents the results of the integrating agents would definitely be better. However, in the case studies the goal is to observe how the integration of hypotheses contributes to the performance of the agent and by not allowing the exchange of raw data we can be sure that the improvement in the performance was caused solely by the integration of hypotheses.

5.1.1 Case studies

The experiments were divided into case studies each of which has slightly different experimental settings and tackle specific objectives. A brief description of each of these case studies and their main objectives is presented below:

- The first case study focuses on the empirical evaluation of the abstract method presented in Section 3.3. Given the formal definition of the method, the main contribution of this case study is to observe if the implementation achieves the expected theoretical results. This comparison, while not proving any results concerning the abstract method, provides

[2] By *off-the-shelf* algorithms I mean that the algorithms are used with the default parametrization without any further tuning.

confidence in the implementation of the algorithm and serves as an introductory case study to the integration of hypotheses in scenarios of complete information.

The case study is divided into three experiments focusing on i) the integration of WEKA hypotheses; ii) the relation between the improvement and the symmetric difference of the sets of correctly classified perceptions of each hypothesis; and iii) the use of the pruning algorithm presented in Section 3.4.3.

- The second case study focuses on the use of our algorithm in more realistic scenarios where complete knowledge is not assumed but still in a fairly simple scenario and dataset. In order to further emphasize the notion of incomplete knowledge, the experiments of the case study concern agents that have been trained to be experts in the classification of different subsets. I.e., after the separation of the dataset into a training and a test set, the actual set provided for each agent to train is a carefully chosen subset of the initial training set.

  The first experiment in the case study is a direct comparison of the performance of both types of agents in an incomplete knowledge scenario. The second experiment studies the relation between the relative performance of integrating and non-integrating agents and the size of the training set. The final experiment focuses on the effect of the use of pruning techniques on the integration of hypotheses.

- The final case study was designed to study the application of the integration of hypotheses algorithm in a real, open problem: the prediction of financial indicators. This case study uses real financial data of eight European countries from a period of ten years and attempts to solve a hard learning problem to any off-the-shelf algorithm due to the high complexity of the domain. The main objectives of this case study are the creation of the best possible predictor and the study of whether the agents performing integration of external hypotheses perform better than the agents that keep their initial hypotheses. In order to find the best predictor several parameters and environment conditions are explored as in the previous case studies.

  The case study is divided into three experiments, each motivated by the previous experimental results. The first experiment considers the use of the dataset in chronological order. I.e., the training set consists of the first instances of the dataset. The second experiment uses the same dataset but with no specific order removing the sequential nature of the problem and dataset. Finally, the last experiment considers a geographic separation of the training dataset to create a scenario analogous to the second case study. In this experiment, each agent is trained as an expert in the financial prediction of a specific country to create a scenario of heterogeneity both in the base hypotheses and on the perceived data which is, as stated before, the main motivation for the development of this work.

5.2 First case study: Empirical study of the method for the integration of hypothesis

The first case study we present aims at the empirical study of the algorithm for the integration of hypotheses as suggested by the abstract method presented in Section 3.3. This means that all the information concerning the scenario will be available and that no heuristic approaches will be used which should allow a comparison with the theoretical results. However, the main goal of the case study is not the empirical validation of theoretical results but rather the confirmation that the implementation of the algorithm was successful and a demonstration of its use, without the need of a heuristic approach, in scenarios with complete information.

The first experiment is a comparison of off-the-shelf WEKA algorithm hypotheses with the integration of those hypotheses to observe the results obtained. Afterwards, two *ad hoc* hypotheses are created and the result of their integration is analysed in terms of the sets of perceptions they correctly classify. Finally, a small experiment using synthetic data is used to demonstrate the effects of the pruning algorithm proposed in Section 3.4.3.

5.2.1 Dataset

This case study uses the famous Iris dataset, [Fis36, FA10], which is probably the most well-known dataset in the machine learning literature. The dataset is composed of 150 instances describing four physical attributes of the flower (petal length, petal width, sepal length and sepal width) where the challenge is the classification of the species (*Iris Setosa, Iris Versicolour* and *Iris Virginica*). One interesting point that contributed to the popularity of the dataset is that the three species are not linearly separable even though this is not a challenge to modern algorithms. The dataset does not have any missing values and is very well balanced with 50 instances of each class. This fact, together with the small number of attributes, makes it a very simple domain commonly used for a first verification of new algorithms.

5.2.2 First experiment: WEKA off-the-shelf algorithm integration

This experiment uses the complete dataset both as training data and test data which are, as explained before, the conditions of the abstract method. The agents use WEKA algorithms as provided by the toolkit without any modification or tuning of the parameters and the integration of hypothesis algorithm uses a decision tree algorithm without any heuristics.

The experiment was run 25 times shuffling the order of the dataset and collecting the ratio of successful classifications of the agents' initial hypotheses (i.e., WEKA algorithms); of the integration of the possible pairs of hypotheses; and of the integration of all three hypotheses. The mean and standard deviation are presented in Table 5.1 and the raw data of all the runs can be found in Appendix B, Table B.1.

As expected, the standard deviation of the J48, Naive Bayes and their integration is zero as they do not depend on the order of the training set. In terms of the integration of hypotheses, it

Table 5.1: First case study, first experiment: classification results (ratio of correct classifications)

	J48	NB	MLP	I12	I13	I23	I123
Mean	0.98	0.96	0.985	0.987	0.997	0.99	0.998
Standard Deviation	0.0	0.0	0.0037	0.0	0.0043	0.0055	0.0032

J48: j48 algorithm; NB: Naive Bayes; MLP: Multilayer Perceptron; I12: integration of J48 and NB; I13: integration of J48 and MLP; I23: integration of NB and MLP; I123: integration of J48, NB and MLP.

manages to improve the results obtained by the single algorithms, even if only slightly, due to the guarantee that it will be at least as good as the best hypothesis being integrated.

In fact, what is really interesting to observe is that I12, I13 and I23 all improve over their constituent algorithms, though they could be equal, which means that there are three subsets of perceptions that are only correctly classified by each of the three algorithms. This implies that I123 should still do better than the single algorithms and the integrations of two algorithms since there is always another hypothesis that correctly classifies some subset not correctly classified by any of the previous. As expected, I123 is the best hypothesis in the experiment.

5.2.3 Second experiment: Integration of hypotheses improvement and relative complement measures

This experiment is designed to investigate the relationship between the improvement in the number of correctly classified instances when two hypotheses are integrated and the relative complement of the sets of perceptions each original hypothesis correctly classifies, i.e., the perceptions that one hypothesis correctly classifies and the other does not.

The motivation is that, with complete information, the set of perceptions the integration of h_1 and h_2 correctly classifies, $\mathscr{C}(\xi(\{h_1,h_2\})$, should be equal to the union of the set h_1 correctly classifies and the set of perceptions that h_2 correctly classifies and h_1 does not.

From Definition 3 we have:

$$\begin{aligned} \forall_{h_1,h_2\in\mathscr{H}} \quad \mathscr{C}(\xi(\{h_1,h_2\})) &= \mathscr{C}(h_1)\cup\mathscr{C}(h_2) \\ &= \mathscr{C}(h_1)\cup[\mathscr{C}(h_2)\setminus\mathscr{C}(h_1)] \end{aligned} \tag{5.1}$$

which, given $\mathscr{C}(h_1)\cap[\mathscr{C}(h_2)\setminus\mathscr{C}(h_1)]=\emptyset$[3], implies that

$$\forall_{h_1,h_2\in\mathscr{H}} \quad |\mathscr{C}(\xi(\{h_1,h_2\}))|-|\mathscr{C}(h_1)| = |\mathscr{C}(h_2)\setminus\mathscr{C}(h_1)| \tag{5.2}$$

In other words, the increase in the number of perceptions correctly classified by the integration of h_1 and h_2 relative to the number of perceptions correctly classified by h_1 should be equal to the cardinality of the relative complement of the set of perceptions h_1 correctly classifies with respect to the set of perceptions h_2 correctly classifies.

[3]From the definition of set difference.

This experiment uses the complete dataset for both training and test data and the two hypotheses used are built *ad hoc* for this particular dataset. The *ad hoc* hypotheses should be plausible but incomplete and neither of their sets of correctly classified perceptions should be a subset of the other. This should promote a good setting for the scenario described above.

In order to create the two *ad hoc* hypotheses that where specialized in two different subsets of the dataset, WEKA's J48 algorithm was run over the entire dataset and the resulting tree, with an accuracy of 98%, is shown in Fig. 5.1.

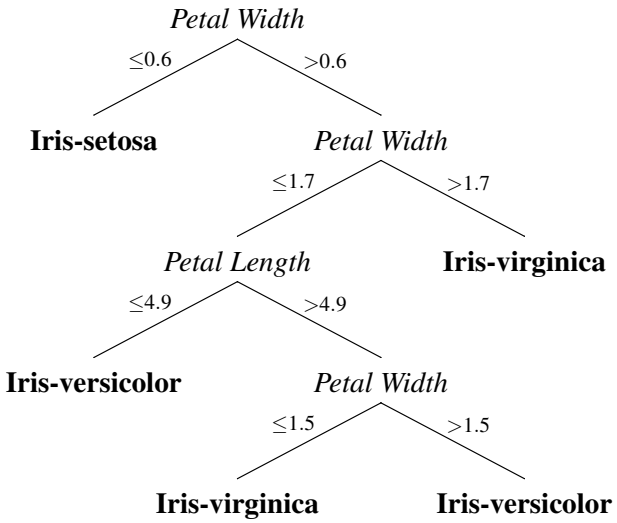


Figure 5.1: Tree resulting from the application of J48 on the Iris dataset.

From this tree, Fig. 5.1, two sub-trees were chosen to form the *ad hoc* hypotheses which are shown in Fig. 5.2. The motivation for this approach is that it would be very difficult to infer appropriate *ad hoc* hypotheses for the Iris dataset from simple observation of the data and decision trees are easy to read and interpret by humans. Additionally, it is straightforward to retrieve incomplete though plausible hypotheses by selecting sub-trees from the original tree.

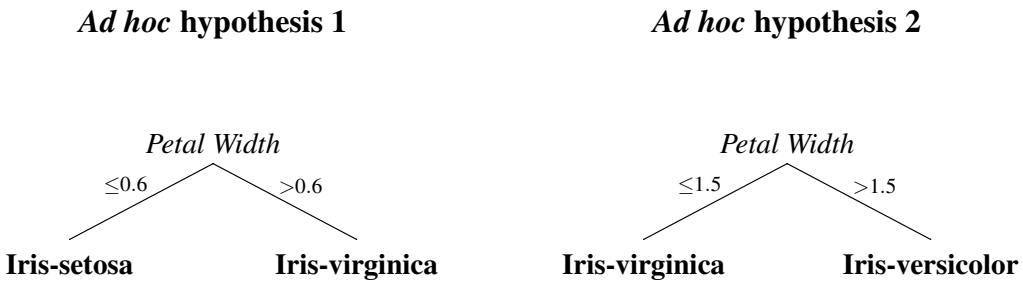


Figure 5.2: Ad hoc hypotheses for the Iris dataset.

Both *ad hoc* hypotheses and the integration of hypotheses were tested against the entire dataset and the number of perceptions classified by each *ad hoc* hypothesis but not by the other was also

recorded. The results of the test are presented as the total number of classified perceptions or, in the case of the relative complements, as the number of perceptions in the set, in Table 5.2.

