UNIVERSIDADE DO PORTO
FACULDADE DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELECTROTECNICA E COMPUTADORES

REDES NEURONAIS
PREVISÃO DE CONSUMOS DE GÁS EM LISBOA

RELATÓRIO DE ESTÁGIO

RUI MANUEL SANTOS RODRIGUES LEITE

1992/93

PRODEP

Parecer Científico sobre o estágio de

Rui Manuel Santos Rodrigues Leite

O estagiário realizou sob minha orientação um trabalho de estágio sobre o tema que lhe foi fixado, tendo sido de uma forma geral alcançados os objectivos científicos inicialmente propostos.

A orientação científica incidiu sobre a definição de objectivos, fornecimento de bibliografia ou acesso à mesma e verificação dos progressos conseguidos.

Esta orientação foi exercida em coordenação com o supervisor do estagiário, na instituição onde fisicamente foi realizada a parte substancial dos trabalhos.

O nível de qualidade do trabalho executado, que se pode deduzir do relatório elaborado pelo estagiário, é bom, ao nível do grau de formação académica do estagiário. O relatório resume e descreve, de forma correcta, as fases dos estudos realizados e dos desenvolvimentos algorítmicos propostos e implementados.

V. Miranda

Vladimiro Miranda

Professor Associado da FEUP/DEEC

REFERÊNCIA DO PROJECTO 4.3/ 1.149/92/93

Parecer Técnico

Acompanhei os trabalhos de estágio realizados no INESC pelo aluno e apreciei o relatório final apresentado, considerando que aquele relatório reflecte o trabalho efectivamente realizado.

Sou de parecer que foram atingidos os objectivos inicialmente propostos no plano de trabalhos.

O Supervisor do Estágio (INESC)

Agradecimentos:

Gostaria de agradecer às pessoas que tornaram possível o meu trabalho, primeiro dando-me a oportunidade de o realizar aceitando o meu pedido de estágio, e segundo por me facilitarem a consulta de livros, paper's e trabalhos realizados anteriormente por outras pessoas. Agradeço também por me porem à disposição todo equipamento informático necessário ao meu projecto.

Durante o meu estágio fiquei bastante satisfeito com o ambiente de trabalho e com a solidariedade dos meus colegas não se escusando a me fornecerem conhecimentos que me fossem úteis, dando-me a visão de que todos os projectos têm sempre ajuda de alguém externo a eles.

Agradeço então ao Prof.Doutor Vladimiro Miranda, ao Eng. Nuno Fidalgo, ao pessoal de secretaria e aos meus colegas de trabalho com um principal destaque ao Celso Martins.

RELATÓRIO DO TRABALHO
SOBRE
REDES
NEURONAIS

AUTOR: Rui Manuel Santos Rodrigues Leite

ÍNDICE

Introdução..... pág. 3
Agradecimento..... " 4
Noções Teóricas..... " 4
Desenvolvimento de Software..... " 6
Definição das estruturas principais..... " 7
Rotinas de cálculo e treino..... " 7
Interpretador de Comandos..... " 8
Módulo de Input/Output..... " 9
Gestor de Máscaras..... " 10
Gestor de Menus tipo janela..... " 10
Menus Impressos..... " 11
Idem..... " 12
Idem..... " 13
Aplicação das redes à previsão..... " 14
Bibliografia..... " 15
Listagens

Introdução:

Foi-me proposto como trabalho de estágio que estudasse os conceitos teóricos das redes neuronais, desenvolvesse software para as calcular e treinar. Paralelamente ao desenvolvimento deste software tive que efectuar testes fazendos treinos de redes relativamente pequenas o que me permitiu não só corrigir os programas mas também confirmar resultados do âmbito teórico sobre redes neuronais já conhecidos nos paper's consultados. Desenvolvi um conjunto de rotinas básicas de cálculo, treino e de Input/Ouput e integrei-as em dois programas principais com filosofias de utilização diferentes, o primeiro ("redes") inclui um interpretador de comandos e é um sistema interactivo em que as acções são requeridas por intermédio da digitação dum comando, com uma determinada sintaxe. O segundo programa ("janelas") é um sistema que difere do primeiro no facto de que os comandos são invocados por opções de menus do tipo janela (pushdown windows), para isso tive que desenvolver software que fizesse a gestão das janelas. Este último programa foi mais apreciado que o primeiro, pela simplicidade de utilização. Desenvolvi ainda, ferramentas gráficas, para verificar duma maneira intuitiva a adaptação da rede ao conjunto de padrões, e também programas para efectuar divisões aleatórias dum ficheiro em dois ficheiros (numa dada proporção para treino e teste), baralhando os padrões.

Finalmente foi-me proposto uma aplicação (treino duma rede) das redes neuronais à previsão de consumos de gás de cidade em Lisboa usando para isso dados diários de consumos, temperaturas e previsões de temperaturas para o dia seguinte, dos anos de 1990 e 1991.

Agradecimento:

Agradeço aos meus colegas estudantes e ainda ao Eng. Nuno Fidalgo e ao Prof. Doutor Vladimiro Miranda o apoio que me deram no desenvolvimento do meu trabalho fornecendo-me paper's sobre o assunto e esclarecendo-me de quaisquer dúvidas que tivesse e agradeço ainda a oportunidade que me deram de tomar contacto, com o mundo apaixonante das redes neuronais.

Noções Teóricas:

No inicio do meu estágio tive que estudar alguns conceitos básicos de redes neuronais. Embora ao nível teórico tenha estudado vários tipos de redes, como as de hopfield, hamming, adaline (adaptative linear), kohonen (self-organizing), este estudo foi só por curiosidade e para me situar na história das redes neuronais. Foi com as redes do tipo feedforward multicamada que eu viria a trabalhar, estudar e implementar os processos de treino e cálculo.

Uma rede neuronal é no fundo uma função com determinados parâmetros (pesos) e que se pretende que esteja adaptada a um determinado conjunto de padrões, ou seja que alimentada com um determinado padrão de input dê uma boa aproximação dum padrão de output. A rede é constituida por células (unidades) ligadas por ramos (pesos), existindo células de input, células hidden (as mais importantes na aprendizagem) e células de output (que irão ser comparadas com os padrões de output). O processo de treino consiste na modificação dos pesos de forma que a rede esteja cada vez melhor adaptada ao conjunto de treino (pares de padrões de input e ouput) que no fundo são valores conhecidos da função que se pretende simular (por exemplo comportamento de consumos, ou mesmo funções binárias). O algoritmo de treino que estudei é chamado de backpropagation, e é um processo em que se faz a alteração dos pesos por forma a que diminua o êrro quadrático total (soma de todos os êrros quadráticos em todos padrões e todos outputs) designado por tss.

O algorítmo backpropagation é baseado no método de descida pelo gradiente e portanto cada alteração de pesos é feita na direcção em que o tss mais diminui. São usados no algorítmo de treino dois parâmetros, a taxa de aprendizagem que é uma constante que conforme o nome indica permite andar mais depressa em cada iteração (não pode ser muito elevada por causa de oscilações e subidas bruscas do erro), e o "momento" que serve para suavizar o movimento na superfície de erro permitindo assim termos taxas de aprendizagem mais elevadas. O algorítmo de treino por vezes é bastante lento visto que para taxas elevadas a convergência é lenta devido à oscilação, e para taxas baixas as modificações são pequenas logo o processo é também lento, o ideal seria arranjar um valor conveniente para a taxa, mas esse valor ideal pode modar também durante o treino. Para resolver este problema tive que implementar o chamado Adaptative backpropagation (A.D.P.) que consistem em ter associado a cada peso uma taxa de aprendizagem e fazer modificações nessas taxas de forma a aumentar a velocidade de convergência (aumentar a taxa se não houve mudanças de sinal nas derivadas parciais de uma iteração para a outra, e diminuir no caso contrário).

As redes neuronais têm a capacidade de se aperceberem de certas regularidades no conjunto de padrões. Quando se pretende usar uma rede neuronal para simular uma função não "completa" (não se conhecem todos os pontos), por exemplo o comportamento de consumos, é comum separar o conjunto de padrões em duas partes, uma parte de teste e outra parte de treino. No processo de treino a rede vai adaptar-se à parte de treino e devido às regularidades do conjunto de padrões, ela vai adaptar-se também à parte de teste, a isto chama-se generalização. A partir de determinado momento o erro no conjunto de teste começa a aumentar significativamente, ou seja, a rede começa a perder capacidade de generalização, isto porque "decora" os padrões de treino. Nesta situação interrompe-se o treino.

Foi ainda estudado por mim o capítulo da interpretação de redes que se baseia na análize de redes treinadas (ou semi-treinadas).

Desenvolvimento de Software:

Como já referi, desenvolvi dois programas principais para treino, que são o programa "redes" e o programa "janelas".

O primeiro é constituido pelos ficheiros:

redes.c	- Constituído só pela função principal (main).
redes.h	- Contém todas as definições de variáveis e estruturas globais.
red_io.c	- Contém um analizador léxico, rotinas para leitura e gravação de ficheiros (de descrição de redes, pesos, padrões) utilizando quando necessário a alocação dinâmica de memória para carregar informação, um interpretador de comandos e respectiva recuperação de erros, e ainda rotinas de apresentação simples de resultados de output's e estado duma unidade.
red_cal.c	- Contém rotinas para cálculo duma rede usando um dado Input, para calcular erros e derivadas usando um ou todos os padrões; rotinas para fazer mudanças de pesos, para adaptar taxas; rotinas fazer treinos por padrão ou por época.

O segundo programa é constituido pelos ficheiros:

janelas.c	- Contém a função principal (main), um gestor de menus que faz a chamada de outras rotinas, isto é, executa comandos e controla todos os Output's e Input's de forma a aparecerem em caixas junto com o pedido de introdução de dados.
jan.h	- Contém todas as definições de variáveis e estruturas globais, assim como uma tabela de mensagens de erro, e uma tabela de "tokens".
jan_io.c	- Contém as mesmas rotinas do módulo de Input/ Ouput do programa "redes", só que neste caso não inclui o interpretador de comandos nem as rotinas de recuperação de erros.
jan_cal.c	- Contém essencialmente o mesmo que o módulo de cálculo do programa "redes", com a possibilidade adicional de durante o treino chamar-se uma rotina de impressão de máscara permitindo observar melhor os pormenores do treino (por exemplo pesos).
lista.c	- Contém rotinas para receber Input's controlados dentro duma janela com possiblidade de scroll, podendo também ser usada para apresentar Output's.

jan_mas.c	- Contém rotinas para ler ficheiros de descrição de máscaras (templates) interpretar os campos para calcular os endereços (rotinas que realizam tanto trabalho sintáctico como semântico), rotinas para imprimir um dado campo e mais geralmente imprimir toda a máscara (texto e campos).

Definição das estruturas principais:

Defini as estruturas para guardar a informação sobre uma rede, i.e., unidades, ramos, padrões, como sendo vectores de estruturas mais pequenas que designam um só elemento, por exemplo para cada unidade terei reservado 3 double, 2 unsigned int e 1 char. Logo que se conheça o número de unidades, o número de inputs e o número de ouputs reserva-se espaço para essas unidades (só as hidden e outputs).

Rotinas de Cálculo e Treino:

O módulo de cálculo é constituido por:

testar()	- Calcula os êrros relativos (em módulo) máximos, minimos e médios no conjunto de padrões de teste.
copia_output()	- Faz uma cópia das células de output para um vector de double's.
treina_epoca()	- Efectua n iterações do A.B.P.
calcnet()	- Calcula a rede usando como input um parâmetro que é um apontador para double's.
activation()	- Calcula a activação duma unidade sendo dada a soma pesada das entradas com os pesos respectivos.
calc_erro_deriv()	- Calcula o êrro para um padrão e acumula derivadas no caso de estarmos a treinar por época.
deriv_zero()	- Coloca todas as derivadas a zero.
aleatorio()	- Retorna um nº aleatório real de módulo menor que range (variável global).
atribui_pesos()	- Atribui aos pesos nºs aleatórios e efectua outras inicializações nos ramos.
guarda_pesos_deriv()	- Faz uma cópia de cada peso e derivada para campos auxiliares existentes em cada ramo.

muda_pesos()	- Faz a modificação dos pesos usando as derivadas, as taxas e os filtros combinados com o "momento".
divide taxas()	- Faz uma redução geral das taxas de aprendizagem porque ocorreu um aumento do tss.
adapta_taxas()	- Faz uma modificação das taxas que é aumentar (multiplicar porfcresci) se o sinal da derivada se manteve caso contrário diminui-se (multiplica-se por fdiminu).
iteracao()	- Efectua uma iteração do A.B.P. não deixando o erro aumentar.
esquece_passado()	- Coloca todos os filtros a zero anulando o efeito de inércia (lanço) do movimento na superfície de erro (usa-se quando o erro aumenta).
calcula_filtros()	- Faz o cálculo das variações das derivadas parciais (calcula filtros).
calc_erro_deriv_epoca()	- Faz o mesmo que calc_erro_deriv() só que neste caso calcula o erro para todos os padrões e acumula todas as derivadas em relação a cada padrão.
calcula_erro()	- Faz Simplesmente o cálculo do tss sem calcular derivadas.

Interpretador de Comandos: Como já referi a constituição básica do interpretador de comandos, vou simplesmente enumerar os comandos, a sua sintaxe e semântica. Como comandos de leitura e gravação de dados temos: para ler redes **readnet(<fich>)**; para ler padrões **readpat(<fich>)**; temos **savepesos(<fich>)** para gravar pesos e **savenet(<fich>)** para gravar redes. Temos um comando chamado **set** que faz atribuições a variáveis de sistema (do meu programa) que são: bin que define se a rede é binária; speed que é o valor atribuido inicialmente a todas as taxas; moment que é a variável "momento"; nepocas indica o nº de iterações a efectuar; ntepocas é um contador de iterações; range define a gama de atribuição aleatória dos pesos; epoca é o nº da iteração presente; critério é a condição de paragem que sucede quando tss for menor que ele; fcresci é o factor de crescimento das taxas; fdiminu é o factor de diminuição; adapt indica se se pretende usar o B.P. ou o A.B.P. como algorítmo de treino.

