Universidade do Minho
Escola de Engenharia
Departamento de Informática

Modelos Inspirados na Natureza para a Previsão de Séries Temporais

Paulo Alexandre Ribeiro Cortez

Tese submetida à Universidade do Minho
para obtenção do grau de Doutor em Informática, elaborada sob a orientação do
Professor Doutor José Carlos Ferreira Maia Neves

2002

Modelos Inspirados na Natureza para a Previsão de Séries Temporais

Paulo Alexandre Ribeiro Cortez
Departamento de Sistemas de Informação
Campus de Azurém
4800-058 Guimarães
Portugal

http://www.dsi.uminho.pt/~pcortez
pcortez@dsi.uminho.pt

ISBN 972-98839-2-0
9 789729 883927

"If you can look into the seeds of time, and say which grain will grow and which will not, speak then unto me."

– William Shakespeare.

Agradecimentos

Fico grato ao Professor Doutor José Carlos Ferreira Maia Neves, meu orientador, por ter influenciado o meu interesse na área da Inteligência Artificial, pelos seus ensinamentos, motivação e aconselhamento, bem como na confiança que sempre depositou em mim.

Não quero deixar de agradecer a estimada contribuição dos colegas do laboratório de Inteligência Artificial, Rui Mendes, Carla Vilela e Luís Brito, por toda ajuda prestada, nomeadamente na partilha de recursos computacionais. Também quero agradecer aos colegas de curso, Orlando Pereira e António Sousa, do Grupo de Sistemas Distribuídos, pelo auxílio fornecido sobre o sistema operativo *Linux*. O *software* de previsão da metodologia de *Box-Jenkins*, *Forecast Pro*, foi cedido gentilmente por Carlos Figueiredo, do Departamento de Produção e Sistemas. Um especial reconhecimento é devido ao amigo e colega Miguel Rocha, que prestou um valoroso contributo com seus conhecimentos sobre *Computação Evolucionária e Genética.*

Este projecto foi suportado pela *Fundação para a Ciência e a Tecnologia (FCT)*, desde 1/12/1998 até 30/9/2001, através da bolsa *PRAXIS XXI/BD /13793/97*, e enquadrado no projecto *PRAXIS/P/EEI/13096/98.*

Finalmente, desejo agradecer à minha família, amigos e à Ângela, por todo o apoio e encorajamento.

Resumo

É possível re-equacionar o(s) problema(s) subjacentes à temática da *Previsão de Séries Temporais (PST)* através de metodologias de resolução de problemas que emanam da área científica da *Inteligência Artificial*, assim como de ferramentas ou artefactos usados na procura das respectivas soluções, como as *Redes Neuronais Artificiais (RNAs)* e os *Algoritmos Genéticos e Evolucionários (AGEs)*. As *RNAs* são candidatas naturais a uma utilização em previsão não linear, enquanto que os *AGEs* providenciam uma procura adaptativa, sendo talhados para uma optimização global.

O presente trabalho descreve uma forma de hibridação de ambos os paradigmas para a *PST*. Foram efectuados testes comparativos com métodos convencionais de previsão (e.g., o *Alisamento Exponencial* e a metodologia de *Box-Jenkins*), em diferentes séries reais e artificiais, constatando-se que os modelos inspirados na Natureza dão corpo a melhores previsões, especialmente quando são considerados sistemas complexos (e.g., séries não lineares e/ou caóticas). Por último, estes modelos foram também aplicados à previsão de eventos que operam em tempo real, onde mais uma vez as *RNAs* apresentaram os melhores resultados, quando comparados com outros procedimentos de previsão.

Temas: Algoritmos Genéticos e Evolucionários; Análise de Séries Temporais; Previsão em Tempo Real; Redes Neuronais Artificiais; Selecção de Modelos.

Abstract

New alternative approaches for *Time Series Forecasting (TSF)* emerged from the *Artificial Intelligence* arena, where optimization methods inspired on natural processes, such as *Artificial Neural Networks (ANNs)* and *Genetic and Evolutionary Algorithms (AGEs)*, are popular. *ANNs* are potential nonlinear candidates for forecasting, while *GEAs* perform an adaptive search, being suited for global optimization.

The present work reports on the use of both paradigms for *TSF*. Comparative tests with conventional methods (e.g., *Exponential Smoothing* and the *Box-Jenkins* methodology), performed on several real and artificial series, favored the bio-inspired models, specially when harder tasks are at stake; i.e., when nonlinear and chaotic series are considered. Finally, the models were also applied to real-time forecasting, being the best results given by the *ANNs*.

Keywords: Artificial Neural Networks; Genetic and Evolutionary Algorithms; Model Selection; Real-Time Forecasting, Time Series Analysis.

Conteúdo

Lista de Figuras

Lista de Tabelas

Notação

Acrónimos

AG *Algoritmo Genético.*

AGE *Algoritmo Genético e Evolucionário.*

AE *Alisamento Exponencial.*

APAGE *Ambiente de Programação de Algoritmos Genéticos e Evolucionários.*

APRNA *Ambiente de Programação de Redes Neuronais Artificiais.*

AR *Auto-Regressivo.*

BJ *Box-Jenkins.*

CGE *Computação Genética e Evolucionária.*

EE *Estratégias Evolutivas.*

F *Fourier.*

IA *Inteligência Artificial.*

IS *Inteligência Social.*

JTD *Janela Temporal Deslizante.*

Meta-AGE *Meta-Algoritmo Genético e Evolucionário.*

ON *Ordenação Numérica.*

PE *Programação Evolucionária.*

PG *Programação Genética.*

POO *Programação Orientada ao Objecto.*

PST *Previsão de Séries Temporais.*

RBF *Radial Basis-Functions.*

RN *Rede Neuronal.*

RNA *Rede Neuronal Artificial.*

RNE *Rede Neuronal Evolucionária.*

RNR *Rede Neuronal Recorrente.*

RNU *Rede Neuronal Unidireccional* (tradução adoptada do inglês *Feedforward Neural Network*).

RP *Retro-Propagação* (do inglês *Back-Propagation*).

RPROP *Resilient backPROPagation.*

RVR *Representações de Valores Reais.*

QP *QuickProp.*

Símbolos Gerais e Abreviaturas

$f(x)$ função sobre a variável x.

D domínio de $f(x)$.

w_{ij} conexão entre o nodo j e o nodo i.

fa função de activação.

u_i valor de integração do nodo i.

s_i valor de saída do nodo i.

$sign(x)$ valor do sinal de x.

E conjunto de nodos de entrada.

I conjunto de nodos de intermédios.

C conjunto de conexões pesadas unidireccionais.

S conjunto de nodos de saída.

F conjunto de funções de activação.

$|C|$ cardinalidade do conjunto C.

ξ função de erro.

n_{ei} o somatório dos nodos de entrada e intermédios.

$\Delta w_{ij}(t)$ ajustamento aos pesos para a iteração t.

$\frac{\delta f(x)}{\delta x}$derivada parcial em ordem a x.

f' derivada da função f.

η taxa de aprendizagem.

$succ(i)$ conjunto dos nodos que recebem conexões a partir do nodo i.

μ taxa de *momentum*.

ν constante do algoritmo de *Quickprop*.

Δ_0, Δ_{max} parâmetros do algoritmo *RPROP*.

ξ_{tr} erro de treino.

ft faixa de treino.

$P_{ft}(t)$ progresso de treino para a iteração t.

ϕ medida de erro de estado estacionário.

φ número máximo de iterações.

$\overline{x}$ valor médio.

$std(x)$ desvio padrão.

min valor mínimo.

max valor máximo.

T numero máximo de gerações.

P_t população para a geração t.

$:=$ atribuição.

$Aptidão$ função de aptidão.

$Operador$ operador genético.

$Cruzamento$ operador de cruzamento.

$Indiv$ indivíduo.

Mut operador de mutação.

λ valor aleatório pertencendo ao intervalo $[0, 1]$.

TP tamanho da população.

TS taxa de substituição.

VE valor de elitismo.

TGD taxa de geração de descendentes.

NP número de progenitores.

x_t série temporal.

$\widehat{x_t}$ modelo de série temporal.

K factor sazonal.

r_k autocorrelação para o deslocamento k.

e_t erro para o período t.

SQE soma do quadrado dos erros.

MQE média do quadrado dos erros.

$RMQE$ raiz da média do quadrado dos erros.

$MNQE$ média normalizada do quadrado dos erros.

F_t alisamento exponencial para o período t.

T_t estimativa para a tendência.

S_t a estimativa sazonal.

α constante do alisamento exponencial.

β constante da estimativa para a tendência.

γ constante para a estimativa sazonal.

med_e média das observações obtidas durante a estação e.

Y número máximo de estações.

$ARMA(P, Q)$ modelo $ARMA$.

μ constante do modelo $ARMA$.

A_i coeficientes do modelo AR.

M_j coeficientes do modelo MA.

$< k_1, ..., k_n >$ janela temporal deslizante contendo os deslocamentos k_i.

g_i valor real do gene i de um cromossoma.

AIC Critério de Informação de Akaike.

BIC Critério de Informação de Bayes.

N número de exemplos de treino.

p número de parâmetros livres de um modelo.

n número de entradas de um modelo.

n_i número de nodos intermédios.

$RMQE_{tr}$ valor de $RMQE$ para os casos de treino.

$RMQE_{pr}$ valor de $RMQE$ para a previsão de curto prazo.

R factor de redução da topologia.

P_{max} ordem máxima do modelo AR.

Q_{max} ordem máxima do modelo MA.

$MNQE_{pr}$ valor de $MNQE$ para a previsão de curto prazo.

HT horizonte temporal.

PF número de previsões em frente.

DJ deslocamento da janela de visibilidade.

CJ comprimento da janela de visibilidade.

I_p índice do indivíduo da população.

z_N nível de significância.

Capítulo 1

Introdução

> "There is nothing more difficult to take in hand, more perilous to conduct, or more uncertain in its success, than to take the lead in the introduction of a new order to things."
>
> – N. Machiavelli.

Motivação, Inteligência Artificial, Movimento Conexionista, Computação Genética e Evolucionária. Objectivos e Organização da Tese.

1.1 Motivação

Desde os tempos primordiais que o ser humano tenta compreender o mundo físico que o rodeia, em termos do como, porquê e do que acontecerá. A selecção natural permitiu o desenvolvimento de seres inteligentes, onde a percepção, antecipação e planeamento de eventos se tornaram factores cruciais para a sobrevivência em ambientes hostis e dinâmicos.

Hoje em dia, com todo o progresso tecnológico e científico, a previsão continua a ter extrema relevância: os efeitos e prejuízos de catástrofes (e.g., inundações ou terramotos) podem ser minorados, caso estes possam ser previstos atempadamente; previsões precisas sobre o número de manchas solares são vitais para a tomada de decisões sobre missões espaciais e a colocação de satélites em órbita; ganhos consideráveis podem ser obtidos caso se consiga prever a evolução de indicadores financeiros (e.g., inflação, índices da bolsa); etc.

Ora, é natural pressupor que as organizações estão interessadas em obter previsões fundamentadas sobre o seu futuro. Quer na tomada de decisões quer no planeamento, torna-se necessário prever em que circunstâncias estas(e) se irão desenvolver. Esta necessidade abrange todas as áreas de Gestão [Makridakis & Wheelwright, 1989]:

- no *Marketing,* há decisões que podem ser significativamente melhoradas caso se conheça a grandeza e características do mercado;
- na *Produção,* a ênfase é dada, não raras vezes, na procura de um novo produto, atendendo a possíveis evoluções do mercado;
- nas áreas *Financeiras* e de *Gestão Empresarial*, na projecção de rácios e ganhos; e
- na gestão dos *Recursos Humanos,* em que há que se lidar com problemas de absentismo ou produtividade, em termos de factores como turnos de trabalho, religião e formação, entre muitos outros.

Contudo, as empresas não devem encarar este fenómeno, ou seja, a previsão, como um santo remédio, mas sim como a melhor maneira de extrapolar as relações existentes entre as diversas variáveis envolvidas.

Os erros de previsão são, como é evidente, inevitáveis, tendo-se como objectivo minimizá-los; i.e., o desejo de compreender o passado e prever o futuro impulsiona a procura de leis que expliquem o comportamento de certos fenómenos ou acontecimentos. Se são conhecidas as equações determinísticas que os explicam, então, e em princípio, estas podem ser utilizadas para prever o resultado de uma dada experiência, desde que sejam conhecidas as condições iniciais. No entanto, na ausência de regras que definam o comportamento de um sistema, procura-se determinar o seu comportamento futuro a partir de observações concretizadas no passado. Nestas situações, uma das técnicas mais comuns é a da *Previsão de Séries Temporais (PST)*, que se baseia em observações cronologicamente ordenadas da variável em estudo [Chatfield, 1989].

Como consequência natural da importância da previsão, variadas técnicas de *PST* foram sendo desenvolvidas, oriundas de disciplinas como a *Investigação Operacional*, a *Estatística* e a *Informática.*

Antes dos anos 20, a previsão era efectuada através de uma simples extrapolação de um valor global ajustado em função do tempo. A era moderna da *PST* começou quando Yule [1927] inventou o modelo *Auto-Regressivo (AR)*, onde o próximo valor dependia de uma soma pesada de valores anteriores, para prever o número anual de manchas solares.

Durante as décadas seguintes, a opinião reinante era a de que o comportamento de um sistema dependia de dois componentes principais: modelos lineares e ruído.

Em meados dos anos 50, dois acontecimentos alteraram dramaticamente o campo da previsão. O primeiro foi a introdução de um conjunto de técnicas de *Alisamento Exponencial (AE),* que eram simples de se utilizar, exigindo pouco cálculo, embora tenha levado cerca de 30 anos até ganharem um reconhecimento e adopção generalizadas. Felizmente, o segundo grande evento nos anos 50 foi o advento do computador, permitindo revolucionar a aplicabilidade dos métodos de previsão, que se tornaram cada vez mais populares nas organizações e nos meios militares, à medida que o custo de computação ia diminuindo [Makridakis & Wheelwright, 1989].

Durante os anos 50 e 60, houve uma procura por uma teoria unificada de previsão. Em 1976, surgiu a metodologia de *Box-Jenkins (BJ)*, que incorporava diversos elementos de tal teoria. Esta metodologia fornecia um procedimento sistemático para a análise de séries temporais que era suficientemente geral para lidar com, virtualmente, qualquer tipo de observações empíricas.

Por outro lado, em Álgebra, tem-se que a linearidade é definida em termos de funções que subscrevem a axiomática:

$$\begin{aligned} f(x+y) &= f(x) + f(y) \\ f(ax) &= af(x) \end{aligned} \tag{1.1}$$

A atracção pelos modelos lineares é justificada pela simplicidade de análise e cálculo. Contudo, existem deveras situações onde o modelo linear se torna inadequado. O surgimento da *Teoria do Caos* veio a destronar o paradigma linear, demonstrando que inúmeros fenómenos complexos e caóticos, tão vulgares no meio físico (e.g., a turbulência em fluídos ou o número de constipações) podem ser gerados a partir de equações simples não lineares [Peitgen et al., 1992].

Por exemplo, os métodos estatísticos convencionais, como o *AE*, poderão não ser os mais adequados para a previsão de curto prazo, por serem incapazes de reconhecer pequenas variações (e.g., variações de preço de acções cotadas na bolsa) que irão ter um grande impacto a longo prazo [Shoneburg, 1990].

Uma alternativa para a *PST* é dada pela *Inteligência Artificial (IA)*, onde foram desenvolvidas um conjunto de técnicas para a resolução de problemas, inspiradas na Natureza, como as *Redes Neuronais Artificiais (RNAs)* e os *Algoritmos Genéticos e Evolucionários (AGEs)*, que apresentaram resultados interessantes num vasto leque de aplicações científi-

cas e de engenharia. Dentro deste contexto, o clausulado central da tese é dado por:

> "Os modelos inspirados na Natureza providenciam uma alternativa mais eficaz para a previsão de séries temporais."

1.2 Inteligência Artificial

O objectivo da *Inteligência Artificial (IA)* passa pelo desenvolvimento de paradigmas e algoritmos para realizar tarefas cognitivas, as quais são actualmente executadas pelos seres humanos de um modo eficiente. Apesar de ser uma disciplina recente (foi formalmente iniciada em 1956), tem induzido desenvolvimentos significativos em diferentes áreas de conhecimento, nomeadamente na compreensão da língua natural, no raciocínio lógico-matemático, na teoria dos jogos ou nos sistemas de apoio à decisão [Russell & Norvig, 1995].

Um sistema de *IA* deve ser capaz de perfazer três requisitos [Haykin, 1999] (Figura 1.1):

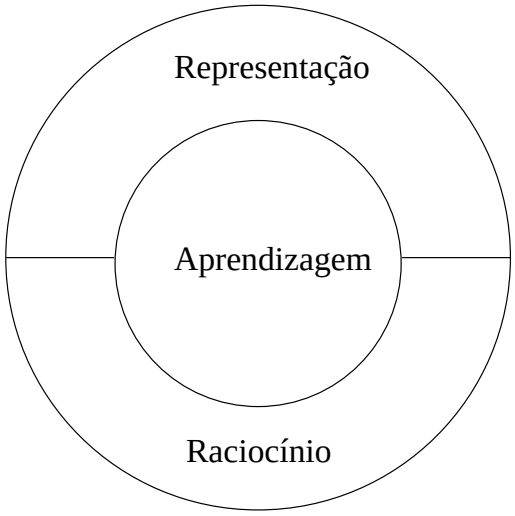


Figura 1.1: Os três componentes fundamentais de um sistema de *IA*.

- armazenar conhecimento (*representação do conhecimento*);
- aplicar o conhecimento adquirido para resolver problemas (*raciocínio*); e
- adquirir novo conhecimento através da experiência (*aprendizagem*).

O primeiro requisito envolve uma *representação do conhecimento*, que provém de dois tipos de informação:

- o estado conhecido, representado por factos sobre o que é e o que se conhece; e
- observações ou medições sobre o ambiente, que podem conter ruído e estar sujeitas a erros.

O segundo requisito diz respeito ao *raciocínio*, definido como a capacidade para a *resolução de problemas,* que pode ser vista como um problema de *procura.* Uma forma de lidar com a procura é o uso de regras, dados, controlo ou raciocínio probabilístico.

Por último, tem-se a *aprendizagem*, permitindo aumentar o conhecimento através da experiência. A aprendizagem pode envolver duas formas diferentes de processamento de informação: *indutiva* e *dedutiva.* Na primeira forma, padrões e regras são determinadas a partir dos dados e experiência (e.g., *aprendizagem baseada em casos*), enquanto que na última, as regras são utilizadas para criar novos factos (e.g., *prova de teoremas*).

Outro aspecto a ter em conta é a forma como um modelo de aprendizagem se relaciona com o seu ambiente. Neste contexto, existem três paradigmas fundamentais em que assenta a aprendizagem [Patterson, 1996]:

- *Supervisionada.* Trata-se de uma técnica bastante popular que envolve um "professor", ou seja, são fornecidas respostas correctas ao sistema. A aprendizagem é efectuada a partir de um conjunto de padrões, onde cada *padrão*, também chamado de *exemplo* ou *caso de treino*, é composto por um vector de entrada e por um vector de *resposta* ou *saída.*
- *De Reforço.* Neste tipo de aprendizagem também se assume a presença de um "professor" embora a resposta correcta não seja apresentada ao sistema. Assim, apenas se fornece uma indicação sobre se a resposta devolvida pelo sistema é correcta ou errada, sendo esta informação utilizada para melhorar a sua eficácia.
- *Não Supervisionada.* Trata-se de uma abordagem diferente, onde não é fornecida ao sistema uma indicação externa acerca da resposta correcta. Por conseguinte, a aprendizagem é feita pela descoberta de características nos dados de entrada, adaptando-se a regularidades estatísticas ou agrupamentos de padrões dos exemplos de treino.

A abordagem inicial seguida na *IA* com vista à obtenção de técnicas para a resolução de problemas passava por se adoptar uma abordagem essencialmente matemática, na crença implícita do raciocínio lógico como um marco para a interpretação da inteligência, onde o conhecimento é representado pelo uso de uma linguagem *simbólica*, através de uma colecção de factos e regras para manipular estes.

Contudo, novos paradigmas para a resolução de problemas, inspirados em modelos biológicos, surgiram dentro da *IA* como uma reacção aos modelos simbólicos, considerados demasiado rígidos e especializados [Luger & Stubblefield, 1998]. O estudo de sistemas presentes em seres vivos, como o *sistema nervoso central* ou o da *evolução biológica*, deu origem aos *Modelos Conexionistas* e à *Computação Genética e Evolucionária (CGE).* Com esta mudança, de métodos baseados em regras para métodos baseados em dados, a *IA* tornou-se apta para lidar com problemas como o da *PST* [Weigend & Gershenfeld, 1994].

1.3 O Movimento Conexionista

As *Redes Neuronais Artificiais (RNAs),* também designadas por *sistemas conexionistas*, são modelos simplificados do sistema nervoso central do ser humano. Uma *RNA* é um processador eminentemente paralelo, composto por simples unidades de processamento, designadas por neurónios ou nodos, que possuem uma propensão natural para armazenar conhecimento empírico e torná-lo acessível ao utilizador. Assemelha-se ao comportamento do cérebro humano em dois aspectos [Haykin, 1999]:

- o conhecimento é adquirido a partir de um ambiente, através de um processo de aprendizagem; e
- o conhecimento é armazenado nas *conexões,* também designadas por *ligações* ou *sinapses*, entre os nodos.

Durante o processo de aprendizagem, dado por um *algoritmo de aprendizagem* ou *de treino*, os *pesos* das conexões são ajustados de forma a se atingir um dado objectivo; i.e., o estado de conhecimento da rede. Embora seja esta a forma tradicional de construir *RNAs,* também é possível modificar a sua própria estrutura interna (ou *topologia*), à semelhança do que se passa no cérebro, onde os neurónios podem morrer e novas sinapses (e mesmo neurónios) se podem desenvolver.

1.3.1 Inspiração Biológica: O Cérebro Humano

A maior parte da investigação em *RNAs* foi inspirada e influenciada pelos sistemas nervosos dos seres vivos, em particular do ser humano. Muitos investigadores acreditam que as *RNAs* oferecem a aproximação mais promissora para a construção de verdadeiros siste-

mas inteligentes, tendo capacidade para ultrapassar a explosão combinatória associada à computação simbólica baseada em arquitecturas de *von Neumann*[1] [Patterson, 1996].

O sistema nervoso central fornece uma forte base de sustentação a esta tese. O cérebro é uma estrutura altamente complexa, não linear e paralela. Possui uma capacidade de organizar os seus constituintes, conhecidos por neurónios, de modo a executarem certas tarefas complexas (e.g., reconhecimento de padrões ou voz), de uma forma inatingível pelo computador mais potente até hoje concebido. Em termos de velocidade de processamento, um neurónio é cerca de 5 a 6 vezes mais lento do que uma porta lógica de silício. Todavia, o cérebro ultrapassa esta lentidão utilizando uma estrutura maciçamente paralela. Estima-se que o *córtex* humano possui cerca de 10 biliões de neurónios e 60 triliões de sinapses [Haykin, 1999].

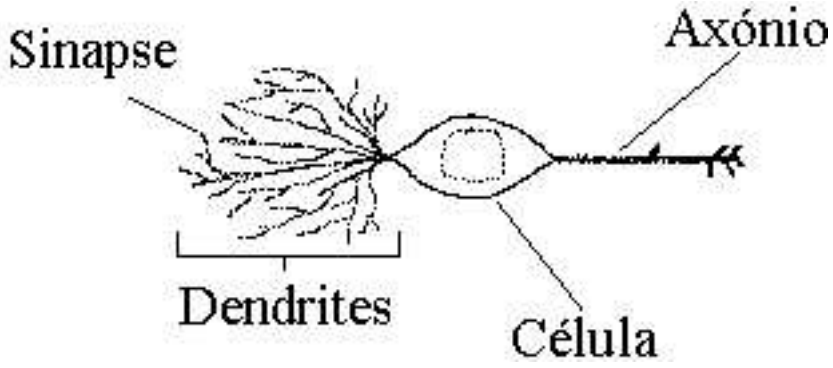


Figura 1.2: Estrutura de um neurónio natural.

Um neurónio é uma célula complexa que responde a sinais electro-químicos, sendo composto por um *núcleo*, por um *corpo celular*, por um numeroso conjunto de *dendrites,* entidades que recebem sinais de outros neurónios via sinapses, e por um *axónio*, que transmite um estímulo a outros neurónios, através das já referidas sinapses (Figura 1.2) [Bose & Liang, 1996].

Um único neurónio pode estar ligado a centenas, milhares, ou mesmo dezenas de milhares neurónios. Num cérebro existem estruturas anatómicas de pequena, média e alta complexidade com diferentes funções, sendo possíveis parcerias. Os neurónios tendem a agrupar-se em camadas, existindo três principais tipos de conexões (Figura 1.3):

- *divergentes*, onde um neurónio pode estar ligado a outros neurónios via uma arborização do axónio;

[1] John von Neumann (1903-1957), matemático húngaro-americano que teve uma grande contribuição na definição da arquitectura de máquinas sequenciais, onde um programa é armazenado na mesma memória de dados que o programa utiliza. Hoje em dia, quase todos computadores são do tipo von Neumann.

- *convergentes*, onde vários neurónios podem estar conectados a um único neurónio; e
- *encadeadas* ou *cíclicas*; as quais podem envolver vários neurónios e formarem ciclos.

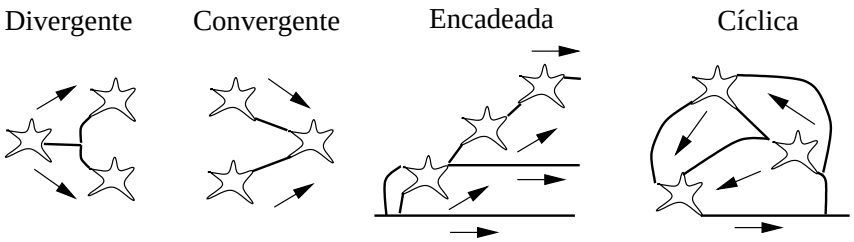


Figura 1.3: Os diferentes tipos de conexões.

Por sua vez, as conexões são constituídas por dois tipos de sinapses: *inibitórias* e *excitatórias*.

1.3.2 Benefícios das *RNA*s

O poder computacional de uma *RNA* alicerça-se em dois aspectos fundamentais: numa topologia que premeia o paralelismo e, por outro lado, na sua capacidade de aprendizagem e generalização; i.e., conseguir responder de modo adequado a novas situações com base em experiências passadas. São estas duas características que tornam possível a resolução de problemas que, de outra forma, seriam intratáveis.

Isto não quer dizer que as *RNAs* sejam caixas mágicas que consigam por si dar resposta a qualquer problema. Em vez disso, precisam não raras vezes de serem integradas com outros sistemas ou paradigmas para a resolução de problemas. Convém, ainda, reconhecer que se está ainda muito longe de atingir uma arquitectura que simule o cérebro humano [Haykin, 1999].

No entanto, as *RNAs* apresentam características únicas, que não se encontram em outros mecanismos ou técnicas [Azoff, 1995; Patterson, 1996; Haykin, 1999]:

- *aprendizagem e generalização*; i.e., conseguindo descrever o todo a partir de algumas partes, constituindo-se como formas eficientes de aprendizagem e armazenamento de conhecimento;
- *processamento maciçamente paralelo*; i.e., permitindo que tarefas complexas sejam realizadas num curto espaço de tempo;

- *não linearidade*; i.e, atendendo a que a maioria dos problemas reais a equacionar e resolver são de natureza não linear;
- *adaptabilidade*; i.e., podendo adaptar a sua topologia de acordo com mudanças do ambiente;
- *robustez e degradação suave*; i.e., permitindo processar o ruído ou informação incompleta de forma eficiente, assim como sendo capazes de manter o seu desempenho quando há desactivação de algumas das suas conexões e/ou nodos; e
- *flexibilidade*; i.e., com um grande domínio de aplicabilidade.

1.3.3 A Inteligência Artificial e as *RNAs*

Os sistemas clássicos da *IA* podem ser comparadas às *RNAs*, a três níveis de complexidade [Memmi, 1989]:

- *Explicação.* Na *IA* clássica, o ênfase está na construção de representações simbólicas, admitindo-se a existência de uma representação mental para a cognição. Por outro lado, a ênfase das *RNAs* está no desenvolvimento de modelos de *Processamento Paralelo Distribuído*, procurando-se uma explicação neurobiológica para os fenómenos cognitivos.
- *Estilo de Processamento.* Na *IA* convencional, o processamento é, não raras vezes, sequencial e, mesmo quando não existe uma ordem pré-definida, as operações são executadas *passo a passo.* Este estilo é influenciado pela natureza sequencial da linguagem natural e lógica, assim como da estrutura da máquina de *von Neumann.* Em contraste, o paralelismo é essencial às *RNAs*, sendo a sua fonte de flexibilidade, robustez e imunidade ao ruído.
- *Estrutura de Representação.* A *IA* clássica utiliza uma linguagem simbólica, com uma estrutura quase linguística, onde surgem expressões complexas a partir de símbolos simples. Ao contrário, nas *RNAs* o conhecimento é armazenado na sua estrutura interna, no valor dos pesos das suas conexões, dados na forma de valores numéricos.

1.3.4 Arquitecturas de Rede

A forma como os nodos se interligam numa estrutura de rede é denominada por *arquitectura* ou *topologia.* Existem inúmeros tipos de arquitecturas de *RNAs* ou *topologias*, cada um

com as suas próprias potencialidades. Em geral, caem dentro de duas categorias principais [Haykin, 1999]:

- *Redes Neuronais Unidireccionais (RNUs)*[2]. São organizadas por camadas, pois não existem ciclos, dado que as conexões se propagam sempre numa só direcção (*convergentes* ou *divergentes*). Na sua forma mais simples uma rede é composta por uma *camada de entrada*, cujos valores de saída são fixados externamente, e por uma *camada de saída*. No entanto, pode possuir uma ou mais camadas intermédias, cujos nodos são designados por nodos intermédios (Figura 1.4).

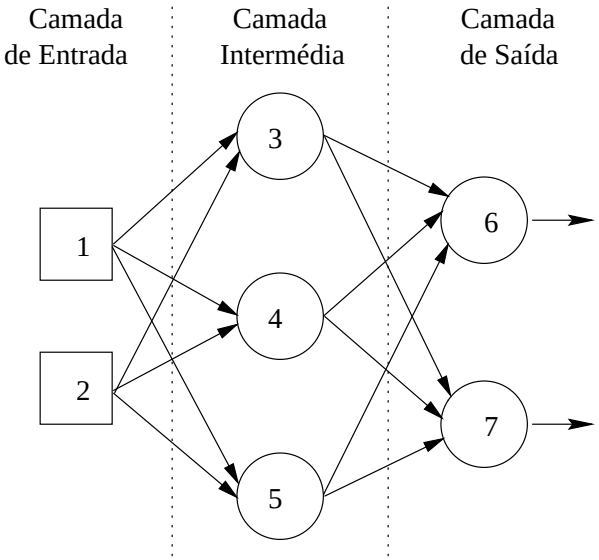


Figura 1.4: Arquitectura de uma *RNU*.

- *Redes Neuronais Recorrentes (RNRs)*. A recorrência existe em sistemas dinâmicos quando uma saída de um elemento influencia de algum modo a entrada para esse mesmo elemento, criando-se assim um ou mais circuitos fechados (Figura 1.5). Ao conter ciclos, as saídas não são função exclusivamente das conexões entre nodos, mas também de uma *dimensão temporal*; i.e., está-se na presença de um cálculo recursivo, que obedecerá naturalmente a uma certa condição de paragem, com a última iteração a ser dada como a saída para o nodo [Rojas, 1996].

[2]Tradução adoptada do inglês *Feedforward*.

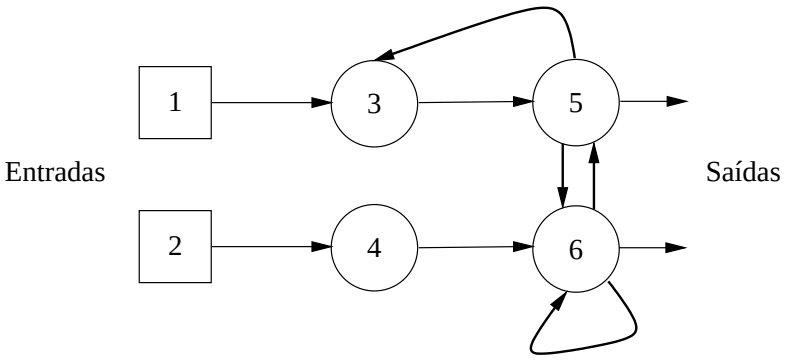


Figura 1.5: Arquitectura de uma *RNR*.

1.3.5 Aprendizagem

Como já foi referido, uma propriedade importante das *RNAs* é a sua capacidade para aprender a partir do seu ambiente. A aprendizagem de uma *RNA* envolve a seguinte sequência de eventos:

- a *RNA* é estimulada por eventos que emanam de um dado ambiente;
- certos parâmetros livres, normalmente *os pesos* das conexões, são alterados em resultado deste estímulo; e
- a *RNA* responde de uma forma diferente a um estímulo que venha desse ambiente, em virtude das alterações sofridas pela sua estrutura interna.

Esta aprendizagem é executada a partir de um conjunto bem conhecido de *regras*, chamado de *algoritmo de aprendizagem* ou *treino*.

A escolha da arquitectura e do método de aprendizagem é influenciada pela tarefa de aprendizagem a ser desempenhada pela *RNA*. Neste contexto existem cinco categorias principais [Patterson, 1996; Haykin, 1999]:

- *Diagnóstico*. Esta é uma tarefa deveras comum em áreas distintas como a *Medicina*, a *Engenharia* ou a *Ecologia*. Trata-se essencialmente de uma tarefa de *classificação*; i.e., exige uma correcta associação entre entradas, que representam dados indicadores de um estado (e.g., sintomas), com o correspondente diagnóstico (e.g., doença). De uma forma geral, as *RNAs* são integradas em sistemas periciais complexos, envolvendo outras técnicas de aprendizagem (e.g., *aprendizagem por regras*).

- *Reconhecimento de padrões.* Formalmente, esta tarefa define-se como o processo pelo qual um sinal/padrão recebido é atribuído a uma de diversas categorias possíveis. Em primeiro lugar, é necessário treinar uma rede, onde os padrões, associados à respectiva categoria, são alimentados à rede de forma repetitiva. Mais tarde, um padrão novo é fornecido à rede, que terá de ser capaz de identificar a categoria correcta, de acordo com a informação assimilada.

- *Regressão/Previsão.* A ideia é conceber uma *RNA* capaz de modelar uma função desconhecida $f(\cdot)$ que se aproxime da função $F(\cdot)$ dada por um conjunto de vectores etiquetados; i.e., compostos por um par entrada-saída, de forma que a distância *euclidiana* seja ínfima para todos as entradas; i.e.:

$$\forall x,\ \|F(x) - f(x)\| < \rho \tag{1.2}$$

  onde ρ representa um valor real próximo de zero. A regressão é uma tarefa perfeita para a aprendizagem supervisionada. Na *previsão*, caso particular da regressão, pretende-se "adivinhar" o comportamento futuro de um dado sistema com base no seu comportamento passado (e.g., *PST*).

- *Controlo.* Esta tarefa envolve um processo ou uma parte crítica de um sistema que tem de ser mantido(a) sobre controle. O principal objectivo do *controlador* é fornecer os apropriados sinais para um dado sistema de controle de forma a que a saída deste acompanhe uma entrada de referência. Como exemplos temos o controlo de *veículos autónomos*, *robôs* ou *processos de fabrico*, onde as *RNAs* têm sido utilizadas com bastante sucesso.

- *Filtragem/Compressão de dados.* O termo *filtro* usualmente designa um dispositivo ou algoritmo que extrai informação de interesse a partir de um conjunto de dados em bruto que apresentam um certo ruído. Por sua vez, a *compressão* envolve uma redução de um espaço n-dimensional para um espaço m-dimensional, sendo $n > m$. Ambas as tarefas se tornam particularmente importantes quando existem quantidades enormes de dados a serem processados (e.g., processamento de imagens).

1.3.6 Perspectiva Histórica

A reflexão filosófica sobre a consciência e qual o órgão que a contém dura desde acerca de dois mil anos, com os filósofos gregos a serem dos primeiros a especular sobre a localização

da alma. O conhecimento sobre como o cérebro funciona é resultado da investigação feita nos últimos 100 anos. Ramón y Cajal[3] em 1894 foi o primeiro a propor a teoria dos neurónios [y Cajal, 1990]. Desde então, numerosos foram os progressos obtidos na compreensão do funcionamento do cérebro humano: os axónios, as dendrites, as sinapses e as activações electro-químicas [Rojas, 1996].

A era moderna das *RNAs* surgiu com o trabalho pioneiro de McCulloch e Pitts [1943], descrevendo um cálculo lógico de *RNAs* que unia os estudos neurobiológicos com a *Lógica-Matemática*. Desde então, um enorme progresso tem sido feito na colaboração de ambas as áreas, com um particular destaque para os anos 80, onde se deu um ressurgimento do interesse em *RNA*s, com contribuições em diversas frentes:

- Hopfield [1982] desenvolve um novo tipo de *RNAs* baseado em *RNRs* com conexões simétricas. No mesmo período, surgem as redes não supervisionadas do tipo *Kohonen* [Kohonen, 1982].
- O famoso algoritmo de *Retro-Propagação* surgiu pelo trabalho de Rumelhart, Hinton e Williams [1986], tornando-se no algoritmo de treino mais popular para as *RNU*s.
- Broomhead e Lowe [1988] descrevem as redes *Radial Basis-Functions (RBFs)*, fornecendo assim uma alternativa às *RNUs*.
- No início dos anos 90, Boser e seus colaboradores [1992] apresentam uma poderosa classe de redes supervisionadas, designadas de *Support Vector Machines*, para a regressão e o reconhecimento de padrões.

Hoje em dia procuram-se não só redes mais eficientes como também melhores algoritmos de treino [Sharda & Rampal, 1996]. Por outro lado, espera-se que a aplicação de *RNAs* a outras áreas do conhecimento se generalize, seja à *Medicina*, à *Economia*, à *Visão por Computador*, à *Robótica* e *Automação*, para além da *Estatística* [Kemsley & Martinez, 1992].

1.4 Computação Genética e Evolucionária

Existe todo um conjunto de estratégias de resolução de problemas que foram influenciadas por processos de evolução das espécies, e em particular pelas descobertas de Darwin[4]. A

[3]Ramón y Cajal (1852-1934). Médico e histologista espanhol, conhecido pelo seu trabalho sobre o cérebro e nervos, isolando o neurónio e descobrindo como os impulsos nervosos são transmitidos às células do cérebro.

[4]Charles (Robert) Darwin (1809-1882). Naturalista inglês, mundialmente célebre pela descoberta da selecção natural, após a análise de espécimes na América do Sul, durante a sua longa viagem a bordo do

famosa *Teoria da Evolução das Espécies* [Darwin, 1859] é definida pela seguinte sinopse:

- Os organismos variam naturalmente em alguns dos atributos que apresentam.
- Tal variação pode levar a diferenças em termos de taxas de sobrevivência ou de reprodução.
- Em todos os organismos existe um excesso do número de progenitores, o que cria uma competição entre si, designada de *luta de sobrevivência*, de modo a produzirem descendência.
- Se a variação que origina diferenças for hereditária, então os indivíduos que produzirem mais descendentes terão também mais filhos que se lhes assemelham, originando uma evolução de espécies, num processo chamado de *selecção natural*.
- Novas espécies surgem a partir de antigas, por mudanças (lentas) nos atributos (incluindo também os de comportamento) e tais mudanças são guiadas pela força estocástica da evolução natural.

A *Teoria da Evolução das Espécies* teve um impacto imediato, não só nas *Ciências Naturais*, como a *Biologia*, mas também na *Filosofia, Religião* e mesmo na sociedade. Várias décadas foram precisas, após o desenvolvimento do computador, para surgir a *Computação Genética e Evolucionária (CGE),* em meados dos anos sessenta. Desde então, vários algoritmos, aplicando os princípios da evolução natural, foram propostos para a resolução de problemas de optimização [Banzhaf et al., 1998].

A *CGE* não resolve problemas via um processo de raciocínio lógico-matemático, do tipo dedutivo; i.e., são antes geradas populações de soluções, candidatas à resolução do problema, que competem entre si. A *CGE* é movida por um processo evolutivo, que à semelhança do que ocorre no meio natural, e não sendo aleatório, progride de modo estocástico. As piores soluções tendem a serem descartadas, enquanto que as que se mostram promissoras na resolução de problemas tendem a sobreviver e a reproduzir-se, contribuindo para a geração de novas soluções [Luger & Stubblefield, 1998].

1.4.1 Inspiração Biológica

Na Natureza, a codificação da informação genética (*genoma*) é feita de forma a permitir uma reprodução *assexuada* ou *sexuada*. No primeiro caso, que abrange a maior parte das

navio "Beagle".

bactérias, são gerados descendentes idênticos. Ao contrário, a reprodução *sexuada,* permite uma mistura de cromossomas, produzindo-se descendentes que contêm uma combinação de informação de cada progenitor. Este tipo de recombinação é normalmente designada de *cruzamento.*

Importa referir ainda que uma das fontes principais de variação genética é o processo de *mutação.* Esta surge de forma probabilística. Por vezes, as mutações são benéficas, embora na maioria dos casos sejam inócuas ou prejudiciais. Com o tempo, a selecção natural tenderá a preservar as primeiras e a eliminar as últimas. A recombinação genética ocorre num ambiente competitivo, onde a *selecção* do indivíduo que acasala é função da sua *aptidão* individual, embora também um certo efeito aleatório esteja envolvido neste processo.

A teoria de Darwin explica o processo de evolução, via pressão da *selecção natural.* No entanto, falta explicar o processo de hereditariedade; i.e., como é que as características dos progenitores são transmitidas aos descendentes. A explicação científica apenas foi encontrada em 1953 com a descoberta por Watson[5] e Crick, da molécula de ADN[6] (Figura 1.6) [Banzhaf et al., 1998].

O ADN, o principal constituinte do genoma, pode ser visto como um conjunto complexo de instruções para criar um organismo. Por exemplo, o ADN humano está comprimido em cerca de três mil milhões de pares de bases. Contudo, o mecanismo pelo o qual o ADN armazena estas instruções é surpreendentemente simples.