Table 5.2: First case study, second experiment: classification results and relative complements

	$\mathscr{C}(h_1)$	$\mathscr{C}(h_2)$	$\mathscr{C}(\xi(\{h_1,h_2\}))$	$\mathscr{C}(h_1)\setminus\mathscr{C}(h_2)$	$\mathscr{C}(h_2)\setminus\mathscr{C}(h_1)$
No. perceptions	100	8	105	97	5

h_1: *ad hoc* hypothesis 1; h_2: *ad hoc* hypothesis 2

Recalling equation (5.2), we have

$$|\mathscr{C}(\xi(\{h_1,h_2\}))| - |\mathscr{C}(h_1)| = |\mathscr{C}(h_2)\setminus\mathscr{C}(h_1)| \Leftrightarrow$$
$$\Leftrightarrow 105 - 100 = 5$$

$$|\mathscr{C}(\xi(\{h_1,h_2\}))| - |\mathscr{C}(h_2)| = |\mathscr{C}(h_1)\setminus\mathscr{C}(h_2)| \Leftrightarrow$$
$$\Leftrightarrow 105 - 8 = 97$$

as expected which suggests that the algorithm is performing the optimal integration.

It is not necessary to run this experiment more than once since the hypotheses are static and do not depend on a learning algorithm; the dataset is always the same and the integration of hypotheses is deterministic. This implies that every run will provide exactly the same results. It is interesting, however, to consider different *ad hoc* hypotheses, especially a new hypothesis that performs better than *ad hoc* hypothesis 2.

A new experiment was run with the hypotheses in Fig. 5.3 whose results are presented in Table 5.3

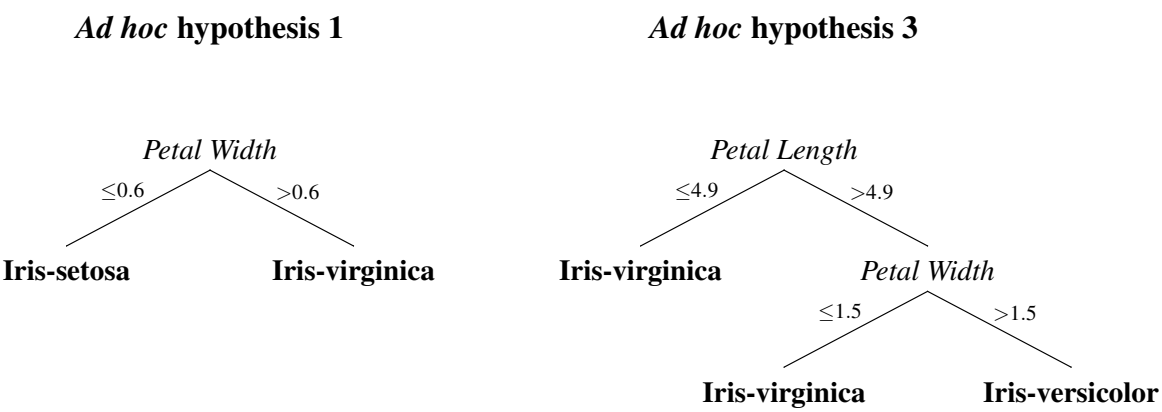


Figure 5.3: Ad hoc hypotheses for the Iris dataset. (2)

Again, we can verify that equation (5.2) holds in this experiment:

$$|\mathscr{C}(\xi(\{h_1,h_2\}))| - |\mathscr{C}(h_1)| = |\mathscr{C}(h_2)\setminus\mathscr{C}(h_1)| \Leftrightarrow$$
$$\Leftrightarrow 150 - 100 = 50$$

Table 5.3: First case study, second experiment: classification results and relative complements (2)

	$\mathscr{C}(h_1)$	$\mathscr{C}(h_3)$	$\mathscr{C}(\xi(\{h_1,h_3\}))$	$\mathscr{C}(h_1)\setminus\mathscr{C}(h_3)$	$\mathscr{C}(h_3)\setminus\mathscr{C}(h_1)$
No. perceptions	100	53	150	97	50

h_1: *ad hoc* hypothesis 1; h_3: *ad hoc* hypothesis 3

$$|\mathscr{C}(\xi(\{h_1,h_2\}))| - |\mathscr{C}(h_2)| = |\mathscr{C}(h_1)\setminus\mathscr{C}(h_2)| \Leftrightarrow$$
$$\Leftrightarrow 150 - 53 = 97$$

This is a closed domain where we know all the perceptions and, therefore, it would always be possible to build a perfect classifier. If J48 does not build a perfect classifier this is because it assumes that this dataset is only the training set and its heuristics prefer to misclassify a small number of perceptions in order to foster the generality of the predictor.

Nevertheless, it is interesting to observe that the original tree built by the J48 algorithm correctly classified 98% of the perceptions, i.e., 147 instances, and the integration of hypotheses applied to two of its sub-trees managed to correctly classify 100% of the perceptions, i.e., 150 instances.

5.2.4 Third experiment: Pruning algorithm

This experiment uses a synthetic dataset to evaluate the integration of two *ad hoc* hypotheses and compare the performance of the resulting hypothesis before and after the application of the pruning algorithm presented in Section 3.4.3.

The synthetic data is generated by creating a random perception and classifying it using a reference classifier. The perception is composed of three attributes: $Att1 \in [0,5]$, $Att2 \in [0.5,1.7]$ and $Att3 \in \{red, blue, green\}$. The reference classifier is presented in Fig. 5.4.

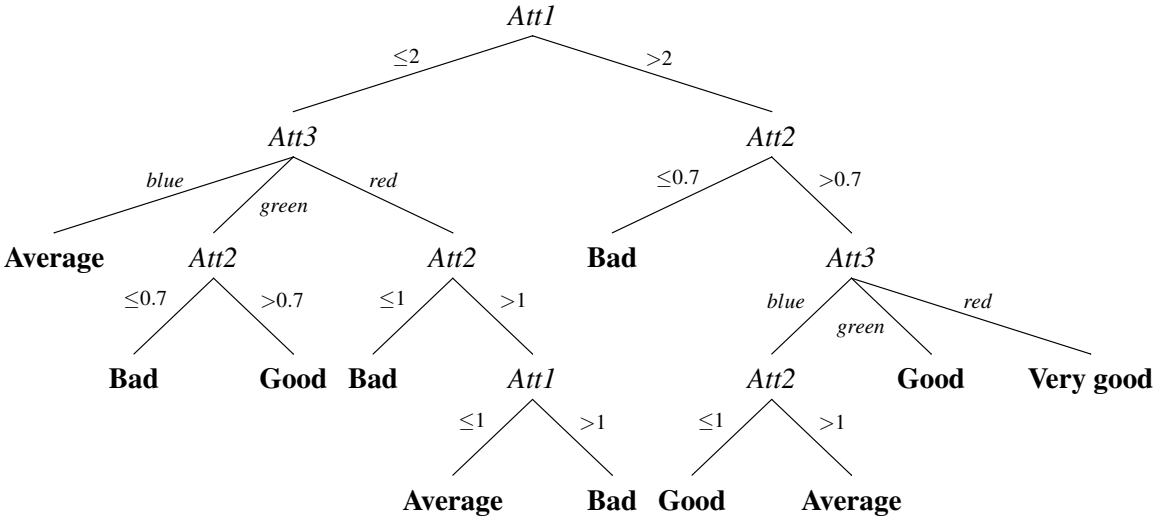


Figure 5.4: Reference tree classifier for the pruning experience.

The two *ad hoc* hypotheses follow the same tree structure but with slight differences in the conditions and classes in the leaves (Appendix C, Fig. C.1).

Table 5.4: First case study, third experiment: classification results (percentage of correct classifications)

	h_1	h_2	Unpruned	Pruned	Diff
Mean	0.562	0.621	0.926	0.926	0
Standard Deviation	0.016	0.015	0.010	0.010	0

h_1: first *ad hoc* hypothesis; h_2: second *ad hoc* hypothesis; Unpruned: unpruned integration of the *ad hoc* hypotheses; Pruned: pruned integration of the *ad hoc* hypotheses; Diff: Difference between the pruned and unpruned integration of the *ad hoc* hypotheses.

Table 5.4 presents the mean and standard deviation of the results of the *ad hoc* hypotheses, the pruned and unpruned integration of the *ad hoc* hypotheses and their difference. The full results for each of the 25 runs, each with a test dataset of 1000 instances, can be found in Appendix B, Table B.1.

This experiment suggests that pruning does not change the resulting hypothesis, as expected from the definition of the pruning algorithm. More importantly, it does considerably reduce the size of the tree representation of the resulting hypothesis as shown in Fig. 5.5. Fig. 5.5 shows the resulting hypothesis trees before and after pruning for a sample run. This result, nevertheless, should be similar in every run given the size and uniform distribution, in terms of feature values of the dataset.

5.2.5 Conclusions from the first case study

The objective of the empirical study of the method for the integration of hypotheses under the conditions of complete knowledge of the domain was not to validate the method but to provide some confidence in the correctness of the implementation of the algorithm since the theoretical implications have already been discussed in Chapter 3.

In this case study it was possible to observe all the expected theoretical results for each of the three experiments conducted: the integration of hypotheses; the relationship between the improvement obtained by integrating and the symmetric differences of the sets of correctly classified perceptions of each hypothesis; and the invariance of the hypothesis when the pruning algorithm is applied. Furthermore, it provides a good example for the direct application of the abstract method which is useful as an introductory case study to precede more realistic approaches presented in the following case studies.

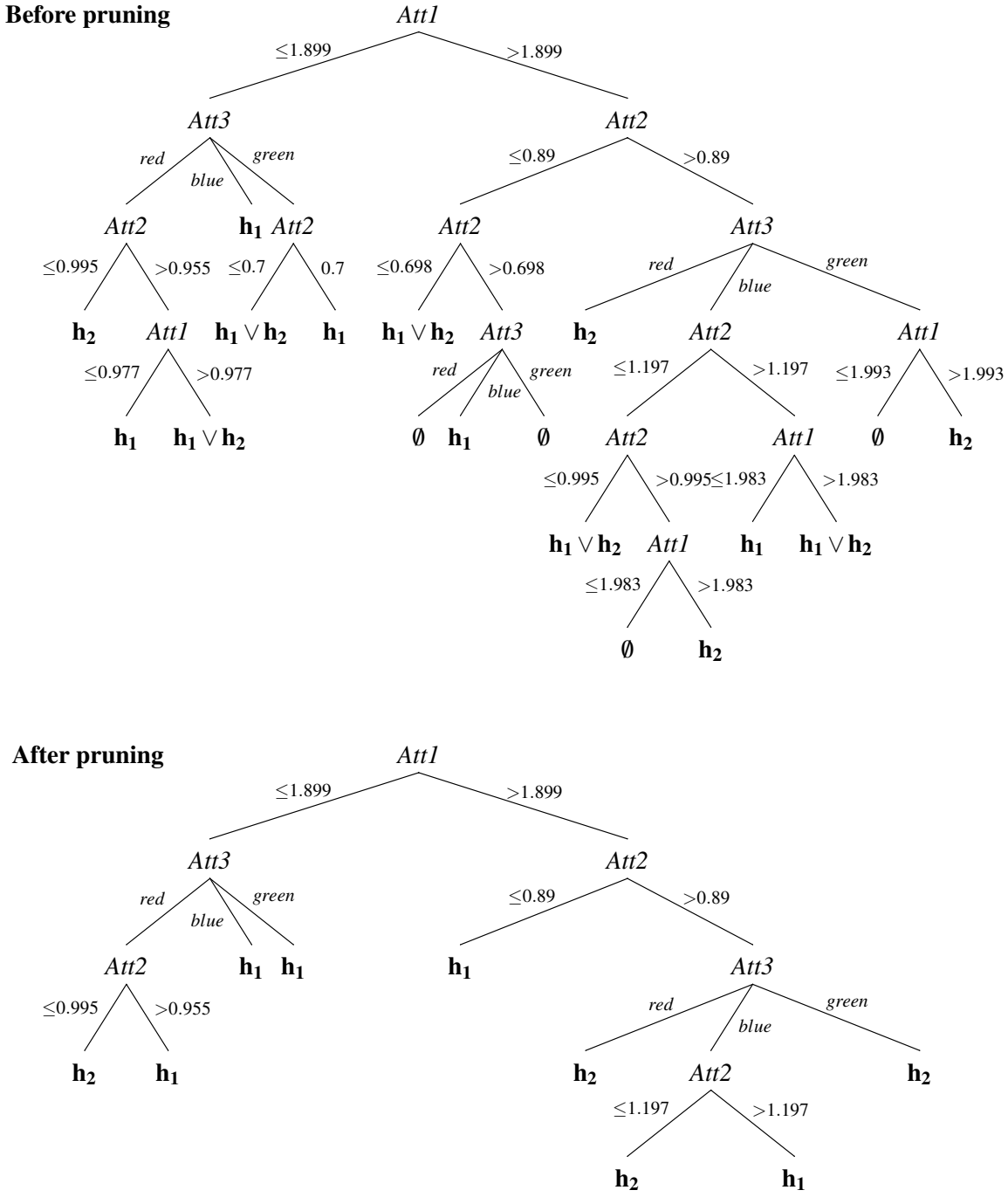


Figure 5.5: Resulting hypothesis tree before and after pruning for a sample run.