Tenho um comando que faz a apresentação de todas as variáveis atrás referidas (o comando Sets), um que acciona a rotina atribui-pesos() e faz com que o sistema considere que está na primeira iteração que é o comando Rand, um comando de saída do programa que é o Exit, o comando Cls apaga o ecrân, o comando State mostra todas as informações sobre uma unidade. Para testar um padrão binário uso o comando teste, e para obter a percentagem de padrões correctos uso o comando Percent. Para calcular a rede para um determinado input escrevo calc(<num>,<num>,..,<num>) e para treinar a rede uso o comando Run que tem duas variantes:o Run by epoca que faz o treino por época e o Run by pattern que faz o treino por padrão.

Módulo de Input/Output:

Sobre este módulo não vou desenvolver em demasiado a descrição das rotinas, visto que as descrições anteriores parecem-me suficientes, vou sim descrever a sintaxe a que obedecem os ficheiros acedidos por este módulo.

Os ficheiros de pesos simplesmente contém valores numéricos.

Os Ficheiros de padrões apresentam no início npatterns=<num> (<num> que é o numero de padrões presentes); depois deste cabeçalho seguem-se os valores numéricos correspondentes aos pares de padrões de input e output, por esta ordem. Os ficheiros de descrição de redes são assim constituidos:

nunidades=<num>
ninputs=<num>
noutput=<num>
{ nlig(<num1>,<num2>), nlig(<num1>,<num2>,<num3>), que significam respectivamente que a unidade <num1> tem <num2> ligações e no segundo caso que as unidades entre <num1> e <num2> têm <num3> ligações }; este bloco pode-se repetir.
{ lig(<num1>,<num2>), que significa que a unidade <num1> está ligada à unidade <num2>;
lig(<num1>,<num2>,<num3>), que significa que as unidades entre <num1> e <num2> estão ligadas à unidade <num3>;
lig(<num1>,<num2>,<num3>,<num4>), que significa que os dois blocos de unidades entre <num1> e <num2> e entre <num3> e <num4> respectivamente, estão ligados exaustivamente}; este bloco pode-se repetir.
{ activ(<num1>,<ident>) significa que a função de activação da unidade <num1> é <ident>, tenho também o caso activ(<num1>, <num2>,<ident>) que me parece inútil explicar }; este bloco pode-se repetir.

Gestor de máscaras:

O gestor de máscaras é um conjunto de rotinas que possibilitam a visualização de campos da rede durante o processo de treino, podendo assim melhor se verificar a convergência. Vou nesta secção descrever como são constituidos os ficheiros de máscaras visto que uma descrição das rotinas seria muito enfastidiosa.

No início do ficheiro (primeiras 20 linhas) o utilizador define que texto aparecerá no ecrân. Usando o caracter #, define-se também a localização e o comprimento dos campos. Posso apresentar blocos de texto em **inverse-video** basta que para isso coloque um caracter % no início e no final do bloco. Depois das 20 primeiras linhas temos que indicar o nome das variáveis, por ordem de leitura no ecrân, e o modo de as apresentar no mesmo. Temos que apresentar essa informação no seguinte formato:

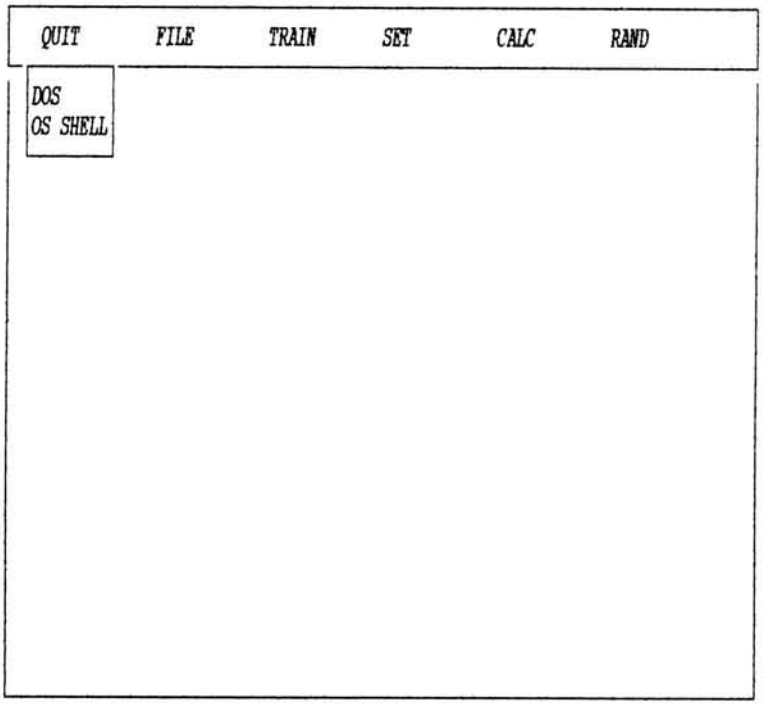
(<tipo>,<mult>,<bloco_da_variável>)

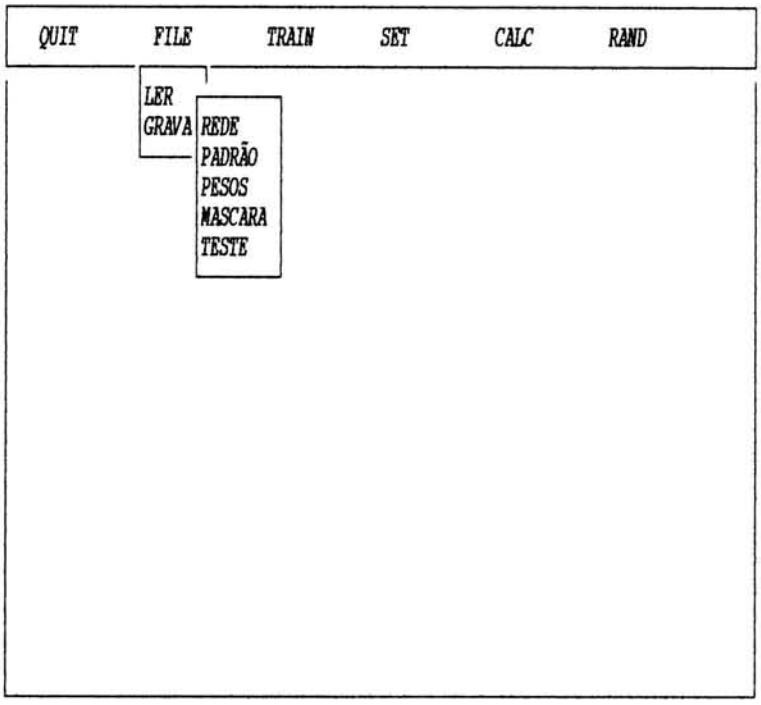
Temos **<tipo>**=1 se for uma variável inteira; 0 se for de vírgula flutuante com apresentação standard; 2 se tivermos uma variável de vírgula flutuante apresentados sem vírgula (parte inteira) previamente multiplicada por **<mult>** e se o nº ocupar mais do que as casas definidas (nº de #'s) então imprimir-se-ão *'s no seu lugar. Temos ainda que se o nº for negativo então aparecerá em **inverse-video**.

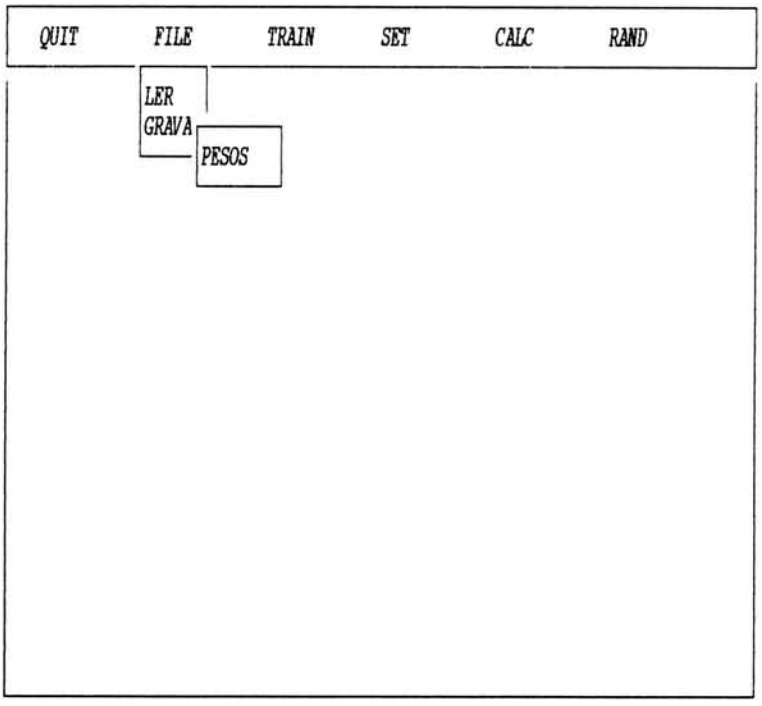
O **<bloco_da_variável>** pode ser [epoca], [tss], [UFO,<nl>], [UDFO,<nl>], [UDELTA,<nl>], [WEI,<nl>,<n2>], [WDER,<nl>,<n2>].

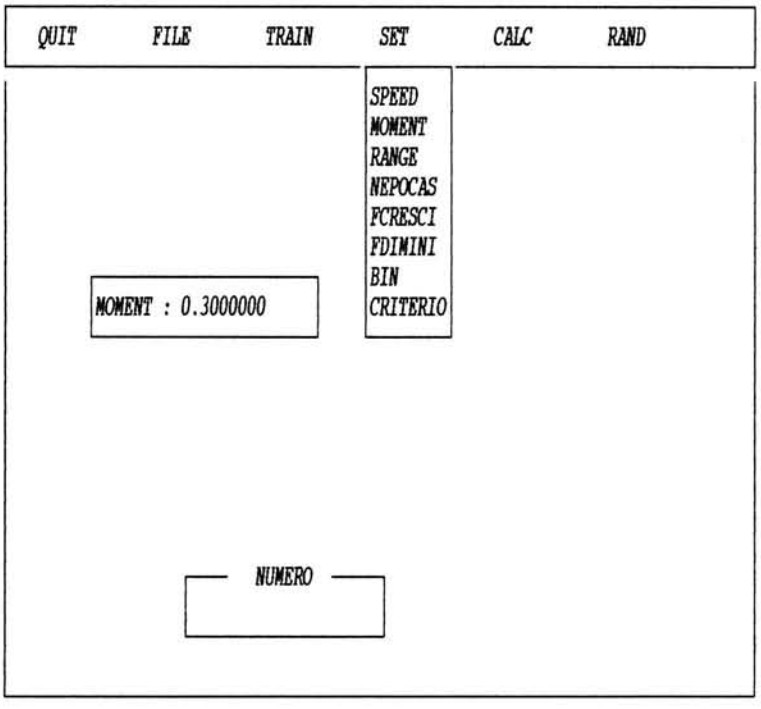
Gestor de Menus tipo janela:

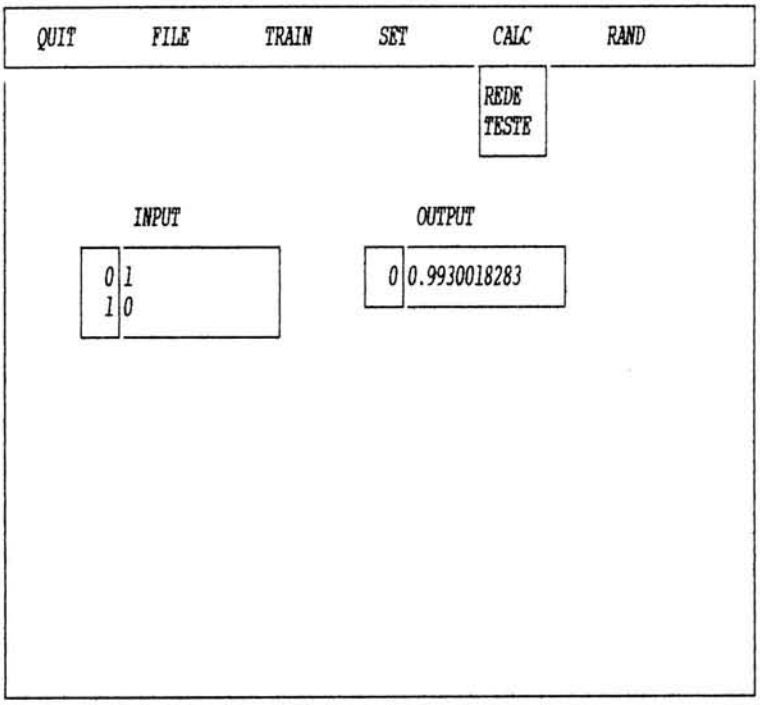
Para utilizar estes menus bastar usar as teclas do cursor para nos deslocarmos pelas opções e utilizar <enter> para executar um comando ou chamar uma sub-janela. Não vou explicar como construí este gestor de menus visto que a listagem parece-me bastante clara. Em páginas seguintes verá os menus impressos.