A unidade básica do ADN é o *par de bases*, que representa parte da instrução para a criação de um aminoácido. O par de bases é composto por dois *nucleótidos*, que podem conter um de quatro tipos: adenina (A), timosina (T), guanina (G) e citosina (C). Assim, o material genético de qualquer ser vivo pode ser representado por uma sequência de símbolos do alfabeto quaternário: {A, C, G, T}.

Em seres vivos complexos, como o ser humano, esta informação genética encontra-se distribuída por sequências distintas, designadas de *cromossomas*, que se agrupam em pares. Por exemplo, o ser humano contém 46 cromossomas, em 23 pares.

Um *gene* pode ser definido como um conjunto de sequências de ADN que contribuem para a ocorrência de um dado evento. Em 1909, Johannsen[7] compreendeu que era impor-

[5]James Dewey Watson (1928 –). Geneticista Norte Americano, que em 1951 se mudou para a Universidade de Cambridge, onde ele e o inglês Francis Crick delinearam a estrutura molecular do ADN, explicando também o seu mecanismo de replicação.

[6]Sigla do ácido desoxirribonucleico.

[7]Wilhem Ludvig Johannsen (1857 - 1927). Botânico e geneticista, nascido em Copenhaga. As suas experiências pioneiras com feijões contribuíram de forma decisiva para o desenvolvimento da Genética.

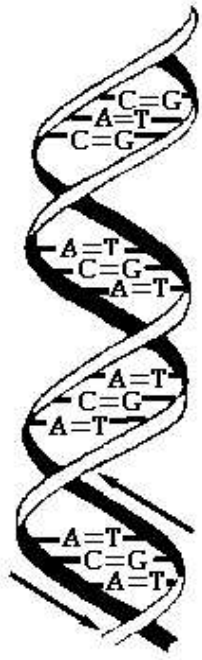


Figura 1.6: Estrutura parcial da molécula de ADN.

tante distinguir entre a aparência de um indivíduo e a sua constituição genética, definindo as palavras *fenótipo* e *genótipo* para os dois conceitos.

Um *genótipo* (ou genoma) compreende todos os genes de um organismo. A hereditariedade processa-se através do genoma, sendo que metade deste é transmitido por um progenitor a um descendente. Por outro lado, o *fenótipo* designa o conjunto de características observáveis de um organismo. A selecção natural actua sobre o fenótipo (e não no genótipo) dado que o fenótipo (o corpo) é necessário para a reprodução biológica.

1.4.2 Variantes da Computação Genética e Evolucionária

A *CGE* contempla toda uma série de técnicas para a resolução de problemas, desenvolvidas de forma independente nos dois lados do Oceano Atlântico, inspiradas no processo de evolução natural próprio dos seres vivos. Por vezes, a diferença entre estes algoritmos não é deveras nítida, distinguindo-se essencialmente pela forma de representação e geração de novas soluções, sendo dada na forma:

- *Algoritmos Genéticos (AGs).* Trata-se da variante de *CGE* mais conhecida, desenvolvida nos Estados Unidos, a partir do trabalho de Holland [1975]. O algoritmo original utilizava uma representação binária fixa, fazendo um uso maciço do opera-

dor de cruzamento. Desde então, várias modificações foram acrescentadas ao longo das três últimas décadas, ao nível da estrutura, forma de representação e operadores genéticos, que contribuíram para o sucesso e esfera de aplicabilidade dos *AGs*. Doravante, o termo *Algoritmos Genéticos e Evolucionários (AGEs)* irá ser utilizado para designar este tipo de modelo mais abrangente.

- *Programação Evolucionária (PE).* Outro algoritmo importante foi desenvolvido por Fogel e seus colaboradores [1966], seguindo uma aproximação em tudo semelhante à seguida em relação aos *AGs*. Aqui, a principal diferença passa pela criação de novas soluções, geradas apenas pelo operador de mutação, não havendo a preocupação de simular os operadores naturais. Historicamente, esta técnica tornou-se única pelo facto não ser limitada pelo tipo de representação, dado que os *AGs* iniciais baseavam-se numa codificação binária. Na *PE*, como não existe recombinação genética, qualquer tipo de representação pode ser facilmente utilizada, desde que se consiga definir um operador de mutação adequado. No entanto, com o processo de generalização que atingiu os *AGs*, as diferenças esbateram-se.

- *Estratégias Evolutivas (EE).* Desenvolvidas nos anos sessenta, devido ao trabalho de Rechenberg [1973], posteriormente aperfeiçoados por Schwefel [1981], dão corpo a outra variante da *CGE*. A primeira versão utilizava apenas um indivíduo, que codificava parâmetros de um problema de optimização. Em cada geração, um novo indivíduo era criado por mutação do indivíduo original. Com o desenvolvimento dos computadores digitais, variáveis contínuas, representadas por números reais, passaram a ser adoptadas. Anos mais tarde, populações de indivíduos foram introduzidas, assim como novos operadores de recombinação genética. As *EE* distinguem-se dos *AGs* originais, fundamentalmente por utilizarem uma representação baseada na numeração decimal e por permitirem a codificação de parâmetros ligados aos operadores de mutação nos indivíduos.

- *Programação Genética (PG).* Nos anos oitenta, dois investigadores, Cramer [1985] e Koza [1989], sugeriram que estruturas em árvores deveriam ser utilizadas para representar genomas. Todavia, foi Koza o primeiro a reconhecer a importância e demonstrar a viabilidade do seu uso em geral. Na *PG,* utiliza-se uma população, constituída por indivíduos de tamanho variável, que codificam um conjunto de instruções para resolver um determinado problema. Cada indivíduo é avaliado pela forma como resolve (ou não) uma dada tarefa. Normalmente, as instruções são codi-

ficadas em árvores, cujos nodos correspondem a dados ou operações sobre estes. As soluções novas são geradas via manipulação de árvores, sendo o operador de mutação raramente utilizado.

1.4.3 Domínios de Aplicabilidade

Desde o seu surgimento, a *CGE* sempre foi (essencialmente) vocacionada para a resolução de problemas de optimização, onde o objectivo é maximizar (ou minimizar) uma dada função. Ora, existe um certo conjunto de problemas denominados de *NP-completos*, onde o domínio das soluções possíveis é tão vasto que a sua apreciação se torna computacionalmente incomportável [Russell & Norvig, 1995]. Dentro deste contexto, a *CGE* apresenta certas diferenças em relação aos métodos convencionais de procura, nomeadamente [Goldberg, 1989]:

- Parte para a procura de uma solução para um dado problema com base num conjunto de possíveis soluções (ou população) e não a partir de uma única solução. Assim, a *CGE* comporta-se como um método de optimização global, capaz de escapar a óptimos locais, sendo indicada para funções multimodais; i.e., que apresentam mínimos (ou máximos) globais. Além disso, o cálculo deste conjunto de soluções é facilmente traduzido em processos de computação paralela.

- Utiliza a informação que apenas provém de uma função objectivo e não de informação auxiliar. Ao contrário das técnicas convencionais de procura, a *CGE* não necessita de derivadas das funções a optimizar, podendo ser aplicada a uma gama mais vasta de problemas.

- Usa regras probabilísticas de transição de estado.

Estas diferenças, quando tomadas em conjunto, contribuem para que a *CGE* encontre soluções onde outros métodos parecem falhar. Assim, a *CGE* tem sido largamente utilizada na resolução de diversos tipos de problemas, em que sobressaem os de:

- *Optimização Numérica de Funções.* Uma instância de um problema de optimização pode ser definida por uma função objectivo, definida na forma $f : D \mapsto \Re$, em que D denota o domínio de f. O objectivo é determinar o valor de $x \in D$ tal que:

$$f(x) \geq f(y), \forall y \in D \tag{1.3}$$

para o caso de um problema de maximização. Nos casos em que D assume características contínuas, está-se na presença de um problema de *Optimização Numérica*. Esta trata-se de uma área por excelência para a aplicação da *CGE*, devido às potencialidades desta lidar com funções multimodais, descontínuas ou com uma componente acentuada de ruído [Michalewicz, 1996].

- *Optimização Combinatória.* Quando D é um conjunto finito (ou infinito mas enumerável), o problema diz-se de *Optimização Combinatória*. Existe um grande número problemas desta natureza que têm sido abordados via a *CGE,* como sejam o *escalonamento de tarefas* [Rocha et al., 2000], a *alocação de cargas com capacidades limite* [Falkenauer, 1994] ou o *encaminhamento de veículos* [Breedam, 1996].

- *Tarefas de Desenho.* Aqui a *CGE* é utilizada para determinar um conjunto de parâmetros no desenho de estruturas (e.g., desenho de pontes ou aviões [Bramlette & Bouchard, 1991], desenho de *RNAs* [Cortez et al., 1999] ou a optimização da arquitectura de multi-processadores [Burgess, 1995]).

- *Aprendizagem.* Um *Sistema de Classificação (SC)* utiliza uma aprendizagem reforçada, que combina um conjunto de regras do tipo *se ... então* (classificadores), um algoritmo de distribuição de créditos e um *AGE*. Os *SC* têm vindo a ser aplicados em variados domínios, nomeadamente no reconhecimento de escrita, diagnóstico médico e vida artificial, entre outros [Santos, 1999].

1.5 Objectivos

A aplicação de modelos computacionais inspirados nos processos de evolução natural próprios dos seres vivos para a *PST* ainda é recente, tendo-se iniciado nos fins dos anos oitenta [Azoff, 1994].

As *RNAs* são candidatas naturais para a *PST*, devido a capacidades como aprendizagem não linear e tolerância ao ruído. Por outro lado, espera-se que o uso da *CGE* para a previsão cresça, motivado por vantagens como uma representação explícita dos modelos e uma procura adaptativa global, que permite escapar de mínimos locais indesejáveis.

Nesta tese, foi decidido adoptar as *RNUs* e os *AGEs* com representações reais, como exemplos de modelos inspirados na Natureza (estas técnicas serão descritas em pormenor no Capítulo 2), dado que também são os modelos mais populares e divulgados.

Uma das dificuldades, que resulta do uso de *RNUs* para a resolução de problemas, reside no tempo dispendido com a procura do melhor tipo de rede ou topologia. Presentemente, a maior parte do uso de *RNAs* para a *PST* ainda se baseia num conjuntos de heurísticas, que dependem um pouco da arte do investigador. Uma via que tem vindo a ser utilizada para tornear este problema, centra-se no uso de *AGEs* para procurar os parâmetros óptimos da *RNA*, nos chamados sistemas de *Redes Neuronais Evolucionárias (RNEs).* Tal solução, também pode ser adoptada aos modelos de *AGEs,* sendo conhecida pelo termo de *Meta-AGE*s, onde um *AGE* de nível superior optimiza os parâmetros de um *AGE* de nível inferior (esta abordagem será descrita no Capítulo 4).

Outra dificuldade relacionada com a adopção dos modelos inspirados na Natureza, surge no processo de aprendizagem, que exige elevados requisitos computacionais. Assim sendo, será que é possível aplicar estes modelos para a previsão em tempo real? No Capítulo 5 é proposto uma arquitectura para este tipo de previsão, que permite utilizar modelos inspirados na Natureza, através da definição de uma janela de visibilidade dos dados.

Dentro deste contexto, a presente tese procura:

- estudar a melhor forma de utilizar os modelos inspirados na Natureza para a *PST*, com especial relevo para o uso das *RNUs* e dos *AGEs*;
- explorar a optimização dos modelos de previsão, em termos de combinação de *AGEs* com *RNAs* e *AGEs* com *AGEs*, tendo em vista a sua aplicação na *PST*;
- analisar os resultados obtidos pelo sistema em séries temporais reais de diferentes tipos e domínios, incluindo a previsão em tempo real; e
- comparar os resultados obtidos com os conseguidos a partir de técnicas convencionais.

1.6 Organização da Tese

A tese encontra-se estruturada em 5 capítulos, organizados da seguinte forma (excluindo o capítulo introdutório):

Capítulo 2

Modelos Inspirados na Natureza

Fornece-se uma descrição sobre dois tipos de modelos de aprendizagem inspirados na Natureza: Redes Neuronais Unidireccionais e Algoritmos Genéticos e Evolucionários. Para cada

um destes modelos apresenta-se a sua constituição e modo de funcionamento. Por fim, são descritos os ambientes de programação que permitem desenvolver aplicações informáticas para a resolução de problemas utilizando os modelos em causa.

Capítulo 3

Modelos de Previsão de Séries Temporais

São explicados os conceitos básicos da análise de séries temporais, sendo apresentadas as séries temporais objecto de estudo. Fornece-se uma descrição sobre os métodos de previsão convencionais e inspirados na Natureza, sendo ambos aplicados na modelação das séries temporais.

Capítulo 4

Optimização de Modelos de Previsão

É explicada a necessidade de optimização de processos computacionais para a resolução de problemas a partir de modelos inspirados na Natureza, sendo apresentadas as principais estratégias de procura. Fornece-se uma descrição das Redes Neuronais Evolucionárias e dos Meta-Algoritmos Genéticos e Evolucionários, sendo ambas estas técnicas aplicadas para a optimização de tais processos. Por último, é realizada uma comparação entre os diferentes modelos considerados como objecto de estudo, em termos de desempenho, assim como dos tempos de computação.

Capítulo 5

Previsão em Tempo Real

São definidos os diferentes horizontes temporais para o processo de previsão, incluindo o caso particular da previsão em tempo real. É apresentada uma arquitectura para sistemas de previsão em tempo real, sendo efectuado um conjunto de experiências com os diferentes modelos de aprendizagem.

Capítulo 6

Conclusões e Trabalho Futuro

Realiza-se o fecho do trabalho desenvolvido, mediante um processo de análise e discussão dos resultados obtidos, lançando-se ainda as bases para trabalho futuro.

Capítulo 2

Modelos Inspirados na Natureza

> "Living organisms are consummate problem solvers. They exhibit a versatility that puts the best computer program to shame."
>
> – John H. Holland, 1975.

Fornece-se uma descrição sobre dois tipos de modelos de aprendizagem inspirados na Natureza: Redes Neuronais Unidireccionais e Algoritmos Genéticos e Evolucionários. Para cada um destes modelos apresenta-se a sua constituição e modo de funcionamento. Por fim, são descritos os ambientes de programação que permitem desenvolver aplicações informáticas para a resolução de problemas utilizando os modelos em causa.

Como foi referido no capítulo anterior, existe todo um conjunto de técnicas computacionais que foram influenciadas por processos de evolução natural, próprios dos seres vivos. O funcionamento do sistema nervoso central motivou o desenvolvimento das *Redes Neuronais Artificiais (RNAs)*, enquanto que a evolução natural influenciou a criação da *Computação Genética e Evolucionária (CGE)*.

Neste capítulo serão descritos em pormenor duas técnicas populares, que servem de exemplo para cada um dos movimentos computacionais inspirados na Natureza: as *Redes Neuronais Unidireccionais (RNUs)* e os *Algoritmos Genéticos e Evolucionários (AGEs)*.

2.1 Redes Neuronais Unidireccionais

2.1.1 Neurónio Artificial ou Nodo

Um *nodo*, termo usado para distinguir entre um neurónio natural e artificial, é a unidade de processamento chave para a operação de uma *RNA*. Embora existam diversos tipos de nodos, em princípio, comporta-se como um comparador que produz uma saída quando o efeito cumulativo das entrada excede um dado valor limite. Um nodo pode ser dissecado de acordo com o exposto na Figura 2.1 [Rojas, 1996]:

- *Um conjunto de conexões* (w_{ij}), cada uma etiquetada por um *peso*; i.e., um número real ou binário (embora seja mais comum na forma real) que tem um efeito excitatório para valores positivos e inibitório para valores negativos. Assim, o *sinal* ou *estímulo* (x_j) como entrada da conexão é multiplicado pelo correspondente peso w_{ij}, onde i denota o nodo objecto de estudo e j o nodo de onde partiu o sinal. Pode ainda existir uma conexão extra, denominada de *bias*, cuja entrada toma o valor $+1$, que estabelece uma certa tendência ou inclinação no processo computacional; i.e., adiciona uma constante (w_{i0}) para que se estabeleçam as correctas condições operacionais para o nodo.

- Um *integrador,* que reduz os n argumentos de entrada (estímulos) a um único valor. Nas *RNUs* é utilizada a função *adição* ($\sum$), pesando todas as entradas numa combinação linear.

- Uma *função de activação* (fa), que pode condicionar o sinal de saída, introduzindo uma componente de não linearidade no processo computacional.

Em termos formais tem-se que este nodo é descrito pelas seguintes equações:

$$u_i = \sum(1 \times w_{io}, x_1 \times w_{i1}, x_2 \times w_{i2}, ..., x_n \times w_{in}) \tag{2.1}$$

$$s_i = fa(u_i) \tag{2.2}$$

para um nodo i com n entradas e uma saída, onde u_i representa o ganho do nodo i e s_i a *saída* do nodo.

A Tabela 2.1 mostra algumas das funções de activação mais utilizadas, onde k denota a *inclinação* da função, *mod* o resto de uma divisão inteira e $sign(x) = \frac{x}{|x|}$, se $x \neq 0$ e

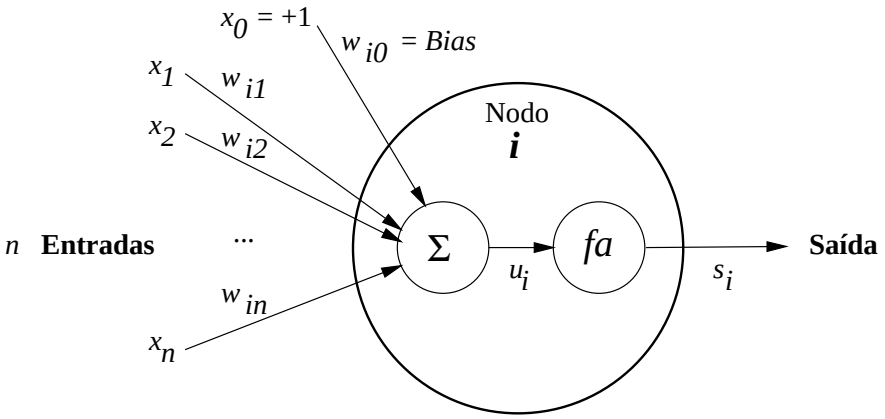


Figura 2.1: Estrutura geral de um nodo.

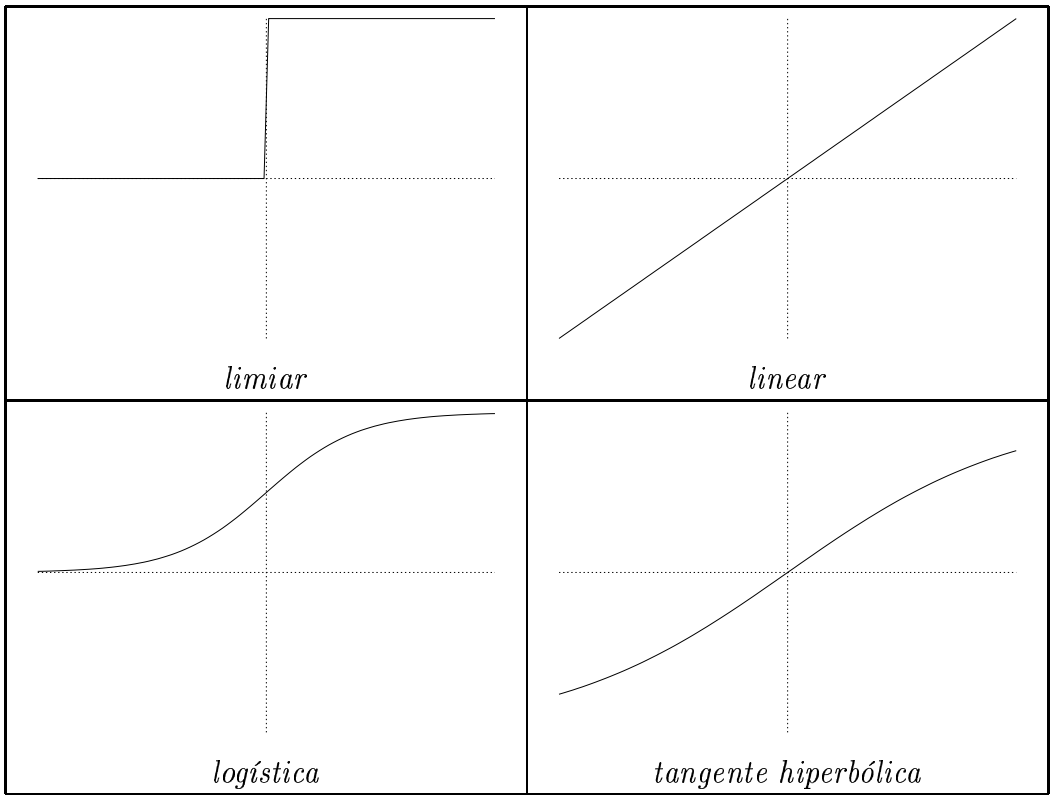


Figura 2.2: Exemplos de funções de activação.

$sign(0) = 0$ (Figura 2.2) [Azoff, 1994]. A primeira função, também designada por função de *Heaviside,* é normalmente utilizada em nodos do tipo McCulloch-Pitts [1943], em que a saída toma o valor +1 apenas se o ganho for não-negativo, de acordo com uma filosofia do *tudo ou nada.* Em seguida, aparecem duas outras funções lineares, com a última a ter como contradomínio o intervalo $[0, 1]$.

Tabela 2.1: Funções de activação.

Nome	**Função**	**Contradomínio**
limiar	$\begin{cases} 1 & , u_i \geq 0 \\ 0 & , u_i < 0 \end{cases}$	$\{0, 1\}$
linear	u_i	$]-\infty, +\infty[$
por troços	$\begin{cases} 1 & , u_i \geq 0.5 \\ ku_i & , -0.5 < u_i < 0.5 \\ 0 & , u_i \leq -0.5 \end{cases}$	$[0, 1]$
logística	$\frac{1}{1+e^{-ku_i}}$	$[0, 1]$
tangente hiperbólica	$\tanh(ku_i)$	$[-1, 1]$
seno	$\sin(u_i \ mod \ 2\pi)$	$[-1, 1]$
coseno	$\cos(u_i \ mod \ 2\pi)$	$[-1, 1]$
gaussiana	$e^{\frac{-u_i^2}{2k^2}}$	$[-1, 1]$
quadrada	$-sign(u_i)u_i^2$	$]-\infty, +\infty[$

A não linearidade é definida em termos das restantes funções, onde uma especial atenção deve ser dada à função *logística*, também conhecida por função *sigmoid*. Esta função, cuja forma é aproximada por um "S", é de longe a função mais utilizada no uso de *RNAs*. É uma função crescente que exibe um balanceamento gracioso entre um comportamento linear e não linear (Figura 2.3) [Jordan, 1995]. Ao se variar a *inclinação* (k) obtêm-se funções com diferentes declives; no limite, em que k se aproxima do infinito, a função tende para a *limiar*.

2.1.2 A Arquitectura Unidireccional

As *RNUs* constituem uma das mais importantes e populares classes de *RNAs*, com um vasto leque de aplicabilidade, utilizáveis em problemas de *memória associativa*, *classificação*, *reconhecimento de padrões* e *regressão*, entre outros [Patterson, 1996].

Em termos formais, estas são definidas pelo tuplo (E, I, S, C, F) [Gallant, 1993; Rojas, 1996] (Figura 2.4), contendo:

- um conjunto de *nodos de entrada* (E), onde pontificam os estímulos enviados a partir

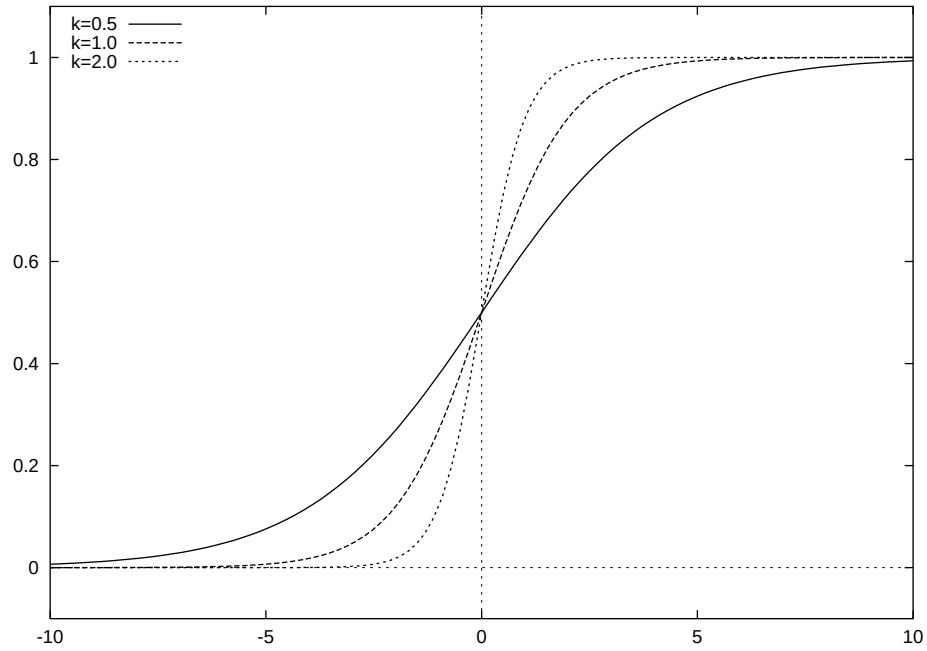


Figura 2.3: A função *sigmoid* para inclinações de $k = 0.5$, $k = 1.0$ e $k = 2.0$.

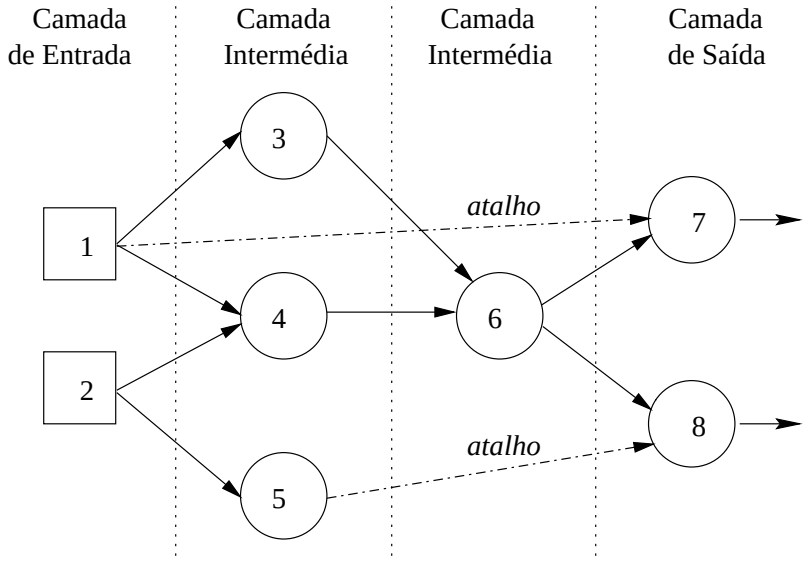


Figura 2.4: Estrutura de uma *RNU* com a topologia *2-3-1-2*.

do ambiente;

- um conjunto de *nodos intermédios* (I), unidades internas de processamento, que aumentam a capacidade de aprendizagem de tarefas complexas;

- um conjunto de *nodos de saída* (S), que devolvem a resposta da rede;

- um conjunto de *conexões pesadas unidireccionais* (C), definidos pelo tuplo (i,j, w) ou abreviadamente w_{ij}, em que $i \in I \cup S$, $j \in E \cup I$, $j < i$ e $w \in \Re$; e

- um conjunto de *funções de activação* (F), normalmente do tipo não linear e diferenciável, sendo a função *logística* uma das mais utilizadas (Tabela 2.1).

O sinal de entrada propaga-se assim para a frente, através da rede, camada por camada, não existindo ciclos. Nas redes multicamada as unidades *Intermédias* (I) dividem-se em c subconjuntos, ou camadas, $(I_1, I_2, \cdots, I_c)$. É comum representar as camadas pela forma $|E| - |I_1| - \cdots - |I_c| - |S|$, onde $|X|$ denota a cardinalidade do conjunto X.

Uma *RNU* designa-se por *completamente interligada* quando contém todas as ligações possíveis entre os nodos de duas camadas adjacentes (Figura 1.4). Caso contrário, designa-se por *parcialmente interligada*. Por vezes existem ligações directas entre as entradas e os neurónios de saída ou que "saltam" camadas (Figura 2.4). Estas conexões designam-se por *atalhos* [Prechelt, 1994].

A não linearidade, a existência de nodos intermédios e o seu alto grau de conectividade tornam a arquitectura *RNU* muito poderosa como uma máquina de aprendizagem. Por outro lado, são estas mesmas características que dificultam um estudo de complexidade ao processo de aprendizagem [Haykin, 1999].

A topologia ideal (em termos de camadas, nodos e conexões) de uma *RNU* depende fundamentalmente do problema a ser resolvido. Em geral, tenta-se reduzir o número de escolhas (ou espaço de procura) adoptando certos pressupostos, como o uso de redes completamente interligadas, evitando-se neste caso a preocupação sobre que conexões devem existir. Também é comum adoptar-se apenas uma camada intermédia, ficando a escolha dos nodos intermédios a ser decidida por regras empíricas. Trata-se de uma estratégia que leva a que se esteja perante diferentes soluções para o problema, sem um critério de decisão contratualizado, dado que não se atribui grande importância ao modelo em termos de desempenho, em parte devido ao enorme poder das *RNUs*. Estratégias mais elaboradas passam por procedimentos automáticos de procura, que serão descritos no Capítulo 4.

2.1.3 O Algoritmo de Retro-Propagação

Dentro dos algoritmos de treino supervisionados, o mais popular e mais usado é o algoritmo de *Retro-Propagação (RP)*[1] ou seus derivados [Gallant, 1993]. Este algoritmo prevalece como um marco para a comunidade das *RNAs,* já que constitui um método eficiente de computação para o treino de *RNUs*, procurando o mínimo da função de erro no espaço de procura dos pesos das ligações entre nodos, baseando-se em métodos de gradiente descendente. A combinação de pesos que minimiza a função de erro é considerada a solução para o problema de aprendizagem. Dado que este método exige o cálculo do gradiente, torna-se necessário que a função de erro (ξ) seja contínua e diferenciável, o que acontece quando se usam funções de activação diferenciáveis [Rojas, 1996].

A aplicação do algoritmo de *RP* pode ser descrita na forma [Rumelhart et al., 1986]:

- *Em frente,* onde o vector de entrada (x) é fornecido aos nodos de entrada da rede, propagando-se em frente, camada por camada, de acordo com as equações 2.1 e 2.2, em que o integrador utiliza a função adição ($\sum$); i.e.,

$$u_i = \sum_j s_j w_{ij} \tag{2.3}$$

  em que $s_0 = 1$ (valor de entrada da conexão de *bias*), $s_1 = x_1, \cdots, s_n = x_n$, para n nodos de entrada ($n = |E|$). Em seguida, calcula-se o erro, com base na função de custo, normalmente dada por:

$$\xi = \frac{1}{2} \sum_{k \in S} (y_{k-n_{ei}} - s_k)^2 \tag{2.4}$$

  onde y denota o vector de saída desejado, S o conjunto de nodos de saída da rede e n_{ei} o somatório dos nodos de entrada e intermédios ($n_{ei} = |E| + |I|$). Durante este passo, os pesos da rede estão fixos.

- *Retro-Propagação,* onde o erro é propagado para atrás, desde a saída até os nodos de entrada. De seguida, os pesos são ajustados segundo a regra de *Widrow-Hoff*. Para um único peso tem-se que:

$$\Delta w_{ij}(t) = -\eta * \frac{\delta \xi}{\delta w_{ij}}(t) \tag{2.5}$$

[1] Tradução do inglês *Back-Propagation.*

em que t refere-se a uma iteração[2] do algoritmo de treino e η denota a taxa de aprendizagem. As derivadas parciais são calculadas segundo o clausulado:

$$\frac{\delta E}{\delta w_{ij}} = \frac{\delta \xi}{\delta s_i}\frac{\delta s_i}{\delta w_{ij}} \tag{2.6}$$

onde

$$\frac{\delta s_i}{\delta w_{ij}} = fa'(u_i)sj \tag{2.7}$$

Para se obter $\frac{\delta \xi}{\delta s_i}$, ou seja a influência da saída s_i do nodo i no erro global ξ, é necessário atender ao tipo de nodo, ou seja:

$$\frac{\delta \xi}{\delta s_i} = \begin{cases} -(y_{i-n_{ei}} - s_i) & , \ i \in S \\ \sum_{j \in succ(i)} \frac{\delta \xi}{\delta s_j} fa'(u_j) w_{ji} & , \ i \notin S \end{cases}$$

onde $succ(i)$ representa o conjunto dos nodos j da camada com que o nodo i se relaciona (ou estabelece ligações).

Antes de se iniciar o treino de uma rede, há que proceder à escolha dos valores iniciais dos pesos associados às ligações entre nodos, que deverão ser pequenos e gerados de forma aleatória. Pode-se então partir para o treino da rede, começando-se por seleccionar um caso de treino, na forma iterativa, ou todos os casos, na forma *em lote*[3]. Normalmente, convém que esta selecção seja aleatória. Em seguida, calcula-se o gradiente e ajustam-se os pesos. Uma *iteração* termina quando todos os casos disponíveis tiverem sido considerados.

O processo é dado como findo pela aplicação de critérios de paragem (e.g., quando as mudanças nos pesos e na função de erro forem insignificantes). De notar que o algoritmo pode convergir para um mínimo local (Figura 2.5); na prática, porém, constata-se que quando se parte de um número elevado de casos de treino, esta questão não se coloca, ou então não se assume como um problema sério.

2.1.4 Alterações ao Algoritmo de Retro-Propagação

A apresentação do algoritmo de *RP* alterou o panorama da investigação em *RNUs* e desde então têm surgido uma miríade de novos algoritmos de treino. Esta explosão deve-se a duas razões. Por um lado, o algoritmo de *RP* é pesado, de convergência lenta. Por outro

[2]Em inglês, utiliza-se o termo *epoch*.
[3]Conhecido em inglês pelo termo *batch*.

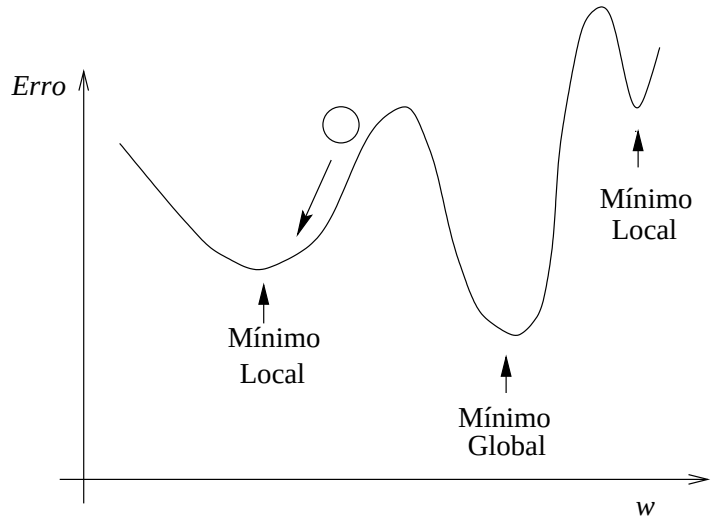


Figura 2.5: Mínimos locais e globais [Gallant, 1993].

lado, baseia-se no gradiente descendente, pelo que todas as técnicas de optimização não linear do gradiente podem ser aplicadas [Bose & Liang, 1996].

No pior dos casos, a aprendizagem de *RNAs* é *NP-completa*; i.e., o esforço computacional envolvido cresce de um modo exponencial com o aumento dos parâmetros livres (pesos). Assim, existe espaço para a proposta de alternativas que possam acelerar o processo de aprendizagem, embora seja sempre possível enganar o "melhor" método com uma dada tarefa de aprendizagem, desde que se conheçam os seus pontos fortes e fracos [Rojas, 1996]. Por conseguinte, diversas variantes rápidas do algoritmo de *RP*, que se baseiam no uso de uma topologia fixa, foram propostas nos últimos anos.

Todavia, também existem formas simples de acelerar o algoritmo de *RP,* nomeadamente as seguintes [Haykin, 1999]:

1. *Taxa de Aprendizagem.* Quanto mais pequena for η menores vão ser as mudanças nos pesos das conexões, de modo que a procura do mínimo global será favorecida pelo uso de saltos mais suaves. O problema é que desta forma tem-se uma aprendizagem mais lenta. Por outro lado, se se aumentar em demasia o valor de η, então os elevados saltos nas mudanças dos pesos poderão provocar instabilidade no treino (e.g., movimento oscilatório). Uma forma de aumentar a taxa de aprendizagem sem provocar instabilidade é alterar a regra de aprendizagem de modo que esta inclua um termo de *momentum*:

$$\Delta w_{ij}(t) = -\eta * \frac{\delta \xi}{\delta w_{ij}}(t) + \mu \Delta w_{ij}(t-1) \tag{2.8}$$

onde μ representa a constante de *momentum,* usualmente dada por um valor positivo escolhido do intervalo $[0, 1]$. Este termo de *momentum* funciona como uma memória que acelera descidas do gradiente, tendo um efeito estabilizador em situações oscilatórias.

2. *Modo de Treino.* Durante a aprendizagem, uma *iteração* corresponde a uma completa apresentação de todos os exemplos de treino à rede. Convém que a apresentação seja aleatória, de forma a que não exista uma tendência para valorizar certos casos de treino. Esta apresentação pode ser efectuada de duas formas:

   - *modo sequencial*, onde se calculam os ajustamentos aos pesos logo após a escolha de um exemplo de treino; e
   - *modo em lote* ou *por iteração*, onde os ajustamentos aos pesos só são efectuados após a apresentação de todos os casos de treino.

   O primeiro modo de treino é menos exigente em termos de memória local, para além da apresentação aleatória dos casos de treino tornar difícil que o algoritmo fique preso em mínimos locais, sendo deveras útil quando o conjunto de dados de treino é elevado e existe bastante informação redundante. Em contraste, o modo *em lote* melhora a estimativa do vector de gradiente, sendo também um processo mais fácil de se paralelizar.

3. *Escolha dos Pesos Iniciais.* Uma boa escolha dos valores iniciais dos pesos pode acelerar bastante o processo de aprendizagem. O problema esta em definir o que é uma boa escolha. Valores elevados podem originar uma saturação da rede, atrasando o processo de aprendizagem. No entanto, valores demasiado pequenos podem levar a que o algoritmo de *RP* opere numa área demasiado plana por volta da origem da superfície de erro. Assim quer valores demasiado elevados ou demasiado pequenos devem ser evitados. Uma boa estratégia é gerar valores aleatórios com uma média de zero que dependam do número de conexões de um nodo. Por exemplo, Gallant [1993] sugere o intervalo $[\frac{-2}{k}; \frac{2}{k}]$, para um nodo com k entradas.

2.1.5 Algoritmos de Adaptação Local

As variantes do algoritmo de *RP* podem ser classificadas em duas categorias: de adaptação global ou local [Riedmiller, 1994]. Os primeiros utilizam um conhecimento global do

estado completo da rede, como a direcção de todo o vector de actualização dos pesos. Em contraste, os últimos são baseados na informação específica de um peso, como o comportamento temporal da sua derivada parcial. Este tipo de estratégia está mais próxima, em termos computacionais, a um modelo neuronal, sendo mais facilmente paralelizável. Para além disso, estas técnicas tendem a ser mais eficazes e robustas, apesar de usarem menos informação.

Neste contexto, importa descrever dois algoritmos importantes de adaptação local: o *QuickProp* e o *RPROP*:

QuickProp

Fahlman [1988] propôs uma variante do *RP* chamada *QuickProp (QP)*, que parece levar a uma rápida convergência na maioria das situações, sendo por isso bastante utilizado. Trata-se de um método de segunda ordem, que se combina com certas heurísticas. Parte de duas assunções que nem sempre se poderão verificar:

- o erro *versus* a curva do peso para cada conexão pode ser aproximado por uma parábola aberta para cima; e
- a mudança no peso não é afectada por todos outros pesos que estão a ser alterados.

O algoritmo comporta-se como o algoritmo tradicional de *RP*, utilizando uma combinação da mesma regra de aprendizagem (equação 2.5) e com a seguinte regra de aprendizagem:

$$\Delta w_{ij}(t) = \frac{DE(t)}{DE(t-1) - DE(t)} \Delta w_{ij}(t-1) \tag{2.9}$$

em que $DE(t) = \frac{\partial \xi(t)}{\partial w_{ij}(t)}$. Para além disso, de modo a evitar elevadas variações, o salto actual dos pesos é limitado a um máximo de ν vezes em relação ao salto anterior. Assim, este algoritmo contém dois parâmetros: η, taxa de aprendizagem para o gradiente descendente e ν, um limitador do tamanho do salto, tendo o valor padrão de 1.75.

RPROP

O nome do algoritmo ***RPROP*** vem do inglês "***R****esilient back****PROP****agation***", tratando-se de uma variante sofisticada do algoritmo de *RP* [Riedmiller & Braun, 1993].

A regra de cálculo é dada por :

$$\Delta_{ij}(t) = \begin{cases} \eta^+ \times \Delta_{ij}(t-1) & , \ \frac{\delta\xi}{\delta w_{ij}}(t-1) \times \frac{\delta\xi}{\delta w_{ij}}(t) > 0 \\ \eta^- \times \Delta_{ij}(t-1) & , \ \frac{\delta\xi}{\delta w_{ij}}(t-1) \times \frac{\delta\xi}{\delta w_{ij}}(t) < 0 \\ \Delta_{ij}(t-1) & , \ \frac{\partial\xi}{\partial w_{ij}}(t-1) \times \frac{\partial\xi}{\partial w_{ij}}(t) = 0 \end{cases} \tag{2.10}$$

para $0 < \eta^- < 1 < \eta^+$. Baseados em considerações de ordem teórica e empírica, estes factores foram fixados em $\eta^+ = 1.2$ e $\eta^- = 0.5$. Os pesos das ligações entre neurónios podem então ser alterados de acordo com a relação:

$$\Delta w_{ij}(t) = \begin{cases} -\Delta_{ij}(t) & , \ \frac{\delta\xi}{\delta w_{ij}}(t) > 0 \\ +\Delta_{ij}(t) & , \ \frac{\delta\xi}{\delta w_{ij}}(t) < 0 \\ 0 & , \ \frac{\delta\xi}{\delta w_{ij}}(t) = 0 \end{cases} \tag{2.11}$$

Estudos comparativos mostraram que este algoritmo converge mais rapidamente que outros algoritmos do género [Riedmiller, 1994]. Além disso, depende apenas de dois parâmetros, Δ_0 e Δ_{max}, cuja escolha de valores não é muito crítica, sendo um algoritmo deveras robusto; i.e., regra geral, não é necessária nenhuma alteração aos valores aconselhados para Δ_0 e Δ_{max} ($\Delta_0 = 0.1$ e $\Delta_{max} = 50.0$).