$\mathbf{h_1}$ and $\mathbf{h_2}$ refer to the first and second *ad hoc* hypotheses. $\emptyset$ means neither hypothesis can correctly classify that particular subset.

5.3 Second case study: Heterogeneous agents with partial access to data

The second case study focuses on the application of the method in domains with incomplete knowledge over the perceptions where the agents behaviour and interactions start to play a greater role in the learning task. In contrast to the first case-study, where the focus was the abstract method in controlled environment, this chapter focuses on the algorithm and how it behaves in more realistic scenarios. Some of the possible heuristic approximation and implementation issues are investigated in a scenario where the autonomy and heterogeneity of the agents will be enforced through different expertise and perception capabilities.

Given the agents' autonomy and individuality, a scenario where they do not share a common perception of the environment and train their hypotheses with different information is common in the field of multiagent systems and multiagent learning appearing as one of the most compelling motivations for the use of the agent abstraction. If it was not for their autonomy, they could be considered simple concurrent processes in a distributed system.

5.3.1 Dataset

The dataset used in this case-study was first presented by Forina *et al* [For, FA10] and consists of chemical attributes and an identifier for the winemaker of 178 wines from a region in Italy. The typical learning task is to build a classifier that can distinguish the winemaker based on the chemical parameters of the wine. In the dataset, there are wines from three different winemakers and each has information concerning thirteen chemical attributes: alcohol concentration, malic acid concentration, ash, alkalinity of the ash, magnesium concentration, total phenol concentration, flavonoid concentration, non-flavonoid phenol concentration, proanthocyanin concentration, colour intensity, colour hue, OD280/OD315 of diluted wines[4] and proline concentration.

5.3.2 First experiment: Heterogeneous expert agents

This experiment focuses on one of the main goals of this thesis which is the sharing of hypotheses in multiagent systems where the agents have heterogeneous hypotheses and different perceptions of the environment. The agents are trained with disjoint subsets of the dataset in order to turn them into experts in those particular subsets. However, they are tested using instances uniformly sampled from the dataset. The goal is to observe if the integration of the hypotheses of experts in different areas leads to a hypothesis with broader applicability and better performance than the one previously held by each expert.

In this experiment, the dataset is divided into training and test sets that respect the ratio between the different classes in the original dataset. The sets that will be used to train the experts are subsets of the training set including all of the instances of the class of expertise and randomly sampled instances amounting to 10% of each non-expertise class. The experiment was run 25

[4] OD280/OD315 is the ratio between absorbencies at 280 and 315 nm.

times with randomly chosen training and test sets, with half of the dataset assigned to the training set and the other half to the test set, and with agents that were willing to interact and integrate external hypotheses and agents that kept their initial hypotheses. The agents may integrate external hypotheses during integration rounds every ten classification tasks. There are additional agents that do not integrate external hypotheses but are trained with the entire training dataset to serve as reference.

There are three types of agents: one with access to the entire training set that does not perform integration; one with access to expert training sets that does not perform integration; and one with access to expert training sets that performs integration. For each type there are agents with hypotheses based on the J48 algorithm, naive Bayes and multilayer perceptron. Finally, for each algorithm in the types related to expert training sets, there is one agent specialised in each of the three winemakers. It is expected that the agents with access to all the information will perform best and serve as an upper bound to the interval where the performance of the agents integrating hypotheses should be. I.e., between the performance of the non-integrating expert agents and the performance of the agents with complete access to the training set. Furthermore, the relative position of the performance of the integrating agents in that interval is also a good indicator of the quality of the method.

The mean and standard deviation of the ratio of successful classifications by each agent type are presented in Table 5.5. In this table, expert agents of the same type and same base algorithm are represented by the mean of the results of all three agents.

Table 5.5: Second case study, first experiment: classification results (percentage of correct classifications)

	MLPall	MLPno	MLPint	NBall	NBno	NBint	J48all	J48no	J48int
Mean	0.978	0.919	0.939	0.960	0.863	0.931	0.940	0.811	0.922
Std Dev	0.009	0.024	0.019	0.013	0.056	0.020	0.0198	0.049	0.022

MLP: multilayer perception; NB: naive Bayes; J48: j48 algorithm.
all: complete training set; no: expert training set without integration; int: expert training set with integration.
Std Dev: Standard Deviation.

As expected, the agents with access to the complete dataset were the ones performing best as they had access to the greatest amount of information directly. Nevertheless, it can be verified that the integration of hypotheses contributed considerably to the increase in performance of the expert agents in the generic test set when compared to the agents under the same conditions who do not integrate external hypotheses. It can be seen that they approach the performances obtained by the agents with access to the complete dataset while showing a rather large improvement over the non-integrating experts. E.g. NBint[5] has a correct classification ratio that is almost 9% better than that of NBno and J48int is over 10% better than J48no which is considerable given the size of the interval between NBno and NBall (approx. 10%) and J48no and J48all (approx. 13%).

[5]NBint, J48int and MLPint, despite being named after the original algorithm are, at the end of the experiment, hypotheses resulting from the integration of various other hypotheses.

5.3.3 Second experiment: Training and test datasets ratio

This experiment focuses on the effects of the size of the training set on the percentage of correct classifications of the integrating and non-integrating agents. It is expected that smaller training sets, when compared to the complete dataset, will lead to hypotheses with worse results than when larger training sets are available. However, the agents integrating external hypotheses are indirectly receiving information from other training subsets which are likely to stem from other areas of expertise. The experimental setting, apart from the variation in the ratio between training and test data, is the same as the previous experiment, Section 5.3.2.

The mean of the mean ratio of correct classification for each of the 25 runs for each train/test dataset ratio are presented in Table 5.6. For each run and each ratio, the experiment is equivalent to the previous experiment, i.e., run 25 times and the result kept is the mean of those 25 runs.

Table 5.6: Second case study, second experiment: results (ratio of correct classifications)

TSet	MLPall	MLPno	MLPint	NBall	NBno	NBint	J48all	J48no	J48int
10%	0.920	0.919	0.924	0.868	0.881	0.910	0.791	0.767	0.897
20%	0.946	0.915	0.930	0.942	0.880	0.921	0.858	0.809	0.911
30%	0.963	0.923	0.935	0.948	0.866	0.925	0.893	0.819	0.916
40%	0.970	0.921	0.935	0.961	0.861	0.921	0.931	0.821	0.912
50%	0.978	0.920	0.939	0.957	0.852	0.923	0.938	0.801	0.916
60%	0.982	0.917	0.938	0.964	0.862	0.929	0.954	0.802	0.919
70%	0.987	0.917	0.943	0.965	0.859	0.928	0.971	0.790	0.918
80%	0.993	0.912	0.948	0.967	0.863	0.938	0.979	0.808	0.927
90%	0.996	0.919	0.950	0.965	0.852	0.937	0.987	0.786	0.927

MLP: multilayer perception; NB: naive Bayes; J48: j48 algorithm.
all: complete training set; no: expert training set without integration; int: expert training set with integration.
TSet: Percentage of the whole dataset used as a training set.
light grey: expert agent with integration was better than the agent with complete access to the training set.

As could be anticipated, the hypotheses with access to the entire training set increase their performance steadily as the training set increases but the expert agents that do not integrate do not seem to follow that pattern, having performances that do not expressively correlate with the size of the training set.

The most interesting observation, however, is that the integration of external hypotheses by expert agents achieved better results than the agents with access to the complete training set (highlighted in Table 5.6) for small training sets suggesting that the integration of hypotheses is surely a method of indirectly exchanging information. When the size of the training set increases, the differences in the access to the information make the agents with access to the complete training set considerably more accurate than the integrating agents as the integrating agents would need good raw data to perform good integrations.

Again, it should be noted that the agents with access to the complete training set are presented in order to give an idea of an upper bound and can be seen as not in the same "problem statement" as the two expert agent settings since they effectively have different conditions of actuation. Under

the same constraints, in this experiment, the integrating agents achieve clearly better results than the non-integrating agents.

5.3.4 Third experiment: J48 pruning and integrated hypothesis pruning

This experiment was designed to observe the behaviour of the system when pruning techniques are used. In particular, it focuses on the use of the pruning method described in Chapter 3 and the use of the traditional J48 pruning[6]. Since pruning techniques are associated with an attempt to increase the generality of the resulting hypotheses and avoid overfitting it is interesting to see their behaviour in different ratios of training to test set sizes. Even the use of our pruning algorithm will influence the final result given that this scenario does not consider complete knowledge on the part of the agents. This is due to the fact that not all perceptions are known which means that a change in the structure of the tree may lead to different predictions on perceptions not know beforehand.

The experimental setting is analogous to that of the previous experiment but, in addition to the variability in the ratio of training to test sets, we also include the use or not of the different pruning algorithms. Therefore, Table 5.7 presents the mean ratio of correct classifications of the various agents when using different pruning techniques and different learning ratios. The best results and the best configurations are highlighted for each training set size.

Table 5.7: Second case study, third experiment: results (ratio of correct classifications)

TSet	Prun	MLPall	MLPno	MLPint	NBall	NBno	NBint	J48all	J48no	J48int
10%	¬P¬J	0.922	0.920	0.924	0.864	0.870	0.888	0.786	0.771	0.841
	¬P J			0.922			0.895			0.849
	P¬J			0.915			0.906			0.895
	P J			0.926			0.908			0.897
30%	¬P¬J	0.957	0.924	0.931	0.953	0.872	0.898	0.898	0.801	0.851
	¬P J			0.934			0.893			0.859
	P¬J			0.936			0.926			0.913
	P J			0.928			0.917			0.906
50%	¬P¬J	0.973	0.919	0.926	0.960	0.865	0.888	0.944	0.807	0.859
	¬P J			0.929			0.891			0.859
	P¬J			0.936			0.922			0.917
	P J			0.933			0.912			0.904

MLP: multilayer perception; NB: naive Bayes; J48: j48 algorithm.
all: complete training set; no: expert training set without integration; int: expert training set with integration.
TSet: Percentage of the whole dataset used as a training set.
Prun: refers to the pruning algorithms; P: use of the pruning algorithm presented in Chapter 3; J: use of the J48 algorithm pruning.
Light grey: best for performance of the integrating agents for each TSet; Dark grey: Best pruning configuration for each TSet.

[6]The decision tree algorithm used in the implementation of the method is J48 as stated in the implementation chapter (Chapter 4).