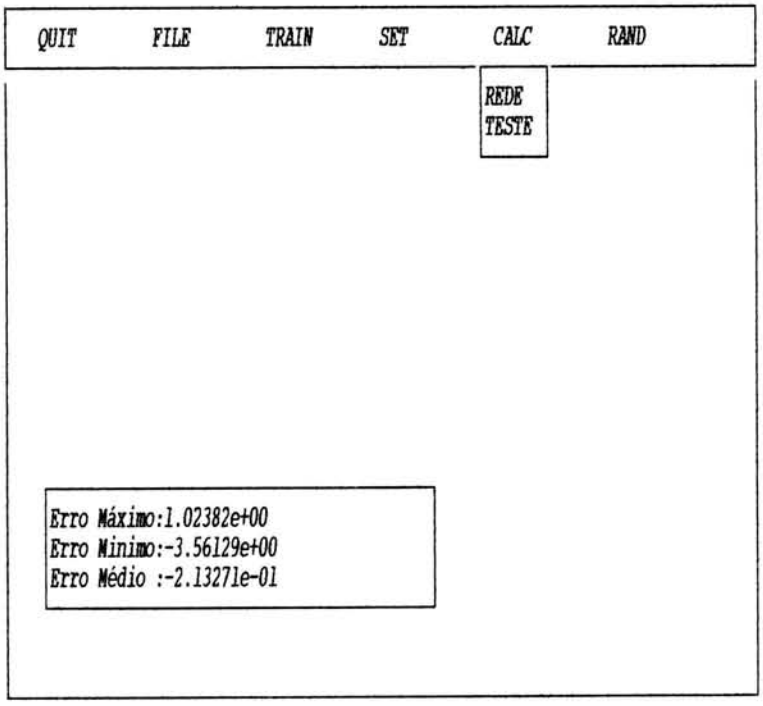

QUIT
FILE
TRAIN
SET
CALC
RAND
DOS
OS SHELL


QUIT
FILE
TRAIN
SET
CALC
RAND
LER
GRAVA
REDE
PADRÃO
PESOS
MASCARA
TESTE


QUIT
FILE
TRAIN
SET
CALC
RAND
LER
GRAVA
PESOS


QUIT
FILE
TRAIN
SET
CALC
RAND
SPEED
MOMENT
RANGE
NEPOCAS
FCRESCI
FDIMINI
BIN
CRITERIO
MOMENT : 0.3000000
NUMERO


QUIT
FILE
TRAIN
SET
CALC
RAND
REDE
TESTE
INPUT
OUTPUT
0 1
1 0
0 0.9930018283


QUIT
FILE
TRAIN
SET
CALC
RAND
REDE
TESTE
Erro Máximo:1.02382e+00
Erro Mínimo:-3.56129e+00
Erro Médio :-2.13271e-01

Aplicação das redes à previsão:

Um outro ponto do meu trabalho era construir uma rede neuronal que fizesse a previsão de consumos de gás de cidade (em Lisboa), para isso utilizei alguns ficheiros que me foram fornecidos, que continham dados sobre consumos, temperaturas e previsões de temperaturas para o dia seguinte, dados esses referentes aos anos de 1990 e 1991. Projectei uma rede neuronal constituida por quatro entradas, 5 células **hidden** e uma de output (correspondente à previsão). Os quatros inputs da rede eram treinados (e mais tarde testados) com a temperatura da antevéspera, consumo da antevéspera, dia da semana (0=Domingo,1=Segunda, etc) e a temperatura prevista para o dia em que queremos fazer uma previsão de consumos. Como método de eficiência usei a comparação com o erro no chamado previsor trivial, usei como previsor trivial o consumo da véspera. Gostaria de concluir dizendo que houve melhores resultados que os meus usando outros métodos mas os métodos faziam a separação das previsões em estações do ano, feriados, ou seja eram previsões mais específicas, diferentes das que eu poderia obter com um treino geral (dados dum ano inteiro), no entanto consegui melhores resultados quando retirei do conjunto de treino os feriados e dias afectados (pontes, férias, etc).

Resultados para 1991 sem feriados:

Previsão trivial: Err.rel.med. = 0.059
Err.rel.max. = 0.25
Err.rel.min. = 8.5 e -5

Previsão usando como conjunto de treino 1990 sem feriados e testando em 1991 sem feriados (para comparar com a hip.trivial):
Err.rel.med. = 0.024
Err.rel.max. = 0.098
Err.rel.min. = 6.8 e -5

Resultados para 1991 com feriados:

Previsão trivial: Err.rel.med. = 0.0798
Err.rel.max. = 0.38
Err.rel.min. = 1.24 e -4

Previsão usando como conjunto de treino 1990 com feriados e testando em 1991 com feriados (para comparar com a hip.trivial):
Err.rel.med. = 0.036
Err.rel.max. = 0.231
Err.rel.min. = 4.7 e -5

Bibliografia:

- Speeding up backprogation; Fernando M.Silva - Luis B. Almeida, In R. Eckmiller (ed.), Advanced Neural Computers, North-Holland,1990.

- An introdution to Computing with Neural Nets; Richard P. Lippman, In IEEE Magazine April 1987.

- Backpropagation in feedforward and recurent networks, Luis B. Almeida, Submited to an IEEE Computer Society Press Book, to be edited by Bruce Shriver.