2.1.6 Critérios de Paragem

O treino de uma *RNA* deveria ser terminado quando se atinge o valor mínimo do erro. Todavia, esquemas de paragem mais elaborados têm de ser adoptados, dado que a função de erro pode apresentar diversos mínimos locais (Figura 2.5). Por exemplo, Prechelt [1998] defende o uso de dois critérios de paragem para *RNAs*:

- *Falha de Progresso do Treino.* A ideia é tentar avaliar o progresso do treino; i.e., qual a diminuição do erro sobre os casos de treino, ξ_{tr}, durante uma dada *faixa de treino*, de ft iterações. A função de progresso, avaliada em cada ft iterações, toma a forma:

$$P_{ft}(t) = 1000 * (\frac{\sum_{t' \in t-ft+1 \ldots t} \xi_{tr}(t')}{ft * min_{t' \in t-ft+1 \ldots t} \xi_{tr}(t')} - 1) \tag{2.12}$$

  O progresso no treino é elevado nas fases instáveis, onde o erro nos casos de treino sobe em vez de descer. No entanto, tende para zero a longo prazo, a não ser que o treino se torne oscilante. O treino é parado se $P_{ft}(t) < \phi$, em que ϕ é uma medida de erro de estado estacionário.

- *Número Máximo de Iterações.* Este critério é aplicado quando o critério anterior falha, de modo a garantir que o treino termine. Assim, o treino é terminado se $t < \varphi$, em que φ define o número máximo de iterações.

2.1.7 Pré-processamento dos Dados

O pré-processamento dos dados deve ser seriamente considerado antes de treinar uma rede. Embora não seja essencial em certos casos, torna-se vital noutros. São várias as operações em que se pode desdobrar o pré-processamento [Azoff, 1995; Sarle, 1999]:

- *Verificação da Integridade dos Dados.* Trata-se de um procedimento de validação dos dados, verificando-se se existem erros de alguma espécie.

- *Representação dos Dados.* Esta fase envolve uma conversão de dados. Dado que as *RNAs* apenas lidam com números, os dados terão de ser codificados, por exemplo, utilizando uma representação binária ou decimal.

- *Escalonamento dos Dados.* O objectivo deste procedimento é realizar uma transformação nos dados de modo a acelerar e melhorar o processo de aprendizagem. Este escalonamento depende do tipo de dados:

  (a) *Entradas.* O escalonamento das variáveis de entrada tem diversos efeitos conforme os distintos algoritmos de aprendizagem. De um modo particular, algoritmos de gradiente descendente, como o conhecido *RP*, são bastante sensíveis ao escalonamento. Assim, cada variável de entrada deve ser escalonada de forma a que a sua média sobre o conjunto de casos de treino esteja à volta do valor zero, pelo que o escalonamento para o intervalo $[-1, 1]$ funcionará melhor do que para o intervalo $[0, 1]$. Existem duas formas eficientes de realizar este escalonamento:

  – *Média Zero e Desvio Padrão 1*

  $$y = \frac{x - \overline{x}}{std(x)} \tag{2.13}$$

  onde $\overline{x}$ denota a média de x e $std(x)$ o desvio padrão de x; i.e., $std(x) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}}$, para n exemplos de treino.

  – *Valor Mínimo -1 e Valor Máximo 1*

  $$y = \frac{x - (max + min)/2}{(max - min)/2} \tag{2.14}$$

em que min e max representam o mínimo e máximo da variável x.

(b) *Saídas.* Existem duas fortes razões para escalonar as saídas. Em primeiro lugar, quando se usa mais de uma saída e se a função de erro é sensível à escala, como acontece no caso da aprendizagem do gradiente descendente, então a diferença de escalas entre as saídas pode afectar a forma como a rede aprende. Por exemplo, se uma saída tem valores entre 0 e 1 enquanto outra tem valores entre 0 e 1000000, então o algoritmo de treino irá dispender a maior parte do seu esforço na aprendizagem da segunda saída. Assim, as saídas que têm a mesma importância devem ser escalonadas para a mesma escala. Neste caso, deve-se proceder a um escalonamento com a mesma escala ou desvio padrão (usando por exemplo a equação 2.13). Em segundo lugar, pode-se querer ajustar as variáveis de saída aos valores do contradomínio da função de activação (e.g., a função *logística* produz valores de saída dentro do domínio $[0, 1]$). Neste caso, é importante conhecer os limites máximos e mínimos da variável em vez dos valores máximos e mínimos dos casos de treino. Neste caso poder-se-á utilizar a seguinte fórmula:

$$y = \frac{(x - L_{min})(B - A)}{L_{max} - L_{min}} + A \tag{2.15}$$

para um escalonamento no domínio $[A, B]$, sendo L_{max} e L_{min} os limites máximo e mínimo da variável. Se a variável de saída tiver limites desconhecidos, então é preferível utilizar a função *linear* (ou outra não limitada) para os nodos de saída.

- *Redução da Dimensão.* O número de elementos num vector de entrada, a sua *dimensão* ou *cardinalidade*, pode atingir valores demasiado elevados, principalmente quando não existe um grande conhecimento sobre a sua natureza e relevância. Em geral, redes com um elevado número de nodos de entrada têm uma aprendizagem mais difícil. Uma das formas de aliviar este problema consiste na redução da dimensão dos vectores de entrada, considerando-se apenas a informação relevante.

- *Filtragem de Ruído.* Por vezes, torna-se útil o uso de técnicas de filtragem para a eliminação de ruído, suavizando a função de aprendizagem, facilitando o treino. Como exemplo tem-se a *análise espectral,* uma técnica muito usada na manipulação de imagens, onde um sinal de entrada é decomposto em ondas sinusoidais, pela *transformada de Fourier.*

2.1.8 Capacidades e Limitações

Uma rede *RNU* treinada por *RP* pode ser vista como uma forma prática para efectuar uma qualquer correspondência não linear. A questão que surge é: dada uma função g desconhecida que faz uma correspondência entre padrões de um espaço n-dimensional para um espaço m-dimensional, $g : \Re^n \rightarrow \Re^m$, será possível encontrar uma *RNU* que realize a correspondência exigida ou pelo menos se aproxime dela com alguma eficácia? A resposta vem do *Teorema Universal de Aproximação de Funções* que estipula que uma camada intermédia é suficiente para uma *RNU* computar uma aproximação de uma qualquer função contínua, embora o teorema não afirme que esta estrutura seja óptima em termos de tempo de aprendizagem e generalização [Haykin, 1999]. Também está provado que com duas camadas intermédias até funções descontínuas podem ser representadas.

Outro aspecto importante diz respeito ao tempo de aprendizagem. A aprendizagem implica a procura dos elementos desconhecidos de uma *RNA,* normalmente pelo ajuste dos pesos. Ora, a aprendizagem numa rede com 100 pesos é bastante mais pesada em termos computacionais do que a de uma rede com 10 pesos, sendo uma relação bem maior do que o factor 1 : 10 poderia sugerir. Seria muito útil que o tempo de aprendizagem fosse limitado por uma função polinomial sobre o número de variáveis. No entanto, tal não se verifica.

Está igualmente provado que o problema geral de aprendizagem em *RNAs* é intratável; i.e., não pode ser resolvido de forma eficiente para todas as instâncias. Não é conhecido um algoritmo que consiga realizar a aprendizagem num tempo polinomial, sendo improvável que tal algoritmo venha a existir. Assim, diz-se que em geral o problema de aprendizagem em *RNAs* é *NP*-completo [Rojas, 1996].

Contudo, devido às capacidades de correspondência não linear entre padrões, as *RNUs* têm sido utilizadas com sucesso em diversas domínios, incluindo *Processos de Controlo Industrial*, *Robótica e Automação*, *Estatística*, *Medicina*, *Gestão e Controlo da Produção*, *Economia*, entre outros [Patterson, 1996; Sharda & Rampal 1996]. Normalmente, para um bom conjunto de dados de treino, uma rede com apenas uma (ou quanto muito duas) camada(s) intermédia(s) serve para aprender uma dada tarefa com mestria.

2.2 Algoritmos Genéticos e Evolucionários

2.2.1 O Algoritmo Original

A ideia original, conhecida por *Algoritmo Genético (AG)* e proposta por Holland, pode ser resumida na Figura 2.6 [Goldberg, 1989]. O *AG* inicia-se com uma *população* de *indivíduos*, soluções possíveis para o problema, gerados de forma aleatória. A informação que um indivíduo disponibiliza, e que atende aos valores dos parâmetros do problema em equação, é representada por um *cromossoma*, de um modo em tudo análogo à estrutura vigente no ADN. Um cromossoma é, por sua vez, composto por um conjunto de *genes* (caracteres). Um valor possível para um *gene* é designado por *alelo*. A qualidade de cada solução (cromossoma) é medida por uma função chamada de *aptidão*[4], sendo os indivíduos avaliados de acordo com esta.

Em cada ciclo, parte-se da população actual (P_t). À semelhança do que se passa no mundo real, os indivíduos da *população* são sujeitos a uma série de operações, tais como *selecção* dos progenitores, *cruzamento* entre pares de progenitores e *mutação* dos descendentes. Como resultado deste processo, uma nova geração de indivíduos é criada. O processo repete-se, então, durante várias gerações, até satisfazer uma dada a condição de paragem, definida, por exemplo, pelo número máximo de gerações (T).

INÍCIO
Iniciar o tempo ($t := 0$)
Gerar a população inicial, de forma aleatória (P_0).
Avaliar P_0.
ENQUANTO ($t < T$) FAZER
Seleccionar os progenitores a partir da população actual (P_t).
Aplicar *cruzamento* aos progenitores para gerar descendentes.
Aplicar *mutação* aos descendentes.
Avaliar a nova população (P_{t+1}).
Actualizar o tempo ($t := t + 1$).
FIM

Figura 2.6: Estrutura genérica de um *AG*.

O processo de representação do fenótipo (solução) em genótipo (cromossoma), é designado de *codificação*, sendo operação crucial para uma boa resolução dos problemas utilizando *AGs* (Figura 2.7). Holland [1975] defendia que a melhor forma de representação seria a codificação binária simples dos valores possíveis para cada parâmetro.

[4] Termo conhecido em inglês por *fitness*.

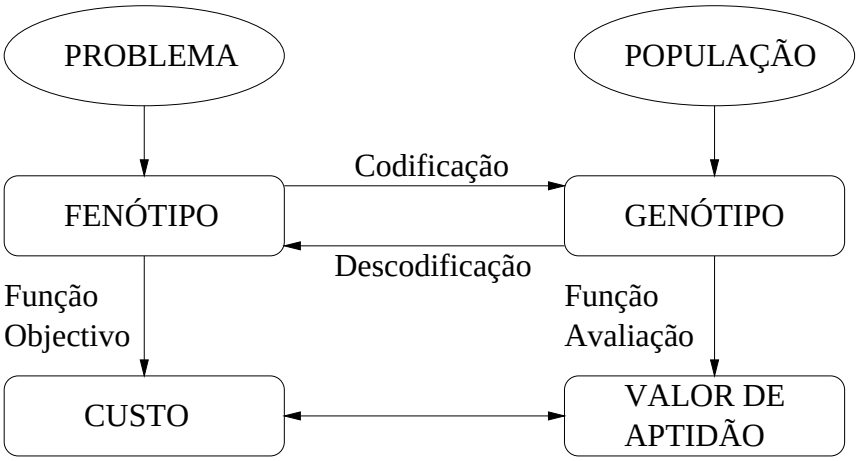


Figura 2.7: Processo de representação do fenótipo [Rocha, 1997].

Outro aspecto importante diz respeito à forma como a função de *aptidão* é calculada, a partir da representação escolhida. Esta função recebe o genótipo de um indivíduo e retorna um valor numérico (*aptidão*), que mede a qualidade da solução para resolver o problema, sendo definida formalmente por:

$$Aptidão : Genótipo \rightarrow \Re \tag{2.16}$$

Na situação ideal, a função de avaliação deverá ser definida de modo suave e regular, de modo a que cromossomas semelhantes tenham valores de aptidão próximos. Além disso, a solução óptima não deve ser um ponto isolado de aptidão elevado; i.e., estar rodeada de soluções com baixos valores de aptidão.

2.2.2 O Modelo Genérico

Desde que foi criado, o modelo original de Holland sofreu diversas alterações, de modo a aumentar a sua gama de aplicabilidade e sucesso. A estrutura deste modelo generalista, designado por *Algoritmo Genético e Evolucionário (AGE)* [Rocha, 1997] é representada na Figura 2.8. Como diferenças principais tem-se que:

- aceitam-se outros tipos de representação genética;
- a população inicial pode ser criada por outros métodos que não o aleatório;
- admite-se uma hipótese de condição de paragem mais flexível;

- generaliza-se o conceito de operador genético, que engloba as operações de cruzamento e mutação;
- são definidos novos parâmetros que controlam o processo de renovação da população, ou seja, como se passa de uma geração de indivíduos para a seguinte; e
- os processos de selecção são generalizados.

INÍCIO
Iniciar o tempo ($t := 0$)
Gerar a população inicial (P_0).
Avaliar P_0.
ENQUANTO (não CONDIÇÃO FINAL) FAZER
Seleccionar os progenitores a partir da população actual (P_t).
Aplicar *operadores genéticos* aos progenitores para gerar novos indivíduos.
Avaliar os novos indivíduos.
Seleccionar os sobreviventes de P_t para a geração seguinte (P_{t+1}).
Seleccionar e inserir os novos indivíduos em P_{t+1}.
Actualizar o tempo ($t := t + 1$).
FIM

Figura 2.8: Estrutura genérica de um *AGE*.

2.2.3 Representações Genéticas

Uma das questões importantes que se vêm discutindo desde a criação dos *AGEs,* é sobre se a representação binária será a mais adequada para um dado tipo de problema. Holland defendia que sim, justificando-se com o argumento de ser aquela que maximiza o número de *esquemas*[5] por bit de informação [Goldberg, 1989]. Por outro lado, este tipo de representação facilita o tratamento teórico, potenciando a definição de operadores genéticos.

Todavia, a representação binária possui alguns inconvenientes. Problemas que requeiram uma alta precisão, ou contenham um grande número de variáveis, originam cromossomas de elevadas dimensões, tornando o espaço de procura demasiado extenso para que um *AGE* tenha sucesso. Mais recentemente, tem sido defendido que a utilização de diferentes alfabetos de representação, mais próximos da estrutura do problema a resolver, facilita o desenvolvimento de operadores genéticos mais eficazes [Michalewicz, 1996].

[5]Um *esquema*, na representação binária, é definido por um padrão de símbolos pertencentes ao conjunto {0,1#}, onde o símbolo # unifica com valor 0 ou 1. Assim, por exemplo, o cromossoma '0010' contém, entre outros, os esquemas '0#10' e '00##'.

Assim, ao longo dos últimos anos tem existido uma proliferação de diferentes representações genéticas, umas mais genéricas, outras mais próximas do problema alvo. De entre estas, tem-se a *Representação de Valores Reais (RVR)* para variáveis contínuas, que tem dado bons resultados em áreas de aplicação distintas (e.g., treino de *RNAs* [Yooni et al., 1994]). Neste caso, a *RVR* está mais próxima do espaço de soluções do problema, não havendo necessidade de conversões, dado que um gene codifica directamente uma variável do problema.

2.2.4 Operadores Genéticos

Na versão abrangente, um operador genético pode ser definido como uma função matemática que aceita pg progenitores e certos parâmetros de controlo, gerando dc descendentes [Neves et al., 1999]:

$$Operador : Progenitores^{pg} \times Controlo \mapsto Descendentes^{dc} \tag{2.17}$$

Como foi referido, no caso dos *AGs*, teríamos dois operadores, o *cruzamento* ($pg = dc = 2$) e a *mutação* ($pg = dc = 1$), aplicados em fases diferentes do algoritmo. No modelo genérico, cada *AGE* pode definir os operadores que utiliza, sendo todos eles aplicados numa mesma fase, designada de *recombinação genética*. Como vantagem deste esquema temos a possibilidade de utilizar diferentes operadores numa mesma aplicação, podendo-se definir operadores diferentes do comum (e.g., definir um cruzamento que retorna apenas um descendente).

A operação de *cruzamento* tenta simular a reprodução sexuada dos seres vivos. O objectivo é conseguir nos descendentes uma combinação do material genético dos progenitores. Deste modo, pretende-se preservar a informação genética, testando-se diferentes combinações. Nesta tese, será adoptada a forma mais comum deste operador, definida através de uma função que recebe dois progenitores, um conjunto de parâmetros de *Controlo*, retornando dois novos indivíduos (descendentes):

$$Cruzamento : Indiv \times Indiv \times Controlo \mapsto Indiv \times Indiv \tag{2.18}$$

onde $Indiv$ denota um indivíduo.

Na representação em binário, consideram-se, em princípio, três possibilidades de cruzamento: cruzamento de *um* e *dois pontos*, e cruzamento *uniforme* [Goldberg, 1989]. O primeiro operador é o adoptado por Holland, para a versão original dos *AGEs*. O *cruza-*

mento de um ponto selecciona, de modo aleatório, uma posição de corte nos cromossomas dos progenitores. Assim, são geradas quatro sub-sequências, que serão ligadas em pares, de modo a criarem-se dois novos cromossomas. No *cruzamento de dois pontos*, os cromossomas são vistos como anéis, com os extremos ligados, sendo o seu funcionamento explicado na Figura 2.9. Por fim, tem-se o cruzamento uniforme, onde é obtida uma máscara binária, gerada aleatoriamente. Nas posições onde a máscara contém o valor 1 (um), o descendente toma o valor do gene do primeiro progenitor, caso contrário, toma o valor do gene do segundo progenitor. O segundo descendente é gerado pela operação inversa.

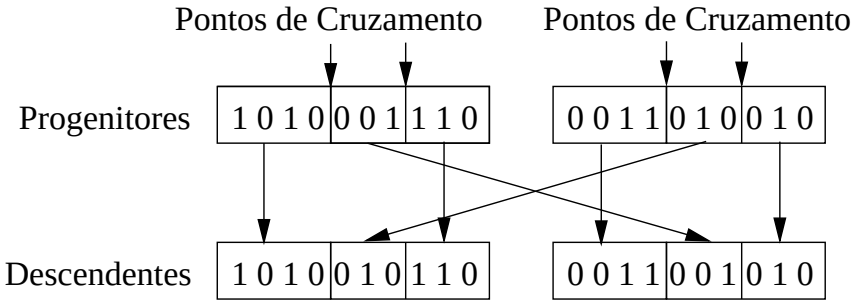


Figura 2.9: Cruzamento de dois pontos.

O estudo sobre os méritos de cada um destes cruzamentos ainda não produziu resultados conclusivos [Beasley et al., 1993], embora seja normalmente aceite que o cruzamento com dois pontos de corte é mais vantajoso do que o cruzamento de apenas um ponto. Estes operadores também podem ser utilizados noutras representações, apesar de terem sido concebidos para lidar com alfabetos binários. Neste cenário, o cruzamento despoleta apenas uma recombinação de genes, dado que nenhum valor de gene é alterado.

No caso de representações reais, é desejável que novos genes sejam gerados via a operação de cruzamento, de modo a explorar melhor o espaço de procura de soluções. O *cruzamento aritmético* é um exemplo deste tipo de operador [Michalewicz, 1996]. Este operador utiliza uma combinação linear dos progenitores:

$$\begin{aligned} x'_1 &= \lambda x_1 + (1-\lambda)x_2 \\ x'_2 &= \lambda x_2 + (1-\lambda)x_1 \end{aligned} \tag{2.19}$$

onde x_1 e x_2 denotam os vectores que representam o cromossoma dos progenitores, x'_1 e x'_2 os vectores dos descendentes, sendo λ um valor aleatório, pertencendo ao domínio $[0, 1]$. Em geral, cada descendente fica com uma constituição genética mais próxima de um dos progenitores. Também existe a possibilidade de ambos descendentes serem iguais

($\alpha = 0.5$), embora tal situação seja deveras improvável.

A *Mutação* (Mut) é dada por um operador unário, que recebe um indivíduo e parâmetros de controlo, de modo a gerar uma nova solução:

$$Mut : Indiv \times Controlo \mapsto Indiv \tag{2.20}$$

Em geral, este operador origina uma pequena alteração na informação genética do progenitor. A mutação contribui para aumentar a diversidade da população, impedindo que haja alelos que sejam perdidos. Assim, mesmo quando o valor de um dado gene é igual para toda a população, situação impossível de se alterar através do cruzamento tradicional, existe sempre a possibilidade do seu valor ser alterado via uma mutação.

Na representação binária, opera-se sobre um ponto de mutação, sendo o valor alterado para o seu inverso (Figura 2.10). Para as representações reais, este operador poderia ser

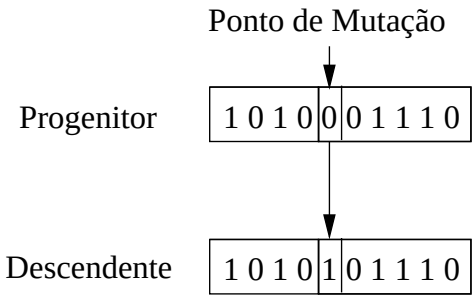


Figura 2.10: Mutação binária.

definido como um valor aleatório, dentro do domínio de valores possíveis para o gene. No entanto, este operador poderá ser demasiado destrutivo, principalmente quando o AGE se encontra num estado adiantado de convergência. Para ultrapassar este inconveniente, Michalewicz [1996] sugeriu o uso de uma perturbação a adicionar (ou subtrair), que deve ser gerada a partir de uma distribuição tomada do foro estatístico.

2.2.5 Renovação da População

O *Tamanho da População (TP)* afecta quer a qualidade da solução final quer o tempo de processamento. Uma *população* pequena é pobre em termos de diversidade dos seus elementos constituintes, pelo que as soluções geradas tendem a não ser as melhores. Por outro lado, uma *população* com um elevado número de indivíduos tem uma maior probabilidade de produzir melhores resultados, embora à custa de um maior esforço computacional.

No modelo original de Holland, é adoptada uma estratégia de *reinserção pura*, onde em cada geração todos indivíduos de uma população eram substituídos por novos indivíduos [Pohlheim, 1996]. Neste caso, a *Taxa de Substituição (TS),* definida como a proporção de indivíduos que é substituída em cada geração, tem um valor de 100%. No lado oposto, temos estratégias que defendem a substituição de um mínimo de indivíduos (e.g., apenas um par[6]) [Whitley, 1989]. A adopção do valor ideal de *TS* ainda se encontra em discussão, não existindo um valor aceite universalmente.

Por vezes, existe o interesse de preservar os melhores indivíduos, ou seja, que estes passem automaticamente para a geração seguinte. O número de indivíduos que auferem deste privilégio é designado de *Valor de Elitismo (VE)*. Outra estratégia consiste na geração de um número excedentário de descendentes, sendo seleccionado um número menor, de acordo com o valor de aptidão de cada indivíduo novo. Este processo é controlado por uma *Taxa de Geração de Descendentes (TGD)*. Por último, também se pode definir o *Número de Progenitores (NP)*, que determina o número de indivíduos seleccionados para reprodução, em cada geração. O modelo geral adoptado engloba então todas estas funcionalidades (*TP, TS, VE, TGD* e *NP),* admitindo a definição de diferentes estratégias (Figura 2.11) [Neves et al., 1999].

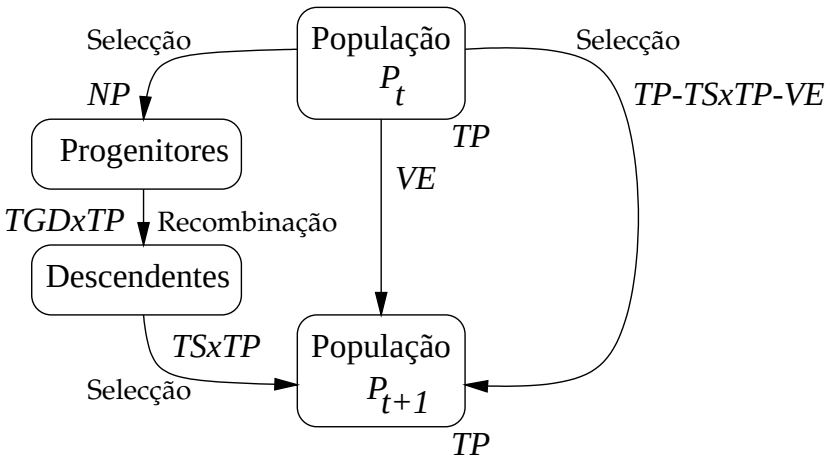


Figura 2.11: Processo de renovação da população.

2.2.6 Processo de Selecção

O processo de selecção deve ser realizado de forma a que os indivíduos com maior aptidão tenham proporcionalmente mais hipóteses de se reproduzir e sobreviver. Existem diversos

[6]Situação que ocorre nos *AGEs* conhecidos por *Steady State*.

métodos de selecção, distinguindo-se pelo modo de conversão do *valor de aptidão* e pelo processo de amostragem.

As técnicas de conversão dos valores de aptidão são utilizadas para calcular um *valor de selecção,* dividindo-se em duas classes principais: de *conversão explícita* e de *conversão implícita*. No primeiro caso é utilizada uma função sobre o valor de aptidão, enquanto que no segundo caso utiliza-se uma transformação indirecta, não obrigando ao cálculo do valor de selecção.

Uma forma simples de converter os valores de aptidão seria utilizar a função identidade. Todavia, à medida que o processo evolutivo decorre, alguns alelos tendem a tornar-se dominantes, pelo que a amplitude dos valores de aptidão diminui, provocando uma convergência lenta. Por outro lado, os indivíduos com um valor de aptidão elevado, constituindo máximos locais mas não valores óptimos, acabarão por dominar a população.

Para colmatar estas dificuldades, desenvolveu-se a *conversão por ordenação*[7] [Whitley, 1989]. Neste método, os indivíduos são ordenados pelo seu valor de *aptidão*, pelo que o valor de selecção depende de uma função, linear ou não, sobre esta ordenação.

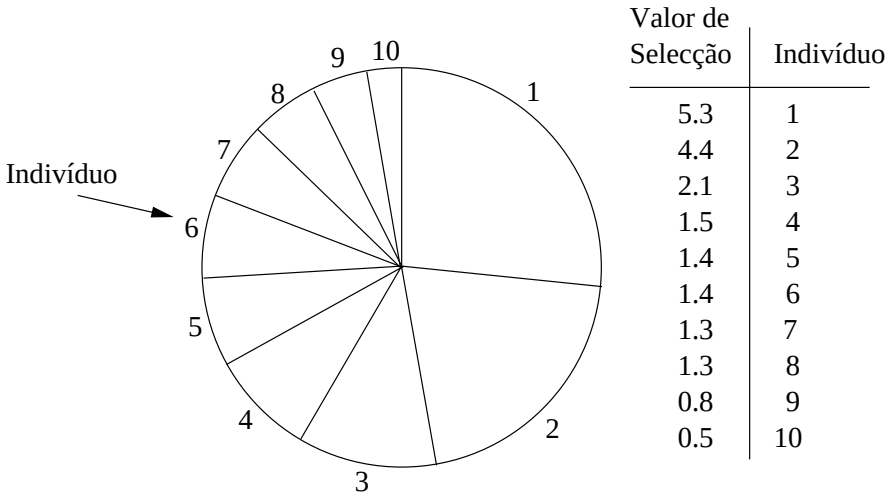
Um dos métodos mais conhecidos para o processo de selecção é a *amostragem estocástica com substituição*[8] [Baker, 1987]. A Figura 2.12 descreve um exemplo desta técnica, para uma população de dez indivíduos. Este método adopta os princípios que estão subjacentes ao funcionamento do jogo da roleta, onde cada indivíduo está associado a um compartimento, com uma área proporcional ao seu *valor de selecção*, sendo estes valores normalizados de modo a que o seu somatório seja a unidade.

Até agora assumiu-se que os indivíduos com maior aptidão são os melhores candidatos à reprodução e sobrevivência. Ora, quando os problemas a resolver são de minimização, devem ser escolhidos os indivíduos com menores valores de aptidão. Assim, neste contexto, torna-se necessário efectuar uma conversão do valor de aptidão, antes de se calcular o seu valor de selecção. Uma forma simples, e das mais utilizadas, consiste em modificar o valor de aptidão para o seu inverso. Quando se utiliza a *selecção por ordenação*, esta conversão poderá ser desnecessária, bastando alterar a ordenação decrescente para crescente.

2.3 Ambientes de Programação

Nesta secção serão explicados os ambientes de programação dos modelos inspirados na Natureza descritos anteriormente, sob um ponto de vista conceptual, guardando-se para o

[7] Conhecido em inglês por *ranking*.
[8] Tradução adoptada do inglês *Roulette Wheel*.



Valor de Selecção	Indivíduo
5.3	1
4.4	2
2.1	3
1.5	4
1.4	5
1.4	6
1.3	7
1.3	8
0.8	9
0.5	10

Figura 2.12: Amostragem estocástica com substituição [Koehn, 1994].

fim da tese uma descrição pormenorizada ao nível de utilização (Apêndice C).

2.3.1 Ambiente de Programação de Redes Neuronais Artificiais

O *Ambiente de Programação de Redes Neuronais Artificiais (APRNA)* foi desenvolvido com o propósito de aumentar a produtividade durante o desenvolvimento de aplicações com *RNAs*. Para tal, decidiu-se adoptar as potencialidades do paradigma da *Programação Orientada ao Objecto (POO)*, que facilita o uso, para além de permitir um desenvolvimento incremental, sendo por isso indicado para o desenvolvimento de sistemas grandes e complexos.

O *APRNA* começou a ser concebido em 1998, como ferramenta de suporte a esta tese. Desde 2000, passou a ser divulgado na disciplina de *Sistemas Inteligentes*[9], tendo sido aplicado pelos seus alunos em diversos projectos (e.g., *Sistemas de Vida Artificial* e o *Reconhecimento Óptico de Caracteres*). Este ambiente foi também utilizado para o *Diagnóstico Médico Baseado em Tomografia Computorizada* (Figura 2.13), obtendo resultados animadores [Alves, 2001].

Em termos de implementação, foi decidido adoptar a linguagem de programação *C++* [Stroustrup, 1991], que embora não seja uma linguagem perfeita de *POO*, foi concebida como uma ferramenta prática para resolver problemas reais, sendo utilizada por uma grande parte da indústria. A razão principal desta escolha deveu-se à sua eficiência (quando

[9]Cadeira do quarto ano lectivo da Licenciatura de Engenharia de Sistemas e Informática, Universidade do Minho.

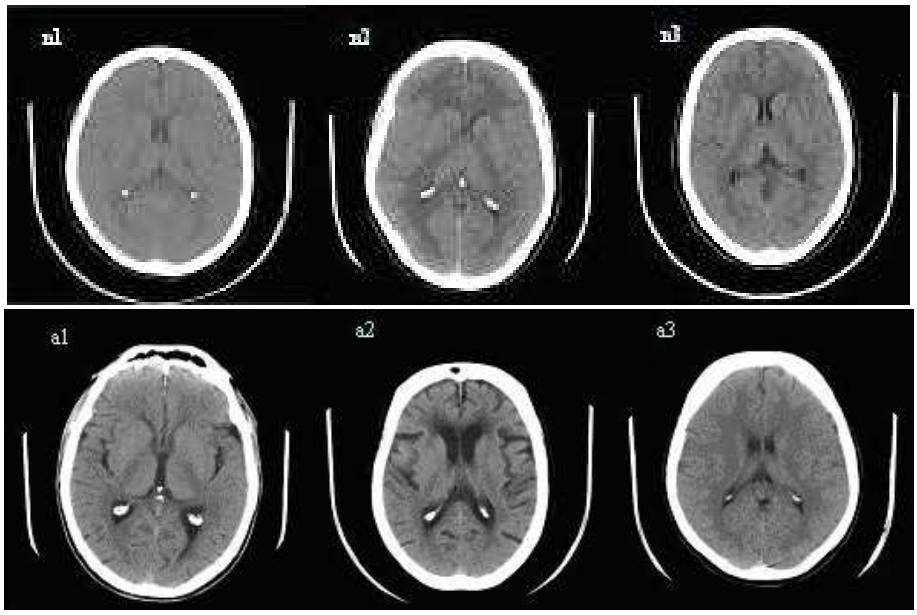


Figura 2.13: Imagens normais (acima) e com atrofia (abaixo).

considerada em relação a outras linguagens *POO,* como por exemplo a linguagem *Java* [Arnow & Weiss, 1998]), dado que a maior parte dos algoritmos de aprendizagem têm grandes requisitos computacionais. Também foi decidido adoptar o sistema operativo *Linux,* por ter uma distribuição gratuita, ser bastante robusto e conter excelentes ferramentas de programação em *C++* (sob licença da *GNU*[10]). Convém ainda referir que todo o código foi escrito em inglês, por uma questão de universalidade.

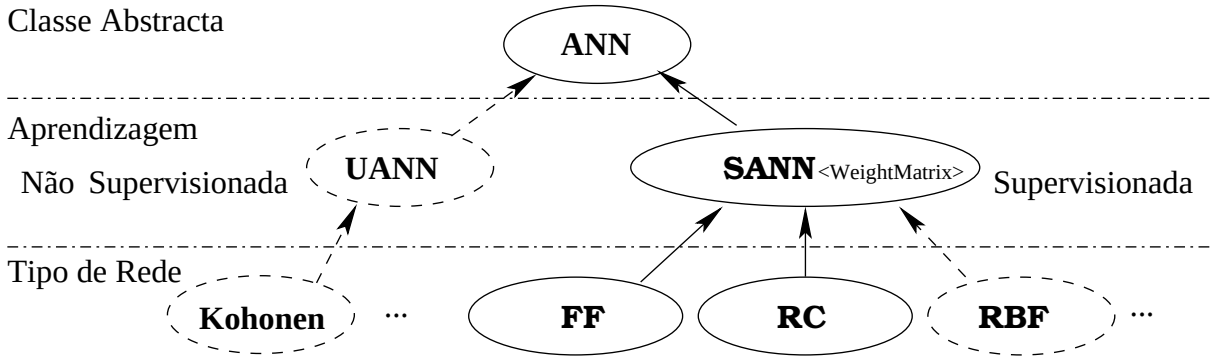


Figura 2.14: Estrutura de classes do *APRNA*.

A hierarquia de classes do *APRNA* está representada na Figura 2.14, onde as classes já

[10] Acrónimo recursivo do inglês "GNU's Not Unix", que designa o primeiro projecto de partilha gratuita de *software*.

desenvolvidas se encontram demarcadas por um tracejado contínuo. Na classe abstracta raiz (**ANN**) são definidas as interfaces comuns. No nível seguinte tem-se o paradigma de aprendizagem, sendo aqui definidos todos os atributos e comportamentos comuns ao paradigma. É de realçar que, apesar de sido contemplada apenas a aprendizagem supervisionada (classe **SANN**), esta estrutura permite uma expansão a outros paradigmas (e.g., aprendizagem não supervisionada). Por fim, no nível inferior, tem-se a definição do tipo de rede. No presente, o sistema é capaz de lidar com *RNUs* (classe **FF**) e *RNRs* (classe **RC**).

Como foi referido na Secção 2.1, uma *RNA* é composta por um conjunto de conexões, nodos e funções de activação, organizados numa dada estrutura. Para representar esta informação, foi decidido adoptar dois componentes básicos de manipulação de dados: vectores e matrizes. O primeiro destes objectos já se encontrava desenvolvido de forma eficaz na biblioteca *STL*[11] [Musser & Saini, 1996], pelo que apenas foi necessário conceber o segundo, definido como um vector de vectores. O *STL* adopta uma funcionalidade do *C++* chamada de *template*, definida pela notação $< Tipo >$, que permite instanciar tipos de dados apenas aquando da construção de um objecto[12].

Dentro deste contexto, uma *RNA* de aprendizagem supervisionada (**SANN**) contém:

- uma *matriz de pesos* (classe **WeightMatrix**) [Gallant, 1993], que define todas as conexões (w_{ij});
- uma *matriz de nodos* (classe **Layers**), que representa a organização entre os nodos ($\{E, I, S\}$); e
- um *vector de funções de activação* (classe **Nodes**), permitindo que diferentes nodos possam utilizar diferentes funções de activação (F).

A Figura 2.15 demonstra como é gerada uma matriz de pesos para uma dada *RNA,* onde o índice dos nodos origem surge na horizontal, pelo que o índice dos nodos destino aparece na vertical.

Esta estrutura de dados é partilhada por todas as subclasses (e.g., **FF** e **RC**). Por uma questão de *especificidade*, ou seja, definir mais informação à medida que se desce de classe, foi associado um *template* à matriz de pesos, de modo que diferentes redes possam utilizar diferentes tipos de matrizes de pesos (e.g., uma *RNU* utiliza uma matriz de pesos

[11] C++ *Standard Template Library.*

[12] Por exemplo, a linha de código: **vector<int> A(5);** define o vector A de números inteiros com um tamanho de 5.

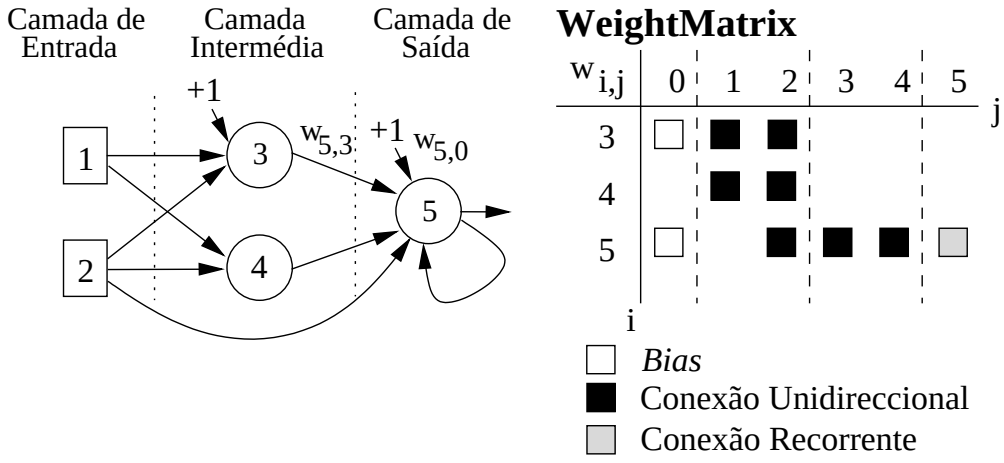


Figura 2.15: Matriz de pesos para uma *RNA*.

unidireccional, enquanto numa *RNR* esta é recorrente). A subclasse **FF** contém todas as operações para a manipulação de *RNUs*, estando disponíveis todos os algoritmos de treino descritos anteriormente (e.g., *RP*, *QP* e *RPROP*).

2.3.2 Ambiente de Programação de Algoritmos Genéticos e Evolucionários

Por sua vez, o *Ambiente de Programação de Algoritmos Genéticos e Evolucionários (APAGE)* [Neves et al., 1999] foi construído dentro de um espírito semelhante; i.e., facilitar a tarefa de desenvolvimento de aplicações com *AGEs* via uma *POO* em *C++*, a correr sobre o sistema operativo *Linux*. Este ambiente foi aplicado com sucesso em áreas tão distintas como o *Problema do Caixeiro Viajante* [Rocha, 1997], *Escalonamento de Tarefas de uma Tipografia* [Rocha et al., 2000] e *a Atribuição de Mesas para Convidados de um Casamento* [Rocha et al., 2001].

O *APAGE* assenta em quatro blocos principais (Figura 2.16), nomeadamente os *indivíduos*, a *população*, os *AGEs* e o *módulo de avaliação*. Cada um destes componentes é caracterizado por uma hierarquia de classes, construídas de tal forma que atributos e comportamentos comuns são definidos nas classes superiores, existindo um processo de especialização, à medida que se desce para as subclasses, redefinindo ou acrescentando novos atributos/comportamentos.

Ao nível da hierarquia do indivíduo, tem-se uma raiz que é uma classe abstracta com um *template* para o genótipo. Assim, as representações genéticas são definidas de uma forma simples, atribuindo-se ao *template* a desejada representação (Figura 2.17).

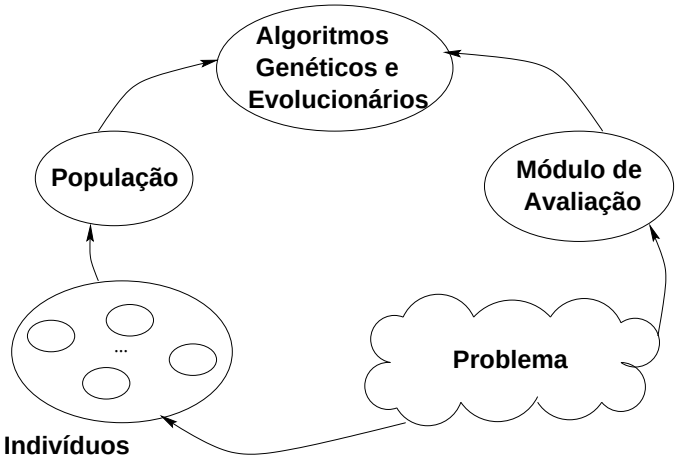


Figura 2.16: O arquétipo do *APAGE* [Neves et al., 1999].