In Table 5.7 we can observe how the performance of the agents integrating hypotheses is affected by whether pruning algorithms are being used or not. This experiment suggests that the use of our pruning algorithm leads to better results in this scenario. In terms of the use of the J48 pruning in the integration of hypotheses, it is interesting to observe that its use leads to better results with small training sets while with larger training sets its absence leads to better results. This might be related to the necessity of generalization in the scenario of a small training set whereas with larger training sets the need for generalization may not be so important as it seems to be better to have a more accurate predictor, even if slightly overfitted.

5.3.5 Conclusions from the second case study

This case study focused on a more realistic scenario, though still using a fairly simple dataset, where agents have different and incomplete perceptions of the environment. Despite these conditions, which render the direct application of the theoretical results impossible, the method for the integration of hypotheses was able to construct hypotheses that performed better than the hypotheses built using off-the-shelf WEKA algorithms.

The experiments focused mainly on investigating the reaction of the method for the integration of hypotheses when under different settings and parametrisations such as partial and selective access to training data, different ratios of training to testing dataset sizes and the use of pruning techniques in the integration algorithm. Overall, agents performing integration performed better than agents that did not perform integration under the same conditions of data access and, in scenarios of small training sets, even better than agents that did not perform integration but had access to the complete training set.

The behaviour of the method in relation to the use of pruning algorithms was consistent with all the integrating agents performing better under the same use of the pruning algorithms for the same conditions of train set size. However, whether to use or not a particular pruning algorithm seems to be too dependent on the characteristics of the scenario to be able to draw a clear conclusion.

5.4 Final case study: Financial forecasting

The final case study focuses on the application of our method in a real-world scenario: the prediction of financial indicators using real data. Unlike the previous case studies, whose objective was to study the method and how it behaved under different conditions, this case study focuses mainly on the construction of a good predictor for the financial indicators.

This scenario presents a hard challenge whose solution could have a great impact on the world's economy. However, the creation of a state-of-the-art financial predictor is deep in the field of machine learning and financial studies and, therefore, out of the scope of this thesis. The main goal of this case study is the creation of the best possible predictor as the integration of WEKA off-the-shelf algorithms and a comparison between the quality of the hypotheses of the agents using our method and those that do not.

5.4.1 Dataset

The dataset used in this case study was built using statistical data provided by the Organization for Economic Co-operation and Development (OECD) [oec]. The dataset consists of 26 monthly statistics of 8 European countries from 2001 to 2010 (10 years). The countries considered in the dataset are Italy, Portugal, Spain, Greece, Germany, France, Sweden and the United Kingdom. For a complete enumeration of the financial indicators of the dataset please refer to Appendix D. This dataset tends to be difficult for off-the-shelf algorithms that manage to perform only slightly better than a random predictor.

5.4.2 First experiment: Chronologically ordered data

The first experiment takes the temporal sequentiality of the instances of the dataset into account and presents the instances, both in the training set and the test set, in chronological order. The task is the prediction of the behaviour of the share price index[7] based on the 27 attributes[8] and the classification of that behaviour as either likely to rise or to fall in the next month.

In this experiment, there are agents willing to integrate external hypotheses and agents that do not perform integration. Both types are represented by three agents each with one of the three WEKA off-the-shelf algorithms (i.e., J48, naive Bayes and multilayer perceptron). The experiment was run 10 times for various configurations concerning the integration of hypotheses algorithm and the amount of training data available. The mean of the ratio of correctly classified instances for each configuration is presented in Table 5.8.

The difficulty of this problem for the off-the-shelf algorithms is apparent from the lack of linearity in the quality as the size of the training set varies. With 10% of the dataset being used as a training set, sJ48 has the worst performance of the three non-integrating agents but with 30% of the dataset being used as a training set, sJ48 boasts the best performance of the three. Furthermore, when 50% of the dataset is used as a training set all three non-integrating agents perform worse than when only 30% of the dataset is being used as a training set. It would be expected that, with the increase of the information available for training, the non-integrating agents would develop better hypotheses.

The main issue seems to be that the learning problem may actually change as time advances and, therefore, training with data from the first five years will lead to overfitted predictors that will not perform well in the following five years. Interestingly, this could perhaps be explained by the fact that the last five years covered by the dataset correspond to the beginning of the current European financial crisis which considerably changed the dynamics of the economical system. These changes could have led to changes in the learning problem itself in which a perception in the year 2003 might lead to different predictions than the same perception in the year 2008, for example.

[7] Average of daily quotations per month of each country.
[8] The 26 monthly statistics plus the country.

It is interesting to note, however, that the agents performing integration of external hypotheses were more resistant to this phenomenon, having performed better than the agents not willing to integrate in every pruning configuration when a training set was 50% of the dataset.

5.4.3 Second experiment: Non-ordered data

As observed in the previous experiment, the use of the chronologically ordered dataset leads to complex learning problem due to the highly dynamic behaviour of the financial environment in relation to time. In this experiment the same dataset is used but without consideration for the chronological order of the instances, i.e., the order is randomized in every run.

If the development of a proper financial predictor was our goal, then the sequential nature of the problem would definitely have to be reflected in the predictor. However, neither of the three initial algorithms is designed to treat time series. Therefore, while this experimental setting slightly deviates from the original financial prediction problem, it should be fairer for our predictors and a better indicator of the comparison between the agents which is, ultimately, the goal of the case-study.

As in the previous experiment, each configuration was run 10 times and the mean of the results is presented in Table 5.9.

The results show that agents integrating external hypotheses are generally better when the training dataset is small and have similar performance to the non-integrating agents with bigger datasets. Again, this is supportive of the idea that the integration of hypotheses can be seen as an effective mean of exchanging information among the agents, which is crucial when the training sets are small.

Also, comparing the results with the first experiment we can see that there was an increase in the overall ratio of correctly classified instances, which supports the idea that the classification problem might actually change with time. I.e. the same perception might lead to different predictions depending on the time of the perception.

5.4.4 Third experiment: Geographic separation

The third experiment expands on the second one in order to create a setting where the integration of hypotheses is more meaningful – the scenario where agents have partial access to the training dataset, which means heterogeneity both in the predictors and in the perceived data. As mentioned before, this is the scenario that motivated the appearance of the agent and multiagent systems abstractions and the development of this work.

In this experiment, we consider the same unordered dataset as in the second experiment but with the difference that there are eight agents willing to integrate external hypotheses and eight that are not willing to. Each agent of each type is an expert in one of the eight countries and is trained only with perceptions relative to that country.

The experiment was run 10 times for each configuration of the integration of hypotheses algorithm and training and test set ratios. The mean of the ratio of correctly classified perceptions

Table 5.8: Final case study, first experiment: results (ratio of correct classifications)

TSet	Prun	m	iJ48	iMLP	iNB	sJ48	sMLP	sNB
10%	¬P¬J	1	0.446	0.430	0.442	0.381	0.499	0.584
		3	0.479	0.458	0.481			
		5	0.455	0.440	0.454			
	¬P J	1	0.449	0.431	0.447			
		3	0.461	0.457	0.470			
		5	0.434	0.429	0.445			
	P¬J	1	0.439	0.442	0.443			
		3	0.446	0.450	0.430			
		5	0.445	0.454	0.446			
	P J	1	0.440	0.445	0.440			
		3	0.449	0.447	0.439			
		5	0.447	0.447	0.446			
30%	¬P¬J	1	0.538	0.552	0.549	0.598	0.505	0.558
		3	0.528	0.533	0.530			
		5	0.530	0.539	0.541			
	¬P J	1	0.528	0.543	0.540			
		3	0.521	0.543	0.531			
		5	0.527	0.541	0.534			
	P¬J	1	0.552	0.549	0.549			
		3	0.520	0.521	0.517			
		5	0.523	0.519	0.520			
	P J	1	0.541	0.541	0.533			
		3	0.527	0.543	0.530			
		5	0.541	0.541	0.542			
50%	¬P¬J	1	0.555	0.494	0.550	0.519	0.483	0.532
		3	0.552	0.487	0.554			
		5	0.561	0.489	0.560			
	¬P J	1	0.561	0.489	0.554			
		3	0.566	0.487	0.556			
		5	0.568	0.483	0.547			
	P¬J	1	0.543	0.500	0.557			
		3	0.550	0.499	0.564			
		5	0.565	0.514	0.569			
	P J	1	0.542	0.494	0.550			
		3	0.556	0.513	0.563			
		5	0.550	0.493	0.561			

MLP: multilayer perception; NB: naive Bayes; J48: j48 algorithm.
i: agent willing to integrate; s: agent not willing to integrate;
TSet: Percentage of the whole dataset used as a training set.
Prun: refers to the pruning algorithms; P: use of the pruning algorithm presented in Chapter 3; J: use of the J48 algorithm pruning; m: the minimum number of instances required to consider a new leaf (J48).
Light grey represents configurations where the agents integrating hypotheses were better than the agents that did not integrate.

Table 5.9: Final case study, second experiment: results (ratio of correct classifications)

TSet	Prun	m	iJ48	iMLP	iNB	sJ48	sMLP	sNB
10%	¬P¬J	1	0.711	0.703	0.708	0.645	0.655	0.681
		3	0.677	0.672	0.679			
		5	0.685	0.676	0.682			
	¬P J	1	0.673	0.672	0.675			
		3	0.687	0.678	0.683			
		5	0.690	0.681	0.684			
	P¬J	1	0.685	0.681	0.682			
		3	0.660	0.663	0.662			
		5	0.682	0.682	0.685			
	P J	1	0.668	0.666	0.670			
		3	0.664	0.662	0.666			
		5	0.673	0.672	0.676			
30%	¬P¬J	1	0.691	0.688	0.695	0.689	0.683	0.688
		3	0.662	0.660	0.675			
		5	0.684	0.668	0.690			
	¬P J	1	0.689	0.672	0.678			
		3	0.689	0.688	0.701			
		5	0.679	0.673	0.685			
	P¬J	1	0.693	0.682	0.688			
		3	0.677	0.670	0.680			
		5	0.670	0.666	0.670			
	P J	1	0.673	0.665	0.674			
		3	0.686	0.683	0.680			
		5	0.709	0.702	0.707			
50%	¬P¬J	1	0.682	0.665	0.691	0.684	0.678	0.690
		3	0.671	0.659	0.674			
		5	0.692	0.690	0.692			
	¬P J	1	0.681	0.692	0.694			
		3	0.685	0.686	0.696			
		5	0.685	0.686	0.687			
	P¬J	1	0.680	0.686	0.686			
		3	0.674	0.675	0.674			
		5	0.688	0.692	0.691			
	P J	1	0.696	0.680	0.692			
		3	0.684	0.675	0.685			
		5	0.681	0.675	0.675			

MLP: multilayer perception; NB: naive Bayes; J48: j48 algorithm.
i: agent willing to integrate; s: agent not willing to integrate;
TSet: Percentage of the whole dataset used as a training set.
Prun: refers to the pruning algorithms; P: use of the pruning algorithm presented in Chapter 3; J: use of the J48 algorithm pruning; m: the minimum number of instances required to consider a new leaf (J48).
Light grey represents configurations where the agents integrating hypotheses were better than the agents that did not integrate.

is presented in Table 5.11. Please note that, contrarily to the previous two tables, in this table the grey shading has the opposite meaning, i.e., highlighted cells represent configurations where the agents integrating external hypotheses were worse than the non-integrating agent.

The results shown in Table 5.11 are very promising, concerning the integration of hypotheses in highly heterogeneous scenarios, with quite significant improvement over the non-integrating agents. The improvement of the average performance of all configurations and of the best configuration over the corresponding non-integrating agent is shown in Table 5.10.