- Neural Networks Pc Tools (a pratical guide) ; Russel C. Eberhart, Roy W. Dobbins, Academic Press, Inc - Harcourt Brace Jovanovich, Publishers.

```
/* Programa principal 'redes' */

#include<stdio.h>
#include<alloc.h>
#include<string.h>
#include<ctype.h>
#include<stdlib.h>
#include "redes.h"

main()
{
 int i;
 ptr=stdin;
 cls();
 randomize();
 lh=fgetc(ptr);
 token=lex();
 command();
}
```

```
/*  REDES.H  */

#include<stdio.h>
#define FIM         1000
#define NUMBER      1001
#define STRING      1002
#define SIGMOID     1
#define LINEAR      2
#define READNET     1005
#define PARESQ      1006
#define PARDIR      1007
#define VIRGULA     1008
#define IGUAL       1009
#define NUNIDADES   1010
#define NINPUTS     1011
#define NOUTPUTS    1012
#define NLIG        1013
#define LIG         1014
#define ACTIV       1016
#define READPAT     1017
#define SAVENET     1018
#define SET         1019
#define SPEED       1020
#define MOMENT      1021
#define RUN         1022
#define CALC        1023
#define ERROR       1024
#define EXIT        1025
#define PONTO       1027
#define NPATTERNS   1028
#define SETS        1029
#define CLS         1030
#define NEPOCAS     1031
#define BY          1032
#define PATTERN     1033
#define EPOCA       1034
#define RANGE       1035
#define CRITERIO    1036
#define RAND        1037
#define LIVRE       65535u
#define BIAS        65534u

struct {
        char *com;
        unsigned int chave;
} tab[]={ "SIGMOID",SIGMOID,"LINEAR",LINEAR,"READNET",READNET,"NUNIDADES",NUNIDADES,"NINPUTS",NINPUTS,
          "NOUTPUTS",NOUTPUTS,"NLIG",NLIG,"LIG",LIG,"ACTIV",ACTIV,"READPAT",READPAT,"SAVENET",SAVENET,
          "SET",SET,"SPEED",SPEED,"MOMENT",MOMENT,"RUN",RUN,"CALC",CALC,"EXIT",EXIT,"WEIGHT",WEIGHT,
          "NPATTERNS",NPATTERNS,"SETS",SETS,"NEPOCAS",NEPOCAS,"CLS",CLS,"BY",BY,"PATTERN",PATTERN,
          "EPOCA",EPOCA,"CRITERIO",CRITERIO,"RANGE",RANGE,"RAND",RAND,"STATE",STATE,NULL,0};
typedef struct {
                float long fo,dfo,delta;
                unsigned int nlig,pos;
                char activ;
        } (far *unit);
typedef struct {
                float long peso,dpeso,deriv;
                unsigned int ligacao;
                char modifica;
        } (far *ramos);
FILE *ptr;
unsigned int nepocas,token,livre,nl,nunidades,ninputs,noutputs,npadroes,lh;
float long toknum,moment,speed,criterio,range=.3,(far *INPUTS),(far *PATTERNS),
          (far *OUTERROS),aleatorio();
char tokstr[100],modo;
unit UNI;
ramos LIGA;
```

```
/* redes_io.c */

cls()
{
 clrscr();
 printf("SISTEMA DE CALCULO E TREINO\n");
 printf("DE REDES NEURONAIS \n");
 printf("\n @ 1992 /1993 \n\n");
}

erro(i) unsigned int i;
 {
  mensagem(i);
  while(!com(token)) match(token);
 }

com(i) unsigned int i;
{
 switch(i)
  {
   case READNET:
   case TESTE:
   case PERCENT:
   case READPAT:
   case SAVENET:
   case SET:
   case STATE:
   case RAND:
   case SETS:
   case RUN:
   case CALC:
   case CLS:
   case EXIT:return 1;
   default:return 0;
  }
}

mensagem(i) unsigned int i;
{
 printf("\n[ERRO]: ");
 switch(i)
  {
   case 1:printf("Falta {nunidades} ");break;
   case 2:printf("Falta {=} ");break;
   case 3:printf("Falta {Numero} ");break;
   case 4:printf("Falta {ninputs} ");break;
   case 5:printf("Falta {noutputs} ");break;
   case 6:printf("Não tem memória ");break;
   case 7:printf("Falta {.} ");break;
   case 8:printf("Falta {NPATTERNS} ");break;
   case 9:printf("Npadrões <=0 ");break;
   case 10:printf("Falta {(} ");break;
   case 11:printf("Unidade inválida ");break;
   case 12:printf("Ficheiro não deveria conter mais nada ");break;
   case 13:printf("Falta unidade ou é inválida ");break;
   case 14:printf("Falta {)} ");break;
   case 15:printf("Ficheiro não existe ");break;
   case 16:printf("Falta {String} ");break;
   case 17:printf("Falta {,} ");break;
   case 19:printf("Rede  mal construida ");break;
   case 20:printf("Ligações > que nlig ");break;
   case 21:printf("Não existe essa ligação ");break;
   case 22:printf("Esperava uma função de activação ");break;
   case 23:printf("Comando desconhecido ");break;
   case 24:printf("Esperava uma variavel para fazer o set ");break;
   case 25:printf("nunidades<=0 ");break;
   case 26:printf("por {epoca} ou {pat} ");break;
   case 27:printf("ninputs<=0 ");break;
```

```
    case 28:printf("noutputs<=0 ");break;
    case 29:printf("nunidades < ninputs+noutputs ");break;
    case 30:printf("Não consigo escrever o ficheiro");break;
    default:printf("[%d]",(unsigned)i);
   }
   printf("\n");
  }

  verify()
  {
   if (nunidades<=0) return 1;
   return 0;
  }

  readpat()
  {
   unsigned int i,j,w;
   double *t;
   first_iter=1;
   if (!match(NPATTERNS)) return 8;
   if (!match(IGUAL)) return 2;
   if (token!=NUMBER) return 3;
   npadroes=toknum;
   match(NUMBER);
   if (nunidades<=0 || ninputs+noutputs<=0) return 18;
   if (npadroes<=0) return 9;
   w=(ninputs+noutputs);
   free(PATTERNS);
   PATTERNS=(double *)calloc(npadroes,w*sizeof(double));
   t=PATTERNS;
   if (PATTERNS==NULL) return 6;
   for(i=0;i<npadroes;i++)
    for(j=0;j<w;j++)  {
                      if (token!=NUMBER) return 3;
                      *t=toknum;
                      match(NUMBER);
                      t++;}
   if (!match(FIM)) return 12;
   return 0;
  }

  liberta()
   {
    unsigned int i;
    if (nunidades==0) return;
    free(UNI);
    free(LIGA);
    free(PATTERNS);
    free(INPUTS);
    free(OUTERROS);
   }

  lex()
  {char *t;unsigned int i;
   while (isspace(lh)) lh=fgetc(ptr);
   if (lh==';') { while(lh!=(unsigned)EOF && lh!=10 && lh!=13) lh=fgetc(ptr);
                  return lex();
                }
   if (lh==(unsigned)EOF) return FIM;
   switch(lh)
    {
     case '(':lh=fgetc(ptr);return PARESQ;
     case ')':lh=fgetc(ptr);return PARDIR;
     case ',':lh=fgetc(ptr);return VIRGULA;
     case '=':lh=fgetc(ptr);return IGUAL;
     case '.':lh=fgetc(ptr);if (!isdigit(lh)) return PONTO;
              ungetc(lh,ptr);lh='.';
    }
```

```
  if (isalpha(lh)) {t=tokstr;
                    for(;isalpha(lh);lh=fgetc(ptr),t++) *t=toupper(lh);
                    *t=0;
                    if (!strcmp("BIAS",tokstr)) {toknum=BIAS;return NUMBER;}
                    for(i=0;tab[i].com!=NULL;i++)
                      if (!strcmp(tab[i].com,tokstr)) return tab[i].chave;
                    return STRING;
                   }
  ungetc(lh,ptr);
  if (fscanf(ptr,"%le",&toknum)) {lh=fgetc(ptr);return NUMBER;}
  while(!isalnum(lh) && lh!=(unsigned)EOF && !isspace(lh)
        && lh!='(' && lh!=')' && lh!='.' && lh!=',' && lh!='=')
              lh=fgetc(ptr);
  return ERROR;
 }

 match(tok) unsigned int tok;
 {
  if (tok!=token) return 0;
  token=lex();
  return 41;
 }

 readnet()
 {
  unit t;
  unsigned int w,i,c;
  if (!match(NUNIDADES)) return 1;
  if (!match(IGUAL)) return 2;
  if (token!=NUMBER) return 3;
  nunidades=toknum;match(NUMBER);
  if (nunidades<=0) return 25;
  if (!match(NINPUTS)) return 4;
  if (!match(IGUAL)) return 2;
  if (token!=NUMBER) return 3;
  ninputs=toknum;match(NUMBER);
  if (ninputs<=0) return 27;
  if (!match(NOUTPUTS)) return 5;
  if (!match(IGUAL)) return 2;
  if (token!=NUMBER) return 3;
  noutputs=toknum;match(NUMBER);
  if (noutputs<=0) return 28;
  if (nunidades<(ninputs+noutputs)) return 29;
  w=nunidades-ninputs;nl=0;
  UNI=(unit)calloc(w,sizeof(*UNI));
  if (UNI==NULL) {nunidades=0;return 6;}
  for(t=UNI,i=0;i<w;i++,t++)
   {
    t->nlig=0;t->pos=-1;t->activ=SIGMOID;
   }
  while(match(NLIG)) {c=nlig();if (c) {free(UNI);nunidades=0;return c;}}
  if (nl==0) {free(UNI);nunidades=0;if (!match(FIM)) return 12;return 0;}
  LIGA=(ramos)calloc(nl,sizeof(*LIGA));
  if (LIGA==NULL && nl!=0) {nunidades=0;free(UNI);return 6;}
  for(i=0;i<nl;i++) LIGA[i].ligacao=LIVRE;
  while(token==LIG || token==ACTIV)
   {
    switch(token)
     {
      case LIG:match(LIG);c=lig();if (c) {free(UNI);free(LIGA);nunidades=0;return c;}
               break;
      case ACTIV:match(ACTIV);c=activ();if (c) {free(UNI);free(LIGA);nunidades=0;return c;}
     }
   }
  if (!match(FIM)) {free(UNI);free(LIGA);nunidades=0;return 12;}
  return 0;
 }
```

```
nlig()
{
 unsigned int n1,n2,n3,i,c;
 if (!match(PARESQ)) return 10;
 if (token!=NUMBER) return 3;
 n1=toknum;match(NUMBER);
 if (!match(VIRGULA)) return 17;
 if (token!=NUMBER) return 3;
 n2=toknum;match(NUMBER);
 if (match(PARDIR)) {
                     c=nlig_atr(n1,n2);
                     if (c) return c;
                     return 0;
                    }
 if (!match(VIRGULA)) return 17;
 if (token!=NUMBER) return 3;
 n3=toknum;match(NUMBER);
 if (!match(PARDIR)) return 14;
 for(i=n1;i<=n2;i++) {c=nlig_atr(i,n3);if (c) return c;}
 return 0;
}

nlig_atr(a,b) unsigned int a,b;
{
 if (a<ninputs || a>nunidades) return 11;
 UNI[a-ninputs].pos=n1;
 UNI[a-ninputs].nlig=b;
 n1+=b;
 return 0;
}

lig()
{
 unsigned int n1,n2,n3,n4,i,j,c;
 if (!match(PARESQ)) return 10;
 if (token!=NUMBER) return 3;
 n1=toknum;match(NUMBER);
 if (!match(VIRGULA)) return 17;
 if (token!=NUMBER) return 3;
 n2=toknum;match(NUMBER);
 if (match(PARDIR)) {
                     c=lig_atr(n1,n2);
                     if (c) return c;
                     return 0;
                    }
 if (!match(VIRGULA)) return 17;
 if (token!=NUMBER) return 3;
 n3=toknum;match(NUMBER);
 if (match(PARDIR)) {
                     for(i=n1;i<=n2;i++) { c=lig_atr(i,n3);
                                           if (c) return c;}
                     return 0;
                    }
 if (!match(VIRGULA)) return 17;
 if (token!=NUMBER) return 3;
 n4=toknum;match(NUMBER);
 if (!match(PARDIR)) return 14;
 for(i=n1;i<=n2;i++)
  for(j=n3;j<=n4;j++)
    {c=lig_atr(i,j);if (c) return c;}
 return 0;
}

lig_atr(a,b) unsigned int a,b;
{
 unsigned int p,n,i;
 if (b<ninputs || b>nunidades) return 11;
 p=UNI[b-ninputs].pos;
```

```
 n=UNI[b-ninputs].nlig;
 for(i=0;i<n;i++)
   if (LIGA[p+i].ligacao==LIVRE)
     {
      LIGA[p+i].ligacao=a;
      return 0;
     }
 return 20;
}

activ()
{
 unsigned int n1,n2,i;
 if (!match(PARESQ)) return 10;
 if (token!=NUMBER) return 3;
 n1=toknum;match(NUMBER);
 if (!match(VIRGULA)) return 17;
 if (token!=NUMBER) {
                    if (token!=SIGMOID && token!=SIGCENT && token!=LINEAR) return 22;
                    if (n1<ninputs || n1>nunidades) return 11;
                    UNI[n1-ninputs].activ=token;
                    match(token);
                    if (!match(PARDIR)) return 14;
                    return 0;
         }
 n2=toknum;match(NUMBER);
 if (!match(VIRGULA)) return 17;
 if (token!=SIGMOID && token!=LINEAR && token!=SIGCENT) return 22;
 for(i=n1;i<=n2;i++)
  {
    if (i<ninputs || i>nunidades) return 11;
    UNI[i-ninputs].activ=token;
  }
 match(token);
 if (!match(PARDIR)) return 14;
 return 0;
}

command()
{
 unsigned int token_a,lh_a,c,e,i;
 char tokstr_a[100],nome[100];
 double toknum_a,*t;
 ramos r;
 while(1)
 {
 switch(token)
  {
   case CLS    :match(CLS);cls();break;
   case STATE  :match(STATE);if (!match(PARESQ)) {erro(10);break;}
                if (token!=NUMBER) {erro(13);break;}
                e=toknum;
                if (e<ninputs || e>=nunidades) {erro(13);break;}
                match(NUMBER);
                if(!match(PARDIR)) {erro(14);break;}
                clrscr();
                printf("Estado da unidade %d\n",e);
                printf("ACTIVACAO :% .15f \n",UNI[e-ninputs].fo);
                printf("DERIVADA AO INPUT TOTAL : %.15f \n",UNI[e-ninputs].dfo);
                printf("DELTA : %.15f \n",UNI[e-ninputs].delta);
                print_act(UNI[e-ninputs].activ);
                c=UNI[e-ninputs].nlig;
                r=&LIGA[UNI[e-ninputs]
```

```
                for(i=0;i<c;i++,r++)
                 {
                  printf("peso[%.15f] filtro[%.15f]\nderiv[%.15f]  \n",
                      r->peso,r->filtro,r->deriv);
                 }
                printf("\n");
                break;
 case SETS    :match(SETS);
                printf("\nBIN=%d",bin);
                printf("\nSPEED=%.15f",speed);
                printf("\nMOMENT=%.15f",moment);
                printf("\nEPOCA=%u",epoca);
                printf("\nNEPOCAS=%u",nepocas);
                printf("\nNTEPOCAS=%u",ntepocas);
                printf("\nFCRESCI=%.15f",fcresci);
                printf("\nFDIMINU=%.15f",fdiminu);
                printf("\nRANGE=%.15f",range);