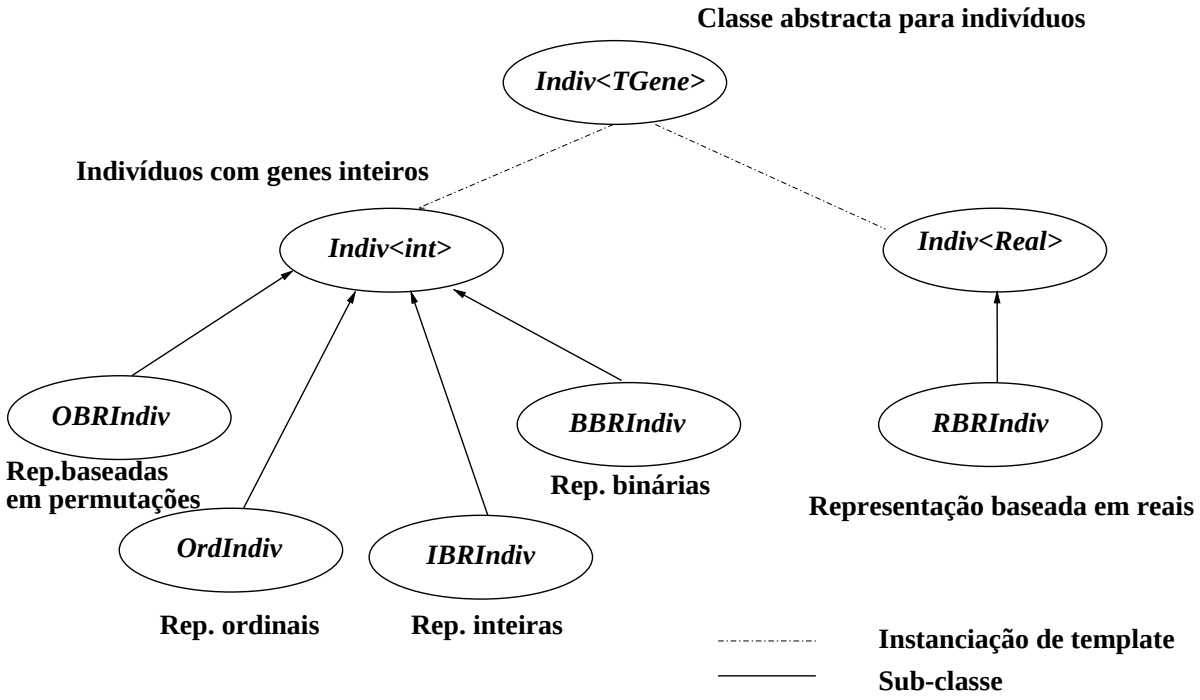


Figura 2.17: A hierarquia de classes do indivíduo [Neves et al., 1999].

Nos restantes módulos, foram seguidas estratégias similares, sendo estabelecidos os comportamentos básicos que podem ser redefinidos, como o processo de renovação da população ou a estrutura do algoritmo geral. Por último, é no modulo de avaliação que o utilizador define o processo de descodificação, ou seja, como construir uma solução com base num cromossoma, e como atribuir o valor de *aptidão* a esta solução.

2.4 Conclusões

Neste capítulo foi descrita a caracterização e funcionamento de duas técnicas inspiradas na Natureza: as *Redes Neuronais Unidireccionais (RNUs)* e os *Algoritmos Genéticos e Evolucionários (AGEs)*.

As *RNUs* são máquinas universais, ideais para uma aprendizagem supervisionada, sendo capazes de aprender uma qualquer correspondência entre conjuntos de entradas e saídas. As *RNUs* contém todas vantagens das *RNAs*, como sejam a aprendizagem não linear, a tolerância ao ruído, a transparência e flexibilidade, sendo aptas para lidar com a resolução de problemas de *classificação* e *regressão*.

A arquitectura, ou topologia, de uma *RNU,* é definida por um conjunto de nodos, agrupados por camadas, conexões unidireccionais pesadas, definidas entre os nodos, e um conjunto de funções de activação. O interesse nas *RNUs* foi estimulado pelo desenvolvimento do algoritmo de aprendizagem de *Retro-Propagação (RP)*, e desde então diversas variantes mais eficientes tem sido obtidas, como o algoritmo de treino *RPROP*. Antes de se treinar uma *RNU,* deve ser considerado um pré-processamento dos dados, por forma a facilitar a aprendizagem.

Por outro lado, os *AGEs* são exemplos de máquinas de aprendizagem que se baseiam em processos de selecção natural próprios dos seres vivos. Os *AGEs* estão talhados para a resolução de problemas de optimização, gerando variantes que competem entre si para sobreviver e procriar novos indivíduos, convergindo para melhores soluções. A ideia original foi proposta por Holland em 1975, e desde então, sofreu diversas alterações, de forma a aumentar o seu leque de aplicações. Os factores centrais para o uso de *AGEs* são: a forma de representação genética, os operadores de recombinação genética, os processos de renovação da população e selecção dos indivíduos, e a função de aptidão.

Finalmente, são descritos os ambientes de programação, ambos concebidos sobre o paradigma da *Programação Orientada ao Objecto,* permitindo um desenvolvimento incremental de aplicações para computador. O *Ambiente de Programação de Redes Neuronais Artifici-*

ais foi desenhado com o intuito de facilitar a concepção de aplicações que utilizem *RNAs*, incluindo as *RNUs*. De um modo similar, o *Ambiente de Programação de Algoritmos Genéticos e Evolucionários* foi delineado para auxiliar o desenho de aplicações que adoptem *AGEs* para a resolução de problemas.

Capítulo 3

Modelos de Previsão de Séries Temporais

> "An unsophisticated forecaster uses statistics as a drunken man uses lamp-posts, for support rather than for illumination."
>
> –After Andrew Lang.

São explicados os conceitos básicos da análise de séries temporais, sendo apresentadas as séries temporais objecto de estudo. Fornece-se uma descrição sobre os métodos de previsão convencionais e inspirados na Natureza, sendo ambos aplicados na modelação das séries temporais.

Hoje em dia assiste-se a uma globalização da economia, e por conseguinte a um acréscimo de concorrência entre as organizações, cujas estruturas, processos e tecnologias, se optimizados, lhe darão vantagens competitivas. No entanto, inovar num ambiente de incerteza pode ser desastroso, pelo que é natural pressupor que as organizações estão interessadas em obter previsões fundamentadas sobre o seu futuro.

Uma das técnicas em uso socorre-se da *Previsão de Séries Temporais (PST)*, que se baseia em observações passadas de uma dada variável. Os métodos convencionais de previsão, desenvolvidos a partir de disciplinas como a *Investigação Operacional* ou a *Estatística*, providenciam previsões precisas, quando estão envolvidos dados lineares. Contudo, quando um elevado grau de não linearidade é introduzido, estas técnicas não são as mais adequadas, pelo que uma alternativa surge com os modelos inspirados na Natureza. Neste capítulo, todos estes modelos serão descritos e aplicados à previsão.

3.1 Análise de Séries Temporais

Uma série temporal é uma colecção de observações registadas $(x_1, x_2, ..., x_t)$ de um modo sequencial. Exemplos de séries temporais surgem nos mais diversos domínios, desde a *Economia*, *Demografia*, *Agricultura* ou *Engenharia*, sendo a *Análise de Séries Temporais* uma área importante da *Estatística* [Chatfield, 1989].

Um modelo de uma série temporal $(\widehat{x_t})$ assume que as observações são dependentes; i.e., padrões passados irão recorrer no futuro, pelo que a série é previsível. Uma série diz-se *determinística* quando é previsível a 100%. Todavia, a maior parte das séries contém um elemento estocástico, sendo o futuro apenas parcialmente determinado por valores passados, pelo que previsões exactas são impossíveis de se obter.

A análise de séries temporais contém vários objectivos, que não são necessariamente idênticos, nomeadamente os indicados a seguir [Weigend & Gershenfeld, 1994]:

- *descrição*, onde se pretende descobrir, com nenhum ou pouco conhecimento à *priori*, determinadas propriedades fundamentais, como o número de graus de liberdade ou quantidade de ruído;

- *explicação,* cujo objectivo é encontrar um modelo que capture características sobre o comportamento de um sistema a longo prazo;

- *previsão,* que almeja prever com precisão a evolução de um sistema a curto prazo; e

- *controlo*, onde se tenta controlar o comportamento de um dado processo, atendendo a certos parâmetros (e.g., qualidade).

De entre estes, a ênfase neste trabalho será dada à *previsão,* uma importante tarefa em áreas tão distintas como vendas, produção ou finanças. Os modelos de previsão tratam o sistema como se de uma caixa negra se tratasse, não tendo como objectivo determinar os factores que afectam o seu comportamento. Existem, porventura, algumas razões para esta postura. A primeira deve-se ao facto de o sistema não ser perfeitamente determinado, sendo difícil identificar as relações de causa-efeito em jogo. A segunda reside no facto de se desejar prever o que acontece e não como acontece; i.e., existem séries temporais reais impossíveis de axiomatizar, tangíveis à previsão (e.g., a previsão da ocorrência de manchas solares ou a previsão do comportamento do mercado de acções [McCluskey, 1993]).

3.1.1 Decomposição

Em geral, as técnicas tradicionais de análise de séries temporais baseiam-se no processo de *decomposição*, onde se identificam os factores que influenciam os valores de uma série temporal, sendo de referir entre estes a tendência e a sazonalidade [Makridakis & Wheelwright, 1989].

A *tendência* dá-nos uma medida do crescimento ou declínio dos valores de uma série. Uma série que possua este tipo de comportamento é chamada de *não estacionária*. Na Figura 3.1 representa-se uma série *não estacionária* típica. Várias forças se conjugam para que esta componente seja comum em situações que têm a ver com o estudo da inflação, das mudanças tecnológicas, do crescimento da população ou dos aumentos de produtividade.

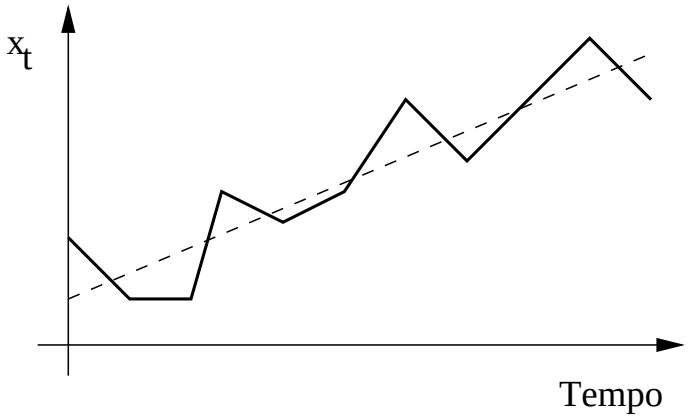


Figura 3.1: Série temporal não estacionária.

A *sazonalidade* corporiza uma flutuação periódica, com uma duração, por exemplo, semanal, mensal ou trimestral. O padrão repete-se, ao longo do tempo, após cada K períodos, sendo K o *factor sazonal*. Esta situação é comum, sendo de referir, por exemplo, a comercialização de produtos que dependem do clima ou que se vendem em determinados períodos do ano. A Figura 3.2 representa uma *série sazonal* com um período de sazonalidade de quatro, correspondendo às quatro estações do ano.

Uma ferramenta estatística importante para a análise de séries temporais é o *coeficiente de autocorrelação*, definido como a correlação entre a série e ela própria (daí o termo auto), deslocada de k períodos de tempo [Box & Jenkins, 1976]:

$$r_k = \frac{\sum_{t=1}^{s-k}(x_t - \overline{x_t})(x_{t+k} - \overline{x_t})}{\sum_{t=1}^{s}(x_t - \overline{x_t})^2} \tag{3.1}$$

onde s representa o tamanho da série e $\overline{x_t}$ o valor médio da série temporal. As autocor-

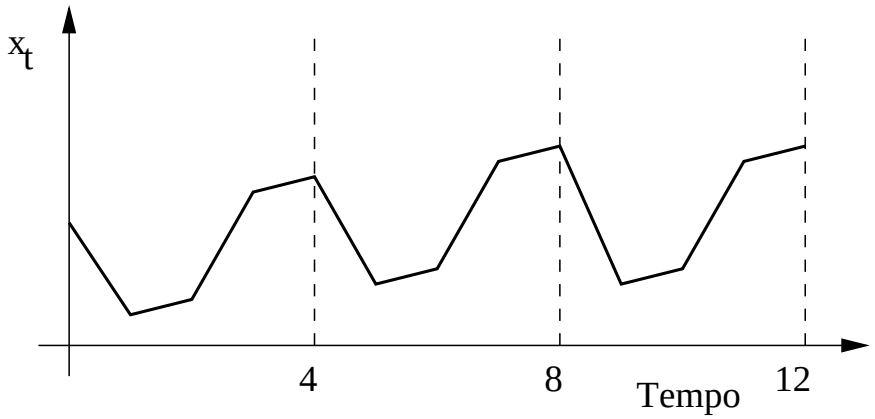


Figura 3.2: Série temporal com factor estação 4.

relações são úteis para testar se uma dada série é previsível e para a decomposição dos principais componentes de uma série, como a *tendência* e a *sazonalidade* (Figura 3.3).

3.1.2 Erro de Previsão

O *erro* de previsão é dado pela diferença entre o valor actual da série e o que foi previsto, na forma:

$$e_t = x_t - \widehat{x_t} \tag{3.2}$$

Para avaliar o desempenho global de um modelo de previsão, torna-se necessário utilizar uma medida do erro, sendo que uma das formas mais utilizas recorre ao quadrado do erro. Em particular, tem-se a *Soma do Quadrado dos Erros (SQE)*, a *Média do Quadrado dos Erros (MQE)*, a *Raiz da Média do Quadrado dos Erros (RMQE)* e a *Média Normalizada do Quadrado dos Erros (MNQE)*:

$$\begin{aligned} SQE &= \textstyle\sum_{i=1}^{L} e^2 \\ MQE &= \frac{SQE}{L} \\ RMQE &= \sqrt{MQE} \\ MNQE &= \frac{SQE}{\sum_{i=1}^{L}(x_t - \overline{x_t})^2} \end{aligned} \tag{3.3}$$

em que L denota o número de previsões a efectuar.

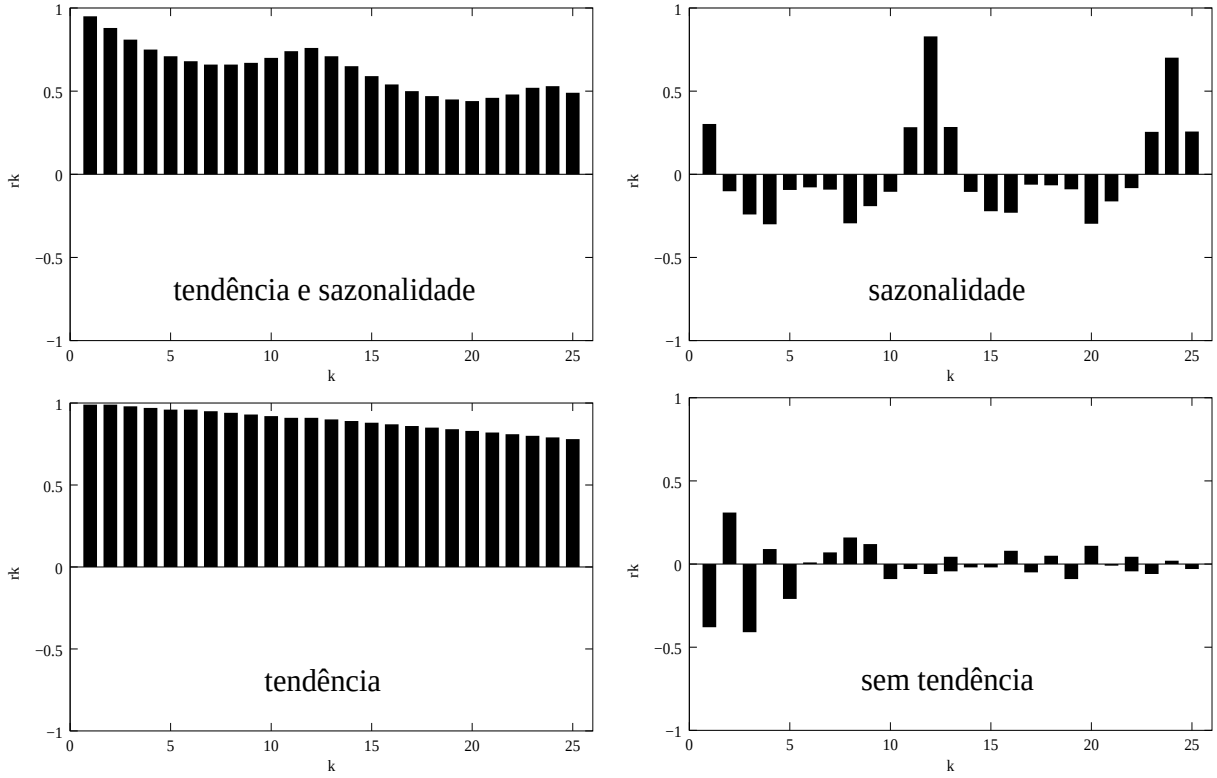


Figura 3.3: Autocorrelações típicas de séries temporais.

3.1.3 Séries Temporais

Para a avaliação dos modelos de previsão, foram escolhidas dez séries temporais, oriundas de domínios e fontes distintas. Estas foram classificadas em cinco categorias, com duas séries por cada classe, que englobam os componentes comuns à maioria das séries temporais, nomeadamente:

Sazonais com Tendência

Trata-se de um tipo muito comum em séries mensais (e.g., vendas). A série turística **passageiros** representa o número mensal de passageiros (em milhares) de uma companhia aérea internacional, entre 1949 e 1960, sendo bastante conhecida e utilizada para demonstrar a validade de métodos de previsão sazonais (Figura 3.4) [Box & Jenkins, 1976]. A segunda série, **papel,** apresenta as mesmas características, reportando-se a vendas de papel em França [Makridakis & Wheelwright, 1989].

Sazonais

Foram escolhidas as séries [Hyndman, 2001] (Figura 3.5):

- **mortes**, relativa ao número mensal de mortes e ferimentos graves em acidentes nas estradas da Grã-Bretanha; e
- **melborne**, uma série meteorológica sobre valores médios mensais, em graus *Celcius*, da temperatura máxima na cidade australiana de Melborne.

Com Tendência

Do livro de Box e Jenkins [1976], foram seleccionadas duas séries não estacionárias (Figura 3.6). A primeira, designada de **preços**, contém os preços diários de fecho das acções da IBM, entre 17 de Maio de 1961 e 2 de Novembro de 1962, enquanto que a segunda, chamada **química**, envolve leituras da temperatura de um processo químico, de minuto em minuto.

Não Lineares

Para esta categoria foram escolhidas séries reais, não lineares, que normalmente surgem em processos físicos [Hyndman, 2001] (Figura 3.7):

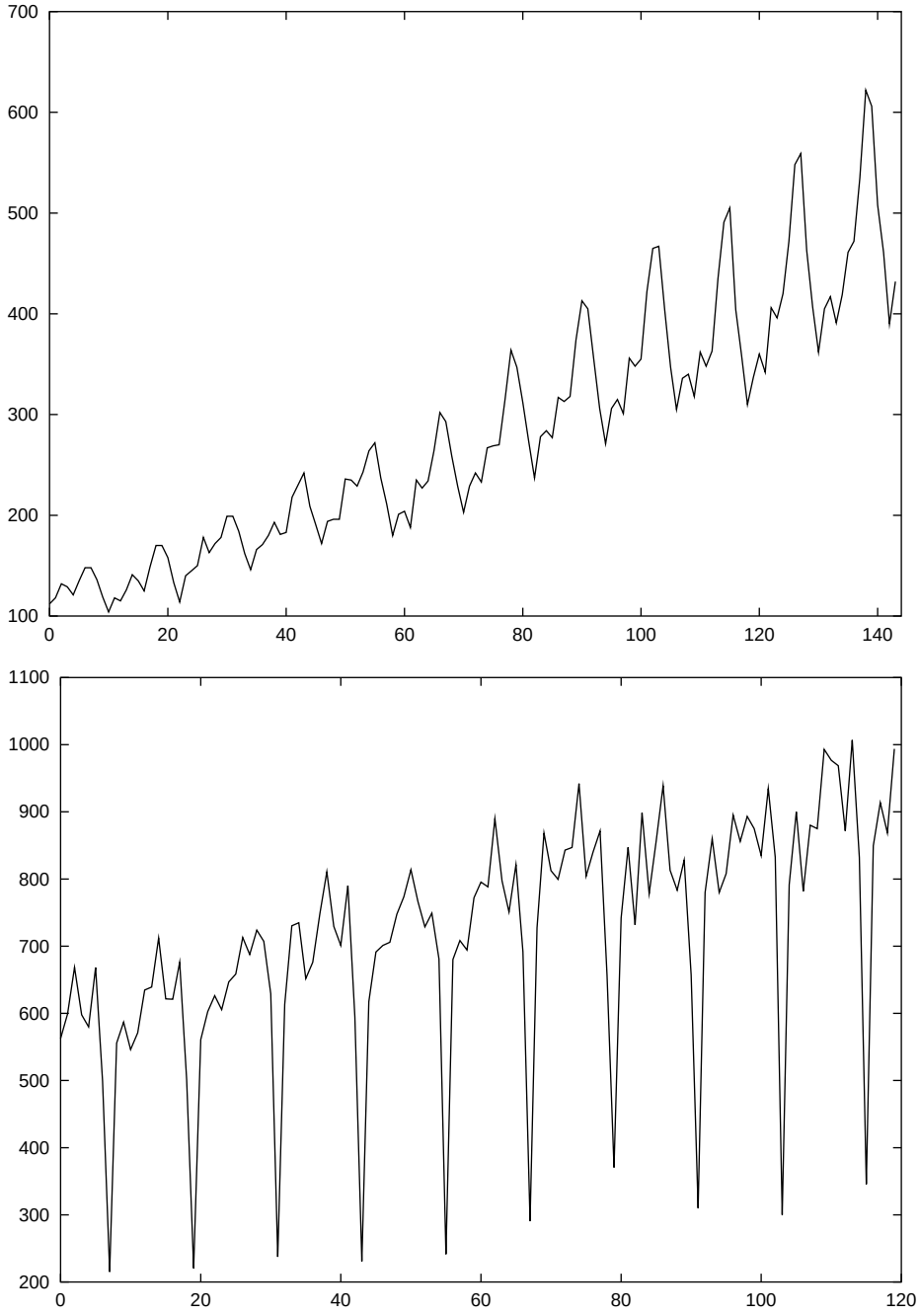


Figura 3.4: Séries sazonais com tendência (**passageiros** e **papel**).

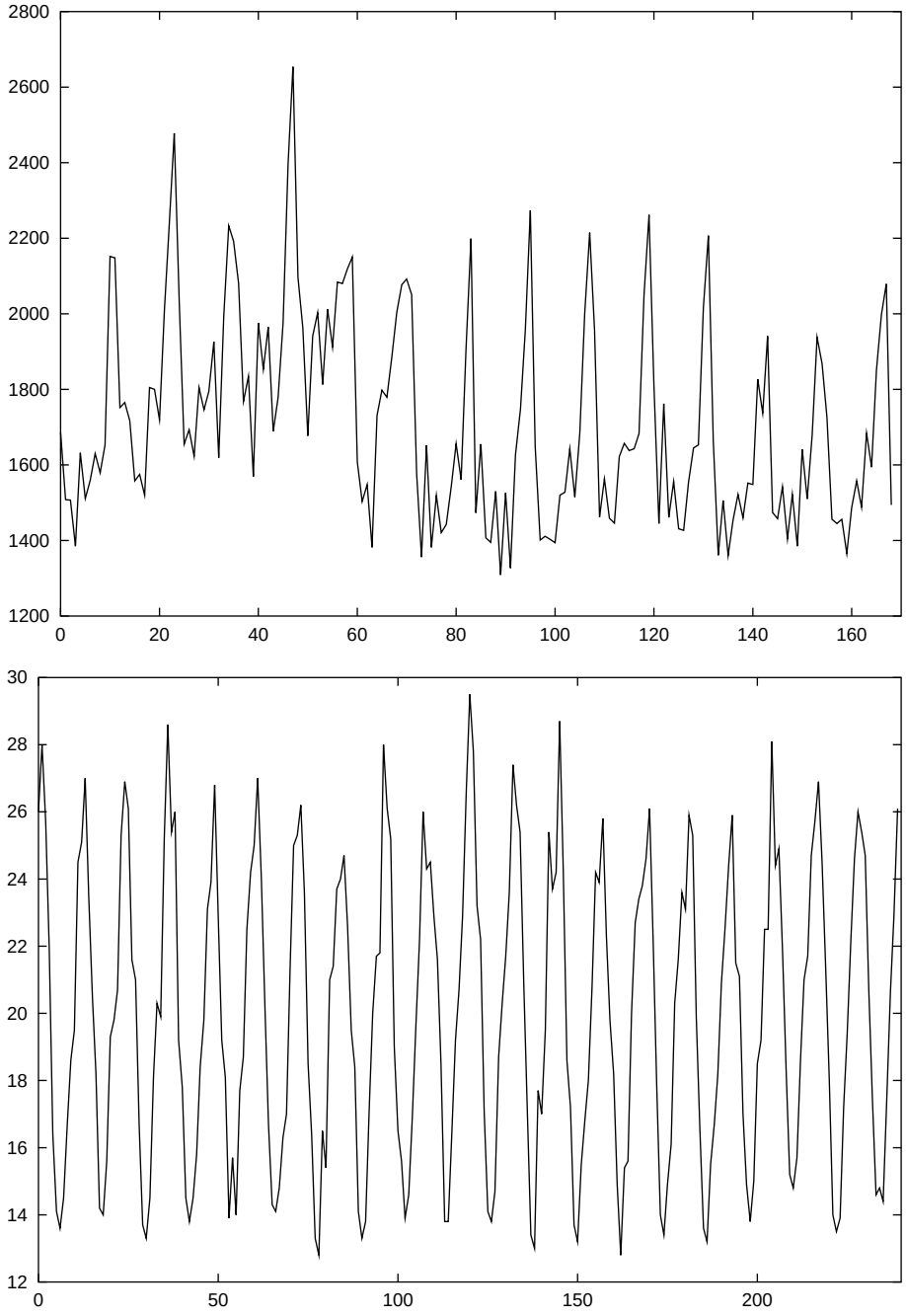


Figura 3.5: Séries sazonais (**mortes** e **melborne**).

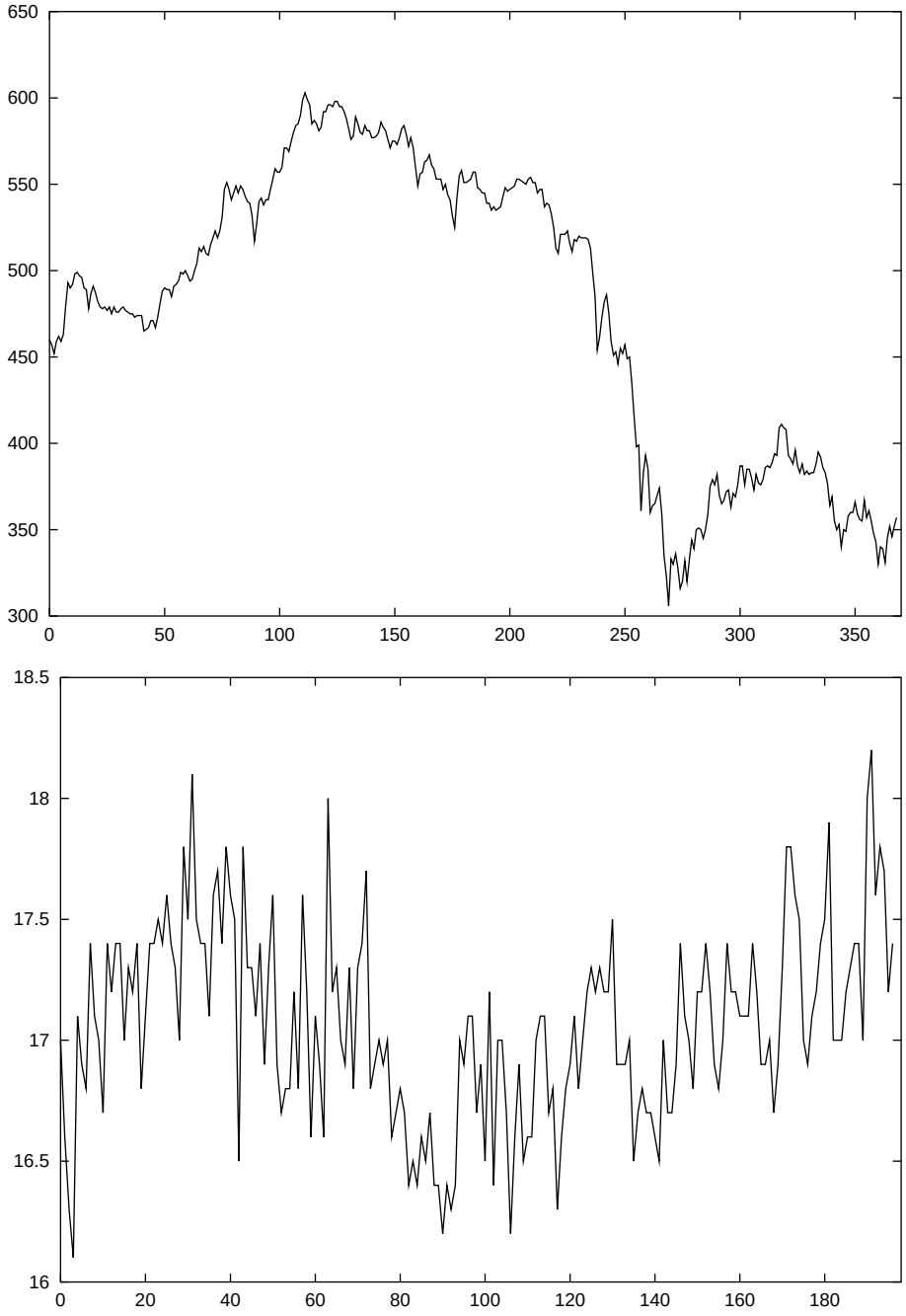


Figura 3.6: Séries não estacionárias (**preços** e **química**).

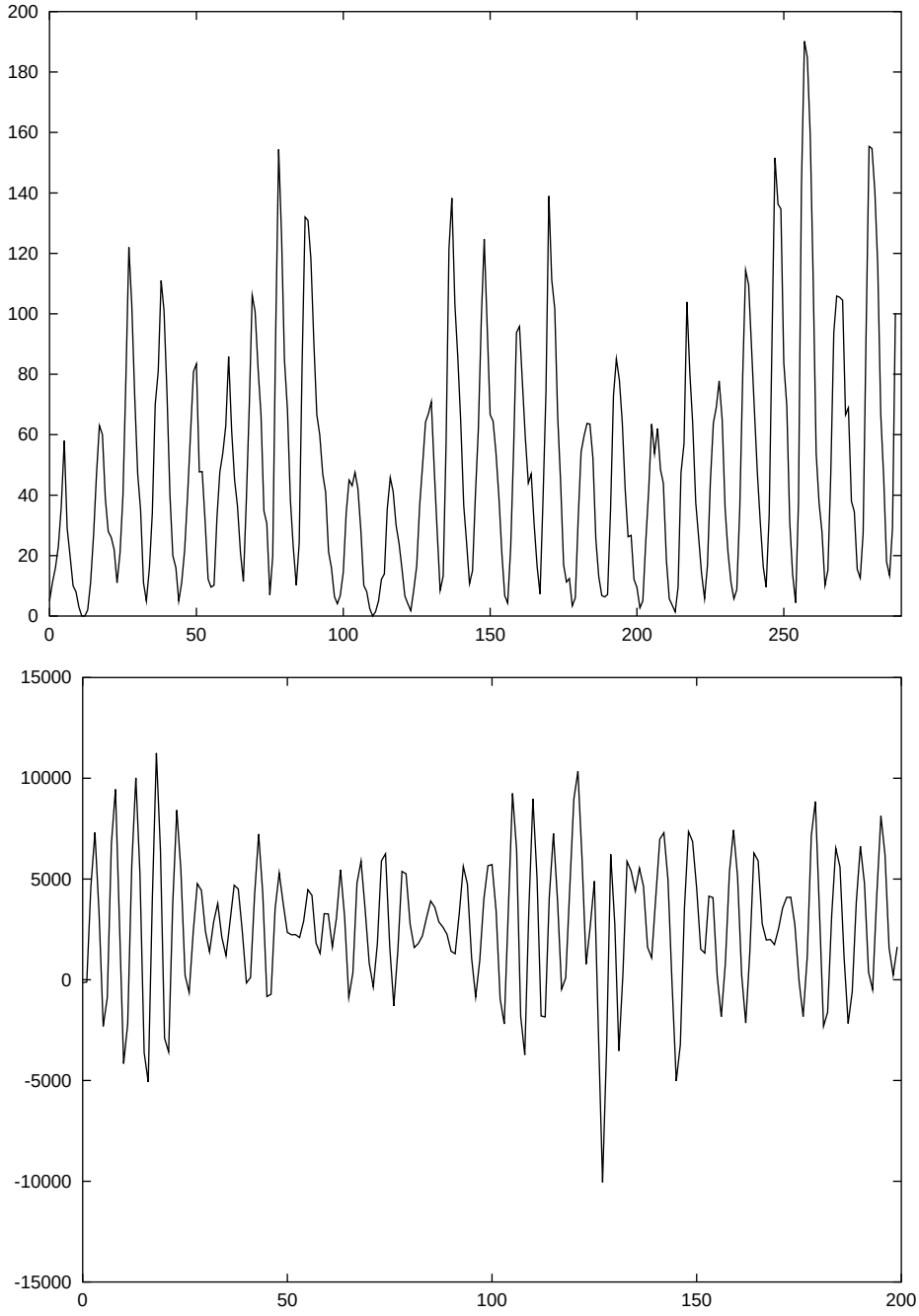


Figura 3.7: Séries não lineares (**manchas** e **kobe**).

- **manchas**, uma série muito conhecida, sobre o número anual de manchas solares ocorridas entre 1700 e 1998, originalmente criada por Wolf[1] em 1848; e

- **kobe**, que contém 200 observações sismográficas sobre o terramoto que atingiu a cidade japonesa de Kobe.

Caóticas

Esta última categoria contém séries com um elevado grau de não linearidade, que foram geradas de um modo artificial, a partir de equações caóticas [Peitgen et al., 1992] (Figura 3.8).

A primeira série, designada de **quadrática**, é uma série determinística deveras conhecida, sendo por exemplo utilizada para modelar processos biológicos (e.g., população de raposas). Esta série é criada a partir da equação $x_t = ax_{t-1}(1 - x_{t-1})$, em que $x_0 = 0.2$ e $a = 4$.

Para segunda série, chamada de **henon**, foram gerados 200 valores a partir da equação de henon $x_t = 1 - ax_{t-1}^2 + bx_{t-2}$, onde $a = 1.4$, $b = 0.3$ e $x_0 = 0.11$, aos quais foi adicionado um ruído gaussiano (distribuição normal com um desvio padrão igual a 0.1), de modo a dificultar a previsão.

3.2 Métodos Convencionais de Previsão

Todos os métodos de previsão partem do princípio de que as experiências do passado serão usadas no futuro. Assume-se, assim, que as condições que condicionaram o passado serão válidas no futuro.

Ao longo das últimas décadas vários tipos de métodos convencionais para a *PST* foram desenvolvidos. De entre estes, importa destacar dois métodos, o *Alisamento Exponencial (AE)* e a metodologia de *Box-Jenkins (BJ)*, que serão descritos em pormenor de seguida.

3.2.1 Alisamento Exponencial

Um dos métodos de *PST* mais em voga é o *AE*, também conhecido por *Holt-Winters*, que se centra em determinadas características da série, como seja a existência de factores

[1] Johann Rudolf Wolf (1818-1893). Astrónomo suíço, responsável pela medição da actividade solar via uma contagem do número e grupos de manchas solares, conhecida pelos números de Wolf.

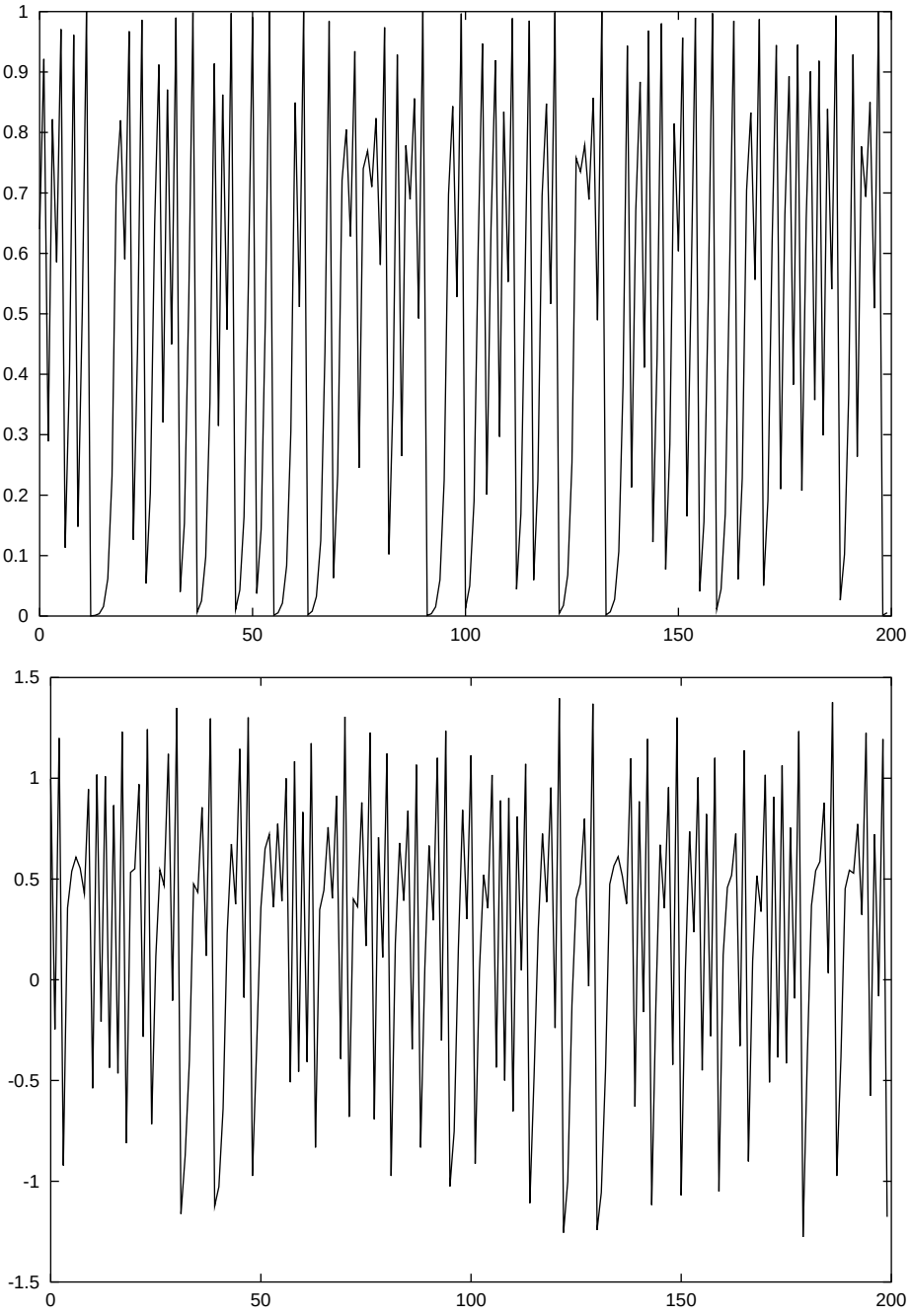


Figura 3.8: Séries caóticas (**quadrática** e **henon**).

de *tendência* ou *sazonalidade*, que são distinguidas do ruído através de uma média (*alisamento*) sobre os valores passados. É vantajoso por ser simples de usar, requerer pouco esforço computacional, e ser bastante eficaz em previsões de curto prazo, em particular com séries sazonais. O modelo base obedece ao clausulado [Makridakis & Wheelwright, 1989]:

$$\begin{aligned} F_t &= \alpha \frac{x_t}{S_{t-K}} + (1-\alpha)(F_{t-1} + T_{t-1}) \\ T_t &= \beta(F_t - F_{t-1}) + (1-\beta)T_{t-1} \\ S_t &= \gamma \frac{x_t}{F_t} + (1-\gamma)S_{t-K} \\ \widehat{x_t} &= (F_{t-1} + T_{t-1}) \times S_{t-K} \end{aligned} \tag{3.4}$$

em que F_t denota o *Alisamento Exponencial* para o período t, T_t a estimativa para a *tendência*, S_t a estimativa *sazonal* e K o *factor sazonal*.

Este método exige que certos parâmetros sejam inicialmente avaliados, como as constantes do modelo (α, β e γ) e as estimativas iniciais de cada componente [Hanke & Reitsh, 1989]. As constantes são normalmente determinadas através de processos de *tentativa-e-erro*[2], em que a minimização do erro é o objectivo perseguido. Uma abordagem diferente é utilizada para as estimativas iniciais dos componentes. Para as séries não sazonais, normalmente utilizam-se os valores $F_1 = x_1$ e $T_1 = 0.0$. No caso de séries sazonais, é aconselhado um esquema mais elaborado [WEB, 2001]:

$$F_{K+1} = x_{K+1} \tag{3.5}$$

$$T_{K+1} = \frac{1}{K}\frac{(x_{K+1} - x_1) + (x_{K+2} - x_2) + ... + (x_{K+K} - x_K)}{K} \tag{3.6}$$

$$\begin{aligned} S_1 &= \frac{\frac{x_1}{med_1} + \frac{x_{K+1}}{med_2} + ... + \frac{x_{KY-K+1}}{med_Y}}{Y} \\ S_1 &= \frac{\frac{x_2}{med_1} + \frac{x_{K+2}}{med_2} + ... + \frac{x_{KY-K+2}}{med_Y}}{Y} \\ &... \\ S_K &= \frac{\frac{x_K}{med_1} + \frac{x_{K+K}}{med_2} + ... + \frac{x_{KY}}{med_Y}}{Y} \end{aligned} \tag{3.7}$$

onde $med_e = \sum_{i=(e-1)K+1}^{(e-1)K+K} \frac{x_i}{K}$ denota a média dos valores para a estação e, sendo Y o número máximo de estações consideradas. De realçar que para a estimativa inicial da tendência, convém existir pelo menos duas estações completas.

[2]Termo conhecido em inglês por *trial-and-error*.