Table 5.10: Final case study, third experiment: Percent point increase of integrating agents over non-integrating

	TSet	J48it	J48pt	J48sp	MLPger	MLPgre	NBfr	NBsw	NBuk	Avg
Mean	10%	8.2	6.8	4.7	8.0	11.7	9.5	7.8	10.8	6.6
	30%	4.5	2.2	3.6	10.4	10.3	5.2	3.9	4.6	
	50%	5.7	4.3	5.9	10.3	11.7	3.6	1.9	2.6	
Max	10%	10.9	10.1	8.5	11.3	15.3	12.7	10.3	14.2	9.8
	30%	8.3	5.5	8.0	13.2	12.4	8.7	7.1	8.4	
	50%	8.9	7.0	8.9	14.0	15.2	6.4	4.6	6.5	

MLP: multilayer perception; NB: naive Bayes; J48: j48 algorithm.
it: Italy; pt: Portugal; sp: Spain; ger: Germany; gre: Greece; fr: France; sw: Sweden; uk: United Kingdom.
TSet: Percentage of the whole dataset used as a training set.
Mean: comparison with the mean; Max: comparison with the maximum.; Avg: average

The improvements shown in Table 5.10 are representative of the usefulness of the method of the integration of hypotheses in heterogeneous scenarios with improvements up to 15 percent points over the corresponding non-integrating agent. More importantly, all the values are positive and the average of the percent point increase is 6.6% in a particularly hard scenario.

5.4.5 Conclusions to the final case study

For the final case study we built a dataset from the raw statistics provided by OECD in order to tackle a particularly hard, real-world learning problem: financial forecasting. The objective was to test the method outside the controlled environment provided by well-balanced, normalized datasets where the characteristics of the dataset are known *a priori* and tackle a problem which we genuinely did not know how to solve.

An expert in finance together with a machine learning expert will definitely be able to build a better financial predictor than the one presented in this thesis but, as mentioned before, that is not our objective. Using off-the-shelf algorithms and the method for the integration of hypotheses we achieved significant results, not only in comparison to the non-integrating agents, but also in the problem itself. Considering the scenarios independent of the time sequence, the second and third experiment, the results obtained were consistently over 65% of correctly classified instances which is a decent result for a binary classifier.

In terms of comparison with non-integrating agents, the first experiment suggested that agents willing to integrate external hypotheses may have better resistance to overfitting than agents that do

not integrate external hypotheses as they completely dominated when the non-integrating agents showed signs of strong overfitting in a scenario that does not seem to pose a constant learning problem. Furthermore, the second experiment has shown that the integration of hypotheses might be a good conveyor of information allowing the integrating agents to perform consistently better with small training sets where information is scarce. Finally, in scenarios characterized by the heterogeneity of the hypotheses and of the perceived environment, agents willing to integrate achieve clearly better results as supported by Table 5.10.

Table 5.11: Final case study, third experiment: results (ratio of correct classifications)

TSet	Prun	m	SJ48it	SJ48pt	SJ48sp	SMLPger	SMLPgre	SNBfr	SNBsw	SNBuk
10%			0.590	0.600	0.619	0.584	0.548	0.571	0.588	0.558
			IJ48it	IJ48pt	IJ48sp	IMLPger	IMLPgre	INBfr	INBsw	INBuk
	¬P¬J	1	0.689	0.659	0.673	0.662	0.646	0.676	0.673	0.666
		3	0.667	0.667	0.658	0.662	0.665	0.667	0.664	0.674
		5	0.673	0.671	0.657	0.654	0.664	0.661	0.665	0.663
	¬P J	1	0.659	0.655	0.648	0.665	0.667	0.663	0.661	0.668
		3	0.668	0.683	0.681	0.678	0.672	0.675	0.671	0.678
		5	0.652	0.645	0.655	0.644	0.658	0.655	0.653	0.652
	P¬J	1	0.662	0.665	0.654	0.653	0.648	0.643	0.653	0.654
		3	0.648	0.638	0.639	0.631	0.642	0.637	0.647	0.630
		5	0.664	0.662	0.660	0.651	0.655	0.657	0.654	0.656
	P J	1	0.685	0.674	0.683	0.676	0.670	0.671	0.677	0.664
		3	0.693	0.695	0.689	0.692	0.701	0.698	0.691	0.684
		5	0.699	0.701	0.704	0.697	0.689	0.693	0.690	0.700
TSet	Prun	m	SJ48it	SJ48pt	SJ48sp	SMLPger	SMLPgre	SNBfr	SNBsw	SNBuk
30%			0.627	0.652	0.640	0.551	0.557	0.617	0.635	0.627
			IJ48it	IJ48pt	IJ48sp	IMLPger	IMLPgre	INBfr	INBsw	INBuk
	¬P¬J	1	0.665	0.670	0.662	0.660	0.665	0.660	0.664	0.657
		3	0.674	0.691	0.678	0.646	0.661	0.672	0.676	0.681
		5	0.678	0.677	0.672	0.660	0.673	0.681	0.683	0.694
	¬P J	1	0.699	0.694	0.706	0.679	0.674	0.691	0.705	0.711
		3	0.673	0.674	0.677	0.656	0.660	0.672	0.680	0.684
		5	0.710	0.707	0.720	0.683	0.675	0.703	0.706	0.703
	P¬J	1	0.656	0.658	0.649	0.631	0.643	0.650	0.652	0.648
		3	0.675	0.662	0.675	0.650	0.660	0.670	0.674	0.665
		5	0.632	0.638	0.643	0.630	0.634	0.640	0.638	0.633
	P J	1	0.651	0.649	0.660	0.646	0.645	0.648	0.653	0.655
		3	0.671	0.670	0.670	0.652	0.650	0.658	0.665	0.656
		5	0.681	0.692	0.693	0.666	0.681	0.677	0.685	0.690
TSet	Prun	m	SJ48it	SJ48pt	SJ48sp	SMLPger	SMLPgre	SNBfr	SNBsw	SNBuk
50%			0.604	0.624	0.606	0.548	0.534	0.626	0.652	0.645
			IJ48it	IJ48pt	IJ48sp	IMLPger	IMLPgre	INBfr	INBsw	INBuk
	¬P¬J	1	0.658	0.650	0.662	0.630	0.650	0.672	0.679	0.681
		3	0.682	0.673	0.678	0.649	0.646	0.675	0.693	0.682
		5	0.671	0.694	0.679	0.674	0.671	0.662	0.697	0.692
	¬P J	1	0.650	0.657	0.637	0.635	0.646	0.646	0.666	0.659
		3	0.633	0.652	0.656	0.635	0.651	0.653	0.642	0.653
		5	0.694	0.688	0.695	0.689	0.686	0.690	0.688	0.710
	P¬J	1	0.652	0.670	0.658	0.659	0.641	0.653	0.665	0.673
		3	0.673	0.687	0.676	0.672	0.656	0.662	0.670	0.664
		5	0.676	0.665	0.679	0.672	0.665	0.682	0.682	0.679
	P J	1	0.671	0.677	0.667	0.654	0.649	0.659	0.666	0.658
		3	0.644	0.669	0.652	0.648	0.644	0.660	0.664	0.664
		5	0.625	0.620	0.640	0.602	0.614	0.629	0.636	0.645

MLP: multilayer perception; NB: naive Bayes; J48: j48 algorithm.
it: Italy; pt: Portugal; sp: Spain; ger: Germany; gre: Greece; fr: France; sw: Sweden; uk: United Kingdom.
I: agent willing to integrate; S: agent not willing to integrate;
TSet: Percentage of the whole dataset used as a training set.
Prun: refers to the pruning algorithms; P: use of the pruning algorithm presented in Chapter 3; J: use of the J48 algorithm pruning; m: the minimum number of instances required to consider a new leaf (J48).
Light grey represents configurations where the agents integrating hypotheses were **worse** that the agents that did not integrate.

Chapter 6

Conclusions

6.1 Summary

This thesis focused on the development of a method for the integration of heterogeneous hypothesis in multiagent learning domains motivated by the work of Tožička *et al* – MALEF [TRPU08] – with special focus on MALEF's formalization of the learning problem and the subsequent proposal of the exchange of learning process descriptions as basis.

Our work can be summarized as follows:

i) We proposed a formal definition of **hypothesis** and of **integration of hypotheses** compatible with the formalization of the learning problem and learning process descriptions proposed by MALEF. Our definition of **hypothesis** is completely independent of the actual reasoning mechanism ensuring that the methods based on this definition can be applied to the treatment of heterogeneous hypotheses.

ii) We derived an abstract method for the construction of the integration of hypotheses for any finite set of hypotheses, from the previous definitions.

iii) We presented an algorithm for the integration of heterogeneous hypotheses based on the abstract method. In order to reduce the complexity of the resulting hypothesis, we also presented a pruning technique that guarantees the integrity of the resulting hypothesis. I.e., the resulting hypothesis is unaltered by the pruning procedure and only the structural complexity is reduced.

iv) Our abstract method is based on the assumption of complete knowledge of the environment for which it guarantees the optimal solution. In order to extend the applicability of the method to more realistic scenarios, this assumption was relaxed from the complete knowledge of the environment to the complete knowledge of the agent and, understanding that the agent has partial knowledge over the environment, heuristic pruning techniques were employed in an attempt to improve the generalization of the resulting hypotheses.

v) While still supported by a theoretical foundation, the extended method could not maintain the theoretical guarantees of the non-relaxed version and various case studies were designed to empirically validate its quality and applicability. The results have shown that agents using our method were consistently better than agents that did not use our method in heterogeneous scenarios where agents have different perceptions of the environment.

6.2 Results

The performance of our method was evaluated in three case studies with a set of specifically designed experiments.

The first case study provided confidence in the implementation of the algorithm through the repetition of some theoretical results. In a scenario of complete knowledge of the environment, using the classic machine learning Iris dataset, the algorithm was shown to perform optimally on the task of the integration of heterogeneous hypothesis. Furthermore, in an experiment designed to verify the theoretically expected improvement by the integration of an external hypotheses the algorithm also performed optimally. Finally, an experiment focusing on the pruning algorithm showed that the implementation produced the expected theoretical results, i.e., the invariance of the hypothesis resulting from the application of our method despite the reduction of the complexity of the structure.

In the second case study, we focused on the application of the method in more realistic scenarios of incomplete knowledge of the environment. In the first experiment, agents specialized on subsets of the domain were asked to classify a test set covering the whole domain and agents using our method outperformed equivalent agents that did not in every run. In the following experiments training to test ratios and different pruning techniques were investigated and, for every configuration, agents using our method also performed better than agents that did not use it.

In terms of the training to test sets ratio, when the training set is very small specialized agents using our method were capable of outperforming non-specialized agents, i.e., with complete access to the training set. This suggests that the integration of external hypotheses can be seen as an efficient means of exchanging information as only through their exchange agents were capable of gathering enough global knowledge to outperform an agent initially trained with the complete training set. In the third experiment, the different pruning configurations had different, though consistent, results that seem to depend on the conditions of the scenario. This suggests that there might be a better configuration for a particular scenario but that configuration will be dependent on the characteristics of that scenario.

Finally, the third case study tackled a real, open problem that proved to be hard to the off-the-shelf algorithms that were the core of our hypotheses. The dataset was real data provided by OECD on 26 financial monthly indicators covering 8 European countries during a period of ten years (2001 to 2010) and the task we proposed to tackle was financial forecasting.

In an initial instance, the dataset was presented in chronological order but the agents, using off-the-shelf algorithms as their core predictors, showed unexpected behaviour. Agents with access to

larger training sets performed worse. This could be an indicator that the actual learning problem could change with time probably due to the high complexity of the financial dynamics. A proper predictor that would take into consideration the time-series nature of the dataset could perhaps be built but that was beyond the scope of our work.

In order to make the problem fairer to our agents we ignored the time-series nature of the dataset and randomly selected the training and test sets from the data. The overall results of the agents increased considerably which suggests that our previous suspicion might be plausible. In this new scenario agents using our method consistently performed better for small training sets and on par for larger training sets for which the results of every agent, using our method or not, started to stall.