                printf("\nADAPT=%d",adapt);
                printf("\nCRITERIO=%.15f\n",criterio);
                break;
 case READNET:match(READNET);
                if (!match(PARESQ)) {erro(10);break;}
                if (token!=STRING) {erro(16);break;}
                strcpy(nome,tokstr);
                strcat(nome,".net");
                match(STRING);
                if (!match(PARDIR)) {erro(14);break;}
                ptr=fopen(nome,"r");
                if (ptr==NULL) {ptr=stdin;erro(15);break;}
                liberta();
                lh_a=lh;token_a=token;toknum_a=toknum;strcpy(tokstr_a,tokstr);
                lh=fgetc(ptr);token=lex();
                c=readnet();fclose(ptr);
                ptr=stdin;lh=lh_a;token=token_a;toknum=toknum_a;
                strcpy(tokstr,tokstr_a);
                if (c) erro(c);
                INPUTS=(double *)calloc(ninputs,sizeof(*INPUTS));
                OUTERROS=(double *)calloc(noutputs,sizeof(*OUTERROS));
                if (INPUTS==NULL || OUTERROS==NULL)
                   {free(UNI);free(LIGA);nunidades=0;erro(6);}
                break;
 case READPAT:match(READPAT);
                ntepocas=0;
                if(!match(PARESQ)) {erro(1);break;}
                if (token!=STRING) {erro(16);break;}
                strcpy(nome,tokstr);
                strcat(nome,".pat");
                match(STRING);
                if (!match(PARDIR)) {erro(14);break;}
                ptr=fopen(nome,"r");
                if (ptr==NULL) {ptr=stdin;erro(15);break;}
                lh_a=lh;token_a=token;toknum_a=toknum;strcpy(tokstr_a,tokstr);
                lh=fgetc(ptr);token=lex();
                c=readpat();fclose(ptr);
                ptr=stdin;lh=lh_a;token=token_a;toknum=toknum_a;
                strcpy(tokstr,tokstr_a);
                if (c) erro(c);
                break;
 case RAND    :match(RAND);
                ntepocas=0;
                atribui_pesos();
                first_iter=1;
                break;
 case CALC    :match(CALC);
                e=0;
                if (!match(PARESQ)) {erro(10);break;}
                for(i=0,t=INPUTS;i<ninputs;i++,t++)
                 {
                   if (token!=NUMBER) {e=1;erro(3);break;}
```

```
                *t=toknum;
                match(NUMBER);
                if (i==ninputs-1 ) {if (!match(PARDIR)) {e=1;erro(14);break;}
                                    continue;}
                    else
                      if (!match(VIRGULA)) {e=1;erro(17);break;}
               }
              if (e) break;
              if (verify()) {erro(19);break;}
              calcnet(INPUTS);
              printoutput();
              break;
case TESTE  :match(TESTE);
              if (!match(PARESQ)) {erro(10);break;}
              if (token!=NUMBER) {erro(3);break;}
              c=toknum;match(NUMBER);
              if (!match(PARDIR))  {erro(14);break;}
              if (testa_padrao(c)) erro(100);
              break;
case PERCENT:match(PERCENT);
              calc_percent();
              break;
case SET    :match(SET);
              token_a=token;match(token);
              if (!match(IGUAL)) {erro(2);break;}
              if (token!=NUMBER) {erro(3);break;}
              switch(token_a)
               {
                case BIN:bin=toknum;break;
                case SPEED:speed=toknum;break;
                case MOMENT:moment=toknum;break;
                case NEPOCAS:nepocas=toknum;break;
                case NTEPOCAS:ntepocas=toknum;break;
                case RANGE:range=toknum;break;
                case EPOCA:epoca=toknum;break;
                case CRITERIO:criterio=toknum;break;
                case FCRESCI:fcresci=toknum;break;
                case FDIMINU:fdiminu=toknum;break;
                case ADAPT:adapt=toknum;break;
                default:erro(24);
               }
              match(NUMBER);
              break;
case RUN    :match(RUN);
              if (!match(BY)) {erro(26);break;}
              if (token!=EPOCA && token!=PATTERN) {erro(26);break;}
              switch(token)
               {
                case EPOCA:match(EPOCA);treina_epoca();break;
                case PATTERN:match(PATTERN);treina_pat();break;
               }
              break;
case SAVEPESOS:match(SAVEPESOS);
              if (!match(PARESQ)) {erro(10);break;}
              if (token!=STRING) {erro(16);break;}
              strcpy(nome,tokstr);
              match(STRING);
              if (!match(PARDIR)) {erro(14);break;}
              ptr=fopen(nome,"w");
              if (ptr==NULL) {ptr=stdin;erro(30);break;}
              c=savepeso();
              fclose(ptr);
              ptr=stdin;
              if (c) {erro(30);break;}
              break;
case READPESOS:match(READPESOS);
              if (!match(PARESQ)) {erro(10);break;}
              if (token!=STRING) {erro(16);break;}
```

```
                strcpy(nome,tokstr);
                match(STRING);
                if (!match(PARDIR)) {erro(14);break;}
                ptr=fopen(nome,"r");
                if (ptr==NULL) {ptr=stdin;erro(15);break;}
                ntepocas=0;
                first_iter=1;
                c=readpeso();
                fclose(ptr);
                ptr=stdin;
                if (c) {erro(15);break;}
                break;
   case EXIT   :match(EXIT);exit(1);
   case SAVENET:match(SAVENET);
                if (!match(PARESQ)) {erro(10);break;}
                if (token!=STRING) {erro(16);break;}
                strcpy(nome,tokstr);
                match(STRING);
                if (!match(PARDIR)) {erro(14);break;}
                ptr=fopen(nome,"w");
                if (ptr==NULL) {ptr=stdin;erro(30);break;}
                savenet();
                fclose(ptr);
                ptr=stdin;
                break;
   default     :erro(23);break;
 }
 match(PONTO);
 }
}

readpeso()
{
 unsigned int i;
 double p;
 ramos L;
 for(L=LIGA,i=0;i<nl;i++,L++)
   {
    if (!fscanf(ptr,"%le",&p) ) return 1;
    L->peso=p;
   }
 for(L=LIGA,i=0;i<nl;i++,L++) L->filtro=0;
 first_iter=1;
 return 0;
}

savepeso()
{
 unsigned int i;
 ramos L;
 for(L=LIGA,i=0;i<nl;i++,L++) fprintf(ptr,"%le\n",L->peso);
 return 0;
}

savenet()
{
 unsigned n,p,j,i,w;
 fprintf(ptr,"nunidades=%d\n",nunidades);
 fprintf(ptr,"ninputs=%d\n",ninputs);
 fprintf(ptr,"noutputs=%d\n",noutputs);
 for(i=ninputs;i<nunidades;i++)
  fprintf(ptr,"nlig(%d,%d)\n",i,UNI[i-ninputs].nlig);
 for(i=ninputs;i<nunidades;i++)
  {
   fprintf(ptr,"activ(%d,",i);
   switch(UNI[i-ninputs].activ)
    {
     case SIGMOID:fprintf(ptr,"SIGMOID");break;
```

```
      case LINEAR :fprintf(ptr,"LINEAR");break;
      case SIGCENT:fprintf(ptr,"SIGCENT");break;
     }
    fprintf(ptr,")\n");
   }
  for(i=ninputs;i<nunidades;i++)
   {
    n=UNI[i-ninputs].nlig;
    p=UNI[i-ninputs].pos;
    for(j=0;j<n;j++)
     {
      fprintf(ptr,"lig(");
      w=LIGA[p+j].ligacao;
      if (w==BIAS) fprintf(ptr,"BIAS,");
        else fprintf(ptr,"%d,",w);
      fprintf(ptr,"%d)\n",i);
     }
   }
}



printoutput()
{
 unsigned int i,j;
 char b;
 for(i=nunidades-noutputs,j=0;j<noutputs;j++,i++)
    {
     if (!bin) printf("\nO[%d] = %.15f ",j+1,UNI[i-ninputs].fo);
      else  { if (UNI[i-ninputs].fo>.5) b='1';
              else  if (UNI[i-ninputs].fo<.5) b='0';
                      else b='*';
            printf("\nO[%d] = %c ",j+1,b);
          }

    }
 printf("\n\n");
}

print_act(u) unsigned int u;
{
 printf("ACTIVAÇÃO : ");
 switch(u)
  {
   case SIGMOID:printf("SIGMOID");break;
   case LINEAR:printf("LINEAR");break;
   case SIGCENT:printf("SIGCENT");break;
  }
 printf("\n");
}
```

```
/* red_calc.c */


treina_pat()
{
 double (far *Input),(far *Target),emax,epp;
 char s[10];
 unsigned int i,j,t,stopin;
 j=nepocas;
 t=ninputs+noutputs;
 clrscr();
 while(1)
  {
   gotoxy(10,10);printf("%d\n",j);
   for(emax=0,i=0;i<npadroes;i++)
    {
     Input=&PATTERNS[i*t];
     Target=&Input[ninputs];
     deriv_zero();
     epp=calc_erro(Input,Target);
     printf("%d %.15f\n",i,epp);
     change_weights();
     if (emax<epp) emax=epp;
    }
   j--;
   if (emax<criterio) {printf("\n\ntreinada \n");return 1;}
   if (j<=0) { printf("\n");scanf("%s",s);if (s[0]=='s') return 0;j=nepocas;}
  }
}


treina_epoca()
{
 unsigned int epoca;
 if (adapt) treina_epoca_adapt();
  else
  {
   calcula_erro();
   for(epoca=1;epoca<=nepocas && (tss>criterio || (bin && ebin!=0));epoca++,ntepocas++)
    {
     calc_erro_deriv_epoca();
     muda_nao_adapta();
     printf("%u %.10f",epoca,tss);
    }
   if (tss<criterio || (bin && ebin==0)) printf("\ntreinada em %u\n",ntepocas);
  }
}

muda_nao_adapta()
{
 double var;
 unsigned int i;
 ramos L;
 L=LIGA;
 for(i=0;i<nl;i++,L++)
  {
   var=L->speed*(L->deriv+moment*L->filtro);
   L->peso+=var;
   L->filtro=L->deriv+moment*L->filtro;
  }
}

treina_epoca_adapt()
{
 unsigned int epoca;
 if (first_iter) { calc_erro_deriv_epoca();
                   first_iter=0;}
```

```
 for(epoca=1;epoca<=nepocas && (tss>criterio || (bin && ebin!=0));epoca++,ntepocas++)
  {
   printf("\n%u   %.10f",epoca,tss);
   iteracao();
  }
 if (tss<criterio || (bin && ebin==0)) {printf("\ntreinada em %u\n",ntepocas);return;}
}



calcnet(I) double *I;
{
 unit x;ramos  y;
 unsigned int i,j,n,l;
 double s;
 for(i=ninputs,x=UNI;i<nunidades;i++,x++)
  {
   n=x->nlig;
   y=&LIGA[x->pos];
   for(j=0,s=0;j<n;j++,y++)
    {
     l=y->ligacao;
     if (l==BIAS) { s+=(y->peso);continue;}
     if (l<ninputs) s+=I[l]*(y->peso);
       else s+=(UNI[l-ninputs].fo)*(y->peso);
    }
   activation(x,s);
  }
}

activation(x,s) unit x;double s;
{
 double w1,w2,w3;
 switch(x->activ)
  {
   case SIGMOID:if (s>=15.935773) {x->fo=9.9999988e-1;
                                   x->dfo=1.2000010e-7;
                                   break;}
                if (s<=-15.935773) {x->fo=1.2000011e-7;
                                    x->dfo=1.2000010e-7;
                                    break;}
                x->fo=1/(1+exp(-s));
                x->dfo=(x->fo)*(1-x->fo);
                break;
   case LINEAR :x->fo=s;
                x->dfo=1;
                break;
  }
}

double calc_erro_deriv(I,T)
          double *I,*T;
{
 unsigned int i,n,w,j;
 double *x,s,inp,del;
 unit p;ramos L;

 calcnet(I);
 for(p=UNI,i=ninputs;i<nunidades;i++,p++)  /* põe deltas a zero */
    p->delta=0;
 p=&UNI[nunidades-noutputs-ninputs];
 for(s=0,i=0,x=T;i<noutputs;i++,p++,x++)
   {p->delta=(*x-(p->fo));
    if (bin) {if (*x && p->fo<.5) ebin++;
                else
               if (*x==0 && p->fo>.5) ebin++;
              }
```

```
    s=s+(p->delta)*(p->delta);}

 p=&UNI[nunidades-ninputs-1];
 for(i=nunidades-1;i>=ninputs;i--,p--)
  {
   p->delta*=p->dfo;
   n=p->nlig;
   L=&LIGA[p->pos];
   for(j=0;j<n;j++,L++)
    {
     w=L->ligacao;
     if (w==BIAS) inp=1;
       else
         if (w<ninputs) inp=I[w];
             else {inp=UNI[w-ninputs].fo;
                   UNI[w-ninputs].delta+=p->delta*L->peso;}
     L->deriv+=p->delta*inp;
    }
  }
 return s;
}


deriv_zero()
{
 unsigned int i;
 ramos r;
 r=&LIGA[0];
 for(i=0;i<nl;i++,r++)
  r->deriv=0;
}

double aleatorio()
{
 double w;
 w=(double)rand()/RAND_MAX;
 w=range*(2*w-1);
 return w;
}

atribui_pesos()
{
 unsigned int i;
 ramos r;
 double w;
 for(i=0,r=LIGA;i<nl;i++,r++)
 {
  w=aleatorio(range);
  r->speed=speed;
  r->peso=w;
  r->filtro=0;
 }
}


muda_pesos()
{
 double var;
 unsigned int i;
 ramos L;
 L=LIGA;
 for(i=0;i<nl;i++,L++)
  {
    var=L->speed*(L->deriv+moment*L->filtro);
    L->peso+=var;
  }
}
```

```
divide_taxas()
{
 unsigned int i;
 ramos L;
 for(L=LIGA,i=0;i<nl;i++,L++)
    {
     L->speed=L->speed/2;
     if (L->speed<TECTOINF) L->speed=TECTOINF;
    }
}


recupera_pesos_deriv()
{
 unsigned int i;
 ramos L;
 for(L=LIGA,i=0;i<nl;i++,L++)
   {L->peso=L->peso_auxi;L->deriv=L->deriv_auxi;}
}

guarda_pesos_deriv()
{
 unsigned int i;
 ramos L;
 for(L=LIGA,i=0;i<nl;i++,L++)
   {L->peso_auxi=L->peso;L->deriv_auxi=L->deriv;}
}

sinal(x) double x;
{
 if (x<0) return -1;
  else return 1;
}


adapta_taxas()
{
  ramos L;
  char s1,s2;
  unsigned int i;
  for(L=LIGA,i=0;i<nl;i++,L++)
  {
   s1=sinal(L->deriv);s2=sinal(L->deriv_auxi);
   if (s1*s2<0) L->speed*=fdiminu;
        else L->speed*=fcresci;
   if (L->speed>TECTOSUP) L->speed=TECTOSUP;
        else if (L->speed<TECTOINF) L->speed=TECTOINF;
   }
}



iteracao()
{
 int i,subiu=0;
 double tss_ant;