3.2.2 A Metodologia de Box-Jenkins

Outro método popular de *PST* é o de *BJ* [Box & Jenkins, 1976], que se baseia num processo iterativo para a síntese de padrões a partir de dados históricos, exigindo etapas como a caracterização formal do sistema (obtenção de um modelo), estimação dos parâmetros a ele associados e a sua posterior validação (Figura 3.9).

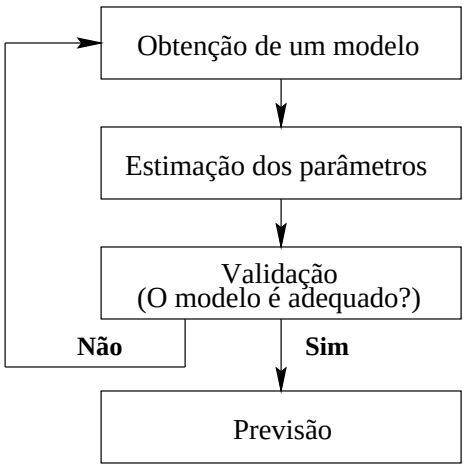


Figura 3.9: Processo iterativo da metodologia *BJ* [Hanke & Reitsch, 1989].

A metodologia de *BJ* tem como vantagem principal o facto de ser bastante precisa sobre uma gama alargada de séries temporais. No entanto, torna-se difícil conciliar os parâmetros com o modelo inicialmente criado à medida que novos dados são adquiridos; i.e., o modelo tem de ser periodicamente revisto ou, pior ainda, um novo modelo criado. Por outro lado, a construção de um modelo exige o uso de um especialista, sendo também mais exigente em termos computacionais que os métodos de *alisamento*.

O modelo designado de *ARIMA*[3], é construído a partir de uma combinação linear finita de valores passados, chamada de componente *Auto-Regressiva (AR)*. Também poderá existir, isoladamente ou em conjunto com o componente anterior, uma combinação linear finita de erros passados, designada de componente *MA*[4].

Assim, o modelo geral, pode ser reescrito como um modelo $ARMA(P,Q)$, sendo definido por:

$$\widehat{x}_t = \mu + \sum_{i=1}^{P} A_i x_{t-i} + \sum_{j=1}^{Q} M_n e_{t-j} \tag{3.8}$$

[3]Do inglês *AutoRegressive Integrated Moving-Average*.
[4]Acrónimo inglês que designa o termo *Moving Average*.

em que P e Q denotam a ordem dos componentes AR e MA, A_i e M_j os coeficientes dos modelos AR e MA, sendo μ um valor constante. Esta constante e os coeficientes do modelo são estimados através de técnicas estatísticas (e.g., métodos de mínimos quadrados). A metodologia também contempla a possibilidade do uso de transformações sobre os dados originais (e.g., transformação logarítmica).

3.3 Redes Neuronais Artificiais

O uso de técnicas inspirados na Natureza para a *PST* começou no fim dos anos oitenta, com o trabalho de Lapedes e Farber [1987], onde *RNUs* foram utilizadas para prever séries caóticas determinísticas. De início, as tentativas de aplicação de *RNAs* ao mercado bolsista revelaram-se algo desapontadoras, situação que se foi progressivamente esbatendo. Por exemplo, já no início dos anos noventa, Shoneberg [1990] refere a inaplicabilidade dos métodos estatísticos convencionais às previsões de curto prazo dos valores em bolsa. Era, então, já óbvio que os métodos de *alisamento* eram incapazes de detectar as pequenas variações de preço dos produtos financeiros no dia a dia, justificando-se assim o uso de novas ferramentas neste domínio.

A aplicação de *RNAs* para a previsão envolve todo um conjunto de etapas como: a análise da série temporal, o pré-processamento dos dados, a escolha do modelo neuronal, o treino da rede e a validação do desempenho desta. Relativamente à escolha do modelo, existem diversos tipos de *RNAs* passíveis de serem aplicadas à *PST*, como sejam as *RBFs* [Shi et al., 1999] ou as *RNRs* [Ulbricht, 1994], embora a maioria dos estudos opte pelas *RNUs* [Tang & Fishwick, 1993; Cortez et al., 1995; Faraday & Chatfield, 1998].

Estudos há que parecem indicar que as *RNUs* são as mais indicadas para a análise do comportamento dos mercados financeiros. Papadourakis, Spanoudakis e Gotsias [1993] utilizam *RNUs* para a previsão de preços de produtos financeiros, com resultados encorajadores. Um estudo dos índices bolsistas do mercado Kuala Lumpur, revelou que elementos de trabalho interessantes podem ser obtidos sem um uso excessivo de dados do mercado ou do conhecimento deste [Yao & Poh, 1995]. Outros casos de interesse são referenciados por Freisleben e Ripper [1995], em que os resultados obtidos se revelaram superiores aos obtidos por regressão linear.

Apesar da maior parte dos estudos adoptarem *RNUs*, há autores a defender o uso de outro tipo de redes. Khotanzad e seus colaboradores [1995] argumentam que esta arquitectura não permite obter resultados fiáveis, pelo que se socorreram de um outro tipo

de redes, as *RNRs*, que aplicaram à previsão da carga de um sistema eléctrico, obtendo excelentes resultados. Resultados idênticos são também apresentados por Ulbricht [1994], que clama que as *RNUs* apenas podem tratar, em simultâneo, excertos da série temporal. Ulbricht constata ainda que as *RNRs* têm a vantagem de permitir a formação de estados de memória, justificando assim o seu uso para a previsão de curto prazo (e.g., na identificação de fluxos de tráfego). Contudo, o estudo empírico de Hallas e Dorffner [1998] contrapõe estas afirmações, ao evidenciar que os modelos *AR* não lineares, fornecidos pelas *RNUs*, apresentam em geral, resultados superiores aos obtidos por vários tipos de *RNRs*.

Competições de previsão, confrontando *RNAs* com métodos tradicionais de *PST*, reportaram maus (e.g., competição M [Makridakis, 1982]) e bons resultados (e.g., competição de Santa Fé [Weigend & Gershenfeld, 1994]). No primeiro caso, o mau desempenho das *RNAs* ficou a dever-se, provavelmente, ao uso de séries muito curtas (a maioria com 50 a 100 observações). Todavia, certos estudos demonstraram que as *RNAs* podem prever tão bem ou ainda melhor que a prática alicerçada na metodologia *BJ*, para várias destas séries.

Por exemplo, Tang e Fishwick [1993] utilizaram *RNUs* para prever 14 séries temporais da competição M, comparando os resultados obtidos via modelos *ARIMA*, com as *RNAs* a fornecer os melhores resultados. Segundo os autores, o desempenho das *RNAs* depende de vários factores, como a estrutura da rede, os parâmetros de treino ou a natureza da série temporal.

Chegou-se igualmente à conclusão de que as *RNU*s com conexões de *atalho* combinam componentes lineares (dado pelas conexões de *atalho*), e não lineares (dada pela camada intermédia), funcionando como um super conjunto dos modelos *ARIMA*. Não é pois de admirar que as *RNAs* possam modelar o comportamento de séries temporais deveras complexas.

A dificuldade no uso de *RNAs* está na falta de processos sistemáticos para construir os modelos de previsão (i.e., escolha correcta dos parâmetros da rede), e no facto das *RNAs* funcionarem como caixas negras, não se podendo tirar ilações a partir do conhecimento da sua estrutura. Para a competição de Santa Fé, já referida em epígrafe, os resultados parecem contraditórios, pois as *RNUs* obtiveram os melhores e os piores resultados, evidenciando que é preciso um certo cuidado quando se ajustam *RNAs* como modelos de previsão.

Dentro deste contexto, para esta tese foi decidido adoptar a arquitectura da *RNU*. A topologia base é definida por uma rede completamente interligada, com apenas uma camada intermédia, com conexões de *bias* e atalhos, dado que estes acrescentam uma componente linear ao modelo (Figura 3.10). A função logística é aplicada nos nodos intermédios, para

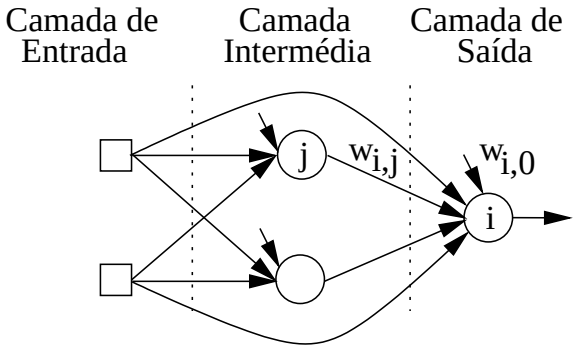


Figura 3.10: Uma *RNU* completamente interligada, com 2 nodos de entrada, 2 nodos intermédios, um nodo de saída, conexões de *bias* e atalhos.

permitir uma aprendizagem não linear. No nodo de saída, é utilizada a função linear, para escalonar os dados de saída, visto que a função logística tem um contradomínio limitado ($[0, 1]$). Esta estratégia tem a vantagem de evitar o uso de técnicas de filtragem, que podem levar a uma perda de informação (e.g., escalonamento).

Uma *Janela Temporal Deslizante (JTD)* define um conjunto de deslocamentos temporais passados (k_i), definidos a partir do tempo t (Figura 3.11), sendo representada pela sequência $JTD =< k_1, k_2, ..., k_n >$, para uma JTD com o comprimento n. Por exemplo, a aplicação da $JTD =< 1, 3 >$ à série $1, 2, 3, 4, 5$ produziria as sequências $< 1, 3 >$ e $< 2, 4 >$, para os instantes $t = 4$ e $t = 5$.

No caso da previsão de curto prazo, o nodo de saída da *RNU* é treinado para efectuar correspondências entre os valores passados, gerados a partir da JTD, com o valor presente. Assim, o modelo geral fornecido pela *RNU* é dado por :

$$\widehat{x_t} = w_{S,0} + \sum_{i=1}^{n} x_{t-k_i} w_{S,i} + \sum_{j=n+1}^{S-1} fa\,(\sum_{i=1}^{n} x_{t-k_i} w_{j,i} + w_{j,0})\, w_{S,j} \tag{3.9}$$

em que S denota o nodo de saída, fa a função logística ($\frac{1}{1+e^{-x}}$) e n o número de nodos de entrada ($n = |E|$).

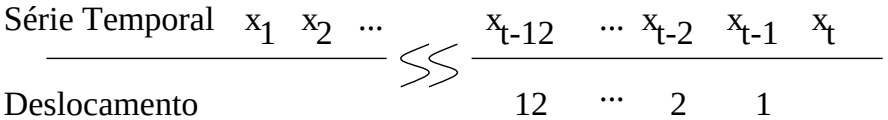


Figura 3.11: Deslocamentos de uma série temporal.

3.4 Algoritmos Genéticos e Evolucionários

Apesar da sua juventude, a *CGE* já produziu um conjunto variado de técnicas que se tornaram particularmente úteis para problemas de optimização, quer numérica quer combinatória. Sendo assim, é surpreendente constatar que a aplicação destas técnicas no domínio da previsão seja tão escasso. De facto, a previsão via a *CGE* apenas surgiu nos anos noventa, sendo o número de aplicações diminuto e com pouco destaque.

O trabalho em curso passa pela optimização de parâmetros de modelos convencionais de previsão, utilizando representações binárias, como se ilustra no que se segue:

- No estudo de Agapie e Agapie [1997], foram utilizados *AGEs* para optimizar o modelo de *Holt-Winters*, obtendo-se resultados encorajadores para séries sazonais curtas.

- O mesmo tipo de *AGEs* também foi utilizado com sucesso para previsões financeiras (e.g., depósitos bancários) por Chiraphadhanakul et al. [1997], onde foram codificados os coeficientes do modelo *AR*.

- Chai e seus colaboradores [1997], optimizaram os parâmetros de modelos *ARMA*, defendendo que os *AGEs* conseguem melhores soluções do que as técnicas estatísticas convencionais (e.g., mínimos quadrados).

Contudo, recentes desenvolvimentos na *CGE,* como sejam o uso de *AGEs* com *Representações de Valores Reais (RVRs)* [Michalewicz, 1996] e a *Programação Genética (PG)* [Banzhaf et al., 1998] alteraram este panorama, alargando a aplicabilidade da *CGE* à previsão. Por exemplo, Kaboudan [1999] aplicou a *PG,* com um alfabeto de símbolos terminais pertencendo ao conjunto $\{+, -, *, \%, exp, sqrt, ln, sin, cos\}$, para a *PST*.

Para esta tese, foi decidido adoptar duas aproximações para a previsão, ambas baseadas em *AGEs* com *RVRs*, dado estes se revestirem de um enorme potencial, em relação às representações binárias, e ainda por, aparentemente, não existirem trabalhos nesta área. Na primeira aproximação, é utilizado uma combinação linear de valores passados (modelo *AR*). Neste cenário, os genes de um cromossoma codificam os pesos pelos quais os valores passados são multiplicados. Na segunda aproximação, ambos os valores e erros passados são considerados, seguindo uma estratégica similar ao modelo *ARMA*, onde os genes codificam os coeficientes do modelo. Por conseguinte, para uma dada *JTD*, as aproximações são definidas pelos modelos de previsão, dados na forma:

G1 - baseado numa combinação linear $AR(n)$; i.e.,

$$\widehat{x_t} = g_0 + \sum_{i=1}^{n} g_i x_{t-k_i} \tag{3.10}$$

G2 - baseado num modelo $ARMA(n, n)$; i.e.,

$$\widehat{x_t} = g_0 + \sum_{i=1}^{n} (g_i x_{t-k_i} + g_{i+n} e_{t-k_i}) \tag{3.11}$$

onde g_i denota o valor do gene i no cromossoma de um indivíduo e n o tamanho da *JTD*.

3.5 Generalização

Diz-se que um modelo de aprendizagem possui uma boa *generalização* quando a correspondência entre entradas e saídas é correcta (ou próxima disso) para *dados de teste*, retirados da mesma população, nunca antes utilizados na criação ou treino da rede. O processo de aprendizagem pode ser visto como um problema de ajustamento de curvas ou de aproximação de funções, onde a rede tenta efectuar uma boa interpolação não linear dos dados [Rojas, 1996].

A Figura 3.12 mostra como podem ocorrer duas generalizações distintas para o mesmo conjunto de dados de treino. Aqui, uma boa generalização ocorre com a curva A, com um erro mínimo para os dados de teste. O mesmo já não se sucede com a curva B, que origina um erro maior para os casos de teste, isto apesar de apresentar um menor erro para os dados de treino. Tal fenómeno, designado de *sobre-ajustamento*[5], ocorre quando um modelo memoriza em demasia os exemplos de treino, tratando-se de um dos problemas mais sérios relacionados com o uso de *RNAs* [Sarle, 1995].

Durante o processo de aprendizagem, o modelo pode captar certas características, como o ruído, que estão presentes nos dados de treino mas não na função implícita a ser aprendida. Este exemplo ilustra os dois objectivos contraditórios da aproximação funcional. Por um lado tem-se a minimização do erro de treino, pelo outro há a minimização do erro para as entradas desconhecidas.

A generalização é influenciada por três factores [Haykin, 1999; Sarle, 1999]:

- *A complexidade do problema a ser aprendido.* Trata-se de um factor de difícil controlo.

[5]Conhecido em inglês pelo termo *overfitting*.

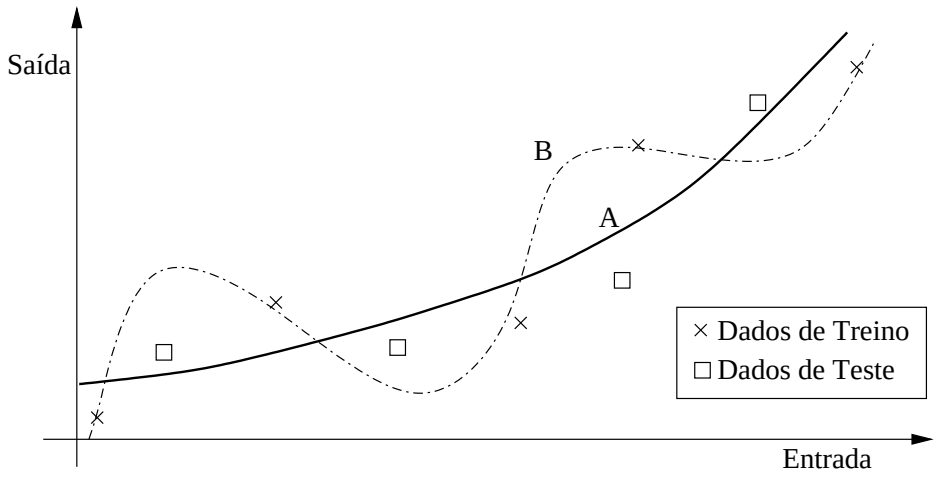


Figura 3.12: Generalização e *sobre-ajustamento.*

As entradas devem conter informação suficiente para permitir a obtenção das saídas desejadas; i.e., tem de existir uma função matemática com algum grau de precisão que relacione as entradas com as saídas. Por outro lado, convém que esta função seja *suave*; i.e., pequenas alterações nas entradas devem provocar pequenas alterações nas saídas, para a maior parte dos casos.

- *Os casos de treino.* A sua cardinalidade deve ser elevada, com exemplos (ou *amostras* na terminologia estatística) que sejam representativos acerca do ambiente. A generalização é sempre efectuada a partir de dois tipos de duas situações: *interpolação* e *extrapolação.* No primeiro caso, um valor é determinado a partir da informação dos valores vizinhos. A segunda situação engloba tudo o resto, ou seja, casos fora do domínio dos dados de treino. Enquanto que a interpolação pode ser efectuada com relativa acuidade, o mesmo já não se passa com a extrapolação, notoriamente menos fiável.

- *Complexidade do modelo de aprendizagem*, medida pelo número de parâmetros livres (p) que contém e pela sua capacidade de aprendizagem. Um modelo que não seja suficientemente representativo do problema em equação irá falhar na aproximação da função a aprender. Por outro lado, um modelo demasiado ajustado aos dados do problema em equação irá fixar o ruído, provocando *sobre-ajustamento.*

A melhor forma de evitar o *sobre-ajustamento* é utilizar uma quantidade elevada de casos de treino. Quando este número for pelo menos 30 (trinta) vezes superior ao número de parâmetros, então é muito improvável que ocorra uma perda de generalização [Sarle, 1999].

O problema é que nem sempre existem suficientes casos de treino disponíveis e não se deve reduzir o número de parâmetros de um modo arbitrário, devido a problemas de falta de representatividade. Assim, dado um número fixo de dados de treino existem duas alternativas para controlar a complexidade ou representatividade de um modelo [Sarle, 1995]: *regularização* e *selecção de modelos.*

A *regularização* baseia-se num controlo dos valores dos parâmetros de um modelo para obter uma boa generalização, envolvendo o uso de restrições ou penalidades sobre estes, de modo a que a rede aprenda funções *mais suaves.* Como exemplo tem-se a *paragem antecipada*, um dos mais populares métodos de regularização, onde os dados de treino são divididos em dois tipos de casos: *de treino* e *de validação.* Os primeiros são utilizados na aprendizagem do modelo, enquanto que os últimos são utilizados para aferir da qualidade deste; i.e., para estimar o erro de generalização. Durante a fase de aprendizagem, calcula-se o erro de validação de forma periódica, parando-se quando este ameaça aumentar.

A regularização diminui o efeito de *perda de generalização* pelo estímulo na aprendizagem de funções *suaves.* No entanto, utiliza uma estrutura fixa, que deve ser especificada em avanço pelo utilizador. Embora se possa utilizar uma grande estrutura, com um grande número de nodos intermédios, na prática, a optimização dos parâmetros torna-se de difícil ajuste, exigindo um grande esforço computacional. Mais ainda, em geral, são métodos que exigem um delicado balanço, controlado por um (ou mais) parâmetro(s) de regularização. Uma alternativa distinta baseia-se na procura de um modelo adequado, em termos de representatividade (i.e., complexidade). Assim, um problema que seja de difícil aprendizagem para um modelo poderá ser facilmente aprendido por outro.

A abordagem estatística para a *selecção de modelos* passa pela estimativa do erro de generalização para cada um dos modelos, ou topologias de rede, escolhendo-se o modelo que minimiza essa estimativa. Existem diversos métodos para estimar a capacidade de generalização de uma *RNA,* como seja a *Validação Cruzada*[6] ou o *Bootstrapping* [Efron & Tibshirani, 1993; Kohavi, 1995], que são deveras exigentes em termos computacionais. Uma alternativa razoável é dada pela teoria da informação, onde o erro de generalização é estimado a partir de estatísticas simples, que acrescentam uma penalidade que depende da complexidade do modelo, nomeadamente:

- *Critério de Informação de Akaike*, conhecido por *AIC*[7], segundo o clausulado [Akaike, 1973]:

$$AIC = N \ln(SQE/N) + 2p \tag{3.12}$$

[6] Técnica conhecida em inglês por *K-fold Validation.*
[7] Do inglês *Akaike Information Criterion.*

- *Critério de Informação de Bayes*, designado de *BIC*[8] ou *SBC*, e dado na forma [Schwarz, 1978]:

$$BIC = N\ln(SQE/N) + p\ln(N) + p \tag{3.13}$$

onde SQE denota o somatório do quadrado dos erros para todos os exemplos de treino, N o número de exemplos de treino e p o número de parâmetros livres, respectivamente. Para os modelos propostos, têm-se que:

$$\begin{aligned} p_{RNU} &= n(n_i+1) + 2n_i + 1 \\ p_{G1} &= n+1 \\ p_{G2} &= 2n+1 \end{aligned} \tag{3.14}$$

onde n_i denota o número de nodos intermédios da rede.

3.6 Selecção dos Modelos de Previsão

Para testar cada um dos métodos de previsão referidos em epígrafe, foi desenvolvida uma aplicação em *C++,* chamada **tsf**[9], que faz a integração da classe **TS** com os ambientes de programação (*APRNA* e *APAGE*). A classe **TS** foi desenvolvida para manipular séries temporais, contendo entre outros:

- todos os atributos da série temporal, definidos por um vector de reais;
- diversas estatísticas, como o cálculo de valores máximos, mínimos e as autocorrelações; e
- métodos de previsão simples (e.g., *modelos AE* e *ARMA*).

Para todos os métodos, foi decidido dividir os elementos de cada série temporal em dois tipos de conjuntos: *dados de aprendizagem*, contendo os primeiros 90% dos valores da série, e *dados de teste*, com os restantes 10%, que serão utilizados para medir o desempenho na previsão de cada método.

3.6.1 Alisamento Exponencial

A Tabela 3.1 descreve os modelos de *AE* para a séries da Secção 3.1.3. Os parâmetros do modelo (α, β e γ) foram escolhidos a partir de uma procura com uma grelha[10] de 0.01, do

[8] Acrónimo tirado do inglês *Bayesian Information Criterion.*
[9] Designação retirada do inglês ***T**ime **S**eries **F**orecasting.*
[10] Do inglês *grid-search.*

menor valor de erro ($RMQE_{tr}$), sendo as estimativas sazonais calculadas pelas equações 3.5, 3.6 e 3.7, onde o número máximo de estações foi estabelecido em $Y = 6$. No caso das séries não sazonais ($K = 0$), apenas foram optimizadas as constantes de alisamento e tendência (α e β). A última coluna, $RMQE_{pr}$, denota o erro de previsão de curto prazo para os últimos 10% dos elementos da série.

Tabela 3.1: Modelos de AE para as séries temporais.

Série	α	β	γ	K	$RMQE_{pr}$
passageiros	0.29	0.03	0.95	12	16.5
papel	0.25	0.01	0.03	12	49.2
mortes	0.36	0.00	0.01	12	135
melborne	0.24	0.00	0.11	12	0.72
química	0.30	0.00	-	0	0.35
preços	1.00	0.02	-	0	7.54
manchas	1.00	0.95	-	0	28.3
kobe	0.05	0.00	-	0	3199
quadrática	0.03	0.00	-	0	0.35
henon	0.02	0.75	-	0	0.73

3.6.2 Metodologia de Box-Jenkins

Como se trata de uma técnica deveras sofisticada, foi decidido utilizar uma aplicação comercial de previsão (*Forecast Pro*) para a obtenção dos modelos óptimos $ARMA$ (Tabela 3.2) [Rycroft, 1993].

Os modelos foram obtidos utilizando os mesmos 90% dos valores iniciais da série, sendo a transformação logarítmica aplicada às séries **passageiros** e **química**. Para as colunas A_i e M_j foi escolhida a notação de sequência $< z_1, z_2, .., z_n >$ onde z_i denota a constante correspondente ao deslocamento i. Por exemplo, a série **preços** é modelada pela equação: $\widehat{x_t} = x_{t-1} - 0.12e_{t-1}$.

3.6.3 Modelos Inspirados na Natureza

Ambos os modelos propostos fazem uso de uma *JTD*, que define qual a informação que será fornecida a um dado modelo, ou seja, o conjunto de deslocamentos de tempo utilizados para construir uma previsão. Para além disso, torna-se também necessário definir um modelo de aprendizagem adequado, ou seja, com uma representatividade (ou complexidade) deveras

Tabela 3.2: Modelos *ARMA* para as séries temporais.

Série	μ	A_i	M_j	$RMQE_{pr}$
pas.	0.00	$< 1_1, 1_{12}, -1_{13} >$	$< -0.35_1, -0.62_{12}, 0.22_{13} >$	17.8
pap.	0.00	$< 1_1, 1_{12}, -1_{13} >$	$< -0.87_1, -0.80_{12}, 0.70_{13} >$	61.0
mor.	0.00	$< 1_1, 1_{12}, -1_{13} >$	$< -0.66_1, -0.90_{12}, 0.59_{13} >$	144
mel.	0.00	$< 1_1, 1_{12}, -1_{13} >$	$< -0.88_1, -0.89_{12}, 0.78_{13} >$	0.91
quí.	0.30	$< 0.90_1 >$	$< -0.56_1 >$	0.35
pre.	0.00	$< 1_1 >$	$< 0.12_1 >$	7.72
man.	14.2	$< 1.39_1, -0.70_2 >$	$<>$	28.4
kob.	3038	$< 0.71_1, -0.81_2 >$	$< 0.77_1, -0.21_2, -0.06_3 >$	582
qua.	0.48	$<>$	$<>$	0.35
hen.	0.25	$<>$	$< -0.30_1, 0.14_2, -0.40_3 >$	0.63

ajustada ao tipo de problema a tratar. Assim, a qualidade de um modelo de previsão dependerá destes dois factores: a janela temporal e a complexidade do modelo.

Qual a melhor *JTD* para uma dada série temporal? Uma janela com alta cardinalidade pode aumentar a complexidade de um sistema, dificultando a aprendizagem de um modelo, enquanto que janelas de baixa cardinalidade podem conter insuficiente informação. A selecção dos deslocamentos temporais relevantes pode melhorar a previsão. Por exemplo, os modelos *ARIMA* costumam utilizar os deslocamentos 1, 12 e 13 para séries sazonais com periodicidade mensal e com tendência.

Uma aproximação empírica a este problema passa por se utilizar informação baseada na análise da série temporal. Por conseguinte, quatro heurísticas serão utilizadas para a selecção da janela temporal, baseadas nos valores de autocorrelação, nomeadamente:

A - uma *JTD* completa, com todos os deslocamentos desde 1 até um máximo m: $JTD =< 1, 2, ..., m >$ (a m foi atribuído o valor de $12+1 = 13$, um valor considerado suficiente para englobar os efeitos de séries com periodicidade mensal e tendência);

B - uma *JTD* com todos os deslocamentos cujas autocorrelações são superiores a um dado valor limite (a que, nos termos presentes, é atribuído o valor de 0.2);

C - uma *JTD* com os quatro deslocamentos correspondentes às autocorrelações com um valor mais elevado (no caso das séries sazonais com tendência, estas foram calculadas após uma *diferenciação*[11], dado que os efeitos de tendência tendem a sobrepor-se aos da sazonalidade); e

[11]Processo simples para eliminar uma tendência, que consiste em definir uma nova série $x'_t = x_t - x_{t-1}$.

D - o uso de informação de decomposição; i.e.,

- $JTD =< 1, K, K+1 >$ se a série for sazonal e não estacionária (com um período sazonal K);
- $JTD =< 1, K >$ se a série for sazonal; e
- $JTD =< 1 >$ e $JTD =< 1, 2 >$ se a série for não estacionária.

A Tabelas 3.3 e 3.4 mostram as janelas, definidas segundo estas heurísticas, para as séries da Secção 3.1.3.

Tabela 3.3: Heurísticas para as séries sazonais.

Série	**Tendência**	**JTD**
passageiros	Sim	A=B=<1,2,...,13> C=<1,11,12,13> D=<1,12,13>
papel	Sim	A=<1,2,...,13> B=<1,3,9,11,12,13> C=<1,11,12,13> D=<1,12,13>
mortes	Não	A=<1,2,...,13> B= <1,2,10,11,12,13> C=<1,11,12,13> D=<1,12>
melborne	Não	A=<1,2,...,13> B =<1,2,10,11,12,13> C=<1,11,12,13> D=<1,12>

Redes Neuronais Unidireccionais

Para avaliar a qualidade de cada modelo, foram testadas diversas *RNUs*, com o número de nodos intermédios a variar entre 0 e 13, de modo a explorar todas as heurísticas de janelas temporais. Os pesos iniciais foram aleatoriamente gerados dentro do domínio $[\frac{-2}{k}, \frac{2}{k}]$, para um nodo com k entradas (ver Secção 2.1.4). Dado que diferentes valores de pesos iniciais podem conduzir a diferentes previsões, trinta aprendizagens independentes foram aplicados a cada modelo, por forma a garantir significância estatística.

Tabela 3.4: Heurísticas para as séries não sazonais.

Série	**Tendência**	**JTD**
química	Sim	A=B=<1,2,...,13> C=<1,2,3,7> D=<1> e <1,2>
preços	Sim	A=B=<1,2,...,13> C=<1,2,3,4> D=<1> e <1,2>
manchas	Não	A=<1,2,...,13> B=<1,2,9,10,11,12> C=<1,2,10,11>
kobe	Não	A=<1,2,...,13> B=C=<1,5,6,10>
quadrática	Não	A=<1,2,...,13> B=C=<8,9,10,11>
henon	Não	A=<1,2,...,13> B=C=<2,4,8,9>

O algoritmo *RPROP* foi adoptado para o treino, dado que permite em geral uma convergência mais rápida, sendo também mais estável em termos do ajustamento dos seus parâmetros (foram utilizados os valores aconselhados para estes parâmetros, com $\Delta_0 = 0.1$ e $\Delta_{max} = 50.0$ [Riedmiller, 1994]). O treino da rede é dado como findo quando o progresso no treino decai (ft =5 e $\phi = 0.1$) ou quando se atinge o número máximo de iterações ($\varphi = 1000$) [Prechelt, 1998].

Em termos da metodologia utilizada, ir-se-á explicá-la em detalhe para a série **manchas** (Tabela 3.5). Aqui, apenas três janelas (diferentes) são utilizadas, dado que se trata duma série estacionária (Tabela 3.4). Para simplificar a explicação, apenas serão visíveis alguns dos resultados relevantes. Os resultados das últimas quatro colunas são dados em termos de uma média sobre os 30 (trinta) treinos. Os intervalos de confiança, para um nível de significância de 95%, também são exibidos para os erros de previsão (Apêndice A) [Flexer, 1996].

O menor erro de treino ($RMQE_{tr}$) é atingido para a janela **A**, com 12 nodos intermédios. Este modelo contém um número demasiado elevado de parâmetros, o que leva a uma perda de generalização. De facto, o aumento do número de nodos intermédios provoca um menor erro de treino. Contudo, quando tal acontece, o desempenho do modelo tende a piorar, devido a uma especialização da rede. O modelo sugerido pelos valores de AIC contém

ainda um número elevado de parâmetros e, por outro lado, também produz um sobre-ajustamento. O mesmo não sucede para o critério BIC, que funciona noutros termos, seleccionando uma rede que fornece uma das melhores previsões de curto prazo (coluna $RMQE_{pr}$). Todavia, o melhor resultado de previsão, correspondente ao menor erro de previsão, parece inatingível.

Tabela 3.5: Resultados das *RNUs* para a abordagem empírica à série **manchas**.

JTD	n_h	p	$RMQE_{tr}$	AIC	BIC	$RMQE_{pr}$
	0	14	14.7	1355	1404	18.2±0.1
A	6	104	11.6	1419	1784	20.5±0.7
	12	194	**9.0**	1474	2155	20.2±0.8
	0	5	14.8	1345	1369	17.8±0.3
B	5	47	11.5	1302	1467	**17.0**±0.6
	13	111	9.4	1328	1717	19.0±0.8
	0	5	15.1	1352	1369	18.1±0.0
C	1	11	14.1	1329	**1368**	17.8±0.3
	8	53	10.7	**1278**	1464	19.4±0.5

Faraday e Chatfield [1998] recomendaram o critério BIC para uma comparação entre diferentes topologias de *RNAs*, embora apenas testassem uma única série (**passageiros**). Um teste mais detalhado, sobre o desempenho de cada critério de selecção de modelos, é fornecido na Tabela 3.6. A *qualidade* de cada *Modelo Seleccionado (MS)* será medida em termos de quão distante está, em termos percentuais, em relação ao melhor modelo obtido (min), sendo dada pela expressão:

$$\frac{RMQE_{pr}(MS) - RMQE_{pr}(min)}{RMQE_{pr}(min)} \tag{3.15}$$

Os resultados confirmam que a medida de $RMQE_{tr}$ origina um sobre-ajustamento com as *RNUs*. As poucas excepções ocorrem com as séries não estacionárias, embora seja de se esperar que gere uma perda de generalização para redes com um número mais elevado de nodos. Ambos os critérios AIC e BIC sugerem modelos idênticos para a maioria das séries. Todavia, quando considerada a média global, a estatística BIC fornece um melhor desempenho, sendo que a diferença surge em modelos não lineares (**mancha**s e **henon**), em que a discrepância é de 3.37%, em média, para o melhor dos modelos.

As melhores *RNUs*, tendo em conta o critério BIC, são apresentadas na Tabela 3.7. Como esperado, o critério sugere a utilização de janelas de pequena cardinalidade, favore-

Tabela 3.6: Desempenho dos critérios de selecção de modelos para as *RNUs*.

Série	$RMQE_{tr}$	AIC	BIC
passageiros	13.5%	**5.58%**	**5.58%**
papel	14.7%	**0.65%**	**0.65%**
mortes	4.00%	**0.66%**	**0.66%**
melborne	11.7%	**6.76%**	**6.76%**
química	**0.63%**	6.91%	15.17%
preços	**0.00%**	**0.00%**	**0.00%**
manchas	18.7%	14.42%	**4.88%**
kobe	12.8%	**0.00%**	**0.00%**
quadrática	27.3%	**0.00%**	**0.00%**
henon	19.3%	20.84%	**0.00%**
Média	12.3%	5.58%	**3.37%**

cendo as estratégias **D** e **C**, assim como os modelos lineares, com $n_i = 0$ nodos intermédios, para todas as série lineares. Para as séries não lineares, a estatística BIC privilegia janelas de maior cardinalidade, contendo entre 6 a 13 entradas. Contudo, continua a favorecer topologias de rede deveras simples (com zero ou um nodos intermédios) para as séries não lineares (**manchas** e **kobe**). Por exemplo, as melhores previsões para a série **manchas**, são obtidas com uma rede que contém 5 nodos intermédios, embora tendo como consequência um maior valor de BIC, devido ao número elevado de conexões (Tabela 3.5).

Tabela 3.7: *RNUs* com o menor valor de BIC.

Série	JTD	n_i	p	$RMQE_{pr}$
passageiros	D=<1,12,13>	0	4	18.4±0.2
papel	D=<1,12,13>	0	4	51.6±0.2
mortes	C=<1,11,12,13>	0	5	134±1
melborne	C=<1,11,12,13>	0	5	0.90±0.02
química	D=<1,2>	0	3	0.40±0.01
preços	D=<1>	0	2	7.49±0.00
manchas	B=<1,2,9,10,11,12>	1	11	17.8±0.3
kobe	A=<1,2,...,13>	0	14	557±4
quadrática	A=<1,2,...,13>	9	149	0.06±0.02
henon	A=<1,2,...,13>	2	44	0.35±0.05

Algoritmos Genéticos e Evolucionários

Os modelos propostos (**G1** e **G2**) foram testados em condições similares às impostas às *RNUs*, ou seja:

- utilizando todas as heurísticas para as janelas temporais; e
- realizando 30 (trinta) simulações para cada modelo, de modo a assegurar uma relevância estatística, sendo os resultados apresentados em termos da média e respectivos intervalos de confiança (na ordem dos 95%).

Em termos da configuração do *AGE*, as populações iniciais foram geradas aleatoriamente, com valores pertencendo ao intervalo $[-1, 1]$. A cardinalidade da população foi definida para o valor de $TP = 50$. A função de aptidão é medida em termos do erro sobre todos os casos de treino ($aptidão = RMQE_{tr}$). O processo de selecção é realizado através duma *conversão por ordenação* (Secção 2.2.6). Para o processo de amostragem, foi utilizada a técnica de *amostragem estocástica com substituição.* Este método é utilizado para seleccionar os 40% dos indivíduos que são mantidos na população, com a excepção do melhor indivíduo, que sempre se mantém ($VE = 1$). Os restantes ($TS = 60\%$) indivíduos são gerados via aplicação dos operadores genéticos.

Para o cruzamento, foi escolhido o operador *aritmético* (Secção 2.2.4), sendo responsável pela geração de dois terços dos descendentes, enquanto os restantes elementos são criados via uma *mutação*, definida através de uma distribuição gaussiana com uma média zero (i.e., em geral, serão geradas perturbações com um valor reduzido). Por último, o AGE é dado como findo após $T = 1000$ gerações.

A título elucidativo, os resultados serão descritos em pormenor para a série **manchas** (Tabela 3.8). Aqui, também o critério BIC se apresenta com melhores resultados, favorecendo a janela temporal **C**, o que já tinha acontecido previamente com as *RNUs,* em que é seleccionado o melhor modelo de previsão.

Este comportamento ocorreu de forma consistente com todas as outras séries, como pode ser comprovado pela Tabela 3.9, onde foi medida a qualidade dos critérios de selecção de modelos para os *AGEs*. De referir que o critério $RMQE_{tr}$ tem um desempenho superior ao obtido para as *RNUs*, devido ao facto de aqui existir uma menor diferença entre os modelos, em termos do número parâmetros. No entanto, o critério BIC apresenta uma qualidade média ainda melhor, falhando apenas em 0.66% para indicar o melhor modelo.

Por fim, a Tabela 3.10 mostra os melhores *AGEs*, correspondendo aos menores valores de BIC. É de realçar que os modelos escolhidos para as séries lineares são em tudo idênticos

Tabela 3.8: Resultados dos *AGEs* para a abordagem empírica à série **manchas**.

JTD	*AGE*	p	$RMQE_{tr}$	*AIC*	*BIC*	$RMQE_{pr}$
A	G1	14	16.5	1412	1461	19.9±0.4
	G2	27	14.6	1379	1474	19.2±0.5
B	G1	5	15.4	1364	1388	18.4±0.1
	G2	11	**14.6**	**1352**	1397	20.1±0.4
C	G1	7	15.4	1360	**1377**	**17.9**±0.0
	G2	13	15.4	1369	1400	18.8±0.1

Tabela 3.9: Desempenho dos critérios de selecção de modelos para os *AGEs*.

Série	$RMQE_{tr}$	*AIC*	*BIC*
passageiros	**2.67**%	**2.67**%	**2.67**%
papel	**2.00**%	**2.00**%	**2.00**%
mortes	**6.26**%	**1.01**%	**1.01**%
melborne	**0.00**%	**0.00**%	**0.00**%
química	**0.21**%	**0.21**%	**0.21**%
preços	3.88%	2.81%	**0.00**%
manchas	7.38%	12.2%	**0.00**%
kobe	**0.00**%	**0.00**%	**0.00**%
quadrática	3.57%	3.57%	**0.37**%
henon	**0.00**%	0.31%	0.31%
Média	2.60%	2.48%	**0.66**%

aos obtidos para as *RNUs*, com a excepção da série **química**. De facto, para uma mesma janela temporal, não existem diferenças entre uma *RNU* sem nodos intermédios e o modelo **G1**, pelo que a diferença de desempenho se deve unicamente ao tipo de aprendizagem utilizado. Tal comportamento já não se reflecte para as séries não lineares, onde embora a janelas seleccionadas sejam semelhantes, os modelos escolhidos são bem diferentes.

Tabela 3.10: *AGEs* com o menor valor de BIC.

Série	JTD	AGE	p	$RMQE_{pr}$
passageiros	D=<1,12,13>	G1	4	21.9±1.2
papel	D=<1,12,13>	G1	4	60.2±2.2
mortes	C=<1,11,12,13>	G1	5	135.9±1.7
melborne	C=<1,11,12,13>	G1	5	0.95±0.02
química	D=<1>	G2	3	0.36±0.00
preços	D=<1>	G1	2	7.49±0.01
manchas	C=<1,2,9,10>	G1	5	17.9±0.0
kobe	A=<1,2,...,13>	G2	27	604±36
quadrática	A=<1,2,...,13>	G2	27	0.37±0.01
henon	A=<1,2,...,13>	G2	27	0.58±0.01

3.7 Discussão e Conclusões

A análise de séries temporais fornece todo um conjunto de indicadores úteis para a *Previsão de Séries Temporais*, como a decomposição dos componentes sazonais e de tendência, coeficientes de auto-correlação e métricas de erro. Para avaliação dos modelos de previsão, foram seleccionadas diversas séries temporais, reais e artificiais, com diferentes características e oriundas de domínios distintos.

Em seguida, são apresentados os métodos convencionais de previsão, nomeadamente o *Alisamento Exponencial* e a metodologia de *Box-Jenkins*, assim como os modelos de previsão inspirados na Natureza, sendo efectuado um levantamento do estado da arte. São propostos dois modelos de previsão, sendo o primeiro baseado numa *RNU* e o segundo num *AGE* com uma representação de valores reais. Ambos estes modelos estão definidos de acordo com uma *Janela Temporal Deslizante*, que compreende um conjunto de deslocamentos passados, a partir do qual o modelo realiza a previsão.