The final experiment focused on the scenario that motivated the whole work: highly-complex domains where agents are heterogeneous both in terms of the reasoning mechanism and in terms of the perceptions of the environment. We considered the agents to be experts in one country, i.e., trained with data from that country only, and ran the experiment with sixteen agents, two for each country, one using our method and the other not. Agents using our method achieved an increase of 6.6 percentual points in the quality when compared to the other agents on average over all the configurations considered. Considering the average of the best configurations for each agent, the increase in the quality when compared to the other agents was 9.8 percentual points. However, in some configurations, the increase in the quality of the agent over the corresponding non-integrating agent, i.e., the agent expert in the same country that does not use our method, was over 15 percentual points.

We believe that these results are significant and suggestive of the relevance of the method in real scenarios characterized by heterogeneity of the initial predictors and of the perceived data. Also, the exchange of hypotheses is much less expensive than the exchange of huge amounts of raw data in highly complex scenarios which is supported even further by the application of our pruning algorithm.

6.3 Future work

When extending to scenarios of incomplete knowledge the method uses the J48 algorithm to create the internal tree. One interesting future work would be the comparison of the performance of the method for different tree-generating algorithms. This is not restricted to decision tree algorithms as there might be other classifiers that, although not explicitly using tree structures, might be easily translated to a tree structure.

However, in our opinion, the main path for future development of the current work is the study of the strategic behaviours in multiagent learning scenarios given the possibilities offered by MALEF and our method. Most interaction mechanisms only assume the exchange of data and we do not know what kind of interactions might arise with the increased expressiveness provided by the exchange of learning descriptors and the use of methods for the integration of heterogeneous hypotheses. These interactions and their relation with the performance of the system and of the

individual agents should be studied in greater detail. Furthermore, the introduction of such mechanisms for the exchange of information in competitive scenarios is very promising, especially when the different agents might not be able to share raw data due to the large size of privacy policies (e.g., clinical domains) but could still benefit from selling part of its reasoning mechanism without disclosing the internal algorithm. To the best of our knowledge, this path of investigation in competitive scenarios is still unexplored.

Perhaps a harder but very interesting perspective is the use, in highly-complex scenarios, of heterogeneous agents that would tackle specific subsets of the domain. It is known that different learning algorithms perform better under different conditions and it might be possible to make use of that proneness to create experts agents in each of the subsets according to their internal algorithm. If this was successful, perhaps through the use of a large number of heterogeneous agents, the agents could use our method to integrate external hypotheses and possibly evolve to a surprisingly good hypothesis.

Appendix A

Proofs to Propositions

This chapter presents the full proofs for the Propositions presented in Chapter 3.

A.1 Proof to Proposition 1

Before we prove Proposition 1, we will first derive some auxiliary results.

Definition 7. *Let $\mathscr{C}_k : \mathscr{H} \to \wp(\mathscr{D}),\ \ k \in \mathscr{H}$, be the mapping of a hypothesis to the set of perceptions it correctly classifies in the set $\mathscr{C}(k)$, that is*

$$\mathscr{C}_k(h) = \{d \in \mathscr{C}(k) : h(d) = g(d)\}$$

We start by noting that $\mathscr{C}_k$, as defined in Definition 7, is such that

$$\mathscr{C}_k(h) = \{d \in \mathscr{C}(k) : h(d) = g(d)\} \Leftrightarrow \mathscr{C}_k(h) = \{d \in \mathscr{D} : h(d) = g(d) \wedge d \in \mathscr{C}(k)\} \tag{A.1}$$

which, from Definition 1, we have

$$\begin{aligned} (A.1) &\Leftrightarrow \mathscr{C}_k(h) = \{d \in \mathscr{D} : d \in \mathscr{C}(h) \wedge d \in \mathscr{C}(k)\} \\ &\Leftrightarrow \mathscr{C}_k(h) = \mathscr{C}(h) \cap \mathscr{C}(k) \end{aligned} \tag{A.2}$$

Proposition 3. $\forall k \in \mathscr{H}, \quad \forall S_1, S_2 \subseteq \mathscr{H}$*:*

$$\bigcup_{h_1 \in S_1} \mathscr{C}(h_1) = \bigcup_{h_2 \in S_2} \mathscr{C}(h_2) \Rightarrow \bigcup_{h_1 \in S_1} \mathscr{C}_k(h_1) = \bigcup_{h_2 \in S_2} \mathscr{C}_k(h_2)$$

Proof. We can rewrite Proposition 3 using A.2:

$$\bigcup_{h_1 \in S_1} \mathscr{C}(h_1) = \bigcup_{h_2 \in S_2} \mathscr{C}(h_2) \Rightarrow \bigcup_{h_1 \in S_1} (\mathscr{C}(h_1) \cap \mathscr{C}(k)) = \bigcup_{h_2 \in S_2} (\mathscr{C}(h_2) \cap \mathscr{C}(k)) \tag{A.3}$$

which, from the distributed law of set algebra, is equivalent to

$$\bigcup_{h_1 \in S_1} \mathscr{C}(h_1) = \bigcup_{h_2 \in S_2} \mathscr{C}(h_2) \Rightarrow \mathscr{C}(k) \cap \left[\bigcup_{h_1 \in S_1} \mathscr{C}(h_1) \right] = \mathscr{C}(k) \cap \left[\bigcup_{h_2 \in S_2} \mathscr{C}(h_2) \right] \tag{A.4}$$

Let $A = \bigcup_{h_1 \in S_1} \mathscr{C}(h_1)$ and $B = \bigcup_{h_2 \in S_2} \mathscr{C}(h_2)$, rewriting (A.4) using A and B we have

$$A = B \Rightarrow \mathscr{C}(k) \cap A = \mathscr{C}(k) \cap B \tag{A.5}$$

Given $A = B$, the right-hand equality is true, proving the proposition. □

Proposition 4.

$$\forall_{h \in \mathscr{H},\ S \subseteq \mathscr{H}}\ \ \mathscr{C}_h(h) \cup \left[\bigcup_{h' \in S} \mathscr{C}_h(h') \right] = \mathscr{C}_h(h)$$

Proof. Using (A.2) we can rewrite the Proposition as (quantifiers omitted)

$$(\mathscr{C}(h) \cap \mathscr{C}(h)) \cup \left[\bigcup_{h' \in S} \mathscr{C}(h') \cap \mathscr{C}(h) \right] = (\mathscr{C}(h) \cap \mathscr{C}(h)) \tag{A.6}$$

which, using the idempotent[1] and distributive laws of set algebra becomes

$$\mathscr{C}(h) \cup \left(\mathscr{C}(h) \cap \left[\bigcup_{h' \in S} \mathscr{C}(h') \right] \right) = \mathscr{C}(h) \tag{A.7}$$

finally, using the absorption law[2] of set algebra, we have

$$\mathscr{C}(h) = \mathscr{C}(h) \tag{A.8}$$

□

Lemma 1. $\forall h', h_1, h_2 \in \mathscr{H}$,

$$\mathscr{C}(h') = \mathscr{C}(h_1) \cup \mathscr{C}(h_2) \Leftrightarrow \forall_{d \in \mathscr{D}}\ h'(d) = \begin{cases} h_1(d) & \text{if } \ d \in \mathscr{C}(h_1) \\ h_2(d) & \text{if } \ d \in \mathscr{C}(h_2) \\ h_1(d) \vee h_2(d) & \text{if } \ d \notin (\mathscr{C}(h_1) \cup \mathscr{C}(h_2)) \end{cases}$$

Proof. We start by proving the implication towards the right-hand side. From Proposition 3 we have that

$$\mathscr{C}(h') = \mathscr{C}(h_1) \cup \mathscr{C}(h_2) \Rightarrow \mathscr{C}_{h_1}(h') = \mathscr{C}_{h_1}(h_1) \cup \mathscr{C}_{h_1}(h_2) \tag{A.9}$$

[1] Idempotent laws: $A \cup A = A$ and $A \cap A = A$ for any set A.
[2] Absorptions laws: $A \cup (A \cap B) = A$ and $A \cap (A \cup B) = A$ for any sets A and B.

and using the results presented in Proposition 4

$$\mathscr{C}_{h_1}(h') = \mathscr{C}_{h_1}(h_1) \cup \mathscr{C}_{h_1}(h_2) \Leftrightarrow \mathscr{C}_{h_1}(h') = \mathscr{C}_{h_1}(h_1) \tag{A.10}$$

which, from the Definition 1 and 2,

$$\begin{aligned} &\mathscr{C}_{h_1}(h') = \mathscr{C}_{h_1}(h_1) \Leftrightarrow \\ \Leftrightarrow \{d \in \mathscr{C}(h_1) : h'(d) = g(d)\} &= \{d \in \mathscr{C}(h_1) : h_1(d) = g(d)\} \Leftrightarrow \\ &\Leftrightarrow h'(d) = h_1(d), \quad \forall d \in \mathscr{C}(h_1) \end{aligned} \tag{A.11}$$

i.e., the first branch of the piecewise function is true.
Similarly, from Proposition 3 we have that

$$\mathscr{C}(h') = \mathscr{C}(h_1) \cup \mathscr{C}(h_2) \Rightarrow \mathscr{C}_{h_2}(h') = \mathscr{C}_{h_2}(h_1) \cup \mathscr{C}_{h_2}(h_2) \tag{A.12}$$

and using the results presented in Proposition 4

$$\mathscr{C}_{h_2}(h') = \mathscr{C}_{h_2}(h_1) \cup \mathscr{C}_{h_2}(h_2) \Leftrightarrow \mathscr{C}_{h_2}(h') = \mathscr{C}_{h_2}(h_2) \tag{A.13}$$

which, from the Definition 1 and 2,

$$\begin{aligned} &\mathscr{C}_{h_2}(h') = \mathscr{C}_{h_2}(h_2) \Leftrightarrow \\ \Leftrightarrow \{d \in \mathscr{C}(h_2) : h'(d) = g(d)\} &= \{d \in \mathscr{C}(h_2) : h_2(d) = g(d)\} \Leftrightarrow \\ &\Leftrightarrow h'(d) = h_2(d), \quad \forall d \in \mathscr{C}(h_2) \end{aligned} \tag{A.14}$$

i.e., the second branch of the piecewise function is true.
We still have to treat $\forall d \notin \mathscr{C}(h_1) \cup \mathscr{C}(h_2)$ which, from $\mathscr{C}(h') = \mathscr{C}(h_1) \cup \mathscr{C}(h_2)$ we have

$$d \notin \mathscr{C}(h_1) \cup \mathscr{C}(h_2) \Leftrightarrow d \notin \mathscr{C}(h') \tag{A.15}$$

Therefore,

$$h'(d) = h_1(d) \vee h'(d) = h_2(d), \quad \forall d \notin (\mathscr{C}(h_1) \cup \mathscr{C}(h_2)) \tag{A.16}$$

is true, proving this implication in all three cases as d is not in $\mathscr{C}(h_1)$ or $\mathscr{C}(h_2)$.
We will now prove the implication towards the left-hand side. Focusing on each case separately:

$$h'(d) = h_1(d), \forall d \in \mathscr{C}(h_1) \Rightarrow \mathscr{C}(h') \supseteq \mathscr{C}(h_1) \tag{A.17}$$

$$h'(d) = h_2(d), \forall d \in \mathscr{C}(h_2) \Rightarrow \mathscr{C}(h') \supseteq \mathscr{C}(h_2) \tag{A.18}$$

$$\begin{aligned} h'(d) = h_2(d) \vee h'(d) = h_2(d), \quad \forall d \notin (\mathscr{C}(h_1) \cup \mathscr{C}(h_2)) \Rightarrow \\ \Rightarrow (\mathscr{C}(h') \cap \overline{\mathscr{C}(h_1)} = \emptyset) \wedge (\mathscr{C}(h') \cap \overline{\mathscr{C}(h_2)} = \emptyset) \Leftrightarrow \\ \Leftrightarrow \mathscr{C}(h') \cap \overline{(\mathscr{C}(h_1) \cup \mathscr{C}(h_2))} = \emptyset \end{aligned} \tag{A.19}$$