 tss_ant=tss;
 guarda_pesos_deriv();
 muda_pesos();
 calcula_erro();
 if (tss<=tss_ant)
      { calc_erro_deriv_epoca();adapta_taxas();}
 else
      {
       esquece_passado();
       subiu=1;
       for(i=0;i<3 && tss>tss_ant;i++)   {
                          calc_erro_deriv_epoca();
```

```
                        adapta_taxas();
                        recupera_pesos_deriv();
                        muda_pesos();
                        calcula_erro();
                    }
        while(tss>tss_ant) {
                        recupera_pesos_deriv();
                        divide_taxas();
                        muda_pesos();
                        calcula_erro();
                       }
        calc_erro_deriv_epoca();
       }
 if (!subiu) calcula_filtros();
}

esquece_passado()
{
 unsigned int i;
 ramos L;
 for(L=LIGA,i=0;i<nl;i++,L++)
    L->filtro=0.0;
}



calcula_filtros()
{
 unsigned int i;
 ramos L;
 L=LIGA;
 for(i=0;i<nl;i++)  L->filtro= L->deriv_auxi + (moment*L->filtro);
}


calc_erro_deriv_epoca()    /* calcula erros e dE/dWij */
{
 double *Input,*Target;
 unsigned int t,i;
 t=ninputs+noutputs;
 deriv_zero();
 for(tss=0,i=0,ebin=0;i<npadroes;i++)
  {
   Input=&PATTERNS[i*t];
   Target=&Input[ninputs];
   epp=calc_erro_deriv(Input,Target);
   tss+=epp;
  }
}


calcula_erro()
{
 unsigned int w,t,i,j,k,n,l;
 double err,s,*Input,*Target;
 unit x;
 ramos y;
 w=nunidades-noutputs;
 t=ninputs+noutputs;
 for(tss=0,i=0,ebin=0;i<npadroes;i++)
  {
   Input=&PATTERNS[i*t];
   Target=&Input[ninputs];
   for(j=ninputs,x=UNI;j<nunidades;j++,x++)
     {
      n=x->nlig;
      y=&LIGA[x->pos];
```

```
        for(k=0,s=0;k<n;k++,y++)
          {
           l=y->ligacao;
           if (l==BIAS) {s+=y->peso;continue;}
           if (l<ninputs) s+=Input[l]*(y->peso);
             else s+=UNI[l-ninputs].fo*(y->peso);
          }
        activation(x,s);
        if (j>=w) {err=Target[j-w]-(x->fo);err=err*err;tss+=err;}
       }
    }
  }


 calc_percent()
 {
  unsigned int ncorrectos=0,i,b;
  double err;
  if (npadroes==0) return;
  if (bin==0) {printf("\ntem de ser binário \n");return;}
  for(i=0;i<npadroes;i++)
   {
    testa(i,&b,&err);
    if (b==0) ncorrectos++;
   }
  err=(double)ncorrectos/npadroes;
  err*=100;
  printf("percentagem = %f",err);
 }



 testa_padrao(i)
 int i;
 {
  unsigned binario,x;
  double continuo,*I;
  if (i<0 || i>=npadroes) return 1;
  I=&PATTERNS[i*(ninputs+noutputs)];
  printf("Padrao :%u",i);
  for(x=0;x<ninputs;x++,I++)
    printf("\n[%f]",*I);
  printf("\n");
  testa(i,&binario,&continuo);
  if (bin)  {
             if (binario==0) printf("\nCorrecto\n");
               else printf("\nIncorrecto\n");
            }
   printf("Erro continuo (soma dos quadrados) = %.15f",continuo);
  return 0;
 }



testa(pad,binario,continuo)
  int pad;
  unsigned int *binario;
  double *continuo;

{
 unsigned int e,i;
 double *I,*T,*x,erro,s;
 unit p;

 I=&PATTERNS[pad*(ninputs+noutputs)];
 T=&I[ninputs];
 calcnet(I);
```

```
 p=&UNI[nunidades-noutputs-ninputs];
 for(s=0,e=0,i=0,x=T;i<noutputs;i++,p++,x++)
   {erro=*x-(p->fo);
    erro*=erro;
    if (bin) {if (*x && p->fo<.5) e++;
                else
               if (*x==0 && p->fo>.5) e++;
              }
    s=s+erro;
   }
 *binario=e;
 *continuo=s;
}
```

```
/* programa principal 'janelas' inclui o gestor de janelas */

#include<conio.h>
#include<string.h>
#include<ctype.h>
#include "jan.h"

#define F1       15104
#define CIMA     18432
#define BAIXO    20480
#define ESQUERDA 19200
#define DIREITA  19712
#define ESC      283
#define ENTER    7181
#define INV      200
#define J_FILE   0
#define J_READ   1
#define J_WRITE  2
#define J_SETS   3
#define J_QUIT   4
#define J_CALC   5

typedef
 struct {
        int x,y,xlen,ylen,nop;
        char opcoes[10][15];
        int filho[10];
 } win;
typedef
 struct {
        int linha,nop;
        int coluna[10];
        int janel[10];
        char opcoes[10][15];
 } horiz;
horiz menu={1,6,
            5,17,29,41,53,65,77,0,0,0,
            J_QUIT,J_FILE,INV,J_SETS,J_CALC,INV,INV,INV,INV,INV,
            "QUIT","FILE","TRAIN","SET","CALC","RAND","","","","",} ;
win w[]={
15,3,6,3,2,"LER   ","GRAVAR","","","","","","","","", /* J_FILE */
J_READ,J_WRITE,INV,INV,INV,INV,INV,INV,INV,INV,
21,4,8,6,5,"REDE    ","PADRÃO ","PESOS   ","MASCARA ","TESTE   ","","","","","", /* J_READ */
INV,INV,INV,INV,INV,INV,INV,INV,INV,INV,
21,5,8,2,1,"PESOS   ","","","","","","","","","", /* J_WRITE */
INV,INV,INV,INV,INV,INV,INV,INV,INV,INV,
39,3,8,9,8,"SPEED   ","MOMENT ","RANGE   ","NEPOCAS ","FCRESCI ","FDIMINI ","BIN     ","CRITERIO","","", /* J_SETS */
INV,INV,INV,INV,INV,INV,INV,INV,INV,INV,
3,3,8,3,2,"DOS     ","OS SHELL","","","","","","","","", /* J_QUIT */
INV,INV,INV,INV,INV,INV,INV,INV,INV,INV,
51,3,6,3,2,"REDE  ","TESTE ","","","","","","","","", /* J_CALC */
INV,INV,INV,INV,INV,INV,INV,INV,INV,INV,
} ;
int opcao=1,refresh=1;

erro(i) int i;
{
 int j;
 char s[100];
 if (i==0) return;
 for(j=0;mensagens[j].num!=-1 && mensagens[j].num!=i;j++);
 if (mensagens[j].num==-1) strcpy(s,"por minha comodidade nao descrimino o erro.");
   else strcpy(s,mensagens[j].frase);
 box(5,17,60,2);gotoxy(7,18);cputs(s);bioskey(0);window(5,17,72,20);clrscr();window(1,1,80,25);
}

show_menu()
{ int i;
  textmode(C80);textcolor(WHITE);textbackground(BLACK);
  clrscr();normal();box(1,1,78,23);box(menu.linha,1,78,2);
```

```
 for(i=0;i<menu.nop;i++) { gotoxy(menu.coluna[i],menu.linha+1);cputs(menu.opcoes[i]);}
 refresh=0;
}
menus()
{
 int i;unsigned int a,o;
 while(1)
 {
  if (refresh)  show_menu();
  inverte();gotoxy(menu.coluna[opcao],menu.linha+1);cputs(menu.opcoes[opcao]);normal();
  a=menu.janel[opcao];
  if (a==INV) {
              do { a=bioskey(0); } while(a!=ESQUERDA && a!=DIREITA && a!=F1 && a!=ENTER);
              if (a==ENTER && opcao==5) {atribui_pesos();continue;}
              if (a==ENTER && opcao==2) {treina_epoca();refresh=1;continue;}
              gotoxy(menu.coluna[opcao],menu.linha+1);cputs(menu.opcoes[opcao]);
              if (a==F1) exit(0);
              if (a==ESQUERDA) opcao--; else opcao++;
              if (opcao<0) opcao=menu.nop-1;
              if (opcao>=menu.nop) opcao=0;
              continue;
    }
   janela(&w[a]);
  }
}

box(x,y,xlen,ylen) int x,y,xlen,ylen;
{
 int i;
 gotoxy(x,y);cputs("┌");
 for(i=0;i<xlen;i++) cputs("-");
 cputs("┐");
 for(i=1;i<ylen;i++)
  {
   gotoxy(x,y+i);cputs("|");
   gotoxy(x+xlen+1,y+i);cputs("|");
  }
 gotoxy(x,y+i);cputs("└");
 for(i=0;i<xlen;i++) cputs("-");
 cputs("┘");
}

lestring(x,y,str,strout)
 int x,y;
 char *str,*strout;
{
 char buffer[200];
 gettext(x,y,x+20+1,y+3,buffer);
 box(x,y,20,2);gotoxy(x+5,y);inverte();cputs(str);normal();
 readstr(x+1,y+1,strout,20);puttext(x,y,x+20+1,y+3,buffer);
}

readstr(x,y,str,len)
int x,y,len;
char *str;
{
 unsigned int w;int i=0;
 unsigned char c;char brancos[81];
 for(i=0;i<len;i++) brancos[i]=' ';
 brancos[i]='\0';
 gotoxy(x,y);cputs(brancos);gotoxy(x,y);i=0;
 do {
     w=bioskey(0);c=w;
     if (c==27) {i=0;break;}
     if (w==21248) {gotoxy(x,y);cputs(brancos);gotoxy(x,y);i=0;continue;}
     if (c=='\r') continue;
     if (i==len && c!=8) {sound(1000);delay(50);nosound();continue;}
     if (c>=' ' && c<=254) {gotoxy(x+i,y);putch(c);str[i]=c;i++;continue;}
```

```
    if (c==8)
        {  if (i==0) {sound(1000);delay(50);nosound();continue;}
           gotoxy(x+i-1,y);putch(' ');gotoxy(x+i-1,y);i--;continue;}
  } while(c!='\r');
 str[i]='\0';
}

processa(m,o)
int m,o;
 {
  char nome[30];int i;
  switch(m)
  {
   case J_QUIT  :if (o==0) {clrscr();exit(1);}
                 clrscr();system("");opcao=0;refresh=1;return;
   case J_READ  :lestring(20,20," FICHEIRO ",nome);
                 switch(o) {
                         case 0:erro(readnet(nome));return;
                         case 1:erro(readpat(nome));return;
                         case 2:erro(readwei(nome));return;
                         case 3:erro(readmas(nome));return;
                         case 4:erro(readtest(nome));return;
                         default:return;
                    }
   case J_WRITE :lestring(20,20," FICHEIRO ",nome);
                 switch(o) {
                         case 0:erro(savewei(nome));return;
                         default:return;
                    }
   case J_CALC:if (o==0) {    gotoxy(15,8);cprintf("INPUT");
                              lista(10,10,INPUTS,ninputs);gotoxy(45,8);cprintf("OUTPUT");                    calcnet(INPUTS);
                              copia_output();lista(40,10,OUTPUTS,noutputs);
                              window(2,7,79,23);clrscr();window(1,1,80,25);
                              return;}
               testar();window(2,7,79,23);clrscr();window(1,1,80,25);
               return;
   case J_SETS  :set(o);return;
  }
}

janela(t)
win *t;
{
 char buffer[400];int i,op=0,aux,x;unsigned int c;
 normal();gettext(t->x,t->y,t->x+t->xlen+1,t->y+t->ylen,buffer);
 box(t->x,t->y,t->xlen,t->ylen);
 for(i=0;i<t->nop;i++) {gotoxy(t->x+1,t->y+i+1);cputs(t->opcoes[i]);}
 while(1)
  {
   if (refresh) show_menu();
   inverte();gotoxy(t->x+1,t->y+op+1);cputs(t->opcoes[op]);normal();
   do { c=bioskey(0); } while(c!=ESQUERDA && c!=DIREITA && c!=ENTER && c!=ESC && c!=CIMA && c!=BAIXO && c!=F1);
   switch(c)
    {
     case F1   :exit(0);
     case CIMA :gotoxy(t->x+1,t->y+op+1);cputs(t->opcoes[op]);op--;
                if (op<0) op=t->nop-1;break;
     case BAIXO:gotoxy(t->x+1,t->y+op+1);cputs(t->opcoes[op]);op++;
                if (op>=t->nop) op=0;break;
     case ESQUERDA:
                while(opcao>=menu.nop) opcao-=100;
                gotoxy(menu.coluna[opcao],menu.linha+1);cputs(menu.opcoes[opcao]);
                puttext(t->x,t->y,t->x+t->xlen+1,t->y+t->ylen,buffer);opcao--;
                if (opcao<0) opcao=menu.nop-1;
                return -3;
     case DIREITA:
                while(opcao>=menu.nop) opcao-=100;
                gotoxy(menu.coluna[opcao],menu.linha+1);cputs(menu.opcoes[opcao]);
```

```
                puttext(t->x,t->y,t->x+t->xlen+1,t->y+t->ylen,buffer);opcao++;
                if (opcao>menu.nop) opcao=0;
                return -2;
      case ENTER:x=opcao;opcao+=100;aux=t->filho[op];
                if (aux==INV) {
                                for(i=0; &w[i]!=t;i++);
                                processa(i,op);opcao=x;
                                if (refresh) {opcao=0;return 1;}
                                break;
                              }
                janela(&w[aux]);opcao=x;
                break;
      case ESC  :if (opcao<menu.nop) break;
                puttext(t->x,t->y,t->x+t->xlen+1,t->y+t->ylen,buffer);
                return -1;
    }
  }
}

inverte()
{
 textbackground(WHITE);textcolor(BLACK);
}
normal()
{
 textbackground(BLACK);textcolor(WHITE);
}

main()
{
 menus();
}


set(var)
int var;
{
 double p;
 switch(var)
  {
   case 0:p=speed;pergunta_var(1,"SPEED",&p);speed=p;return;
   case 1:p=moment;pergunta_var(1,"MOMENT",&p);moment=p;return;
   case 2:p=range;pergunta_var(1,"RANGE",&p);range=p;return;
   case 3:p=nepocas;pergunta_var(0,"NEPOCAS",&p);nepocas=(int)p;return;
   case 4:p=fcresci;pergunta_var(1,"FCRESCI",&p);fcresci=p;return;
   case 5:p=fdiminu;pergunta_var(1,"FDIMINU",&p);fdiminu=p;return;
   case 6:p=bin;pergunta_var(0,"BIN",&p);bin=(int)p;return;
   case 7:p=criterio;pergunta_var(1,"CRITERIO",&p);criterio=p;return;
  }
}

pergunta_var(modo,str,pointer)
int modo;
char *str;double *pointer;
{char buffer[400],s[25];
 gettext(10,10,34,14,buffer);box(10,10,23,2);gotoxy(11,11);
 cputs("                    ");inverte();gotoxy(11,11);cputs(str);
 normal();cputs(" : ");
 if (modo==1) cprintf("%.7f",*pointer);
   else cprintf("%d",(int)(*pointer));
 lestring(20,20,"   NUMERO  ",s);puttext(10,10,34,14,buffer);
 if (s[0]=='\0') return;
 con_str_doub(s,pointer);
}

con_str_doub(str,num)
char *str;double *num;
{int i=0,sinal=1;double s=0,pot=.1;
```

```
 while(str[i]==' ' || str[i]=='\t') i++;
 if (str[i]=='+') i++;
  else if (str[i]=='-') {i++;sinal=-1;}
 if (!isdigit(str[i])) return;
 while(isdigit(str[i])) { s*=10;s+=str[i]-'0';i++;}
 if (str[i]=='.')
  {
   i++;
   while(isdigit(str[i])) {s+=pot*(str[i]-'0');pot*=.1;i++;}
   while(str[i]==' ' || str[i]=='\t') i++;
   if (str[i]!='\0') return;
  }
 s*=sinal;*num=s;
}
```

```
/*  jan.h */

#include<stdio.h>
#include<alloc.h>
#include<ctype.h>