Um dos problemas que surge com a aprendizagem supervisionada é a perda de generalização, que ocorre quando um modelo aprende em demasia, memorizando características

particulares, como o ruído. Uma das formas de evitar esta perda de generalização, passa pela *selecção de modelos*, onde diferentes modelos são comparados de acordo com uma estimativa para o erro de generalização. Para se estimar este podem-se adoptar estatísticas simples, que penalizam a complexidade de um modelo, como os critérios AIC e BIC.

Por último, foi elaborado um estudo sobre a selecção dos modelos de previsão inspirados na Natureza, para as séries temporais apresentadas anteriormente. Aqui, foram definidas quatro regras heurísticas, baseadas na análise de séries temporais, para a construção das janelas temporais. O estudo permite concluir que o desempenho dos modelos de previsão propostos depende de dois factores principais: a janela temporal e a complexidade do modelo de aprendizagem. Mais ainda, uma comparação entre diferentes critérios de selecção, revelou que a estatística BIC origina uma melhor generalização, quer para as *RNUs* quer para os *AGEs*.

Capítulo 4

Optimização de Modelos de Previsão

> "Everyone designs who devises courses of action aimed at changing existing situations into preferred ones."
>
> – H. A. Simon, 1969.

É explicada a necessidade de optimização de processos computacionais para a resolução de problemas a partir de modelos inspirados na Natureza, sendo apresentadas as principais estratégias de procura. Fornece-se uma descrição das Redes Neuronais Evolucionárias e dos Meta-Algoritmos Genéticos e Evolucionários, sendo ambas estas técnicas aplicadas para a optimização de tais processos. Por último, é realizada uma comparação entre os diferentes modelos considerados como objecto de estudo, em termos de desempenho, assim como dos tempos de computação.

No capítulo anterior referiu-se que o desempenho de um modelo de previsão depende de dois factores principais: a janela temporal e a complexidade do modelo. Foi também demonstrado que a estatística *BIC* é um bom critério para a selecção destes factores.

No entanto, foi igualmente revelado que para as *RNUs,* este critério tende a favorecer modelos simples, mesmo para séries não lineares. Tal sucede porque ocorre um significativo aumento no número de parâmetros da rede, quando se acrescentam nodos intermédios, devido à adopção de uma topologia de rede completamente interligada.

Mais ainda, as heurísticas para as janelas deslizantes foram desenvolvidas com base nos valores de autocorrelação. Ora, estes apenas medem interacções lineares, que não são as

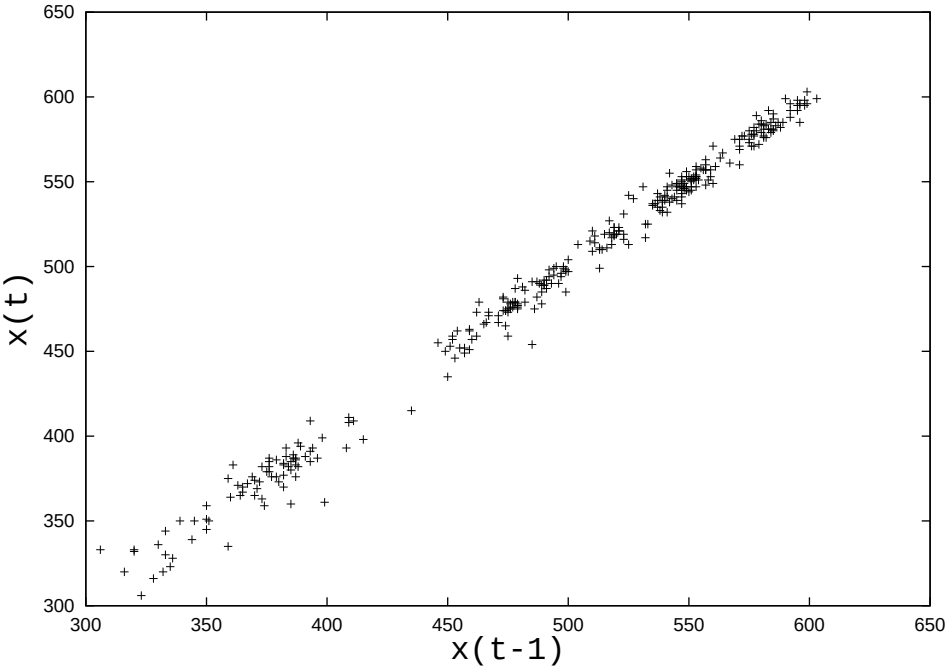


Figura 4.1: Interacções entre os deslocamentos $t-1$ e t para a série **preços**.

mais adequadas para séries complexas. Por exemplo, a autocorrelação de deslocamento um para a série **preços** tem um valor próximo de um ($r_1 = 0.99$), medindo correctamente a elevada relação linear existente (Figura 4.1). Todavia, o mesmo não se sucede para a série **quadrática**, onde o valor de autocorrelação é quase nulo ($r_1 = 0.03$), apesar de existir uma forte interacção não linear, visível na forma de uma parábola (Figura 4.2).

Ao invés de se utilizar heurísticas, o ideal seria a adopção de métodos automáticos para a procura da melhor janela temporal e modelo de previsão. Assim, este capítulo irá abordar a optimização automática, sob a forma de um *AGE*, para o desenho de cada um dos modelos inspirados na Natureza.

4.1 Estratégias de Procura

Existem diferentes tipos de estratégias de procura para máquinas de aprendizagem, podendo estas ser classificadas segundo três classes principais [Banzhaf et al., 1998]:

Procura Cega - Nesta estratégia, o espaço de soluções é explorado sem ser influenciado pelos resultados de procuras anteriores. Em geral, a procura utiliza uma estrutura em árvore, onde cada nodo contém uma solução candidata. Como exemplos desta

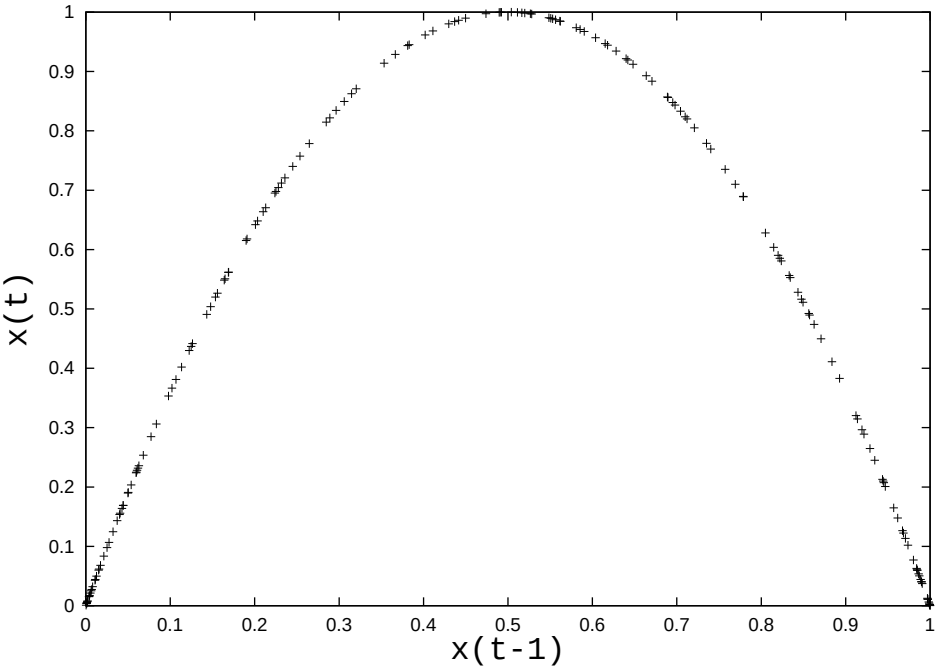


Figura 4.2: Interacções entre os deslocamentos $t-1$ e t para a série **quadrática**.

estratégia tem-se as procuras de *primeiro em largura*[1] e *primeiro em profundidade*[2] .

Hill-Climbing - Aqui é utilizado um único ponto de procura, criando uma nova solução a partir da anterior, sendo geralmente esta a adoptada (no caso de ser melhor que a anterior). Nenhum registo é mantido sobre o caminho percorrido. Várias são as técnicas que utilizam esta estratégia, sendo de referir:

- *Simulated Annealing* - Trata-se de um método genérico de procura, que tem a ver com fenómenos que ocorrem com o processo de arrefecimento de metais. A procura envolve o uso de saltos aleatórios a partir de um ponto, adoptando-se a solução alterada de acordo com um parâmetro de temperatura, que decresce, à medida que o processo decorre.

- *Algoritmos de corte* - Desenvolvidos especificamente para optimizar quaisquer tipos de topologias para *RNAs*, a partir da remoção de nodos e conexões desnecessárias, segundo o princípio de *Ockham*[3]; i.e., tenta-se obter a rede mais simples que melhor

[1] Conhecida em inglês por *breadth-first*.

[2] Do inglês *depth-first*.

[3] William of Ockham – 1349. Filósofo e frade franciscano inglês, conhecido pela defesa da aplicação da regra económica à teologia: "entidades não são multiplicadas para além da necessidade", tão frequentemente utilizada que se tornou famosa como a *navalha de Ockham*.

se adapte à resolução de um dado problema. As principais funções de um algoritmo de corte passam pela escolha da(s) unidade(s) a ser cortada(s), quando efectuar o corte e quando parar o treino da rede [Thimm & Fiesler, 1997].

- *Algoritmos construtivos* - Igualmente desenvolvidos para optimizar quaisquer tipos de topologias para *RNAs,* recorrem a uma estratégia algo diferente, começando com uma rede simples, à qual são acrescentados nodos e conexões, até que uma solução satisfatória seja encontrada [Kwok & Yeung, 1999].

Procura por Feixe - Este tipo de procura passa por um compromisso entre uma procura exaustiva do espaço de soluções e o *hill-climbing*, sendo mantida uma população de pontos de procura. Em cada ponto é utilizado uma métrica para seleccionar os pontos promissores, sendo os restantes descartados. Como exemplos têm-se todas as técnicas da *CGE*, tais como os *AGEs* ou a *PG*.

Em princípio, a *procura cega* exige a consideração de todas as soluções possíveis, sendo por isso apenas aplicável quando o espaço de procura é diminuto, o que não se verifica na maioria dos problemas de aprendizagem. Há variantes que não exigem que se tome em linha de conta todas as soluções, actuando através de um compromisso entre o tempo e o espaço de procura (e.g., técnicas de *aprofundamento progressivo*[4] [Korf, 1985]). No entanto, tais variantes apenas procuram soluções para o problema num sub-conjunto deveras restrito do espaço de soluções.

Por outro lado, os métodos de *hill-climbing*, como os algoritmos de corte ou construtivos, são susceptíveis aos mínimos locais [Angeline et al., 1994]. Mais ainda, não raras vezes, estes métodos tendem igualmente a fazer incidir a sua atenção apenas num subconjunto restrito do espaço de procura.

Neste trabalho ir-se-á adoptar a última estratégia, a *procura por feixe*, que se irá materializar através dos *AGEs*. Estes utilizam uma procura global multi-ponto, dando origem a soluções de elevada qualidade, que escapam a mínimos locais [Branke, 1995].

4.2 Optimização do Modelo Neuronal

4.2.1 Redes Neuronais Evolucionárias

A combinação de paradigmas, tais como o conexionista e o evolucionário para a resolução de problemas, tem como forte inspiração as formas de evolução que ocorrem nos seres vivos,

[4] Tradução adoptada para o termo *iterative deepening*.

estando a ser objecto de atenção por aqueles que se dedicam à análise, especificação e desenvolvimento de sistemas computacionais de alto desempenho. Em particular, a combinação de *RNAs* com *AGEs*, dando origem às *Redes Neuronais Evolucionárias (RNEs),* pode-se processar de diversas formas, algumas das quais são indicadas a seguir [Whitley, 1995]:

- no *treino*, para determinar os valores dos pesos para topologias de rede fixadas à priori;
- no *desenho de novas topologias*, para evoluir topologias de rede, em termos de nodos intermédios e conexões; e
- na *selecção dos dados para treino*, em termos de uma redução da cardinalidade do conjunto dos valores de entrada ou selecção de casos de treino.

A atenção irá ser virada para as últimas duas tarefas, de forma a resolver as questões levantadas no capítulo anterior, ou seja, qual é a melhor janela temporal e melhor topologia da *RNU* para uma dada série temporal. Estas questões têm vindo a ser resolvidas pela combinação de heurísticas com processos de *tentativa-e-erro*, como foi visto anteriormente, mas que tendem a não ser as mais eficazes soluções para o problema, quando o espaço de procura envolvido é substancial.

As *RNEs* começaram a ser utilizadas no final dos anos oitenta, com o trabalho de Miller, Todd e Hedge [1989]. Um ano mais tarde, Whitley e seus colaboradores demonstraram que os *AGEs* podem ser utilizados para encontrar novas topologias de rede com velocidades de aprendizagem superiores às das redes completamente interligadas, não se descurando também a pressão para a selecção de redes de baixa cardinalidade. Desde então, o campo tem crescido e diversos tipos de *RNEs* têm sido apresentados, distinguindo-se pelo tipo de representação, dando corpo a duas visões sobre o tema: a *forte* e a *fraca* [Branke, 1995]. Na primeira, e em termos de codificação, feita ao baixo nível, cada gene codifica directamente a existência de um nodo ou conexão. No outro extremo tem-se a visão *frac*a, de alto nível, onde existem regras para a construção da rede (e.g., *desenvolvimento celular* [Gruau & Whitley, 1993]).

Uma das desvantagens da codificação directa está no aumento explosivo do comprimento do cromossoma, à medida que a cardinalidade da rede aumenta, o que afecta a sua escalabilidade. Assim, é recomendada apenas para redes de pequena cardinalidade, sendo esta previamente definida pelo utilizador. Este procedimento tem a vantagem de impedir que uma rede cresça em demasia e, com isso, que venha a perder capacidade de generalização.

Por outro lado, na codificação indirecta, nem todas as estruturas são igualmente prováveis, sendo favorecidas as estruturas regulares. Embora tal possa ser útil para certas aplicações, também corre-se o risco de eliminar as soluções porventura mais adequadas e que escapem à intuição humana. É de referir ainda que a codificação indirecta exige um maior esforço computacional, devido ao processo de descodificação das soluções.

4.2.2 Modelo Proposto

Alguns estudos sobre *RNAs* aplicadas à previsão recorrem a heurísticas para a definição das topologias da rede [Cortez et al., 1995], embora a maioria utilize a intuição ou métodos de *tentativa-e-erro* [Tang & Fishwick, 1993; Faraday & Chatfield, 1998]. As *RNEs* corporizam uma alternativa promissora para automatizar este processo, eliminando o procedimento fastidioso de *tentativa-e-erro*.

A grande maioria dos trabalhos sobre *RNEs* aplicadas à previsão opta por uma codificação directa, sendo representados no cromossoma uma panóplia de parâmetros tais como o número de nodos de entrada e intermédios, os pesos iniciais, as funções de activação e taxas de aprendizagem [Cortez et al., 1996; Falco et al., 1998; Chen & Lu, 1999]. Embora cada um destes parâmetros possa influenciar o modelo final de previsão, à luz do espírito do critério BIC e das experiências relatadas no capítulo anterior, parece ser mais importante um ajustamento correcto da janela temporal e da representatividade ou complexidade do modelo, em termos do número total de conexões. No trabalho de LeBaron [1995], apenas esta última parte foi considerada, sendo explorado o espaço de procura de conexões, para um janela fixa, para prever a série caótica **henon**.

Ora, nesta tese é proposto este modelo, onde se adopta um AGE binário, em que cada indivíduo denota uma nova topologia para uma *RNU*, sendo utilizada como topologia base a descrita na Secção 3.3. Foi decidido utilizar uma codificação directa, dado que as experiências anteriores demonstraram que se conseguem bons resultados em previsão com redes de baixa cardinalidade. Se o valor de um dado gene for 1, então a conexão correspondente existirá, caso contrário não é considerada. A excepção ocorre nas conexões entre os nodos intermédios e o nodo de saída, uma constante do modelo adoptado, dado que esta estratégia favorece a geração de redes válidas.

A construção da topologia de rede é realizada via uma matriz de pesos, estando os genes organizados segundo as linhas da matriz, que denotam as ligações que entram num dado nodo (Figura 4.3). De referir que este tipo de codificação tem a vantagem de facilitar a junção de nodos de diferentes redes, via o operador de cruzamento. Quanto à cardinalidade

do cromossoma, esta será dada pela expressão:

$$(n+1) \times (n_i+1) \tag{4.1}$$

para uma rede com n nodos de entrada e n_i nodos intermédios.

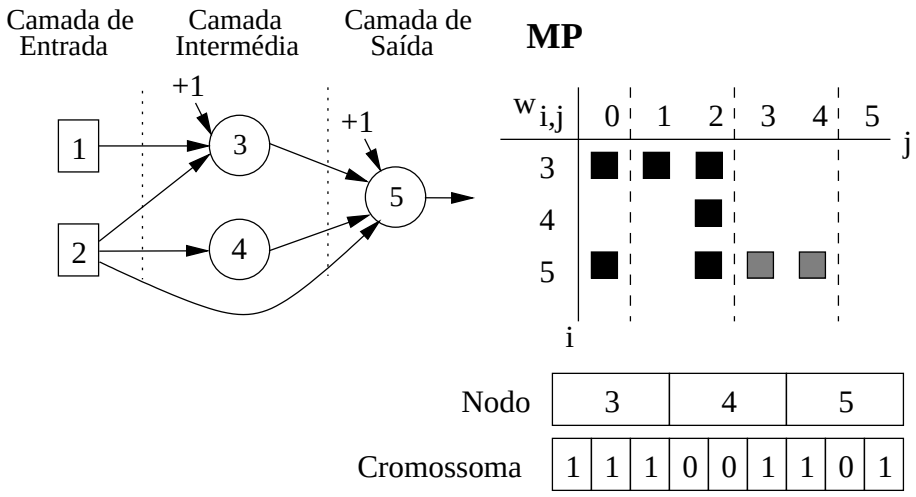


Figura 4.3: Esquema de codificação de uma *RNU*.

Dentro deste cenário, ocorrerá um corte num nodo intermédio, quando este não receber conexões a partir das entradas (Figura 4.4). De um modo similar, ocorrerá um corte num nodo de entrada, e também no correspondente deslocamento temporal, quando o nodo não tiver conexões de saída (Figura 4.5). Esta estratégia permite explorar um espaço de procura que contém o pleno das topologias de rede, desde a simplesmente linear, sem nodos intermédios, até à completamente interligada, assim como faz a exploração de qualquer subconjunto de deslocamentos temporais.

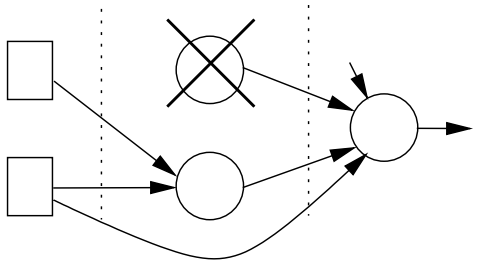


Figura 4.4: Corte num nodo intermédio.

A função de aptidão é obtida através de uma descodificação do cromossoma para a *RNU*, que por sua vez é treinada, sendo finalmente calculado o seu valor de BIC. Assim,

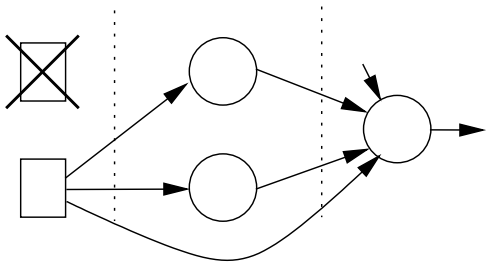


Figura 4.5: Corte num nodo de entrada.

o objectivo do *AGE*, passa pela a procura da topologia e janela temporal que minimizam este valor. A adopção do critério *BIC* apresenta três vantagens fundamentais sobre as demais (e.g., *Validação Cruzada*) :

- exige um cálculo simples, sendo bastante menos exigente em termos computacionais do que outros estimadores dos parâmetros de generalização da rede;
- é um bom estimador da generalização da rede, conforme o que foi demonstrado anteriormente (Secção 3.6.3); e
- aumenta a pressão para a selecção de topologias de rede deveras simples, acelerando o processo de convergência do *AGE*.

O funcionamento global deste sistema é descrito na Figura 4.6. Ao nível da programação, foram adoptados os ambientes de programação *APAGE* e *APRNA*, com a interacção do *AGE* com a *RNU* a ser definida no módulo de avaliação, sendo este sistema integrado na aplicação **tsf**.

4.2.3 Experiências

Neste caso, o *AGE* irá comportar-se como um procedimento de segunda ordem, pelo que a escolha dos seus parâmetros não é tão crítica, em relação ao modelo de *AGE* para previsão (Secção 3.6.3). Assim, e dado tratar-se de uma representação binária, foi decidido adoptar o modelo original de Holland (Figura 2.6), por uma questão de simplicidade.

A população inicial é gerada de forma aleatória, com o valor dos genes a pertencer ao alfabeto {0,1}. O número máximo de nodos de entrada (n) e intermédios (n_i) foi ajustado aos valores 13 e 6, dado que as experiências anteriores apontam para o uso de redes de baixa cardinalidade. Uma outra política iria aumentar o espaço de procura sem benefícios

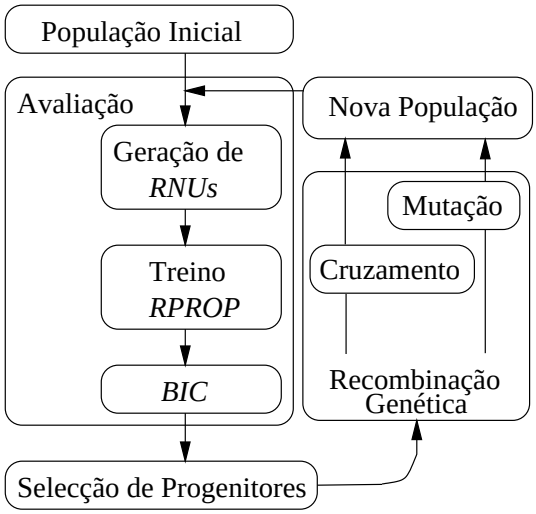


Figura 4.6: Esquema de funcionamento do sistema de optimização neuronal.

acrescidos. Por conseguinte, o cromossoma tem um comprimento de $14 \times 7 = 98$ genes (equação 4.1), o que aponta para um espaço de procura de 2^{98} soluções distintas.

A cardinalidade da população foi fixada em $TP = 100$. Foi utilizado um cruzamento de dois pontos, responsável pela criação de 80% dos novos indivíduos, sendo a mutação binária responsável pela criação dos restantes elementos. O processo de selecção é realizado através da utilização do valor de aptidão por forma a seriar as soluções (conversão por ordenação), sendo depois aplicada uma amostragem estocástica com substituição, que garante uma supremacia estocástica dos mais aptos.

A invocação do *AGE* é dada como finda quando $T = 500$ gerações, dado que os melhores indivíduos foram sempre encontrados em gerações anteriores. Por último, ao melhor indivíduo encontrado pelo *AGE,* são aplicados trinta (30) treinos independentes, para providenciar uma maior relevância estatística. De referir ainda que, para as *RNUs,* foram considerados todos os parâmetros constantes da Tabela 4.1.

As melhores topologias para as *RNUs,* obtidas a partir *RNEs,* são apresentadas nas Figuras 4.7 e 4.8. Verifica-se que em todas as redes existem ligações directas entre as entradas e saídas (atalhos), o que atesta a utilidade destas. O mesmo não sucede com as conexões de *bias*, estando estas apenas presentes em quatro séries (**papel**, **kobe**, **quadrática** e **henon**). Por sua vez, os nodos intermédios tendem a receber um número diminuto de conexões a partir das entradas (entre 1 e 4).

A Tabela 4.2 descreve os melhores modelos de *RNUs* encontrados pelo *AGE* binário, para todas as séries da Secção 3.1.3, os quais são definidos em termos da sua janela tempo-

Tabela 4.1: Valores dos parâmetros do sistema de optimização neuronal.

	Sistema de Optimização	**Modelo**
Características	*AGE* binário	*RNU*
Estrutura	$TP = 100$	$n - n_i - 1$
Parâmetros	98	$(n+1)(n_i+1)$
Inicialização	Aleatória {0,1}	Aleatória $[\frac{-2}{k}, \frac{2}{k}]$
Aprendizagem	Algoritmo de Holland	Algoritmo *RPROP*
	Cruzamento de 2 pontos (80%)	$\Delta_0 = 0.1$
	Mutação binária (20%)	$\Delta_{max} = 50.0$
Termo	$T = 500$	$ft = 5$, $\phi = 0.1$
		$\varphi = 1000$

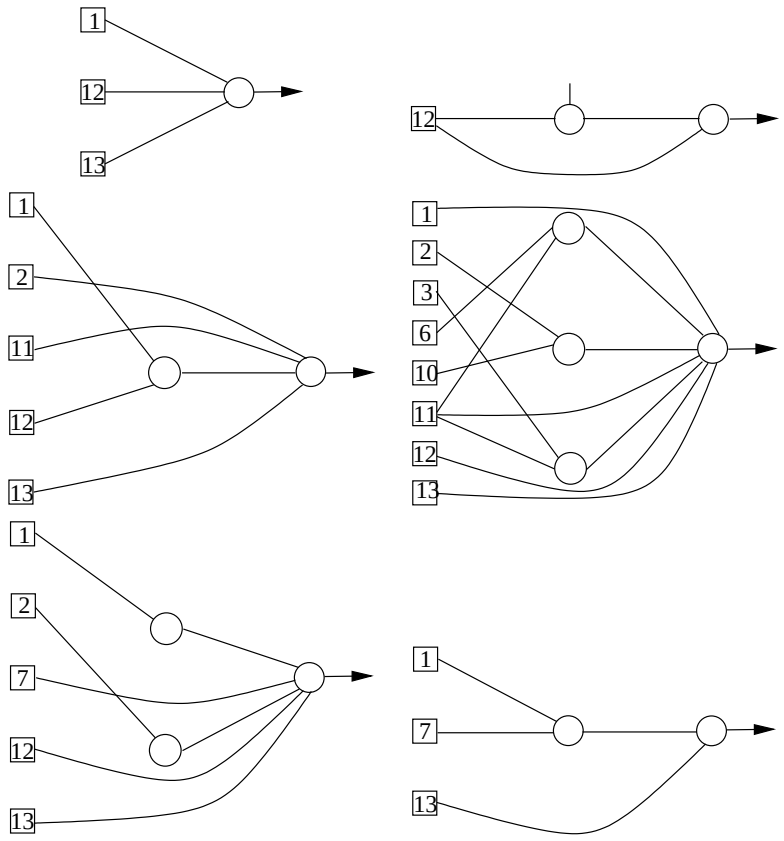


Figura 4.7: Topologias de rede obtidas para as séries sazonais e com tendência **passageiros**, **papel**, **mortes**, **melborne**, **química** e **preços**.

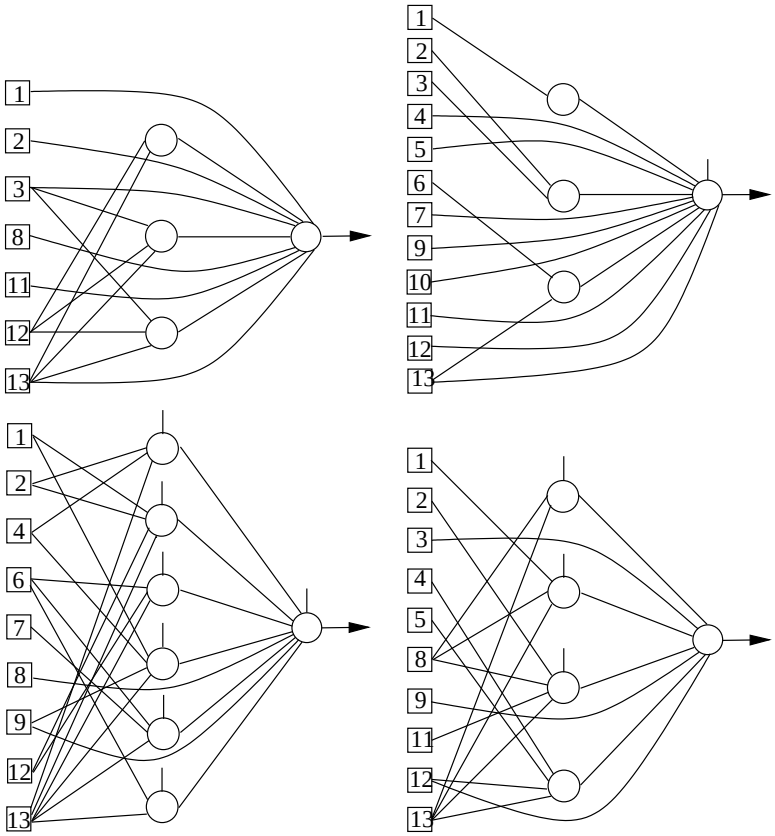


Figura 4.8: Topologias de rede obtidas para as séries não lineares **manchas**, **kobe**, **quadrática** e **henon**.

ral (coluna *JTD*), número de nodos intermédios (coluna n_i) e número total de conexões (p). A coluna R denota o factor de redução da topologia (em percentagem), quando se compara a topologia obtida pelo sistema de *RNEs* com a topologia completamente interligada equivalente, ou seja:

$$R = 1 - \frac{p}{p_{ci}} \tag{4.2}$$

onde p_{ci} denota o número total de conexões da topologia de rede completamente interligada. Por exemplo, para a série **manchas** tem-se que $R = 1 - 17/35 = 51\%$. Por último, a coluna $RMQE_{pr}$ apresenta o erro de previsão para cada modelo, em termos da média dos trinta treinos, com um grau de significância de 95%.

Tabela 4.2: Características dos modelos de previsão obtidos via a optimização neuronal.

Série	*JTD*	n_i	p	R	$RMQE_{pr}$
passageiros	<1,12,13>	0	3	25%	18.2±0.3
papel	<12>	1	4	20%	52.5±0.6
mortes	<1,2,11,12,13>	1	6	54%	132±1
melborne	<1,2,3,6,10,11,12,13>	3	13	67%	0.87±0.02
química	<1,2,7,12,13>	2	7	65%	0.36±0.01
preços	<1,7,13>	1	4	56%	7.49±0.01
manchas	<1,2,3,8,11,12,13>	3	17	51%	17.4±0.5
kobe	<1,2,3,4,5,6,7,9,10,11,12,13>	3	17	69%	498±8
quadrática	<1,2,4,6,7,8,9,12,13>	6	34	55%	0.01±0.00
henon	<1,2,3,4,5,8,9,11,12,13>	4	23	61%	0.24±0.02

Em geral, a incidência do factor de redução (R) sobre as topologias de rede (o ganho em percentagem na redução das ligações entre nodos) ocorre com valores entre os 50% e 70% (a excepção ocorre nas topologias de rede de menor cardinalidade, como no caso das séries **passageiros** e **papel**). Tal facto demonstra que o *AGE* consegue efectuar um corte substancial no número total de conexões possíveis, o que resulta em modelos mais simples.

Comparando estes resultados com os obtidos pelas heurísticas utilizadas no capítulo anterior (Tabela 3.7), conclui-se que:

- existe uma diferença ao nível das janelas temporais seleccionadas, com a excepção da primeira série, **passageiros**;
- o sistema de optimização neuronal tende a escolher as arquitecturas mais simples, que contêm um menor número de conexões, apesar de possuírem mais nodos intermédios; e

- o sistema actual apresenta melhores previsões, especialmente para as séries não lineares (a excepção ocorre com a série **paper**).

Aqui, a flexibilidade do *AGE* permite seleccionar modelos com menores valores de *BIC* para topologias de rede com mais nodos intermédios e menos parâmetros, resultando em melhores previsões.

Convém referir ainda que as topologias de rede escolhidas parecem também estar ajustadas ao grau de complexidade da série. Para as séries lineares, estes contêm entre 3 a 13 conexões, um valor que aumenta para 17 para as séries não lineares, elevando-se até às 23 e 34 conexões para as séries do tipo caótico.

A título demonstrativo, a Figura 4.9 apresenta os últimos 10% dos elementos da série **manchas**, bem como as respectivas previsões fornecidas pelo sistema de *RNEs*, considerando a média das trinta aprendizagens. Como se pode verificar, os valores de saída da *RNU* encontram-se próximos das observações reais, revelando uma boa generalização por parte do modelo.

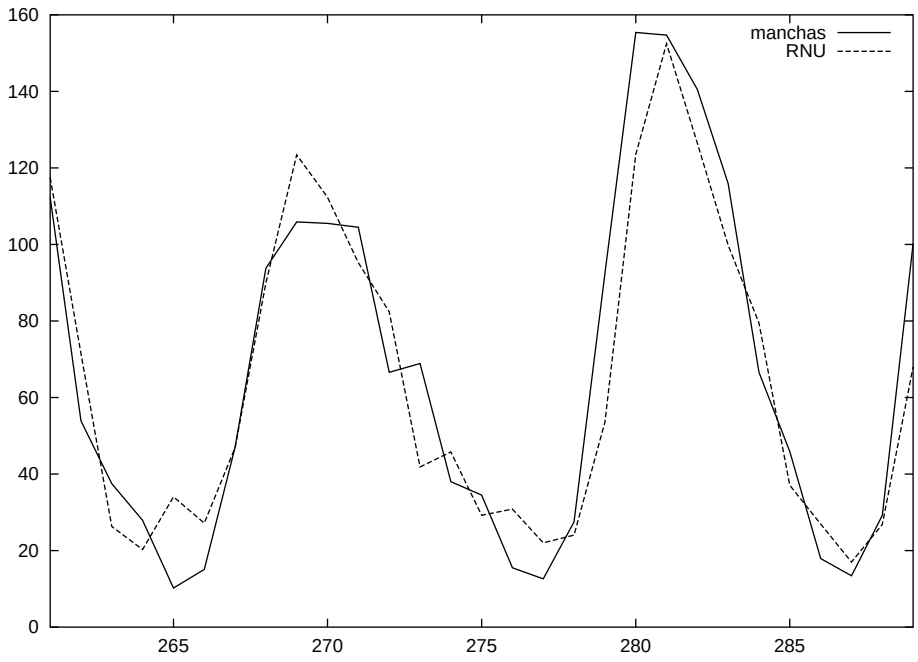


Figura 4.9: Previsões via sistema de *RNEs* para a série **manchas**.

4.3 Optimização do Modelo Evolucionário

4.3.1 Meta-Algoritmos Genéticos e Evolucionários

O desempenho dos *AGEs* em problemas de optimização é afectado por um certo número de factores. Alguns destes factores são determinados pelo problema a resolver, sendo não raras vezes incontroláveis. Outros são factores que determinam o próprio funcionamento do *AGE*, podendo ser ajustados para melhorar a eficácia do *AGE*. Estes parâmetros incluem, entre outros, a cardinalidade da população, as taxas de aplicação dos operadores de cruzamento e mutação, a taxa de renovação da população, o processo de selecção.

A influência destes factores no desempenho de um *AGE* não pode ser descrita por um processo analítico, dado que um *AGE* é por definição um modelo estocástico. Isto significa que os métodos convencionais da *Teoria de Controlo* ou de deveras comuns técnicas de optimização, não são os mais eficazes para o tratamento destes factores. Assim, nos últimos anos assistiu-se ao desenvolvimento de novas técnicas para a optimização de *AGEs*, através de duas correntes, que utilizam ou um *ajustamento adaptativo*[5] ou *Meta-AGEs* [Petrovski & McCall, 1997].

Os métodos de ajustamento adaptativo determinam os valores dos parâmetros do *AGE* de uma forma dinâmica, durante a sua execução, através de, por exemplo, processos de *hill-climbing*. Embora sejam potencialmente promissores, estes métodos apresentam uma série de desvantagens. Em primeiro lugar, o auto-ajustamento só é aplicável aos factores activos, como a taxa de mutação. A dificuldade surge com os factores que não estão directamente envolvidos na produção de novas soluções (e.g., a cardinalidade da população). Por outro lado, a própria afinação dos métodos de ajustamento depende da intuição do programador, dificultando uma análise objectiva do desempenho do *AGE*.

A alternativa à utilização dos métodos de ajustamento passa pela utilização dos *Meta-AGEs*, propostos por Grefenstette [1986]. Os *Meta-AGEs* utilizam *AGEs* a um nível meta para determinar valores fixos de parâmetros do *AGE* ao nível objecto. Grefenstette demonstrou empiricamente que os *Meta-AGEs* conseguem um melhor desempenho, em relação à aplicação de regras *ad hoc*, quando se trata de *AGEs* binários.

Os *Meta-AGEs* são atractivos sob o ponto de vista da optimização de *AGEs* devido a dois factores: permitem explorar um elevado número de parâmetros, independentemente da cardinalidade do espaço de procura; e por outro lado porque realizam uma procura automática não tendenciosa. Todavia, têm a desvantagem de exigir um elevado esforço

[5]Conhecido em inglês por *Adaptive Tuning*.

computacional.

4.3.2 Modelo Proposto

Os promissores resultados da optimização de sistemas computacionais baseados em *RNAs* são um bom indicador de que também poderão existir benefícios numa optimização do modelo evolucionário. De notar que a gama dos modelos de *AGEs,* testados no capítulo anterior, é porventura redutora, já que envolve apenas as heurísticas de janelas temporais e os modelos **G1** e **G2**. Não foram testadas nem diferentes combinações de janelas, nem os modelos $ARMA(P,Q)$, em que $P \neq Q$.

Ora, para colmatar estas limitações, é proposto um *Meta-AGE* binário, que ao invés de optimizar os próprios parâmetros do *AGE* ao nível objecto, irá codificar modelos $ARMA(P,Q)$, a serem optimizados via *AGEs* com recurso a representações na base 10. Assim, o cromossoma que codifica possíveis soluções para o problema apresenta uma cardinalidade (uma medida dos seus genes) dada por:

$$1 + P_{max} + Q_{max} \tag{4.3}$$

e que corresponde à existência (ou não) dos coeficientes do modelo, μ, A_i e M_j, e em que P_{max} e Q_{max} denotam o limite superior dos coeficientes admitidos para os modelos AR e MA (Figura 4.10).

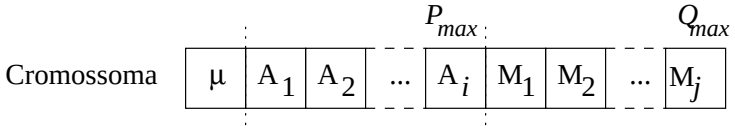


Figura 4.10: Esquema de codificação do *Meta-AGE*.

Se valor de um dado gene for 1, então o coeficiente correspondente será codificado ao nível do cromossoma, sendo descartado no caso contrário. No capítulo anterior, as janelas temporais AR e MA coincidiam, pelo que se tornava apenas necessário definir uma delas. No presente contexto, tal não será a situação comum, na medida em que muito frequentemente se terá $P \neq Q$, pelo que será conveniente diferenciar as janelas. Sendo assim, é possível agora apresentar o procedimento de descodificação de um cromossoma do *Meta-AGE* (Figura 4.11).

Os parâmetros do modelo definido pelo *Meta-AGE* são posteriormente estimados pelo *AGE* de baixo nível (ou nível objecto), que codifica as soluções na base 10, com cada gene

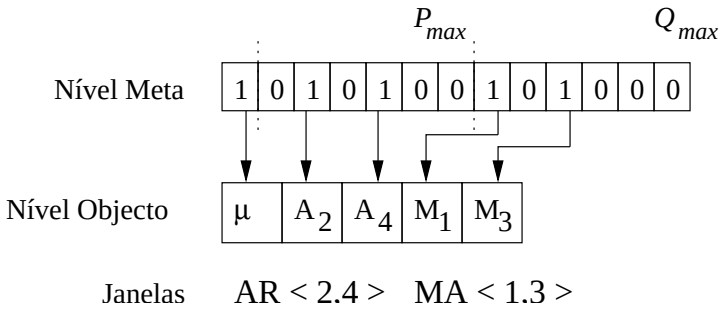


Figura 4.11: Descodificação de um indivíduo do *Meta-AGE*.

a referir-se a um coeficiente do modelo $ARMA$, em que é considerada uma configuração em tudo análoga à utilizada na Secção 3.6.3.

Por sua vez, o funcionamento do sistema do *Meta-AGE* é em tudo semelhante ao do processo subjacente à optimização neuronal (Figura 4.6). A aptidão de cada meta-indivíduo é então obtida após a descodificação do seu cromossoma no AGE de baixo nível, invocando-se o respectivo processo de aprendizagem e, finalmente, calculando-se o valor de BIC do modelo optimizado.

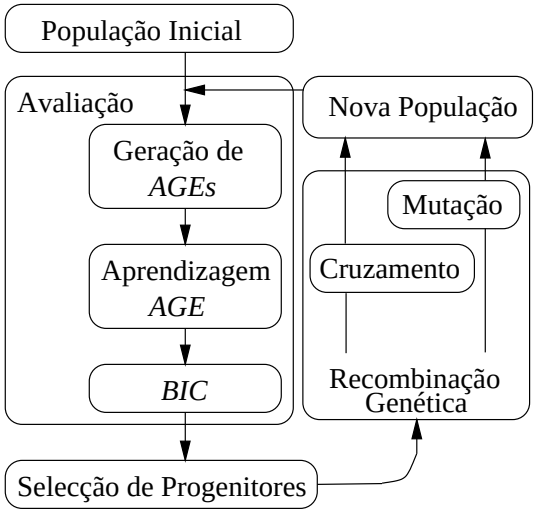


Figura 4.12: Esquema de funcionamento do sistema de optimização evolucionária.

Em termos de programação, este sub-sistema foi adicionado à aplicação **tsf**, na medida em que a interacção do *Meta-AGE* com o AGE ao nível objecto é também definida no módulo de avaliação do primeiro.

4.3.3 Experiências

À semelhança do sistema de *RNEs*, este *Meta-AGE* binário funcionará igualmente como um algoritmo de optimização de segunda ordem, pelo que a escolha dos seus parâmetros não é deveras crucial. Por uma questão de simplicidade e de comparação, foi decidido adoptar os mesmos parâmetros do AGE anteriormente utilizado para optimizar o modelo neuronal, ou seja, foi considerado(a);

- o AGE binário proposto por Holland;
- a população inicial é gerada de forma aleatória, tomando valores do conjunto {0,1};
- a cardinalidade da população é ajustada para o inteiro 100;
- uma recombinação genética via cruzamento de dois pontos e mutação binária;
- uma função de aptidão definida pelo valor de BIC;
- uma selecção baseada na ordem e amostragem estocástica com substituição; e
- um *Meta-AGE* cuja invocação é dada como finda ao fim de $T = 500$ gerações.