The conjunction of the three cases should imply the left-hand side of the Lemma. We will join the first two and then the result with the third.
From (A.17) and (A.18) we have

$$[\mathscr{C}(h') \supseteq \mathscr{C}(h_1)] \wedge [\mathscr{C}(h') \supseteq \mathscr{C}(h_2)] \Leftrightarrow \mathscr{C}(h') \supseteq \mathscr{C}(h_1) \cup \mathscr{C}(h_2) \tag{A.20}$$

and from (A.20) and (A.19) we have

$$[\mathscr{C}(h') \supseteq \mathscr{C}(h_1) \cup \mathscr{C}(h_2)] \wedge [\mathscr{C}(h') \cap \overline{(\mathscr{C}(h_1) \cup \mathscr{C}(h_2))} = \emptyset] \tag{A.21}$$

which, by a well known result in set algebra stating that $A \cap \overline{B} = \emptyset \Leftrightarrow A \subseteq B$[3],

$$(A.21) \Leftrightarrow [\mathscr{C}(h') \supseteq \mathscr{C}(h_1) \cup \mathscr{C}(h_2)] \wedge [\mathscr{C}(h') \subseteq \mathscr{C}(h_1) \cup \mathscr{C}(h_2)] \tag{A.22}$$

and, finally, on the definition of the equality of sets,

$$(A.22) \Leftrightarrow \mathscr{C}(h') = \mathscr{C}(h_1) \cup \mathscr{C}(h_2) \tag{A.23}$$

We have proved both implications therefore proving the Lemma. □

Proposition 1. *The integration of hypotheses $\xi(S)$ can be constructed for any finite set $S \in \mathscr{H}$ as a finite composition of piecewise functions whose general form, for an arbitrary set of cardinality $n > 1$, is:*

$$\forall_{d \in \mathscr{D}} \;\; \xi(S_n)(d) = \begin{cases} h_n(d) & \text{if } \; d \in \mathscr{C}(h_n) \\ \xi(S_{n-1})(d) & \text{if } \; d \notin \mathscr{C}(h_n) \end{cases}$$

where $S_i = \{h_1, h_2, ..., h_{i-1}, h_i\}, \;\; S_i \subseteq \mathscr{H}$.
And whose base case, for a set of cardinality 1, is: $\xi(S_1) = h_1$

Proof. As in the statement, let S_i be such that $S_i \subseteq \mathscr{H} \wedge |S_i| = i$. The hypotheses in S_i are enumerated and will be called $S_i = \{h_1, h_2, ..., h_{i-1}, h_i\}$.[4]
We will start by proving the construction of $\xi(S)$, $\forall S \subseteq \mathscr{H}$, using a proof by mathematical induction on the cardinality of S.

[3] Presented as Theorem 5.3 in [Sto79].

[4] As stated when Proposition 1 was first presented, it should be noted that the enumeration of the hypotheses in the set does not affect the generality of the proposition and is used only to simplify the presentation of the result. Recalling Definition 3 and knowing that the union of sets is a commutative operator, one can see that any possible enumeration or ordering of the hypotheses in the set S_i leads to the same resulting hypothesis and, therefore, does not compromise the validity of the proposition.

i) First we consider a set with cardinality equal to one: $S_1 = \{h_1\}$

From the Definition 3 we have

$$\mathscr{C}(\xi(S_1)) = \bigcup_{h \in S_1} \mathscr{C}(h) \Leftrightarrow \mathscr{C}(\xi(S_1)) = \mathscr{C}(h_1) \tag{A.24}$$

and, from the definition of equality, Definition 2,

$$\mathscr{C}(\xi(S_1)) = \mathscr{C}(h_1) \Leftrightarrow \xi(S_1) = h_1 \tag{A.25}$$

Therefore, it is possible to construct the integration of hypotheses for a set with cardinality equal to one.

ii) We will now show that the integration of hypotheses can be constructed for S_{k+1} if it can be constructed for S_k (i.e., if $\xi(S_k)$ can be constructed), $\forall k \in \mathbb{N}$.

From the Definition 3 we have

$$\mathscr{C}(\xi(S_{k+1})) = \bigcup_{i=1}^{k+1} \mathscr{C}(h_i) \tag{A.26}$$

$$\mathscr{C}(\xi(S_{k+1})) = \bigcup_{i=1}^{k} \mathscr{C}(h_i) \cup \mathscr{C}(h_{k+1}) \tag{A.27}$$

$$\mathscr{C}(\xi(S_{k+1})) = \mathscr{C}(\xi(S_k)) \cup \mathscr{C}(h_{k+1}) \tag{A.28}$$

using Lemma 1 we have

$$\forall_{d \in \mathscr{D}}\ \xi(S_{k+1})(d) = \begin{cases} h_{k+1}(d) & \text{if} \quad d \in \mathscr{C}(h_{k+1}) \\ \xi(S_k)(d) & \text{if} \quad d \in \mathscr{C}(\xi(S_k)) \\ h_{k+1}(d) \vee \xi(S_k)(d) & \text{if} \quad d \notin (\mathscr{C}(h_{k+1}) \cup \mathscr{C}(\xi(S_k))) \end{cases} \tag{A.29}$$

Given that for the third case both hypotheses are equally valid, we can opt for one rewriting (A.29) as

$$\forall_{d \in \mathscr{D}}\ \xi(S_{k+1})(d) = \begin{cases} h_{k+1}(d) & \text{if} \quad d \in \mathscr{C}(h_{k+1}) \\ \xi(S_k)(d) & \text{if} \quad d \notin \mathscr{C}(h_{k+1}) \end{cases} \tag{A.30}$$

Having proved the basis (S_1) and the inductive step, we showed that the integration of hypotheses can be constructed for any finite set $S \in \mathscr{H}$. From the proof itself it can be seen that base case is $\xi(S_1) = h_1$ and that the general form, for an arbitrary set of cardinality $k+1$, is (A.30). Since the general form for the integration of hypotheses is a piecewise function whose arguments are either atomic hypotheses or integration of hypotheses of a set of smaller cardinality, we have that the integration of hypotheses can be constructed as a finite composition of piecewise functions for any finite set. □

A.2 Proof to Proposition 2

Proposition 2. *Any $D \subseteq \mathscr{D}$ can be described by a finite composition of binary feature partitions.*

Proof. We will prove the statement using mathematical induction over the number of features, i.e., the cardinality of $\mathscr{F}$.

i) First we consider the case in which $|\mathscr{F}| = 1$.

Let $\mathscr{F} = \{f_1\}$, we claim that

$$\exists_{k \subseteq \Delta(f_1)} \;\; P_{f_1}(k, \mathscr{D}) = D \tag{A.31}$$

Let $k' = \{f_1(d), d \in D\}$, then k' is such that

$$P_{f_1}(k', \mathscr{D}) = D \tag{A.32}$$

Given the construction of k', we know that $\forall_{d \in D} \;\; f_1(d) \in k'$. The only possibility (A.32) would be false was if $\exists_{d \notin D} \;\; f_1(d) \in k'$, i.e., there was some d not in D whose value for the feature f_1 was in k'.
However, there are not two different perceptions with the same value for feature f_1

$$\neg \exists_{d_1, d_2 \in \mathscr{D}} : d_1 \neq d_2 \wedge f_1(d_1) = f_1(d_2) \tag{A.33}$$

as from Definition 5 we have

$$\begin{aligned} \forall_{d_1, d_2 \in \mathscr{D}} \quad d_1 = d_2 &\Leftrightarrow \forall_{f \in \mathscr{F}} \quad f(d_1) = f(d_2) \\ &\Leftrightarrow f_1(d_1) = f_1(d_2) \end{aligned} \tag{A.34}$$

From this we know that $\neg \exists_{d \notin D} : f_1(d) \in k'$.

Therefore, $\exists_{k \subseteq \Delta(f_1)}$ that satisfies (A.31), proving the proposition for $|\mathscr{F}| = 1$.

ii) We will now show that the proposition holds for $|\mathscr{F}| = n+1$ if it holds for $|\mathscr{F}| = n$

Let $\mathscr{F} = \{f_1, f_2, f_3, ..., f_n, f_{n+1}\}$.
Also, let p^k be a composition of k binary feature partitions, in no particular order, over $f_i \in \mathscr{F}$ such that $1 \leq i \leq k$.
We have two possibilities:

a) we can construct p^n such that $p^n(\mathscr{D}) = D$. I.e., we do not need feature f_{n+1} to discriminate among $d \in \mathscr{D}$ as members, or not, of set D.

b) p^n cannot be constructed which means that

$$\exists_{d_1 \in D, d_2 \notin D} \, \forall_{f_i, 1 \leq i \leq n} \quad f_i(d_1) = f_i(d_2) \tag{A.35}$$

i.e., if p^n cannot be constructed it means there is at least two different perceptions whose values for all features $f_i, 1 \leq i \leq n$ are the same.
However, we have that

$$(A.35) \Rightarrow f_{n+1}(d_1) \neq f_{n+1}(d_2) \tag{A.36}$$

otherwise

$$\exists_{d_1,d_2 \in \mathscr{D}} \forall_{f \in \mathscr{F}(\mathscr{D})} \quad f(d_1) = f(d_2) \wedge d_1 \in D \wedge d_2 \notin D \tag{A.37}$$

which from the definition of equality of perceptions (Definition 5) we would obtain

$$\exists_{d_1,d_2 \in \mathscr{D}} \quad d_1 = d_2 \wedge d_1 \in D \wedge d_2 \notin D \tag{A.38}$$

which is impossible implying that (A.36) holds.
Since (A.36) holds, there is a $p_{f_{n+1}}$ that can correctly part

$$D' = \{d_1 \in \mathscr{D} : \exists_{d_2 \in \mathscr{D}} \forall_{f_i, 1 \leq i \leq n} \quad f_i(d_1) = f_i(d_2)\} \tag{A.39}$$

i.e. the set of perceptions that p^n could not characterize.
From this we know that there is $p_{f_{n+1}}$ such that $p_{f_{n+1}}(k^*, p^n(\mathscr{D})) = D$, where k^* is such that satisfies $p_{f_{n+1}}(k^*, p^n(\mathscr{D})) = D$, using an argument analogous to the one presented in the proof for $|\mathscr{F}| = 1$.

This composition is finite since for the base case it is only necessary one partition and at each step at most one new partitioning is added to the composition. This means that for $|\mathscr{F}| = m$, the maximum number of partitions that need to be used in the composition in order to characterize a set $D \in \mathscr{D}$ is m. From Definition 4 we have that a perception is characterized by a finite number of features and, therefore, $|\mathscr{F}|$ will always be finite and, accordingly, the composition will also be finite. □

Appendix B

Experimental Results

This appendix contains the tables with the full experimental results of the first case study. The other case studies experiments contain raw data that was too large to include. For an explanation and interpretation of the data refer to respective the case-study chapter.

Table B.1: First case study, first and third experiments. Raw data.