#include<stdlib.h>
#include<float.h>
#include<math.h>
#define FIM         1000
#define NUMBER      1001
#define STRING      1002
#define SIGMOID     1
#define LINEAR      2
#define C_INICIAL   5
#define L_INICIAL   3
#define PARESQ      1006
#define PARDIR      1007
#define VIRGULA     1008
#define IGUAL       1009
#define NUNIDADES   1010
#define NINPUTS     1011
#define NOUTPUTS    1012
#define NLIG        1013
#define LIG         1014
#define ACTIV       1016
#define ERROR       1024
#define WEIGHT      1026
#define PONTO       1027
#define NPATTERNS   1028
#define WDERIV      1045
#define WPESO       1046
#define WSPEED      1047
#define UFO         1048
#define UDFO        1049
#define UDELTA      1050
#define TSS         1051
#define EPOCA       1052
#define WEI         1061
#define WDER        1062
#define TECTOSUP    10e30
#define TECTOINF    10e-30
#define LIVRE       65535
#define BIAS        65534
#define DIREITA  19712
#define ESQUERDA 19200
#define CIMA     18432
#define BAIXO    20480
#define ESC      283
#define BS       3592
#define TAB      3849
#define ENTER    7181

struct {
        char *com;
        unsigned int chave;
} tab[]={ "SIGMOID",SIGMOID,"LINEAR",LINEAR,"NUNIDADES",NUNIDADES,"NINPUTS",NINPUTS,"NOUTPUTS",NOUTPUTS,
          "NLIG",NLIG,"LIG",LIG,"ACTIV",ACTIV,"NPATTERNS",NPATTERNS,"WDERIV",WDERIV,"WPESO",WPESO,
          "WSPEED",WSPEED,"UFO",UFO,"UDFO",UDFO,"UDELTA",UDELTA,"EPOCA",EPOCA,
          "TSS",TSS,"WEI",WEI,"WDER",WDER,NULL,0};

struct {
        char *frase;
        int num;
}mensagens[]=
  {
   "Falta {nunidades} ",1,"Falta {=} ",2,"Falta {Numero} ",3,
   "Falta {ninputs} ",4,"Falta {noutputs} ",5,"Não tem memória ",6,
   "Falta {.} ",7,"Falta {NPATTERNS} ",8,"Npadrões <=0 ",9,"Falta {(} ",10,
```

```
    "Unidade inválida ",11,"Ficheiro não deveria conter mais nada ",12,
    "Falta unidade ou é inválida ",13,"Falta {}} ",14,"Ficheiro não existe ",15,
    "Falta {String} ",16,"Falta {,} ",17,"Rede  mal construida ",19,
    "Ligações > que nlig ",20,"Não existe essa ligação ",21,
    "Esperava uma função de activação ",22,"Comando desconhecido ",23,
    "Esperava uma variavel para fazer o set ",24,
    "nunidades<=0 ",25,"por {epoca} ou {pat} ",26,"ninputs<=0 ",27,
    "noutputs<=0 ",28,"nunidades < ninputs+noutputs ",29,
    "Não consigo escrever o ficheiro",30,"",-1
};

typedef struct {
                double fo,dfo,delta;
                unsigned int nlig,pos;
                char activ;
        } *unit;

typedef struct {
                double speed,peso,peso_auxi,
                       deriv,deriv_auxi,filtro;
                unsigned int ligacao;
        } *ramos;

FILE *ptr;
unsigned int epoca,nepocas=100,token,livre,nl,nunidades=0,ninputs,noutputs,npadroes=0,ntest=0,lh;
long int ebin;
double toknum,moment=.3,speed=.05,criterio,range=.3,fcresci=1.2,fdiminu=.7,tss,epp;
double *INPUTS,*OUTPUTS,*PATTERNS,*TEST;
char tokstr[100],bin,first_iter,adapt;
unit UNI;
ramos LIGA;
double aleatorio();
char quadro[10][20];

typedef
 struct
  {
   int tipo,linha,coluna,comp;
   unsigned long int scale;
   void *end;
  }
 campo_mascara;

campo_mascara camp[20];
int nc;
char ecran[20][60];
```

```
/* jan_io.c */

extensao(s) char *s;
{
 char *u;
 for(u=s;*u!='\0';u++) if (*u=='.') return 1;
 return 0;
}

readwei(s) char *s;
{
 unsigned int i;
 if (!extensao(s)) strcat(s,".wei");
 ptr=fopen(s,"r");if (ptr==NULL) return 100;
 for(i=0;i<nl;i++)
  if(!fscanf(ptr,"%le\n",&LIGA[i].peso)) {fclose(ptr);return 100;}
 for(i=0;i<nl;i++)  {LIGA[i].filtro=0;LIGA[i].speed=speed;}
 first_iter=1;fclose(ptr);
 return 0;
}

readpat(s) char *s;
{
 unsigned int i,j,w;double *t;
 if (!extensao(s)) strcat(s,".pat");
 ptr=fopen(s,"r");if (ptr==NULL) return 15;
 lh=fgetc(ptr);token=lex();first_iter=1;
 if (!match(NPATTERNS)) {fclose(ptr);return 8;}
 if (!match(IGUAL)) {fclose(ptr);return 2;}
 if (token!=NUMBER) {fclose(ptr);return 3;}
 npadroes=toknum;match(NUMBER);
 if (nunidades<=0 || ninputs+noutputs<=0) {fclose(ptr);return 18;}
 if (npadroes<=0) return 9;
 w=(ninputs+noutputs);free(PATTERNS);
 PATTERNS=(double *)calloc(npadroes,w*sizeof(double));t=PATTERNS;
 if (PATTERNS==NULL) {fclose(ptr);free(PATTERNS);return 6;}
 for(i=0;i<npadroes;i++)
  for(j=0;j<w;j++)  {
                    if (token!=NUMBER) return 3;
                    *t=toknum;match(NUMBER);t++;}
 return 0;
}

readtest(s) char *s;
{
 unsigned int i,j,w;
 double *t;
 if (!extensao(s)) strcat(s,".tst");
 ptr=fopen(s,"r");if (ptr==NULL) return 15;
 lh=fgetc(ptr);token=lex();first_iter=1;
 if (!match(NPATTERNS)) {fclose(ptr);return 8;}
 if (!match(IGUAL)) {fclose(ptr);return 2;}
 if (token!=NUMBER) {fclose(ptr);return 3;}
 ntest=toknum;match(NUMBER);
 if (nunidades<=0 || ninputs+noutputs<=0) {fclose(ptr);return 18;}
 if (ntest<=0) return 9;
 w=(ninputs+noutputs);free(TEST);
 TEST=(double *)calloc(ntest,w*sizeof(double));t=TEST;
 if (TEST==NULL) {fclose(ptr);free(TEST);return 6;}
 for(i=0;i<ntest;i++)
  for(j=0;j<w;j++)  {
                    if (token!=NUMBER) return 3;
                    *t=toknum;match(NUMBER);t++;}
 return 0;
}

liberta()
 {
   unsigned int i;
```

```
 if (nunidades==0) return;
 free(UNI);free(LIGA);free(PATTERNS);free(INPUTS);
}

lex()
{char *t;unsigned int i;
 while (isspace(lh)) lh=fgetc(ptr);
 if (lh==';') { while(lh!=(unsigned)EOF && lh!=10 && lh!=13) lh=fgetc(ptr);
                return lex();
              }
 if (lh==(unsigned)EOF) return FIM;
 switch(lh)
  {
   case '(':lh=fgetc(ptr);return PARESQ;
   case ')':lh=fgetc(ptr);return PARDIR;
   case ',':lh=fgetc(ptr);return VIRGULA;
   case '=':lh=fgetc(ptr);return IGUAL;
   case '.':lh=fgetc(ptr);if (!isdigit(lh)) return PONTO;
            ungetc(lh,ptr);lh='.';
  }
 if (isalpha(lh)) {t=tokstr;
                   for(;isalpha(lh);lh=fgetc(ptr),t++) *t=toupper(lh);
                   *t=0;
                   if (!strcmp("BIAS",tokstr)) {toknum=BIAS;return NUMBER;}
                   for(i=0;tab[i].com!=NULL;i++)
                     if (!strcmp(tab[i].com,tokstr)) return tab[i].chave;
                   return STRING;
                  }
 ungetc(lh,ptr);
 if (fscanf(ptr,"%le",&toknum)) {lh=fgetc(ptr);return NUMBER;}
 while(!isalnum(lh) && lh!=(unsigned)EOF && !isspace(lh)
       && lh!='(' && lh!=')' && lh!='.' && lh!=',' && lh!='=')
             lh=fgetc(ptr);
 return ERROR;
}

match(tok) unsigned int tok;
{
 if (tok!=token) return 0;
 token=lex();
 return 41;
}

readnet(s) char *s;
{
 unit t;unsigned int w,i,c;
 if (!extensao(s)) strcat(s,".net");
 ptr=fopen(s,"r");lh=fgetc(ptr);token=lex();
 if (ptr==NULL) return 15;
 if (!match(NUNIDADES)) {fclose(ptr);return 1;}
 if (!match(IGUAL)) {fclose(ptr);return 2;}
 if (token!=NUMBER) {fclose(ptr);return 3;}
 nunidades=toknum;match(NUMBER);
 if (nunidades<=0) {fclose(ptr);return 25;}
 if (!match(NINPUTS)) {fclose(ptr);return 4;}
 if (!match(IGUAL)) {fclose(ptr);return 2;}
 if (token!=NUMBER) {fclose(ptr);return 3;}
 ninputs=toknum;match(NUMBER);
 if (ninputs<=0) {fclose(ptr);return 27;}
 if (!match(NOUTPUTS)) {fclose(ptr);return 5;}
 if (!match(IGUAL)) {fclose(ptr);return 2;}
 if (token!=NUMBER) {fclose(ptr);return 3;}
 noutputs=toknum;match(NUMBER);
 if (noutputs<=0) {fclose(ptr);return 28;}
 if (nunidades<(ninputs+noutputs)) {fclose(ptr);return 29;}
 w=nunidades-ninputs;nl=0;UNI=(unit)calloc(w,sizeof(*UNI));
 if (UNI==NULL) {nunidades=0;fclose(ptr);return 6;}
 for(t=UNI,i=0;i<w;i++,t++)
```

```
  {
   t->nlig=0;t->pos=-1;t->activ=SIGMOID;
  }
 while(match(NLIG)) {c=nlig();
                     if (c) {free(UNI);nunidades=0;fclose(ptr);return c;}}
 if (nl==0) {free(UNI);nunidades=0;
             if (!match(FIM)) {fclose(ptr);return 12;}
             return 0;}
 LIGA=(ramos)calloc(nl,sizeof(*LIGA));
 if (LIGA==NULL && nl!=0) {nunidades=0;free(UNI);fclose(ptr);return 6;}
 for(i=0;i<nl;i++) LIGA[i].ligacao=LIVRE;
 while(token==LIG || token==ACTIV)
  {
   switch(token)
    {
     case LIG:match(LIG);c=lig();if (c) {free(UNI);free(LIGA);fclose(ptr);nunidades=0;return c;}
              break;
     case ACTIV:match(ACTIV);c=activ();if (c) {free(UNI);free(LIGA);fclose(ptr);nunidades=0;return c;}
    }
  }
 INPUTS=(double *)calloc(ninputs,sizeof(double));OUTPUTS=(double *)calloc(noutputs,sizeof(double));
 if (INPUTS==NULL || OUTPUTS==NULL)
                  {free(INPUTS);free(OUTPUTS);free(UNI);free(LIGA);nunidades=0;return 1;}
 for(i=0;i<ninputs;i++) INPUTS[i]=0.0;
 atribui_pesos();
 return 0;
}

nlig()
{
 unsigned int nl,n2,n3,i,c;
 if (!match(PARESQ)) return 10;
 if (token!=NUMBER) return 3;
 nl=toknum;match(NUMBER);
 if (!match(VIRGULA)) return 17;
 if (token!=NUMBER) return 3;
 n2=toknum;match(NUMBER);
 if (match(PARDIR)) {
                     c=nlig_atr(nl,n2);
                     if (c) return c;
                     return 0;
                    }
 if (!match(VIRGULA)) return 17;
 if (token!=NUMBER) return 3;
 n3=toknum;match(NUMBER);
 if (!match(PARDIR)) return 14;
 for(i=nl;i<=n2;i++) {c=nlig_atr(i,n3);if (c) return c;}
 return 0;
}

nlig_atr(a,b) unsigned int a,b;
{
 if (a<ninputs || a>nunidades) return 11;
 UNI[a-ninputs].pos=nl;UNI[a-ninputs].nlig=b;nl+=b;
 return 0;
}

lig()
{
 unsigned int nl,n2,n3,n4,i,j,c;
 if (!match(PARESQ)) return 10;
 if (token!=NUMBER) return 3;
 nl=toknum;match(NUMBER);
 if (!match(VIRGULA)) return 17;
 if (token!=NUMBER) return 3;
 n2=toknum;match(NUMBER);
 if (match(PARDIR)) {
                     c=lig_atr(nl,n2);
                      if (c) return c;
```

```
                    return 0;}
 if (!match(VIRGULA)) return 17;
 if (token!=NUMBER) return 3;
 n3=toknum;match(NUMBER);
 if (match(PARDIR)) {
                    for(i=n1;i<=n2;i++) { c=lig_atr(i,n3);
                                          if (c) return c;}
                    return 0;}
 if (!match(VIRGULA)) return 17;
 if (token!=NUMBER) return 3;
 n4=toknum;match(NUMBER);
 if (!match(PARDIR)) return 14;
 for(i=n1;i<=n2;i++)
  for(j=n3;j<=n4;j++)
    {c=lig_atr(i,j);if (c) return c;}
 return 0;
}

lig_atr(a,b) unsigned int a,b;
{
 unsigned int p,n,i;
 if (b<ninputs || b>nunidades) return 11;
 p=UNI[b-ninputs].pos;n=UNI[b-ninputs].nlig;
 for(i=0;i<n;i++)
   if (LIGA[p+i].ligacao==LIVRE)
     {
      LIGA[p+i].ligacao=a;
      return 0;
     }
 return 20;
}

activ()
{
 unsigned int n1,n2,i;
 if (!match(PARESQ)) return 10;
 if (token!=NUMBER) return 3;
 n1=toknum;match(NUMBER);
 if (!match(VIRGULA)) return 17;
 if (token!=NUMBER) {
                    if (token!=SIGMOID && token!=SIGCENT && token!=LINEAR) return 22;
                    if (n1<ninputs || n1>nunidades) return 11;
                    UNI[n1-ninputs].activ=token;match(token);
                    if (!match(PARDIR)) return 14;
                    return 0;
         }
 n2=toknum;match(NUMBER);
 if (!match(VIRGULA)) return 17;
 if (token!=SIGMOID && token!=LINEAR && token!=SIGCENT) return 22;
 for(i=n1;i<=n2;i++)
  {
    if (i<ninputs || i>nunidades) return 11;
    UNI[i-ninputs].activ=token;
  }
 match(token);
 if (!match(PARDIR)) return 14;
 return 0;
}

savewei(s) char *s;
{
 unsigned int i;ramos L;
 if (!extensao(s)) strcat(s,".wei");
 ptr=fopen(s,"w");if (ptr==NULL) return 30;
 for(L=LIGA,i=0;i<n1;i++,L++) fprintf(ptr,"%le\n",L->peso);
 fclose(ptr);
 return 0;
}
```

```
/* lista.c   'faz a gestão duma janela de entradas numéricas permitindo scroll ' */


conv_doub_str(d,s) double d;char *s;
{
 char *p;
 sprintf(s,"%-16.10f",d);
 for(p=s;*p;p++);p--;
 for(;*p==' ' || *p=='0';p--) *p='\0';
 if (*p=='.') *p='\0';
}

lista(x,y,Lis,compl) unsigned int x,y,compl;double *Lis;
{
 unsigned int t,comp=10;char car,buff[20];
 int px,py=1,pri=0,i;
 if (compl<10) comp=compl;
 box(x-1,y-1,20,comp+1);box(x-1,y-1,3,comp+1);
 for(i=0;i<comp;i++)  {gotoxy(x,y+i);cprintf("%3d",i);}
 for(i=0;i<comp;i++)   conv_doub_str(Lis[i],quadro[i]);
 for(i=0;i<comp;i++)  {gotoxy(x+4,y+i);cprintf("%s",quadro[i]);}
 px=strlen(quadro[0])+1;
 while(1)
  {gotoxy(x+3+px,py+y-1);
   do {t=bioskey(0);car=t;}
     while(t!=TAB && t!=BS && t!=CIMA && t!=BAIXO && t!=ENTER
          && (car<' ' || car>'z'));
   if (car>=' ' && car<='z') {cprintf("%c",car);quadro[py-1][px-1]=car;
                              quadro[py-1][px]='\0';
                              if (px<16) px++;continue;}
   switch(t)
    {
     case TAB:Lis[pri+py-1]=atof(quadro[py-1]);return;
     case BS:if (px>1) {gotoxy(x+2+px,py+y-1);cprintf(" ");px--;}
             break;