Seguindo o esquema mental adoptado anteriormente, ambos os limites de ordem foram definidos no valor de $P_{max} = Q_{max} = 13$, o que dá origem a um cromossoma com um comprimento de 27 genes (1 para a constante e 13 para os coeficientes AR e MA).

Para o AGE ao nível objecto, foram utilizados os valores padrão da Secção 3.6.3 (Tabela 4.3).

A Tabela 4.4 descreve os melhores modelos de $ARMA$ encontrados pelo sistema de *Meta-AGE*s, para todas as séries da Secção 3.1.3. Cada modelo é definido pelo seu conjunto de coeficientes AR e MA, sendo também apresentado o número total de parâmetros (coluna p). Ao melhor modelo encontrado foram aplicadas trinta optimizações independentes do AGE de baixo nível, para garantir uma maior evidência estatística, sendo os erros de previsão definidos em termos da média, com uma significância de 95% (coluna $RMQE_{pr}$). De uma comparação destes resultados, com os obtidos anteriormente via as janelas heurísticas (Tabela 3.10), pode afirmar-se que:

- as janelas temporais seleccionadas (AR e MA) são bem diferentes; aqui, regra geral, a cardinalidade da janela AR é a menor, e o modelo MA é, na maioria dos casos, o mais considerado (aqui utilizado em 7 séries);

Tabela 4.3: Valores dos parâmetros do sistema em termos do processo de optimização evolucionária.

	Sistema de Optimização	**Modelo**
Características	AGE binário	AGE com $RVRs$
População	100	50
Parâmetros	27	$\mu + P + Q$
Aptidão	BIC	$RMQE_{tr}$
Inicialização	Aleatória {0,1}	Aleatória $[-1, 1]$
Aprendizagem	Algoritmo de Holland	Algoritmo Generalista
	$VE = 0$, $TS = 100\%$	$VE = 1$, $TS = 60\%$
Cruzamento	Dois pontos (80%)	Aritmético (67%)
Mutação	Binária (20%)	Gaussiana (33%)
Terminação	$T = 500$	$T = 1000$

Tabela 4.4: Características dos modelos $ARMA$ obtidos via optimização evolucionária.

Série	AR	MA	p	$RMQE_{pr}$
passageiros	<12>	<1,2,3,9,12>	7	17.2±0.2
papel	<12>	<>	2	52.5±0.1
mortes	<1,11,12>	<13>	4	137±2
melborne	<1,7,11,12>	<>	4	0.93±0.04
química	<1,2>	<1,2,3,4,7,11>	8	0.34±0.00
preços	<1>	<>	1	7.48±0.00
manchas	<1,2,3,9,10>	<1,9>	8	17.6±0.2
kobe	<1,2,3,7,8,9,13>	<1,3,5,6,12>	12	493±10
quadrática	<7,10>	<4,5>	5	0.35±0.00
henon	<2,3>	<1,8,10>	6	0.57±0.00

- o *Meta-AGE* tende a escolher os modelos mais simples (i.e., com menor cardinalidade) para as séries não lineares (a excepção ocorre com a série **manchas**); e
- a optimização evolucionária selecciona *AGEs* que levam às melhores previsões, com a excepção da série **mortes**.

Por fim, será apresentado um exemplo de previsão para o sistema de *Meta-AGEs*, sendo escolhida a série **kobe**. A Figura 4.13 exibe os valores médios das previsões realizadas via o *AGE* para os últimos 20 (vinte) elementos desta série (i.e., os elementos de teste). Neste caso, ambas as curvas se encontram próximas, revelando um bom ajuste por parte do modelo escolhido.

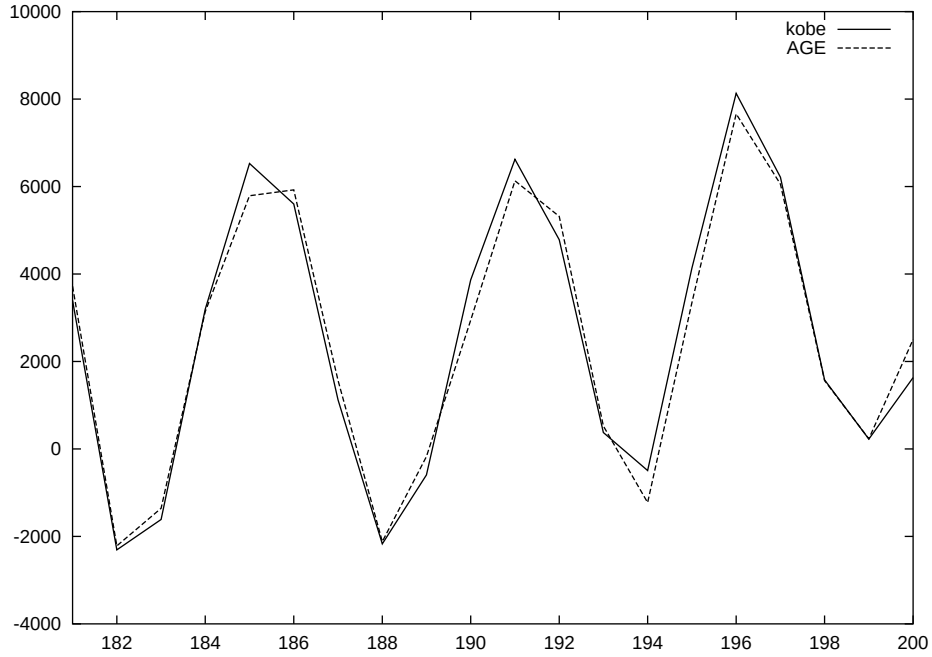


Figura 4.13: Previsões via o modelo óptimo *ARMA* para a série **kobe**.

4.4 Comparação entre Modelos

4.4.1 Eficácia de Previsão

Uma vez que se procedeu a uma optimização dos modelos inspirados na Natureza, eis que se chegou ao ponto ideal para se efectuar uma comparação com os modelos designados por convencionais.

Na Tabela 4.5 é apresentado o desempenho de cada modelo para todas as séries da Secção 3.1.3, em termos de duas métricas para a avaliação do erro: o $RMQE_{pr}$ e o $MNQE_{pr}$ (em parêntesis). Esta última medida foi incluída para facilitar a comparação entre séries e modelos diferentes.

Tabela 4.5: Comparação entre os diferentes métodos de previsão.

Série	**AE**	**BJ**	**RNEs**	**Meta-AGE**
passageiros	**16.5** (0.70%)	17.8 (0.81%)	18.2 (0.84%)	17.2 (0.75%)
papel	**49.2** (4.4%)	61.0 (6.8%)	52.5 (5.0%)	52.5 (5.0%)
mortes	135 (37%)	144 (42%)	**132** (36%)	137 (38%)
melborne	**0.72** (2.5%)	1.07 (5.6%)	0.87 (3.8%)	0.93 (4.3%)
química	0.35 (51%)	0.36 (53%)	0.36 (54%)	**0.34** (48%)
preços	7.54 (0.39%)	7.72 (0.41%)	7.49 (0.38%)	**7.48** (0.38%)
manchas	28.3 (35%)	21.4 (20%)	**17.4** (13%)	17.6 (14%)
kobe	3199 (105%)	582 (3.5%)	498 (2.6%)	**493** (2.5%)
quadrática	0.38 (101%)	0.35 (101%)	**0.01** (0.07%)	0.35 (103%)
henon	0.73 (112%)	0.63 (83%)	**0.24** (13%)	0.57 (67%)

A partir dos dados presentes nesta Tabela pode-se concluir que:

- O método de *Alisamento Exponencial* (coluna *AE*) apresenta um melhor desempenho para as séries sazonais e com tendência (séries **passageiros** e **papel**). Tal facto não é surpreendente, dado que este método foi especificamente desenvolvido para lidar com este tipo de séries. O sucesso do *AE* é devido a um ajustamento sucessivo das estimativas sazonais e de tendência, não dependendo de pesos fixos. Por outras palavras, o *AE* utiliza uma memória de longo prazo, bem superior à fornecida pela janela temporal que é aplicada nos outros métodos. Também convém realçar que o *Meta-AGE* consegue superar, em termos de desempenho, a metodologia de *Box-Jenkins* (coluna *BJ*), mesmo apesar desta utilizar uma transformação logarítmica para a série **passageiros**, que lhe é favorável.

- Contudo, e de forma algo inesperada, a vantagem do *AE* já não é tão visível para as séries sazonais mensais (séries **mortes** e **melborne**). De facto, o modelo de optimização neuronal chega a apresentar melhores resultados para a série **mortes**, indicando que a memória de longo prazo não será tão crucial, quando se retira a componente de tendência às séries sazonais.

- Para as séries com forte tendência (séries **química** e **preços**), a metodologia de *Box-Jenkins (BJ)* apresenta os piores resultados, com o método de *AE* e o sistema de *RNEs* com erros de previsão em tudo semelhantes, e com o *Meta-AGE* a apresentar o melhor desempenho.

- Como esperado, o modelo de *AE* não está apto para lidar com séries não lineares (séries **manchas** e **kobe**), produzindo os piores resultados. A metodologia de *BJ* consegue melhores previsões, mas é suplantada, por uma margem considerável, por ambos os métodos inspirados na Natureza, que têm um desempenho similar.

- O modelo *ARMA* torna-se ineficaz na previsão das séries caóticas (série **quadrática** e **henon**), pelo que nem a metodologia de *BJ*, nem o sistema de *Meta-AGE,* conseguem previsões melhores do que as obtidas através da leitura do valor médio da série (conforme indicado pelo valor de $MNQE_{pr}$, que está próximo dos 100%). No entanto, o sistema de optimização neuronal capta perfeitamente estes efeitos não lineares, produzindo previsões de rara qualidade.

4.4.2 Tempos de Computação

Até aqui, apenas se considerou o desempenho dos modelos de previsão, medido em termos da sua eficácia, através do recurso a métricas de erro. Sendo assim, o que se pode afirmar em termos de esforço computacional? Pela descrição dos modelos inspirados na Natureza, parece ser óbvio que exigem um poder de computação significativamente superior.

Para se ter uma ideia da quantificação dos custos de computação de cada modelo, será utilizada como exemplo a série mensal **passageiros**, com 144 elementos, dos quais os últimos 14 são utilizados para a previsão. A Tabela 4.6 descreve os tempos de computação, em segundos, calculados pela aplicação **tsf**, para um processador *Intel Pentium III* a 450MHz.

A operação de **previsão** consiste em estimar o valor de catorze (14) operações de cálculo (casos de teste), sendo por isso uma operação deveras rápida, seja qual for o modelo considerado.

Em seguida, surgem os tempos de **aprendizagem**, aplicáveis apenas aos modelos inspirados na Natureza, definida como o ajustamento dos parâmetros de modelos fixos. Foram utilizados os modelos mais complexos para a janela temporal **A**=<1,2,...,13>, definidos pela topologia completamente interligada 13-13-1 para o modelo neuronal e o pelo modelo **G2** para o *AGE*.

Os valores obtidos confirmam que a aprendizagem evolucionária é mais exigente em termos computacionais, requerendo aproximadamente o dobro do tempo, quando comparado com o treino *RPROP*. No entanto, convém realçar que a aprendizagem neuronal utiliza um critério extra de paragem, a falta de progresso no treino, que é activado na maior parte das situações.

Por último, surge a operação de **optimização**, que consiste na procura do modelo óptimo de previsão. Não é apresentado o valor para a metodologia de *BJ*, dado que o modelo foi optimizado através da aplicação comercial *Forecast Pro*, para o sistema operativo *Windows*. O tempo descrito para o método de *AE* corresponde ao uso de uma procura em grelha de 0.01. Para os métodos inspirados na Natureza, foram escolhidos os tempos de obtenção do melhor modelo, correspondendo à geração 193 do sistema de *RNEs* e geração 45 do *Meta-AGE* (Tabela 4.6).

É neste tipo de operação que as diferenças de computação entre os modelos se acentuam. De facto, o sistema de *RNEs* tem um custo computacional de cerca de 36 vezes superior ao exigido pelo método de *AE*. Como seria de esperar, o sistema de *Meta-AGE* apresenta os maiores requisitos computacionais, aproximadamente 22 vezes superior à exigida pelo sistema de *RNEs*. Trata-se de um tipo de comportamento que é comum na resolução de problemas *NP-completos*.

Tabela 4.6: Custo computacional de cada modelo para a série **passageiros**.

Operação	**AE**	**BJ**	**RNU**	**AGE**
previsão	$\ll 0$	$\ll 0$	$\ll 0$	$\ll 0$
aprendizagem	-	-	20	47
optimização	324	-	17184	375037

4.5 Discussão e Conclusões

Este capítulo centrou-se na automatização do processo de procura, dos dois factores cruciais para a previsão dos modelos de previsão inspirados na Natureza: a janela temporal e complexidade do modelo.

São apresentadas as principais estratégias de procura, como a *procura em feixe*, que mantém uma população de pontos de procura. Os *AGEs* são um produto típico desta estratégia, sendo considerados devido aos algoritmos de procura que implementam, o que lhes

permite localizar rapidamente soluções de elevada qualidade, estando assim vocacionados para problemas de natureza combinatória.

A combinação de *AGEs* e *RNAs* é motivada pelos processos de selecção próprios dos seres vivos, onde a *evolução* ocorre ao nível das populações, determinando as estruturas básicas dos organismos, sendo a aprendizagem responsável pela adaptação de um indivíduo ao ambiente que o rodeia.

De facto, a combinação de ambos os paradigmas, conhecida pelos sistemas de *Redes Neuronais Evolucionárias (RNEs)*, tem ganho uma atenção crescente pela comunidade científica, em particular para o desenho de topologias de *RNAs*. Neste domínio, e em termos de representação do conhecimento disponível ao nível do universo de discurso, posicionam-se duas visões com vista à solução do mesmo problema: uma *forte*, onde é utilizada uma codificação directa para os nodos e conexões da rede, outra *fraca*, onde são definidas regras para a construção da rede.

Foi efectuado um breve levantamento do estado da arte, sobre a aplicação de *RNEs* à previsão, sendo proposto um modelo de codificação directa diferente do usual, por permitir definir uma qualquer topologia de rede e janela temporal. O sistema de *RNEs* proposto foi aplicado às séries temporais objecto de estudo, sendo comparado com o sistema de *tentativa-e-erro* descrito no capítulo anterior. Desta comparação conclui-se que a optimização neuronal apresenta um melhor desempenho, seleccionando modelos ajustados à complexidade da série temporal.

Os bons resultados obtidos indiciam que também devem existir benefícios numa optimização do modelo evolucionário. Os *Meta-AGEs* utilizam *AGEs* num nível meta, para a optimização dos parâmetros de *AGEs* do nível inferior. Esta combinação beneficia de uma exploração automática, não tendenciosa, de um número elevado de parâmetros. Assim, foi proposto um sistema de *Meta-AGE* para a procura dos modelos evolucionários considerados óptimos, que deu origem a melhores modelos de previsão.

Finalmente, foi efectuada uma comparação entre os modelos convencionais e os inspirados na Natureza. Os primeiros, apenas revelaram melhores resultados na previsão de séries sazonais, através do método de *AE*. Apesar de utilizar o mesmo modelo $ARMA$ subjacente, o sistema *Meta-AGE* obteve um desempenho superior à metodologia BJ em todas as séries, revelando excelentes previsões para as séries com tendência e não lineares.

Por sua vez, e de forma algo surpreendente, o sistema de optimização neuronal obteve bons resultados na previsão das séries sazonais sem tendência, demonstrando também um excelente desempenho na previsão das séries não lineares, especialmente para as séries com comportamentos caótico, sendo de realçar que para estas últimas todos os outros modelos

se revelaram ineficazes.

Todavia, o bom desempenho obtidos pelos modelos inspirados na Natureza tem um senão, que se manifesta no seu acrescido custo computacional.

Capítulo 5

Previsão em Tempo Real

> "There is nothing remarkable about it. All one has to do is hit the right keys at the right time and the instrument plays itself."
>
> –Johann Sebastian Bach.

São definidos os diferentes horizontes temporais para o processo de previsão, incluindo o caso particular da previsão em tempo real. É apresentada uma arquitectura para sistemas de previsão em tempo real, sendo efectuado um conjunto de experiências com os diferentes modelos de aprendizagem.

Como foi demonstrado anteriormente, os modelos de previsão inspirados em fenómenos naturais assumem-se como uma alternativa eficaz aos métodos convencionais de previsão, principalmente quando uma dada série temporal apresenta componentes não lineares.

Contudo, este desempenho é obtido à custa de um aumento da complexidade dos modelos de previsão, ou seja, existe um maior número de parâmetros que necessitam de ser ajustados. Esta complexidade reflecte-se num aumento do custo computacional. Para um vulgar computador pessoal, o processo de aprendizagem exige um poder de computação na ordem das dezenas de segundos, enquanto que o processo de optimização contém requisitos ainda superiores, na ordem dos milhares de segundos.

Embora este tempo de computação não seja um factor relevante, na presença de períodos temporais longos (e.g., trimestrais ou mensais), o mesmo já não sucede quando as sequências temporais são curtas (e.g., minutos ou segundos). Assim sendo, será que é possível aplicar os modelos inspirados na Natureza para a previsão de séries em tempo real?

Este capítulo procurará responder a esta questão, através da proposta de uma arquitectura para a previsão em tempo real, que permite adoptar modelos inspirados na Natureza a estas situações.

5.1 Horizontes Temporais

De uma forma geral, em sistemas de gestão e controlo da produção, a previsão potencia uma melhoria dos processos de controlo. O período de tempo durante o qual se pretende que prevaleça uma dada tomada de decisão, designado por *horizonte temporal*, afecta naturalmente o processo de selecção do método de previsão.

As previsões podem ser classificadas em quatro categorias principais, dependendo da relação existente entre o período temporal do sistema que lhe está subjacente e da escala de tempo considerada para a previsão. Assim, e para um dado contexto (i.e, situação em que se desenvolve o problema em equação), as seguintes categorias de previsão devem ser consideradas, tendo em conta diferentes horizontes temporais [Ding et al., 1995]:

- *Previsões de Longo Prazo*, correspondendo, por exemplo, a evoluções anuais de aplicações financeiras, necessárias para o planeamento de investimentos;

- *Previsões de Médio Prazo,* tipicamente efectuadas para períodos mensais, de modo a auxiliar o planeamento de recursos;

- *Previsões de Curto Prazo,* normalmente associadas a sequências temporais que variam entre dias ou horas. Tratam-se de previsões essenciais para um controlo óptimo, para decisões em situações críticas ou para a detecção de situações anormais; e

- *Previsão em Tempo Real,* que lida com amostras temporais que, em média, não excedem o minuto. Nesta situação, o sistema de previsão deve operar de forma automática, sendo a segurança um factor crítico.

Convém referir ainda que embora esta classificação se mantenha válida para uma diferente gama de ambientes, o mesmo poderá não suceder com respeito aos períodos temporais. Por exemplo, um horizonte temporal de três horas em avanço é considerado como de médio prazo, na previsão de precipitação pluviométrica, através da análise de imagens de radar [Denoeux & Rizand, 1995].

5.2 Séries Temporais em Tempo Real

Para avaliar cada um dos modelos de previsão, foram escolhidas duas séries temporais com uma dada cadência temporal (Figura 5.1), oriundas das Ciências da *Saúde e Geologia,* e que no que se segue se apresentam:

kobe-r Trata-se de uma extensão à série **kobe** apresentada no Capítulo 3, agora com um total de 1200 elementos [Hyndman, 2001]. A série diz respeito às leituras de um sismógrafo, em termos da sua aceleração vertical, durante um terramoto na cidade japonesa de Kobe. As observações foram registadas pela universidade da Tasmânia, Austrália, em 16 de Janeiro 1995, com um período regular de um segundo. Quando se analisa a série e quando se parte para a sua decomposição, vê-se que esta apresenta os mesmos componentes da série **kobe**, ou seja, trata-se de uma série estacionária, não linear.

coração Esta série contém uma forte tendência, referindo-se a 2400 registos de batimento cardíaco, obtido durante o sono de um paciente no hospital de Boston, Massachusetts [Weigend & Gershenfeld, 1994]. As observações seguem-se a cada 0.5 segundos, a partir da leitura dos dados constantes de um electrocardiograma.

De referir que ambas estas séries apresentam oscilações ao longo do tempo. No primeiro caso, devido ao surgimento do terramoto e, no segundo caso, devido aos estados de apneia[1] do paciente.

Por exemplo, a Figura 5.2 mostra as autocorrelações para dois períodos de observação, diferentes, da série **kobe-r**, ambos com 100 observações. As diferenças encontradas espelham as alterações que ocorreram no comportamento das séries.

5.3 Sistema de Previsão em Tempo Real

A previsão em tempo real assume-se como crucial, em particular, no que diz respeito a sistemas de controlo. Em boa verdade podem-se referir os mais variados exemplos que vão desde a gestão em tempo real dos caudais dos cursos de água [Mathias et al., 1998], da poluição em termos dos esgotos urbanos [Gong & Denceux, 1996], da gestão e controlo da produção [Chatfield, 1989] ou do fornecimento de energia eléctrica [Charytoniuk & Chen, 2000].

[1] Suspensão temporária da respiração.

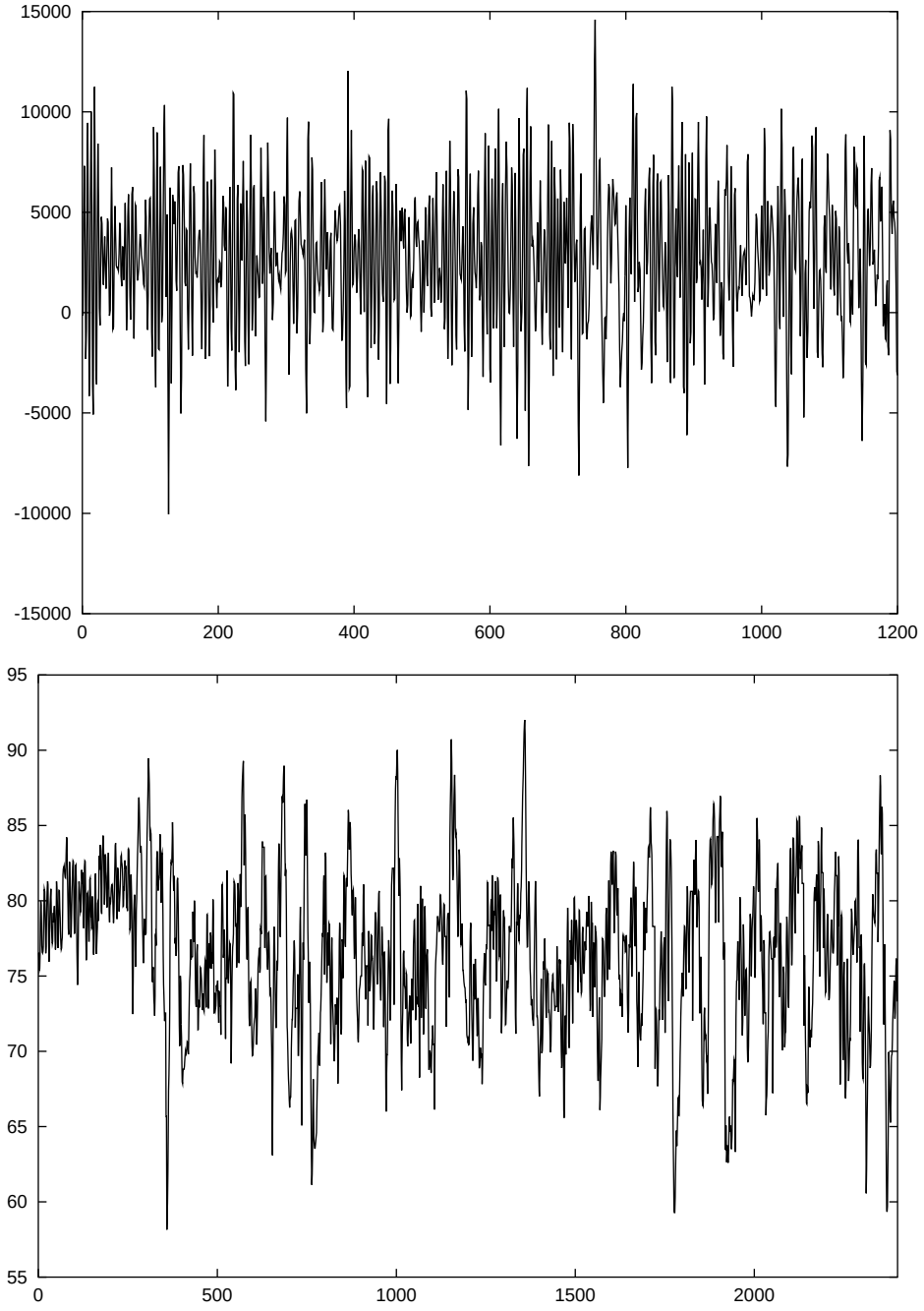


Figura 5.1: Séries temporais em tempo real (**kobe-r** e **coração**).

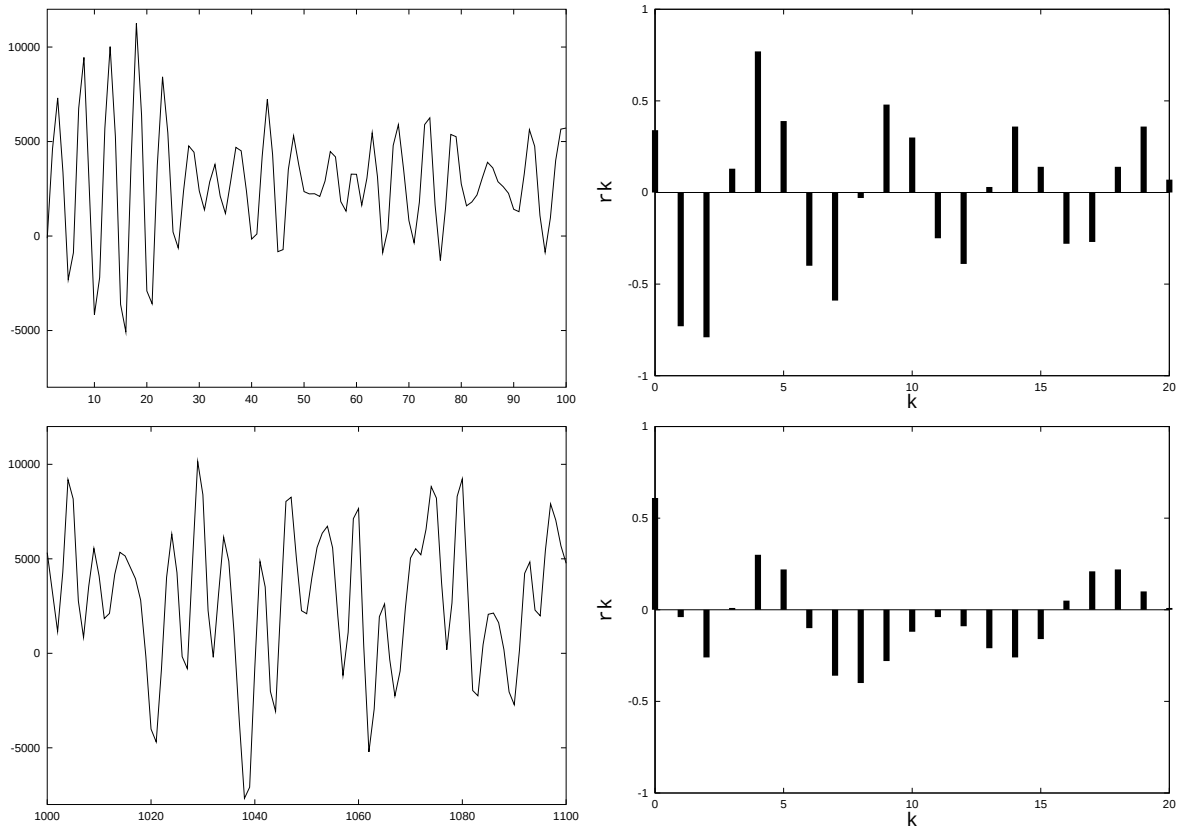


Figura 5.2: Autocorrelações para diferentes períodos da série **kobe-r**.

Para a previsão em tempo real podem-se utilizar os métodos de previsão de curto prazo, quer sejam convencionais (e.g., *AE* ou a metodologia de *BJ*) ou os inspirados na Natureza. Contudo, existe uma vantagem no uso de métodos que consigam extrair informação a partir dos dados, sem qualquer conhecimento à priori. Daí que diversos estudos privilegiem o uso de *RNAs* [Ding et al., 1995; Charytoniuk & Chen, 2000; Barnett, 2001], defendendo que estas conseguem modelar relações não lineares, mesmo em situações dinâmicas, demasiado complexas para serem descritos por métodos analíticos.

A diferença entre a previsão em tempo real e a de curto prazo assenta em dois factores principais:

- o sistema de previsão tem de ser automatizado, de modo a que este consiga dar uma resposta em tempo real; e
- o modelo de previsão tem de ser estimado em tempo real, mesmo que tal implique uma perda no seu desempenho.

O primeiro requisito implica o desenvolvimento de uma arquitectura para a previsão em tempo real, enquanto o segundo define um limite temporal, durante o qual o modelo computacional considerado para a previsão deve ser ajustado. Ambos estes requisitos irão ser tratados em detalhe nas secções seguintes.

5.3.1 Arquitectura de Previsão em Tempo Real

Um sistema para a previsão em tempo real deverá envolver quatro etapas: a *aquisição de dados históricos*, a *validação e/ou redução dos dados*, a *construção do modelo de previs*ão, e a *extrapolação a partir desse modelo* [Hanke & Reitsch, 1989].

A primeira etapa exige um particular cuidado na leitura dos dados do problema em equação, de forma a se evitarem distorções. A etapa seguinte, *valida*ção, consiste em determinar os dados relevantes para a resolução do problema em equação. Posteriormente, vem a *construção do modelo de previs*ão, que envolve um ajustamento dos dados a um modelo, de forma a se minimizar o erro. Uma vez definido o modelo, segue-se a última etapa, extrapolar para períodos futuros.

Com base neste processo, é proposta uma arquitectura a usar em sistemas de previsão em tempo real que envolve dois componentes funcionais da maior importância (Figura 5.3): o módulo de *pré-processamento* e o módulo de *previsão*.

O módulo de pré-processamento contempla um sistema de *aquisição* de dados e um sistema de *validação* dos mesmos. A aquisição de dados poderá ser materializada através de,

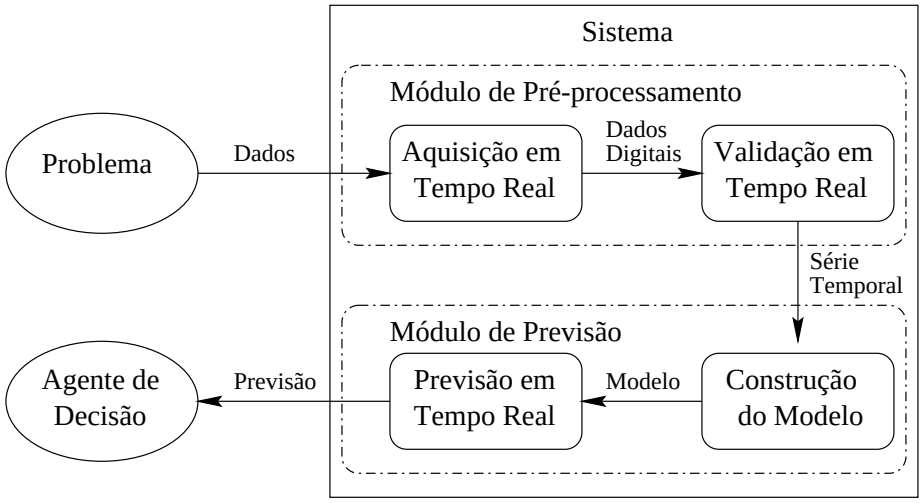


Figura 5.3: Arquitectura de previsão em tempo real.

por exemplo, um sensor ou um conversor analógico-digital, tendo por objectivo a captura dos dados reais do problema. Em seguida, os dados gerados são processados pelo sistema de validação, cujo objectivo é filtrar e armazenar a informação relevante, produzindo uma série temporal com um determinado período regular (e.g., um sistema de amostragem ou de filtragem de ruído).

Neste trabalho, este módulo não será considerado, por estar intimamente dependente do problema a ser solucionado (e.g., a adopção de componentes específicos de *hardware,* como um sismógrafo), partindo-se do princípio que a aquisição e validação dos dados foi realizada de forma correcta.

No que se segue, abordar-se-ão as duas últimas etapas do processo: a construção do modelo de previsão, que envolve a definição e aprendizagem deste em tempo real, e a previsão a partir do mesmo.

Por último, os valores previstos são enviados a um *agente de decisão*, que poderá ser um sistema de controlo ou sistema inteligente, que activará, caso seja necessário, um conjunto de acções relevantes para a resolução do problema em equação.

5.3.2 Módulo de Previsão

No capítulo anterior, equacionou-se o processo de previsão em séries temporais a partir de um dado modelo, seja este do tipo convencional ou não, tendo ficado demonstrado que se trata de um processo expedito. Ora, este facto induz, no que concerne ao processo computacional de previsão, que se centre o maior esforço exclusivamente na fase de construção, e

não na sua operação.

Por outro lado, embora a optimização dos modelos inspirados na Natureza via *Algoritmos Genéticos* produza modelos para a previsão de séries temporais deveras eficientes (como foi provado anteriormente), trata-se de um processo demasiado lento (na ordem dos milhares de segundos) para que possa ser utilizado em todos os processos de previsão em tempo real. Por conseguinte, optar-se-á pela utilização de modelos pré-definidos. Esta decisão permite simplificar o processo, reduzindo os requisitos computacionais ao processo de aprendizagem dos modelos de previsão.

Todavia, mesmo quando se considera apenas o processo de aprendizagem dos modelos inspirados na Natureza, as exigências computacionais são da ordem das dezenas de segundos, o que poderá configurar um tempo demasiado longo, para certos tipos de séries temporais. Em geral, o processo de aprendizagem converge de uma forma rápida nas primeiras iterações mas, à medida que o tempo passa, mais lenta se torna a diminuição do erro.

Para exemplificar este fenómeno, foi feita uma simulação, utilizando um processador *Intel Pentium III* a 450MHz, do processo de aprendizagem para a série **passageiros**. Os últimos 10% dos valores desta série foram utilizados para a previsão, sendo a série modelada através de uma *RNU*, na forma 13-13-1. A Figura 5.4 dá-nos a evolução das duas curvas de erro, o $RMQE_{tr}$ e o $RMQE_{pr}$, para esta simulação, onde no eixo das abcissas tem-se o tempo, medido em segundos.

Esta experiência indica que o tempo de aprendizagem pode ser diminuído, sem que haja uma grande degradação em termos do desempenho do modelo de previsão (Figura 5.5). De facto, ao fim de 4 segundos, o valor do erro de previsão é de 21.7, um valor não muito distante do valor de 19.5, que é obtido aos 50 segundos.

Outro aspecto importante para a construção do modelo de previsão tem a ver com a evolução da própria série temporal. De facto, as séries com centenas ou milhares de observações, não raras vezes apresentam comportamentos dinâmicos, ou seja, existem padrões que se alteram ao longo do tempo devido a influências exteriores (como foi demonstrado na Secção 5.2 para a série **kobe-r**).

Daí que, quando se passa para a previsão em tempo real, poderá ser mais eficaz o uso dos acontecimentos mais recentes, através de uma janela temporal deslizante, ao invés de se considerar a totalidade da série. O objectivo que se persegue através da utilização desta janela temporal, doravante designada por *janela de visibilidade*, passa por definir quais são os dados acessíveis num dado momento para a aprendizagem de um modelo de previsão, sendo diferente da janela temporal referida nos capítulos anteriores, em que se definem

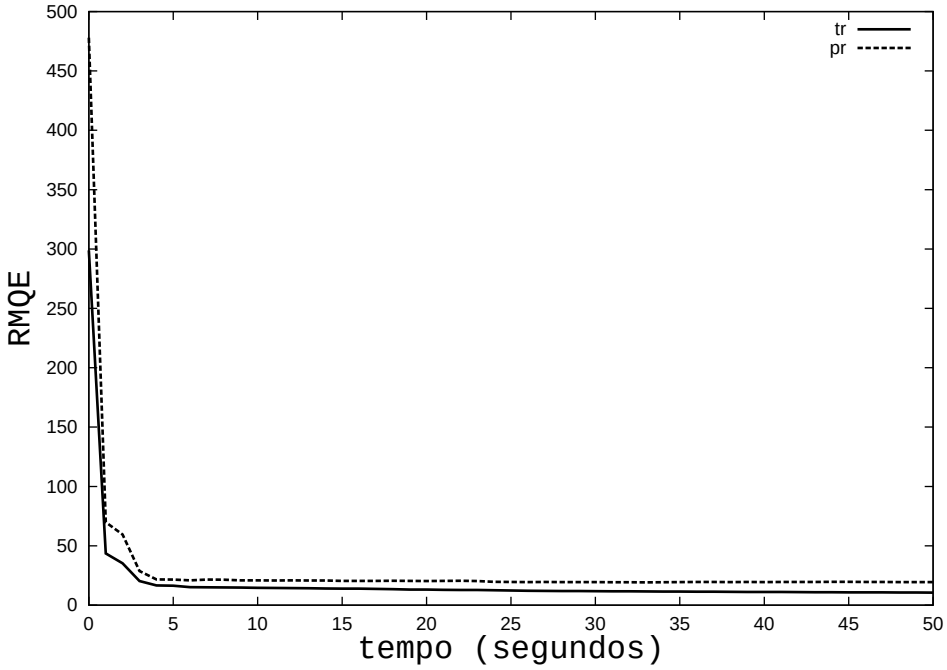


Figura 5.4: Progresso da aprendizagem para a série **passageiros**.

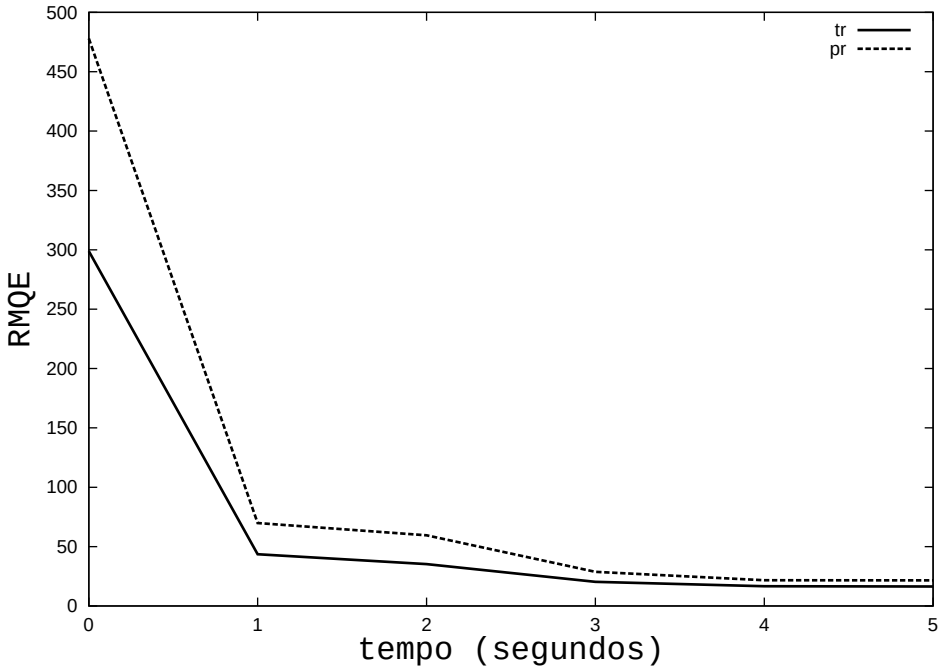


Figura 5.5: Instantes iniciais do processo de aprendizagem para a série **passageiros**.

quais são as entradas para o mesmo.

O uso de uma janela de visibilidade reveste-se de duas vantagens adicionais, que aceleram o processo de aprendizagem:

- reduz o número de casos de treino a ser aprendidos em cada instante; e
- permite que o modelo mantenha uma memória, pois em cada instante de tempo, a maioria dos casos de treino não irá sofrer alteração.

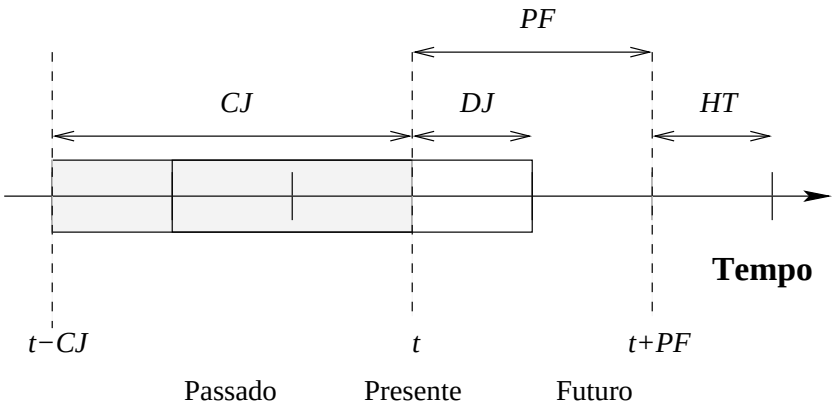


Figura 5.6: Processo dinâmico de previsão em tempo real.

Outro aspecto importante tem a ver com a capacidade de generalização, ou seja, a adopção de uma janela de visibilidade (que se desloca ao longo do tempo) impede que um dado modelo de previsão se fixe em demasia em determinados tipos padrões, evitando-se com isto o sobre-ajustamento.

Esta estratégia de aprendizagem é definida em termos dos factores: o *Horizonte Temporal (HT),* o número de *Previsões em Frente (PF)*, o *Deslocamento da Janela (DJ)* e o *Comprimento da Janela (CJ)* (Figura 5.6).