First Experiment								Third Experiment				
J48	NB	MLP	I12	I13	I23	I123		h_1	h_2	Unpruned	Pruned	Diff
0.98	0.96	0.987	0.987	1	0.993	1		0.582	0.618	0.932	0.932	0
0.98	0.96	0.98	0.987	0.993	0.987	0.993		0.553	0.615	0.916	0.916	0
0.98	0.96	0.98	0.987	0.993	0.98	0.993		0.57	0.599	0.929	0.929	0
0.98	0.96	0.987	0.987	1	0.993	1		0.582	0.601	0.928	0.928	0
0.98	0.96	0.98	0.987	1	0.993	1		0.568	0.627	0.926	0.926	0
0.98	0.96	0.98	0.987	0.993	0.993	0.993		0.562	0.609	0.923	0.923	0
0.98	0.96	0.98	0.987	0.993	0.98	0.993		0.544	0.614	0.928	0.928	0
0.98	0.96	0.987	0.987	1	0.993	1		0.558	0.625	0.915	0.915	0
0.98	0.96	0.987	0.987	1	0.993	1		0.554	0.622	0.9	0.9	0
0.98	0.96	0.987	0.987	1	0.993	1		0.545	0.635	0.926	0.926	0
0.98	0.96	0.987	0.987	1	0.993	1		0.542	0.617	0.92	0.92	0
0.98	0.96	0.987	0.987	1	0.993	1		0.57	0.607	0.921	0.921	0
0.98	0.96	0.987	0.987	1	0.993	1		0.576	0.627	0.94	0.94	0
0.98	0.96	0.987	0.987	1	0.993	1		0.557	0.632	0.932	0.932	0
0.98	0.96	0.987	0.987	1	0.993	1		0.569	0.594	0.919	0.919	0
0.98	0.96	0.987	0.987	1	0.993	1		0.571	0.602	0.921	0.921	0
0.98	0.96	0.987	0.987	0.987	0.98	0.993		0.548	0.636	0.923	0.923	0
0.98	0.96	0.987	0.987	1	0.993	1		0.554	0.633	0.938	0.938	0
0.98	0.96	0.987	0.987	0.987	0.98	0.993		0.603	0.609	0.937	0.937	0
0.98	0.96	0.973	0.987	0.993	0.987	0.993		0.547	0.624	0.933	0.933	0
0.98	0.96	0.987	0.987	1	0.993	1		0.573	0.62	0.929	0.929	0
0.98	0.96	0.987	0.987	1	0.993	1		0.565	0.646	0.941	0.941	0
0.98	0.96	0.98	0.987	0.993	0.98	0.993		0.568	0.655	0.947	0.947	0
0.98	0.96	0.987	0.987	1	0.993	1		0.522	0.638	0.914	0.914	0
0.98	0.96	0.987	0.987	1	0.993	1		0.557	0.624	0.914	0.914	0

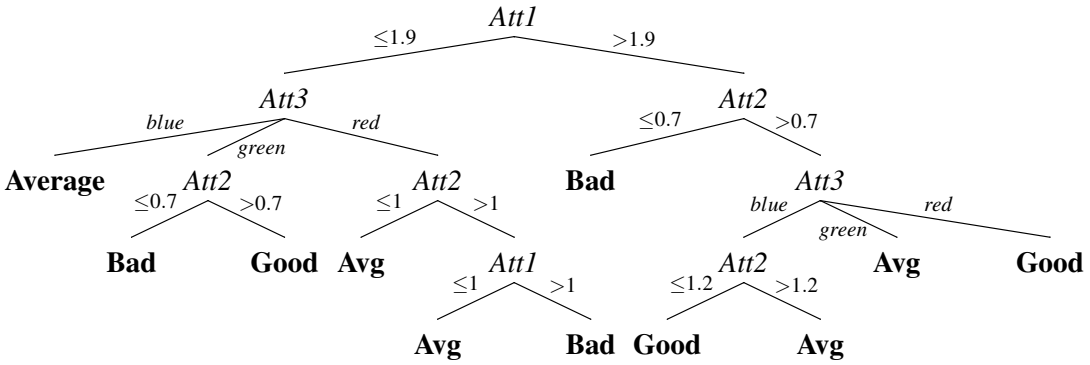
NB: Naive Bayes; MLP: Multilayer Perceptron; I12: integration of J48 and NB; I13: integration of J48 and MLP; I23: integration of NB and MLP; I123: integration of J48, NB and MLP.
h_1: *ad hoc* hypothesis 1; h_2: *ad hoc* hypothesis 2; Unpruned: unpruned integration of the *ad hoc* hypotheses; Pruned: pruned integration of the *ad hoc* hypotheses; Diff: Difference between the pruned and unpruned integration of the *ad hoc* hypotheses.

Appendix C

Additional material

This appendix contains diagrams and information relevant to the thesis but that were considered to be accessory and too large considering the natural flow of the chapters.

***Ad hoc* hypothesis 1**



***Ad hoc* hypothesis 2**

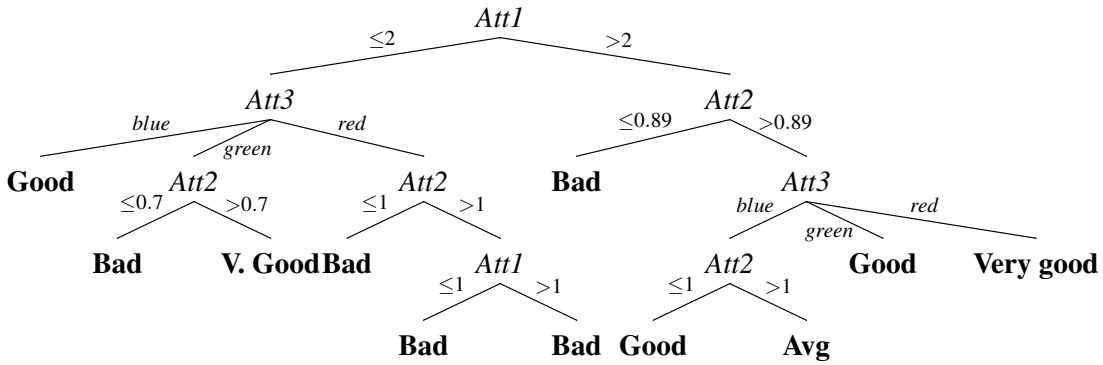


Figure C.1: First case-study, third experiment: *ad hoc* hypotheses

Appendix D

Financial Dataset

The financial dataset was created using monthly statistic indicators provided by the Organization for Economic Co-operation and Development (OECD) [oec]. The dataset covers ten years (2001 to 2010) and eight countries: Italy, Portugal, Spain, Greece, Germany, France, Sweden and the United Kingdom. The monthly statistical indicators are enumerated below.

- Retail Trade Volume
- Harmonised Unemployment Rate
- OECD Standardised BCI, Amplitude Adjusted (Long-term Average 100sa)
- Index of Industrial Production
- Share Prices, Index 2005=100
- BLSA Balance Seasonally Adjusted CSCICP02 Economic Confidence Indicator
- BLSA Balance Seasonally Adjusted CSESFT Future Tendency
- BLSA Balance Seasonally Adjusted BVEMFT Future Tendency
- BLSA Balance Seasonally Adjusted BREMFT Future Tendency
- Long-term Interest Rates, Per Cent Per Annum
- Trend GDP
- Normalised Composite Leading Indicators
- Relative Consumer Price Indices
- Normalised GDP
- Trend Restored Composite Leading Indicators
- BLSA Balance Seasonally Adjusted BVCICP Composite Indicators

- Amplitude Adjusted Composite Leading Indicators
- BLSA Balance Seasonally Adjusted BRBUFT Future Tendency
- Reserve Assets SDR Millions
- Short-term Interest Rates, Per Cent Per Annum
- BLSA Balance Seasonally Adjusted BCEMFT Future Tendency
- Consumer Price Index
- 12-month Rate of Change Of The Trend Restored Composite Leading Indicators
- OECD Standardised CCI Amplitude Adjusted Long Term Averages
- BLSA Balance Seasonally Adjusted BRCICP Composite Indicators
- Ratio to Trend GDP

References

[BFOS84] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and regression trees*. Wadsworth, Belmont, 1984.

[BGST99] S. Bailey, R. Grossman, H. Sivakumar, and A. Turinsky. Papyrus: a system for data mining over local and wide area clusters and super-clusters. *Proceedings of the Conference on Supercomputing*, pages 13–18, 1999.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.

[BK99] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: bagging, boosting and variants. *Machine Learning*, pages 105–139, 1999.

[Cha01] N. Chawla. Creating ensembles of classifiers. *Proceedings of ICDM 2001*, pages 580–581, 2001.

[DC04] D. Dash and G.F. Cooper. Model averaging prediction with discrete bayesian networks. *Journal of Machine Learning Research*, pages 1177–1203, 2004.

[Die00] Thomas Dietterich. Ensemble methods in machine learning. *MCS 2000, LNCS 1857*, pages 1–15, 2000.

[FA10] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[Fis36] R. A. Fisher. The use of multiple measurements in taxonomic problems. In *Annual Eugenics, 7, Part II*, pages 179–188, 1936.

[For] M. Forina. Parvus – an extendible package for data exploration, classification and correlation. In *Institute of Pharmaceutical and Food Analysis and Technologies*.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns*. Addison Wesley, 1995.

[HBS73] Carl Hewitt, Peter Bishop, and Richard Steiger. A universal modular actor formalism for artificial intelligence. In *IJCAI*, 1973.

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian Witten. The weka data mining software: An update. In *SIGKDD Explorations, Volume 11, Issue 1*, 2009.

[Loh08] Wei-Yin Loh. Classification and regression tree methods. In *Encyclopedia of Statistics in Quality and Reliability*, pages 315–323, 2008.

[Mit97] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[Nun06] Luis Nunes. *Learning From Multiple Sources in Heterogeneous Groups of Agents*. PhD in computer science, Faculty of Engineering, University of Porto, 2006.

[Oa04] Martin Odersky and al. An Overview of the Scala Programming Language. Technical Report IC/2004/64, EPFL, Lausanne, Switzerland, 2004.

[oec] Organization for economic co-operation and development statistics. `http://stats.oecd.org/Index.aspx`.

[OSV10] Martin Odersky, Lex Spoon, and Bill Venners. *Programming in Scala: A Comprehensive Step-by-step Guide*. Artima Incorporation, USA, 2nd edition, 2010.

[PL05] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

[Qui93] Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[RN10] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, Inc, third edition, 2010.

[Rov08] Michael Rovatsos. Collaborative and strategic multiagent learning. `http://homepages.inf.ed.ac.uk/ckiw/rpml/rpml-talk2008.pdf`, 2008.

[RPR09] Xavier Rafael-Palou and Michael Rovatsos. Collaborative agent-based learning with limited data exchange (short paper). In *Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, pages 1191–1192, 2009.

[RPRMPLA09] Xavier Rafael-Palou, Michael Rovatsos, Mariola Mier-Perez, and Magi Lluch-Ariet. Collaborative agent-based learning for brain tumour diagnosis. In *Proceedings of the 2009 conference on Artificial Intelligence Research and Development: Proceedings of the 12th International Conference of the Catalan Association for Artificial Intelligence*, pages 128–137. IOS Press, 2009.

[SFC97] S. Stolfo, D. Fan, and P. Chan. Jam: java agents for meta-learning over distributed databases. *Proceedings of the KDD-97*, pages 74–81, 1997.

[SLB09] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, first edition, 2009.

[SPG07] Yoav Shoham, Rob Powers, and T. Grenager. If multi-agent learning is the answer, what is the question? In R. Vohra and M. Wellman, editors, *Artificial Intelligence 171 (7)*, pages 365–377, 2007.

[Sto79] Robert R. Stoll. *Set Theory and Logic*. Dover Publications, 1979.

[SV00] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. In *Autonomous Robots*, 2000.

REFERENCES

[TJP06] J. Tozicka, M. Jakob, and M. Pechoucek. Market-inspired approach to collaborative learning. *Cooperative Information Agents X*, pages 213–227, 2006.

[TRPU08] Jan Tožička, Michael Rovatsos, Michal Pechoucek, and Stepan Urban. Malef: Framework for distributed machine learning and data mining. *Int. J. Intelligent Information and Database Systems*, pages 6–24, 2008.

[WMJ03] Z. Wei, L. Moreau, and N. Jennings. Recommender systems: a market-based design. *Proceedings of AAMAS-03*, pages 600–607, 2003.