     case BAIXO:
     case ENTER: Lis[pri+py-1]=atof(quadro[py-1]);
                 if (py<comp) {py++;px=strlen(quadro[py-1])+1;break;}
                if (pri>=(compl-comp)) break;
                movetext(x,y+1,x+2,y+comp-1,x,y);gotoxy(x,y+comp-1);cprintf("%3d",pri+comp);
                movetext(x+4,y+1,x+19,y+comp-1,x+4,y);
                for(i=0;i<comp-1;i++) strcpy(quadro[i],quadro[i+1]);
                gotoxy(x+4,y+comp-1);cprintf("                    ");
                conv_doub_str(Lis[pri+comp],buff);strcpy(quadro[comp-1],buff);
                gotoxy(x+4,y+comp-1);cprintf("%s",buff);
                px=strlen(buff)+1;pri++;
                break;
    case CIMA:
         Lis[pri+py-1]=atof(quadro[py-1]);
                 if (py>1) {py--;px=strlen(quadro[py-1])+1;break;}
               if (pri==0) break;
               movetext(x,y,x+2,y+comp-2,x,y+1);gotoxy(x,y);cprintf("%3d",pri-1);
               movetext(x+4,y,x+19,y+comp-2,x+4,y+1);gotoxy(x+4,y);cprintf("                    ");
               for(i=comp-1;i>0;i--) strcpy(quadro[i],quadro[i-1]);
               conv_doub_str(Lis[pri-1],buff);strcpy(quadro[0],buff);
               gotoxy(x+4,y);cprintf("%s",buff);
               px=strlen(buff)+1;pri--;
               break;
    }
  }
}
```

```
/* jan-mas.c */

imprime(c) campo_mascara c;
{
 double w,*d;
 unsigned int *i,j;
 unsigned long int p;
 char str[30];
 normal();
 gotoxy(c.coluna+C_INICIAL,c.linha+L_INICIAL);
 if (c.tipo!=2) for(j=0;j<c.comp;j++) str[j]=' ';
 str[j]=0;
 printf("%s",str);
 gotoxy(c.coluna+C_INICIAL,c.linha+L_INICIAL);
 switch(c.tipo)
  {
   case 0:d=(double *)c.end;
          cprintf("%le",*d);
          break;
   case 1:i=(unsigned int *)c.end;
          cprintf("%u",*i);
          break;
   case 2:d=(double *)c.end;
          w=*d;
          if(w<0) {inverte();w=w*-1;}
          p=w*c.scale;
          if (excede(p,c.comp)) {for(j=0;j<c.comp;j++) str[j]='*';
                                 str[c.comp]='\0';}
           else
            {
             conv(p,str);
             for(j=0;str[j]!='\0';j++);
             for(;j<c.comp;j++) str[j]=' ';
             str[c.comp]='\0';
             ajusta(str);
            }
          cprintf("%s",str);
          normal();
          break;
  }
}

ajusta(s) char *s;
{
 char *i,*j;
 for(i=s;*i!='\0';i++);
 i--;if (*i!=' ') return;
 for(j=i;*j==' ';j--);
 while(j>=s) {*i=*j;*j=' ';i--;j--;}
}


excede(p,n) unsigned long int p;int n;
{
 int i;
 unsigned long int s;
 for(s=0,i=0;i<n;i++)
  {
   s=s*10;
   s=s+9;
  }
 if (p>s) return 1;
 return 0;
}

conv(n,s) unsigned long int n;char *s;
{
 int i,j;
```

```
 char w;
 if (n==0) {s[0]='0';s[1]='\0';return;}
 for(i=0;n!=0;i++,n=n/10)
   s[i]=(n%10)+'0';
 s[i]='\0';i--;
 for(j=0;j<i;i--,j++)
  {w=s[j];s[j]=s[i];s[i]=w;}
}

lelinha(s) char *s;
{
 int i;
 for(i=0;lh!=10;lh=fgetc(ptr),i++)
   s[i]=lh;
 lh=fgetc(ptr);
 s[i]=' ';s[i+1]='\0';
}

analisa_campos()
{
 int i,j,fora;
 nc=0;
 for(i=0;i<20;i++)
  {
   fora=1;
   for(j=0;ecran[i][j]!='\0';j++)
    {
     if (ecran[i][j]=='#' && fora==1) {fora=0;
                                       camp[nc].coluna=j;
                                       camp[nc].linha=i;
                                       continue;
                                       }
     if (ecran[i][j]!='#' && fora==0)
        {camp[nc].comp=j-camp[nc].coluna;fora=1;nc++;}
    }
   for(j=0;ecran[i][j]!='\0';j++) if (ecran[i][j]=='#') ecran[i][j]=' ';
  }
}

print_ecran()
{
 int i,j,troc=0;
 inverte();
 box(C_INICIAL-2,L_INICIAL-2,62,23);
 normal();
 for(i=0;i<20;i++)
  {
   gotoxy(C_INICIAL,L_INICIAL+i);
   for(j=0;ecran[i][j]!='\0';j++)
    {
     if (ecran[i][j]=='%') { if (troc==0) {inverte();troc=1;}
                                 else {normal();troc=0;}
                               continue; }
     cprintf("%c",ecran[i][j]);
    }
  }
}

print_mascara()
{
 int i;
 for(i=0;i<nc;i++)
   imprime(camp[i]);
}

readmas(s) char *s;
{
 unsigned int i,t,n,m,j,nlig;
  ramos L;
```

```
  if (!extensao(s)) strcat(s,".mas");
  ptr=fopen(s,"r");
  if (ptr==NULL) return 15;
  lh=fgetc(ptr);
  for(i=0;i<20;i++)
    lelinha(ecran[i]);
  analisa_campos(); /*  olha para o ecran que leu e coloca já os comprimentos */
  token=lex();
  for(i=0;i<nc;i++)
   {
    if (!match(PARESQ)) return 100;
    if (token!=NUMBER) return 100;
    camp[i].tipo=toknum;match(NUMBER);
    if (!match(VIRGULA)) return 100;
    if (token!=NUMBER) return 100;
    camp[i].scale=toknum;match(NUMBER);
    if (!match(VIRGULA)) return 100;
    if (token==TSS) { match(TSS);camp[i].end=&tss;}
    else
    if (token==EPOCA) { match(EPOCA);camp[i].end=&epoca;}
    else
    if (token==UFO || token==UDFO || token==UDELTA)
     {
      t=token;match(token);
      if (!match(VIRGULA)) return 100;
      if (token!=NUMBER) return 100;
      n=toknum;match(NUMBER);
      if (n<ninputs || n>nunidades) return 100;
      n-=ninputs;
      switch(t)
       {
        case UFO:camp[i].end=&(UNI[n].fo);
                 break;
        case UDFO:camp[i].end=&(UNI[n].dfo);
                 break;
        case UDELTA:camp[i].end=&(UNI[n].delta);
                 break;
       }
     }
    else
   if (token==WEI || token==WDER)
    {
     t=token;match(token);
     if (!match(VIRGULA)) return 100;
     if (token!=NUMBER) return 100;
     n=toknum;match(NUMBER);
     if (n!=BIAS && (n>nunidades)) return 100;
     if (!match(VIRGULA)) return 100;
     if (token!=NUMBER) return 100;
     m=toknum;
     match(NUMBER);
     if (m<ninputs || m>nunidades) return 100;
     m-=ninputs;
     nlig=UNI[m].nlig;
     L=&LIGA[ UNI[m].pos ];
     for(j=0;j<nlig && L->ligacao!=n;j++,L++);
     if (j==nlig) return 100;
     switch(t)
      {
       case WEI:camp[i].end=&(L->peso);
                break;
       case WDER:camp[i].end=&(L->deriv);
                break;
      }
    }
   if (!match(PARDIR)) return 100;
}
return 0;
}
```

```
/* jan_cal.c */

testar()
{
 double media,maior,menor,e,s;
 unsigned int w,i,j;
 unit u;
 if (ntest==0) return;
 w=ninputs+noutputs;
 media=0;
 maior=-9999e100;
 menor=9999e100;
 for(i=0;i<ntest;i++)
 {
  calcnet(&TEST[w*i]);
  for(s=0,j=0,u=&UNI[nunidades-w];j<noutputs;j++,u++)
   {
    e=TEST[w*i+ninputs+j]-u->fo;
    if (e<0) e*=-1;
    e/=TEST[w*i+ninputs+j];s+=e;
   }
  if (s>maior) maior=s;
  if (s<menor) menor=s;
  media+=s;
 }
 media=media/ntest;
 normal();box(5,17,40,4);
 gotoxy(6,18);cprintf("Erro Máximo:%le",maior);
 gotoxy(6,19);cprintf("Erro Mínimo:%le",menor);
 gotoxy(6,20);cprintf("Erro Médio :%le",media);
 bioskey(0);
 window(5,17,46,22);clrscr();window(1,1,80,25);
}

copia_output()
{
 unit u;
 unsigned int i;
 u=&UNI[nunidades-noutputs-ninputs];
 for(i=0;i<noutputs;i++,u++)
   OUTPUTS[i]=u->fo;
}

treina_epoca()
{
 clrscr();
 if (nunidades==0 || npadroes==0) return;
 print_ecran();
 if (first_iter) { calc_erro_deriv_epoca();
                   first_iter=0;}
 for(epoca=1;epoca<=nepocas && (tss>criterio || (bin && ebin!=0));epoca++)
  {
   iteracao();
   print_mascara();
  }
 bioskey(0);
}

calcnet(I) double *I;
{
 unit x;ramos  y;
 unsigned int i,j,n,l;
 double s;
 for(i=ninputs,x=UNI;i<nunidades;i++,x++)
  {
   n=x->nlig;
   y=&LIGA[x->pos];
   for(j=0,s=0;j<n;j++,y++)
```

```
    {
     l=y->ligacao;
     if (l==BIAS) { s+=(y->peso);continue;}
     if (l<ninputs) s+=I[l]*(y->peso);
       else s+=(UNI[l-ninputs].fo)*(y->peso);
    }
   activation(x,s);
  }
}

activation(x,s) unit x;double s;
{
 double w1,w2,w3;
 switch(x->activ)
  {
   case SIGMOID:if (s>=15.935773) {x->fo=9.9999988e-1;
                                   x->dfo=1.2000010e-7;
                                   break;}
                if (s<=-15.935773) {x->fo=1.2000011e-7;
                                    x->dfo=1.2000010e-7;
                                    break;}
                x->fo=1/(1+exp(-s));x->dfo=(x->fo)*(1-x->fo);
                break;
   case LINEAR :x->fo=s;x->dfo=1;
                break;
  }
}

double calc_erro_deriv(I,T)
          double *I,*T;
{
 unsigned int i,n,w,j;
 double *x,s,inp,del;
 unit p;ramos L;
 calcnet(I);
 for(p=UNI,i=ninputs;i<nunidades;i++,p++)  /* põe deltas a zero */
    p->delta=0;
 p=&UNI[nunidades-noutputs-ninputs];
 for(s=0,i=0,x=T;i<noutputs;i++,p++,x++)
   {p->delta=(*x-(p->fo));
    if (bin) {if (*x && p->fo<.5) ebin++;
               else
              if (*x==0 && p->fo>.5) ebin++;
             }
    s=s+(p->delta)*(p->delta);}
 p=&UNI[nunidades-ninputs-1];
 for(i=nunidades-1;i>=ninputs;i--,p--)
  {
   p->delta*=p->dfo;n=p->nlig;L=&LIGA[p->pos];
   for(j=0;j<n;j++,L++)
    {
     w=L->ligacao;
     if (w==BIAS) inp=1;
       else
         if (w<ninputs) inp=I[w];
             else {inp=UNI[w-ninputs].fo;
                   UNI[w-ninputs].delta+=p->delta*L->peso;}
     L->deriv+=p->delta*inp;
    }
 }
 return s;
}

deriv_zero()
{
 unsigned int i;
 ramos r;
 r=&LIGA[0];
```

```
 for(i=0;i<nl;i++,r++)
  r->deriv=0;
 }

 double aleatorio()
 {
  double w;
  w=(double)rand()/RAND_MAX;w=range*(2*w-1);
  return w;
 }

 atribui_pesos()
 {
  unsigned int i;
  ramos r;
  double w;
  for(i=0,r=LIGA;i<nl;i++,r++)
  {
   w=aleatorio(range);
   r->speed=speed;r->peso=w;r->filtro=0;
  }
 first_iter=1;
 }


 muda_pesos()
 {
  double var;
  unsigned int i;
  ramos L;
  L=LIGA;
  for(i=0;i<nl;i++,L++)
   {
     var=L->speed*(L->deriv+moment*L->filtro);
     L->peso+=var;
     if (L->peso>TECTOSUP) L->peso=TECTOSUP;
   }
 }

 divide_taxas()
 {
  unsigned int i;
  ramos L;
  for(L=LIGA,i=0;i<nl;i++,L++)
     {
      L->speed=L->speed/2;
      if (L->speed<TECTOINF) L->speed=TECTOINF;
     }
 }

 recupera_pesos_deriv()
 {
  unsigned int i;
  ramos L;
  for(L=LIGA,i=0;i<nl;i++,L++)
    {L->peso=L->peso_auxi;L->deriv=L->deriv_auxi;}
 }

 guarda_pesos_deriv()
 {
  unsigned int i;
  ramos L;
  for(L=LIGA,i=0;i<nl;i++,L++)
    {L->peso_auxi=L->peso;L->deriv_auxi=L->deriv;}
 }

 sinal(x) double x;
 {
   if (x<0) return -1;
```

```
    else return 1;
  }

  adapta_taxas()
  {
    ramos L;
    char s1,s2;
    unsigned int i;
    for(L=LIGA,i=0;i<nl;i++,L++)
    {
     s1=sinal(L->deriv);s2=sinal(L->deriv_auxi);
     if (s1*s2<0) L->speed*=fdiminu;
          else L->speed*=fcresci;
     if (L->speed>TECTOSUP) L->speed=TECTOSUP;
          else if (L->speed<TECTOINF) L->speed=TECTOINF;
     }
  }

  iteracao()
  {
   int i,subiu=0;
   double tss_ant;
   tss_ant=tss;
   guarda_pesos_deriv();
   muda_pesos();
   calcula_erro();
   if (tss<=tss_ant)
        { calc_erro_deriv_epoca();adapta_taxas();}
   else
        {
         esquece_passado();
         subiu=1;
         for(i=0;i<3 && tss>tss_ant;i++)   {
                              calc_erro_deriv_epoca();
                              adapta_taxas();
                              recupera_pesos_deriv();
                              muda_pesos();
                              calcula_erro();
                    }
         while(tss>tss_ant) {
                              recupera_pesos_deriv();
                              divide_taxas();
                              muda_pesos();
                              calcula_erro();
                             }
         calc_erro_deriv_epoca();
        }
   if (!subiu) calcula_filtros();
  }

  esquece_passado()
  {
   unsigned int i;
   ramos L;
   for(L=LIGA,i=0;i<nl;i++,L++)
      L->filtro=0.0;
  }

  calcula_filtros()
  {
   unsigned int i;
   ramos L;
   for(i=0,L=LIGA;i<nl;i++)  L->filtro= L->deriv_auxi + (moment*L->filtro);
  }

  calc_erro_deriv_epoca()     /* calcula erros e dE/dWij */
  {
   double *Input,*Target;
    unsigned int t,i;
```

```
 t=ninputs+noutputs;
 deriv_zero();
 for(tss=0,i=0,ebin=0;i<npadroes;i++)
  {
   Input=&PATTERNS[i*t];
   Target=&Input[ninputs];
   epp=calc_erro_deriv(Input,Target);
   tss+=epp;
  }
}

calcula_erro()
{
 unsigned int w,t,i,j,k,n,l;
 double err,s,*Input,*Target;
 unit x;
 ramos y;
 w=nunidades-noutputs;t=ninputs+noutputs;
 for(tss=0,i=0,ebin=0;i<npadroes;i++)
  {
   Input=&PATTERNS[i*t];Target=&Input[ninputs];
   for(j=ninputs,x=UNI;j<nunidades;j++,x++)
     {
      n=x->nlig;y=&LIGA[x->pos];
      for(k=0,s=0;k<n;k++,y++)
        {
         l=y->ligacao;
         if (l==BIAS) {s+=y->peso;continue;}
         if (l<ninputs) s+=Input[l]*(y->peso);
           else s+=UNI[l-ninputs].fo*(y->peso);
        }
      activation(x,s);
      if (j>=w) {err=Target[j-w]-(x->fo);err=err*err;tss+=err;}
     }
  }
}
```