Os primeiros três factores são definidos pelos requisitos do problema, onde *HT* corresponde ao período regular da série, *PF* ao número de previsões desejadas e *DJ* ao comprimento do deslocamento da janela temporal. Por conseguinte, a afinação do processo depende do valor de *CJ*, que se for muito superior a zero, poderá dificultar a capacidade de aprendizagem do modelo de previsão, enquanto que um valor demasiado pequeno poderá fornecer a este insuficiente informação.

5.4 Experimentação

5.4.1 Configuração do Sistema

Para estudar o comportamento de cada um dos modelos de previsão em tempo real, foi acrescentado à aplicação **tsf** uma das valências descritas na secção anterior; i.e., foi adoptada a biblioteca **time.h** da linguagem de programação *C* [Kernigham & Ritchie, 1988], que inclui um conjunto de procedimentos para a manipulação da coordenada temporal. Em particular, a função **time** que devolve o tempo do sistema, medido em segundos (s).

Ambas as séries temporais referidas na Secção 5.2 foram consideradas, tendo em vista atender à previsão de curto prazo em tempo real. Assim, para a primeira série, **kobe-r**, têm-se os valores de $HT = 1s$, $PF = 1$ e $DJ = 1$, que reflectem a cadência natural do registo sismográfico.

Quanto à segunda série, esta apresenta uma cadência de registos demasiado rápida (na ordem dos $0.5s$), que é inferior ao menor dos tempos dado pela função **time** ($1s$). Daí que para o estudo do modelos de previsão se tenha optado por manter uma cadência equivalente, através dos valores: $HT = 1s$, $PJ = 2$ e $DJ = 2$. Estes valores fazem com que, em cada segundo, o modelo aprenda a partir de CJ elementos anteriores, efectuando em cada período duas previsões, dado que a cadência de previsão é de $0.5s$ e o período é de $1s$.

Foram escolhidos três modelos distintos para a previsão em tempo real, que denotam diferentes abordagens para a solução do mesmo problema:

- o modelo linear, fornecido pelo método de *AE*;
- o modelo *ARMA,* sendo os coeficientes estimados via *AGEs*; e
- o modelo conexionista não linear, suportado pelas *RNUs*.

O *AE* foi incluído para servir de comparação com os métodos não convencionais, sendo também a sua aplicação em tempo real de fácil implementação. Tratando-se de séries não sazonais, apenas se torna necessário estimar dois parâmetros α e β, que denotam, respectivamente, os coeficientes de alisamento e de tendência da série temporal, que serão obtidos a partir de uma procura em grelha.

Conforme explicado previamente, para os modelos inspirados na Natureza, torna-se necessário o uso de modelos pré-definidos. Estando-se na presença de séries não sazonais, foi decidido utilizar uma janela deslizante com uma cardinalidade máxima de 10 posições ($JTD =< 1, 2, ..., 10 >$), para ambos os modelos de previsão em consideração.

Para a *RNU*, foi escolhida a topologia completamente interligada de $10-3-1$, dado que as experiências anteriores demonstraram que bons resultados são obtidos com poucos nodos intermédios. No caso do *AGE*, optou-se pela escolha de um modelo que incluísse as mesmas características, adoptando-se assim um modelo $ARMA(10, 10)$. Por último, foi decidido manter os mesmos parâmetros de aprendizagem utilizados nos capítulos precedentes (e.g., algoritmo *RPROP* para a aprendizagem da *RNU*).

Convém referir ainda que a metodologia de *BJ* não foi abordada devido a duas razões principais. Em primeiro lugar, porque a sua utilização exige a presença de um perito ou sistema pericial (como o fornecido pelo programa *Forecast Pro*), sendo a sua automatização de difícil execução em tempo real (o tempo limite é $1s$). Em segundo lugar, porque a metodologia de *BJ* utiliza uma estimação por mínimos quadrados, que levou a piores resultados do que os obtidos via os *AGEs*. Contudo, o modelo base subjacente a esta metodologia, o *ARMA*, continua representado no modelo evolucionário alternativo.

Relativamente ao processo dinâmico, foi decidido ajustar a janela de visibilidade ao valor de $CJ = 100$, um valor razoável para permitir que o modelo consiga uma rápida aprendizagem com recurso a um conjunto de padrões de treino deveras significativo (e.g., para um processador Pentium III a 450MHz, este valor leva a que em cada segundo se executem aproximadamente 90 iterações do algoritmo *RPROP* e 35 do algoritmo evolucionário).

Finalmente, no que diz respeito ao registo dos valores de previsão, foi decidido contabilizar estes apenas ao fim de $200s$, para garantir que haja tempo suficiente para a afinação inicial de cada um dos modelos que está a ser objecto de estudo. Assim, o período da avaliação da previsão em tempo real perdura ao longo dos restantes $1000s$, num total de 1000 previsões para a série **kobe-r** e 2000 para a série **coração**.

5.4.2 Resultados

Os três modelos de previsão mencionados em epígrafe foram testados na previsão das duas séries em tempo real. Para cada modelo inspirado na Natureza, foram equacionados trinta (30) casos para estudo (i.e., simulações), para assegurar uma maior relevância estatística.

A título demonstrativo, uma atenção especial será devotada à série **kobe-r**, através da análise das últimas cinquenta (50) previsões em tempo real, considerando os resultados obtidos por meio de um processador *Pentium III* a 450MHz (foram utilizadas apenas as últimas cinquenta (50) previsões para facilitar a leitura dos dados).

Em primeiro lugar, serão exibidos os resultados para o método de *AE* (Figura 5.7). Da

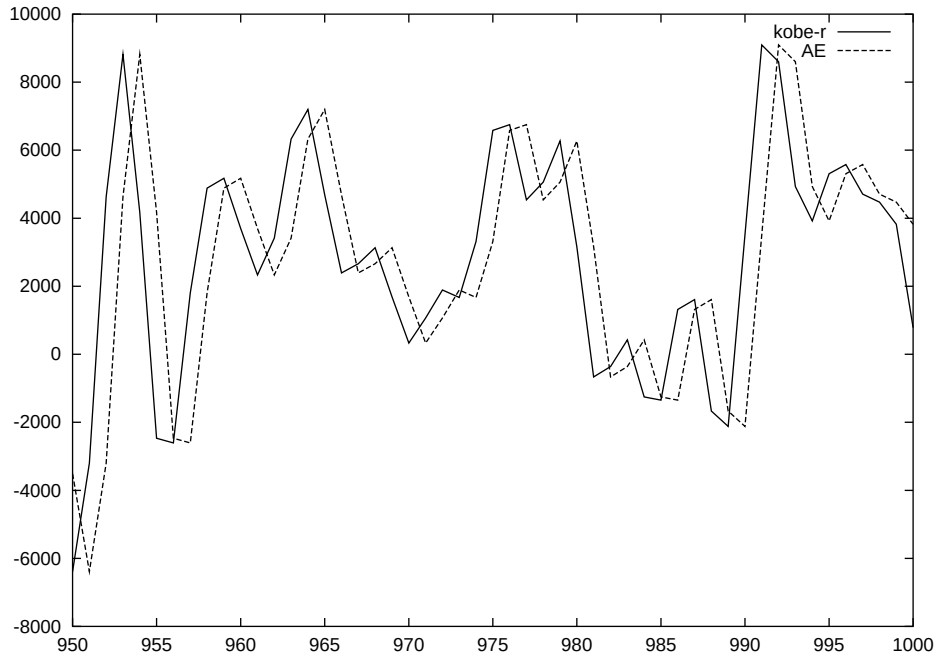


Figura 5.7: Valores reais e previsões via *AE* para a série **kobe-r**.

figura depreende-se que existe um nítido desfasamento entre ambas as curvas, ou seja, o método de *AE* utiliza o valor do instante passado para prever o valor seguinte: $\widehat{x_t} = x_{t-1}$. Ora, tal acontece porque se trata de um método linear que se torna nitidamente inadequado para previsão desta série.

Este tipo de comportamento já não ocorre com os modelos não convencionais, como pode ser verificado nas Figuras 5.8 e 5.9. Aqui, optou-se por escolher o valor médio das trinta (30) simulações, para a representação das previsões. Em ambas as situações, a curva de previsão encontra-se próxima dos valores reais da série, revelando uma boa adaptação por parte dos modelos de previsão em tempo real.

As diferenças entre os métodos de previsão são visualizadas mais facilmente na Figura 5.10, que descreve os erros de previsão absolutos, para cada modelo contemplado. De facto, a curva que descreve os erros obtidos via *AE* contém uma amplitude com um valor elevado, bem distante do eixo zero (que denota uma previsão perfeita). Em comparação, os modelos inspirados na Natureza apresentam valores de erro significativamente menores.

Para averiguar se os resultados dependem do poder computacional disponível, foi decidido realizar todo o conjunto de simulações em três computadores, que apenas diferem no tipo de processador, sendo o primeiro um Pentium II a 350 MHz, para além de dois

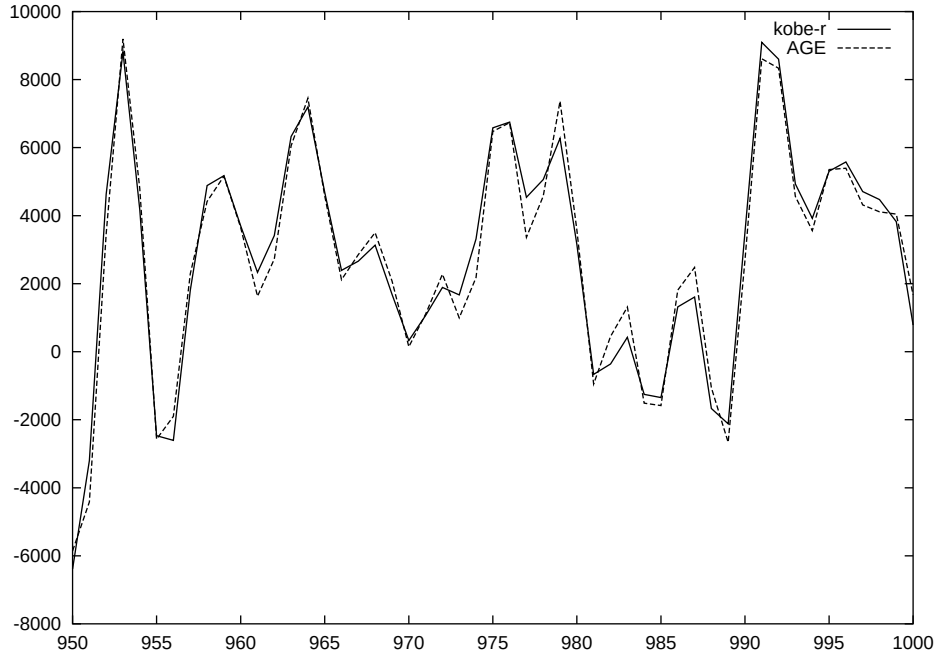


Figura 5.8: Valores reais e previsões via o *AGE* para a série **kobe-r**.

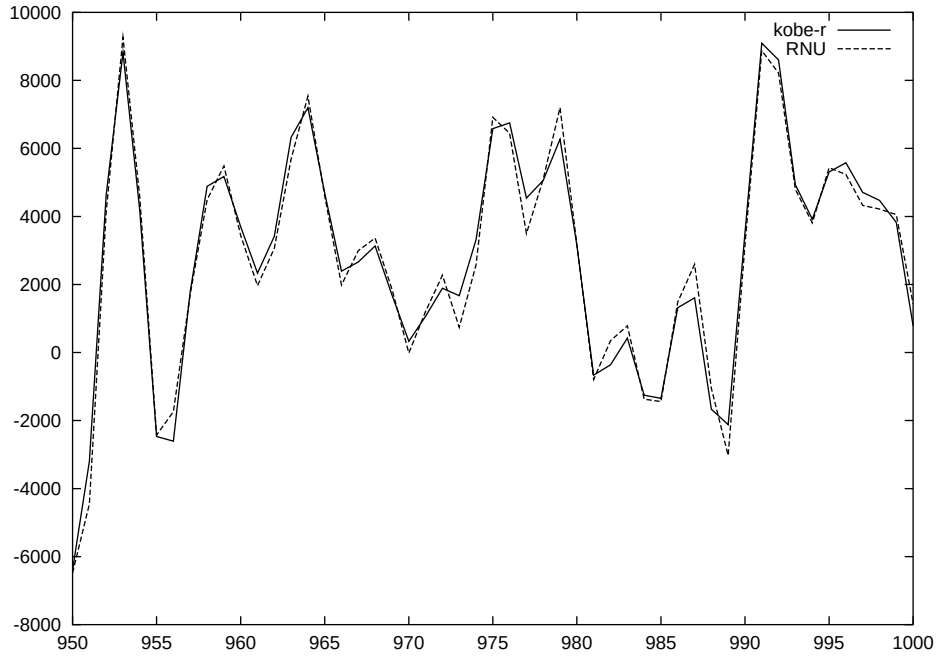


Figura 5.9: Valores reais e previsões via a *RNU* para a série **kobe-r**.

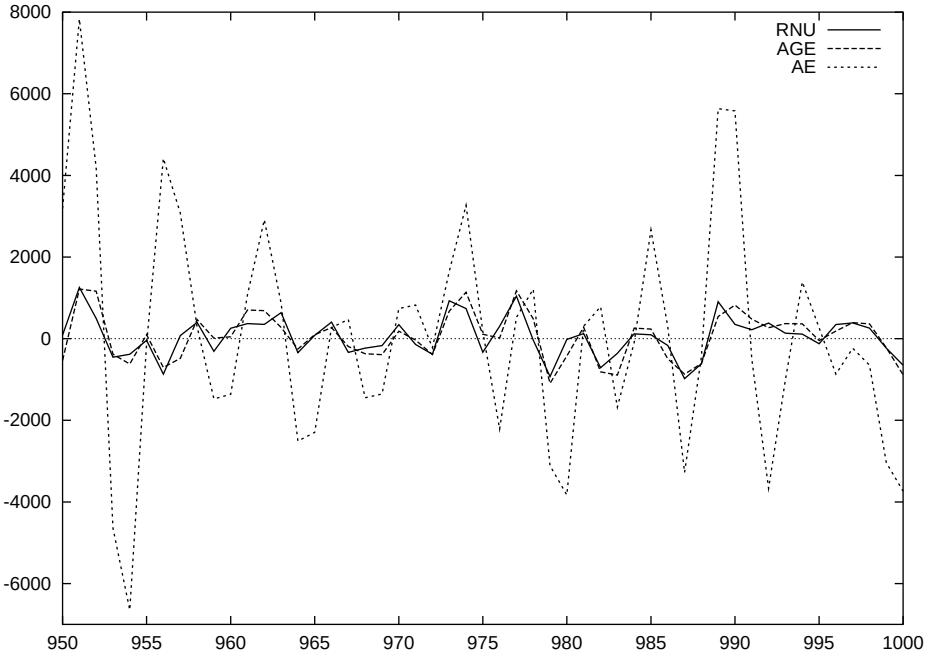


Figura 5.10: Valores de erro para as diferentes previsões em tempo real para a série **kobe-r**.

Pentium III, a 450 MHz e 933 MHz. (Tabela 5.1).

Tabela 5.1: Características dos diferentes computadores.

Máquina	**Processador**	**Frequência**	**Cache**
A	Pentium II	350Mhz	512KB
B	Pentium III	450Mhz	512KB
C	Pentium III	933Mhz	256KB

A Tabela 5.2 reporta o desempenho de cada uma das máquinas. A coluna *Bogomips* refere-se à métrica utilizada pelo sistema operativo *Linux*, que mede a velocidade com que certos ciclos são executados pelo processador. Contudo, esta métrica dá apenas uma indicação aproximada sobre a velocidade do processador. Assim, foi decidido adoptar os testes padrão de desempenho fornecidos pelo programa **nbench** [Mayer, 2001], e em particular as métricas:

- *Ordenação Numérica (ON)*, que mede a velocidade de processamento utilizando valores inteiros;

Tabela 5.2: Desempenho dos diferentes computadores.

Máquina	*Bogomips*	*ON*	*F*	*RN*
A	350	150	3875	3.83
B	900	198	4788	4.26
C	1860	414	8639	8.72

- *Fourier (F)*, que mede a velocidade de processamento utilizando valores reais; e
- *Rede Neuronal (RN)*, que simula um treino simples, utilizando o algoritmo de *RP*.

Estes indicadores confirmam que os desfazamentos, em termos de capacidade de computação, são sensivelmente equivalentes aos dados pelas diferenças de velocidade de processamento de cada processador. Tal significa que a máquina **C** é aproximadamente duas vezes mais rápida do que a máquina **B**, sendo esta cerca de 30% mais veloz do que a máquina **A**.

Os resultados das experiências efectuadas encontram-se resumidos na Tabela 5.3, onde se consideram os valores dos erros de previsão em tempo real, em termos do $RMQE_{pr}$ e o $MNQE_{pr}$ (em parêntesis). De referir ainda que, para os métodos inspirados na Natureza (colunas **AGE** e **RNU**), os valores denotam a média de trinta (30) simulações (e.g., treino de uma *RNU*).

Tabela 5.3: Comparação entre os diferentes modelos de previsão em tempo real.

Série	**Máquina**	**AE**	**AGE**	**RNU**
	A	3269 (93%)	736 (5.0%)	590(3.0%)
kobe-r	**B**	3278 (93%)	696 (4.3%)	572 (2.8%)
	C	3286 (93%)	**652** (3.7%)	**547** (2.6%)
	A	1.93 (13.7%)	1.91 (13.1%)	1.68 (10.4%)
coração	**B**	1.93 (13.7%)	1.87 (13.0%)	1.68 (10.3%)
	C	1.93 (13.7%)	**1.73** (11.0%)	**1.66** (10.1%)

Em termos da estimação dos parâmetros do *AE*, foram utilizadas três configurações para a procura em grelha, onde a discretização é feita através de amostras com um passo de:

- 0.1 para a máquina **A**;

- 0.05 para a máquina **B**; e
- 0.01 para a máquina **C**.

o que cria um espaço de procura com 100, 400 e 10000 soluções distintas, respectivamente. Contudo, os resultados demonstram que o aumento do espaço de procura não se traduz em diferenças ao nível dos erros de previsão (os valores para o $MNQE_{pr}$ mantêm-se).

Pela análise dos dados apresentados na Tabela 5.3 comprova-se que o poder de computação disponível influencia a qualidade dos resultados obtidos, com a máquina mais veloz (**C**) a produzir os melhores resultados. No entanto, a variação do desempenho é reduzida para o caso das *RNUs* (entre 0.4% e 0.3%, tal que $0.4 = 3.0 - 2.6$ e $0.3 = 10.4 - 10.1$). Neste caso, o aumento obtido em termos de eficácia de previsão não corresponde ao aumento de poder computacional. Por exemplo, para a série **kobe-r**, existe uma melhoria do erro de previsão de 0.2% quer quando se muda da máquina **A** para a **B**, quer quando se muda desta para a **C**. Contudo, no primeiro caso o aumento do poder computacional é de 30%, enquanto que no segundo caso é de 100%.

Os resultados obtidos estão de acordo com os dados na Figura 5.4, onde a progressão da aprendizagem é bastante lenta, após os instantes iniciais. Contudo, este cenário não ocorre com os *AGEs*, que apresentam variações superiores (1.3% e 2.1%, tal que $1.3 = 5.0 - 3.7$ e $2.1 = 13.1 - 11.0$), o que poderá ser explicado por uma aprendizagem mais lenta neste caso.

Comparando os resultados obtidos pelos diferentes modelos de previsão em tempo real, têm-se que:

- o método de *AE* é inadequado para previsão da série não linear **kobe-r**, com um valor de *MNQE* próximo dos 100% (como seria de se esperar);
- para a segunda série (**coração**), o método de *AE* apresenta uma melhoria significativa (explicada devido à forte componente de tendência da série), continuando, no entanto, a ser ultrapassado pelos modelos inspirados na Natureza (excepto quando comparado com o *AGE* da máquina **A**); e
- as *RNUs* apresentam o melhor desempenho na previsão em tempo real de ambas as séries, embora para a máquina **C**, os modelos evolucionários apresentem valores que lhes estão próximos, sendo bem melhores dos que os obtidos pelo método de *AE*.

As diferenças de desempenho dos modelos inspirados na Natureza poderão ser, em parte, explicadas pelas diferenças de aprendizagem destes (a aprendizagem dos *AGEs* é mais

lenta). No entanto, também podem indicar que as *RNUs* têm uma melhor capacidade de adaptação ao meio, conforme referido na Secção 1.3.2.

5.5 Discussão e Conclusões

O *horizonte temporal* é definido pelo período de tempo, durante o qual tem de ser tomada uma decisão, afectando a escolha do método de previsão. Por conseguinte, dentro de uma organização pode existir a necessidade de diversos tipos de previsão, tendo em conta diferentes horizontes temporais.

Em particular, tem-se a *previsão em tempo real*, que se recorre de amostras temporais curtas (e.g. em segundos ou minutos). Trata-se de um tipo de previsão deveras útil em sistemas de controlo, onde determinadas acções de correcção dos parâmetros do sistema têm que ser efectuadas, quando há ruído.

O objectivo deste capítulo foi de abordar a previsão em tempo real via modelos inspirados na Natureza. Para tal, foi feita a proposta de uma arquitectura de previsão em tempo real, que engloba dois módulos, o de *pré-processamento* e o de *previsão*. Apenas este último foi considerado neste trabalho, que envolve as etapas de construção do modelo e sua posterior previsão.

A construção dos modelos inspirados na Natureza foi simplificada através da utilização de estruturas fixas, centrando-se o problema apenas no processo de aprendizagem. Para que esta seja obtida dentro dos requisitos temporais, foi decidido adoptar de uma janela temporal deslizante, chamada de *janela de visibilidade*, que reduz o número de dados de treino disponíveis em cada instante.

Foram efectuadas experiências com duas séries, oriundas de diferentes universos de discurso, para a previsão em tempo real, sendo utilizadas três máquinas com capacidades de computação distintas. Para cada uma das séries, foram definidos três modelos de previsão, baseados no método de *AE*, na optimização do modelo *ARMA* via *AGEs*, e nos modelos não lineares fornecidos pelas *RNUs*.

Os resultados obtidos revelam que para os modelos inspirados na Natureza ocorre uma melhoria na eficácia de previsão, quando se aumenta o poder computacional disponível, embora esta não seja proporcional a este. Quando se comparam os diferentes modelos de previsão, verifica-se que o método de *AE* é ineficaz, principalmente para a série **kobe-r**, que é não linear. O melhor desempenho vai para as *RNUs*, embora os resultados obtidos pelos *AGEs* estejam próximos, no caso da máquina mais rápida.

Capítulo 6

Conclusões e Trabalho Futuro

> "All science is concerned with the relationship of cause and effect. Each scientific discovery increases man's ability to predict the consequences of his actions and thus his ability to control future events."
>
> – Laurence J. Peter

> *Realiza-se o fecho do trabalho desenvolvido, mediante um processo de análise e discussão dos resultados obtidos, lançando-se ainda as bases para trabalho futuro.*

Após a conclusão do trabalho desenvolvido, importa agora realizar o fecho do mesmo. Este será efectuado em termos de uma breve sinopse, discutindo os resultados atingidos e perspectivando trabalho futuro.

6.1 Síntese

O processo de *previsão* envolve o estudo de dados históricos, com o intuito da descoberta dos padrões subjacentes, de modo a utilizar este conhecimento para projectar os dados actuais para o futuro.

Este trabalho centrou-se numa forma usual de previsão, designada de *Previsão de Séries Temporais (PST)*, em que se recorre à análise da sequência dos valores passados de uma dada variável. A *PST* almeja somente a previsão do comportamento de sistemas e não a compreensão do seu funcionamento. Tal acontece porque, não raras vezes, é-se confrontado

com sistemas complexos impossíveis de especificar, onde se torna extremamente difícil distinguir as relações de causa-efeito.

Dada a sua importância, diversos métodos, aqui designados por convencionais, foram desenvolvidos para a *PST*. Em particular, destacam-se os seguintes:

- o *Alisamento Exponencial (AE)*, que se baseia em determinados componentes da série como a tendência e a sazonalidade; e
- a metodologia de *Box-Jenkins (BJ)*, que se fundamenta em modelos $ARMA$, sendo bastante precisa sobre um amplo leque de séries temporais.

Em alternativa aos métodos convencionais utilizados na *PST*, há que mencionar, entre outros, certos modelos que modulam a evolução dos seres vivos, desenvolvidos no seio da *Inteligência Artificial (IA)*. De facto, o estudo de processos biológicos como o sistema nervoso central ou a evolução natural, contribuiu para a afirmação de ferramentas a utilizar no processo de resolução de problemas como as *Redes Neuronais Artificiais (RNAs)* e a *Computação Genética e Evolucionária (CGE)*. Neste trabalho, optou-se por considerar duas técnicas populares, que personificam cada um destes movimentos, nomeadamente:

- as *Redes Neuronais Unidireccionais (RNUs)*, modelo conexionista organizado por camadas, com *conexões* que se desenvolvem em apenas um sentido; e
- os *Algoritmos Genéticos e Evolucionários (AGEs),* uma variante da *CGE* que parte de uma população de soluções para um problema em estudo, que se vão alterando ou evoluindo por recurso a operadores genéticos, como os de *cruzamento* e *mutação*.

Ambos estes modelos, descritos em pormenor no Capítulo 2, apresentam um conjunto de propriedades úteis para a tarefa de previsão. As *RNUs* seguem uma aprendizagem não linear, sendo imunes ao ruído, para além de se adaptarem facilmente a mudanças do ambiente. Por outro lado, os *AGEs* utilizam uma procura global multi-ponto, o que lhes permite localizar rapidamente soluções de elevada qualidade, evitando ficar reféns de mínimos locais.

No Capítulo 3 foram propostos estes dois modelos (alternativos) de previsão, na forma de:

- *RNUs* com uma *camada intermédia*, ligações de *atalho* e *bias*; e
- *AGEs* com uma *Representação dos dados segundo Valores Reais (RVRs)*, a utilizar na estimação dos coeficientes de modelos $ARMA$.

Por outro lado, verificou-se que a aplicação destes modelos à *PST* depende de dois elementos fundamentais, a saber:

- a *janela temporal deslizante*, que define o conjunto de entradas do modelo; e
- a *complexidade do modelo*, que define a sua estrutura.

Mais ainda, demonstrou-se empiricamente que a estatística BIC, que penaliza a complexidade dos modelos de *PST*, é um bom critério para a selecção dos modelos propostos.

Em seguida, procedeu-se à automatização do processo de desenho dos modelos a considerar na *PST*, utilizando um AGE binário como motor de procura de soluções (Capítulo 4), tendo-se recorrido a:

- *Redes Neuronais Evolucionárias (RNEs),* que assentam na evolução de topologias de *RNUs* por meio de um AGE; e
- *Meta-Algoritmos Genéticos e Evolucionários (Meta-AGEs),* onde um AGE binário de alto nível optimiza modelos $ARMA$, sendo os coeficientes destes posteriormente estimados via *AGEs* de baixo nível, que utilizam uma representação de base 10 (i.e., decimal).

Em quaisquer das situações foram obtidos modelos para a *PST* mais ajustados à complexidade da série, o que se traduziu numa melhor eficácia de previsão.

Por último, foi considerada a *previsão em tempo real*, que se associa a sequências temporais curtas, na ordem dos segundos ou minutos, e que é não raras vezes utilizada em sistemas de controlo (e.g., gestão do fornecimento de energia eléctrica [Charytoniuk & Chen, 2000]). Por conseguinte, foi definida uma arquitectura de previsão em tempo real, em que pontificam como componentes principais (Capítulo 5):

- o *pré-processamento*, que envolve etapas como a *aquisição* e a *validação* de dados; e
- a *previsão em tempo real*, através da *construção de um modelo para a PST*, e sua posterior utilização.

Relativamente à construção do modelo, os requisitos temporais foram respeitados pela adopção de modelos de *PST* pré-definidos e de uma janela de visibilidade dos dados, que diminui a quantidade de dados a serem considerados. As experiências revelaram que esta estrutura possibilita a aplicação dos modelos de evolução próprios dos seres vivos à previsão de séries com cadências temporais curtas.

6.2 Discussão

À medida que o mundo dos negócios se tornou cada vez mais complexo, a necessidade de uma previsão fundamentada em bases racionais aumentou, e a previsão passou a assumir uma posição proeminente no processo da gestão das organizações. De facto, todas organizações operam numa atmosfera de incerteza e, mesmo nesta situação, decisões que afectam o futuro das organizações têm de ser tomadas [Makridakis & Wheelwright, 1989].

Por isso, nas últimas décadas uma certa ênfase tem sido colocada no melhoramento dos processos de tomada de decisão, de forma a substituir-se a gestão baseada em palpites ou intuição. Em particular, três factores contribuíram de forma decisiva a ênfase que se coloca nos meios de previsão, a qual pode ser colocada na forma [Hanke & Reitsch, 1989]:

- o *crescimento da indústria dos semi-condutores*, permitindo disponibilizar artefactos com um elevado poder computacional e a baixos custos;
- o *aumento exponencial da informação disponível*, em linha directa com a diminuição dos seus custos de armazenagem, a que se alia um elevado nível de automação; e
- o *desenvolvimento de uma panóplia de ferramentas*, oriundas de disciplinas como a *Investigação Operacional*, a *Estatística* ou a *Informática*.

Hoje em dia, com o advento do computador pessoal, qualquer gestor tem um fácil acesso a técnicas sofisticadas na área da previsão, de modo que novas técnicas têm vindo a ser propostas para alimentar este potencial. Existe, por conseguinte, uma necessidade de escolha do melhor método de previsão, que deverá depender da sua exactidão, custo computacional e complexidade de utilização.

Dentro da *IA*, a emergência de novas ferramentas inspiradas nos processos de evolução próprios dos seres vivos, como as *RNAs* e os *AGEs*, possibilitou todo um conjunto de alternativas estimulantes neste campo. Actualmente, a aplicação destas técnicas exige um esforço por parte de um perito, envolvendo etapas tais como:

- a *análise de dados*, englobando, entre outros, o pré-processamento e a caracterização da informação; e
- a *selecção de modelos de previsão*, que envolve a procura do modelo ideal para o tratamento dos dados.

Ora, no presente trabalho foi adoptada uma abordagem deveras diferente, em que se assume nenhum conhecimento à priori sobre cada série temporal (e.g., uso específico de transformações conhecidas, como do logaritmo). Mais ainda, os sistemas propostos funcionam de modo autónomo, não exigindo qualquer tipo de processamento estatístico.

Estes sistemas apresentam porém um senão, que se manifesta no seu elevado custo computacional. Contudo, à medida que o custo de computação decresce de forma exponencial[1], tal inconveniente passará a ter cada vez menos importância.

Uma comparação de resultados obtidos com os modelos de *PST* inspirados na Natureza e os convencionais, para diversas séries reais e artificiais, permite concluir que:

- O método de *AE*, embora simples, tem o melhor desempenho com séries lineares, especialmente quando estas contêm componentes de sazonalidade e tendência. Tal não é de se estranhar, dado que o *AE* foi desenvolvido especificamente este tipo de previsão; i.e., séries bem comportadas. Daí que, neste caso, o *AE* deverá ser o método considerado, principalmente se se pretende uma reduzida sofisticação e simplicidade de cálculo.

- Apesar de utilizar o mesmo modelo geral subjacente à metodologia de *BJ* ($ARMA$), a flexibilidade do sistema de *Meta-AGEs* permite-lhe ultrapassar este método, para todas as séries consideradas. Isto não obstante esta utilizar uma transformação logarítmica para duas séries (**passageiros** e **química**). Atendendo à popularidade da metodologia de *BJ*, estes resultados mostram que uma maior atenção deve ser dirigida à optimização de modelos $ARMA$ via *AGEs*, principalmente quando se utilizam séries com tendência ou não linearidade.

- As *RNAs* apresentam bons resultados em algumas das séries lineares, como no caso das séries **preços** ou **mortes**. Contudo, as *RNAs* encontram-se nitidamente talhadas para a modelação de problemas não lineares, onde o seu desempenho se destaca. Em particular, nas séries caóticas (**quadrática** e **henon**), onde todos os outros métodos falham.

- À medida que a complexidade das séries aumenta, com comportamentos não lineares, os métodos convencionais revelam-se claramente inadequados. Nesta situação, a diferença de desempenho justifica o investimento no uso dos modelos de previsão inspirados nos processos de evolução próprios dos seres vivos.

[1] Conforme a lei de Moore, que estipula que a capacidade de um processador duplica a cada 18-24 meses.

- A rápida aprendizagem, fornecida pelo algoritmo de *RPROP*, e a capacidade adaptativa das *RNAs*, faz com que estas sejam especialmente indicadas para a previsão em tempo real, ultrapassando todos os restantes modelos. Contudo, com o aumento do poder de computação, os *AGEs* poderão se impor como uma alternativa a considerar.

6.3 Perspectivas de Trabalho Futuro

O trabalho efectuado reflecte todo um esforço que tem vindo a ser desenvolvido nos últimos anos, começando primeiro pelo uso de *RNAs* para a previsão e depois evoluindo para a adopção dos *AGEs*.

Neste momento, que se atingiu uma base sólida em termos de conhecimento, importa indicar pistas para a consideração de novos caminhos, nomeadamente no que respeita a:

Sistemas de *RNEs* para o Desenho de Topologias de *RNAs*

Nas experiências efectuadas, foi possível verificar que as *RNAs* são poderosas ferramentas, embora dependam fortemente do correcto desenho da sua topologia. Estruturas pobres providenciam uma capacidade de aprendizagem insuficiente, enquanto estruturas demasiado complexas tendem a perder capacidade de generalização.

Existe, assim, uma necessidade de métodos automáticos para a selecção de topologias para *RNAs* que dêem corpo a um melhor potencial de generalização, quando se trata de resolver um dado problema. O sistema de *RNE*s proposto, baseado no critério *BIC*, revelou-se como uma solução adequada para a definição de topologias de RNAs a utilizar na área da previsão embora, devido ao seu potencial, possa vir a ser utilizado em quaisquer outras aplicações.

Por outro lado, embora se tenha apenas considerado o desenho de topologias de *RNUs*, os mesmos princípios mantêm-se quando se consideram outros tipos de arquitecturas (i.e., o *AGE* de ordem superior poderia ser facilmente adaptado para a optimização de topologias de diferentes classes de *RNAs*).

Adopção de Diferentes Arquitecturas de *RNAs* para a Previsão

Conforme descrito no Capítulo 3, dentro dos modelos conexionistas existem diferentes arquitecturas de *RNAs* que se podem utilizar nos processos de previsão. Ora, seria interessante considerar outros tipos de *RNAs* passíveis de serem aplicadas à *PST*, a saber [Patterson, 1996]:

- *Redes Neuronais Recorrentes*;
- *Radial Basis-Functions*; e
- *Redes de Regressão Generalizada.*

Um estudo comparativo entre todas estas, assim como das *RNUs*, permitirá tirar ilações sobre as vantagens e desvantagens de cada tipo de rede, em termos de factores como a velocidade de aprendizagem, a capacidade de generalização ou a eficácia de previsão.

Melhoria na Aprendizagem de *RNAs*

No Capítulo 2 foram apresentados diversos algoritmos de aprendizagem supervisionada, baseados no gradiente descendente, tendo-se optando pelo algoritmo *RPROP* para a aprendizagem dos modelos neuronais de previsão, devido à sua rápida convergência e robustez em relação ao ajustamento dos seus parâmetros (Secção 3.6.3).

Contudo, os algoritmos baseados no gradiente tendem a revelar dificuldades de convergência, quando a superfície de erro do problema é deveras rugosa, ficando presos em mínimos locais. E quanto maior for a complexidade da função a aprender (i.e., não linearidade), maior é a probabilidade desta situação ocorrer. Assim, foi proposto o uso de *AGEs* para a aprendizagem de *RNAs*, dado que estes realizam uma procura global multiponto, rapidamente alcançando soluções de elevada qualidade, sendo capazes de escapar a mínimos locais [Branke, 1995]. Todavia, no passado, os *AGEs* têm demonstrado uma convergência lenta, com dificuldades de ajustamento final, sendo ultrapassados pelos métodos de gradiente descendente [Schaffer et al., 1992].

Uma alternativa reside na combinação de ambos os paradigmas, por forma a aproveitar as valências de cada método, tomando partido do efeito evolucionário proposto por *Lamarck*[2] [Ackley & Littman, 1994]. Neste tipo de aprendizagem, uma população de *RNAs* evolui através um *AGE*, sendo cada indivíduo composto pelo conjunto dos pesos da *RNA* (Figura 6.1). A cada indivíduo é aplicado uma aprendizagem local, via métodos de gradiente descendente, para melhorar a sua aptidão. Por fim, os novos pesos são codificados de volta para o cromossoma, permitindo assim uma herança do conhecimento adquirido.

De facto, experiências efectuadas com a aprendizagem via efeito de *Lamarck* sobre *RNUs* com topologias fixas, em tarefas tão distintas como a *classificação* ou a *regressão*, revelaram que é possível superar os métodos de gradiente descendente (e.g., algoritmo de

[2]Jean Baptiste Lamarck (1744-1829). Naturalista francês, conhecido pela sua obra *"Philosophie zoologique"*, onde postulava que os atributos adquiridos poderiam ser herdados pelas gerações futuras.

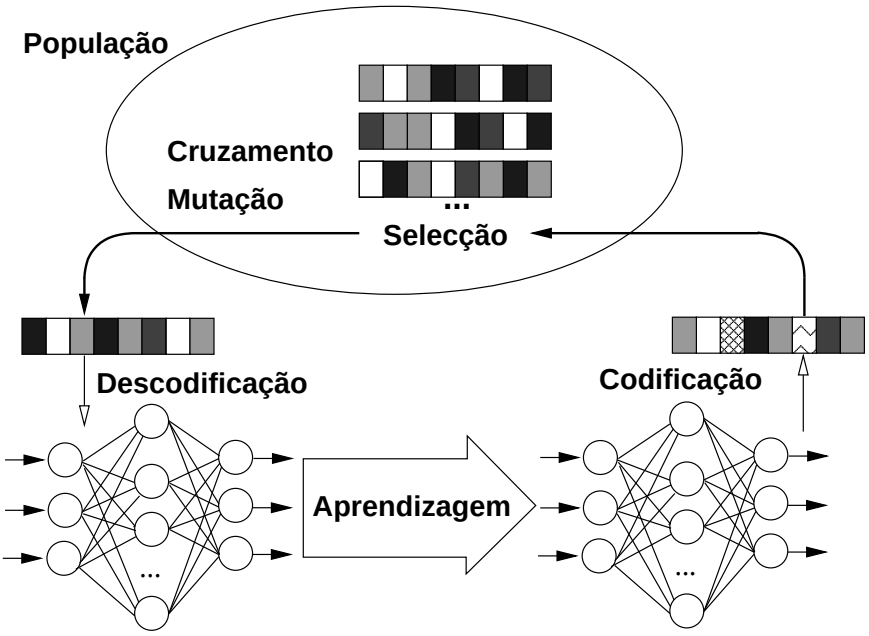


Figura 6.1: Aprendizagem de *RNAs* via modelo proposto por *Lamarck*.

RPROP) [Cortez et al., 2002]. Por conseguinte, esta estratégia poderá ser utilizada para a optimização do modelo neuronal. Neste cenário, poder-se-á extender esta visão do processo de aprendizagem, dando corpo a uma evolução simultânea da topologia da rede e dos pesos das conexões.

Aperfeiçoamento do Modelo Evolucionário

Embora se tenham obtidos bons resultados com a estimação dos modelos $ARMA$ via *AGEs* com *RVRs*, há situações em que uma combinação linear, de valores e erros passados, se revela deveras insuficiente. Por exemplo, tal comportamento foi verificado na previsão das séries caóticas (**quadrática** e **henon**), pelo que o modelo base poderá ser enriquecido com a integração de funções não lineares (e.g. logarítmicas ou trigonométricas).

Outra alternativa promissora, poderá passar pela amálgama de diferentes técnicas associadas à *Computação Evolucionária*, como a *Programação Genética* [Banzhaf et al., 1998]; i.e., através da utilização de modelos de previsão gerados pela aplicação de operadores matemáticos (e.g., $+, -, *$ ou $/$) sobre elementos passados de uma série temporal.

Exploração de Outros Modelos de Previsão Inspirados na Natureza

Neste trabalho foram apenas considerados dois modelos de previsão inspirados nos processos de evolução próprios dos seres vivos, as *RNUs,* em representação dos modelos conexionistas, e os *AGEs*, um ramo da família da *CGE*.

Contudo, nos últimos anos tem-se assistido a um enorme crescimento de outras técnicas, sustentadas no mesmo paradigma. Em particular, destaca-se um conjunto de métodos com potencial para a optimização combinatória e numérica, oriundos da nova disciplina da *Inteligência Social*[3] *(IS)* [Bonabeau et al., 1999]. A *IS* centra-se no estudo do comportamento de agentes simples (e.g., abelhas ou formigas), que interagem com o seu ambiente local, do qual emerge uma inteligência colectiva. Como exemplo deste tipo de estratégia, têm-se os *Algoritmos de Exames de Partículas* [Kennedy & Eberhart, 1995].

Previsão de Médio e Longo Prazo

Como foi referido no Capítulo 5, existem quatro classes principais de previsão, atendendo a diferentes horizontes temporais: *em tempo real*, *de curto prazo*, *de médio prazo* e *de longo prazo*.

Ora, neste trabalho abordaram-se apenas as duas primeiras categorias de previsão, embora as restantes sejam igualmente úteis para as organizações (e.g., para o planeamento de investimentos ou gestão da produção). Porém, ambos os modelos propostos podem ser adaptados a previsões de médio e longo prazo, existindo duas alternativas [Faraday & Chatfield, 1998; Neves & Cortez, 1998]:

- *adição de múltiplas saídas*, cada uma treinada para corresponder à previsão de diferentes períodos em frente; e
- *recorrência do valor de saída*, ou seja, o valor previsto para um período à frente é realimentado ao modelo, de modo a este fornecer a previsão de dois períodos à frente, prosseguindo-se este processo em cascata, até que se obtenha o número desejado de previsões em avanço.

Previsão Multi-Variável

A *PST* recorre-se dos valores passados de uma dada variável para a previsão do comportamento de sistemas complexos. Porém, existem situações em que é preferível incorporar

[3]Tradução adoptada para o termo *Swarm Intelligence*.

diversos factores. Este tipo de previsão, denominado de *Previsão de Séries Temporais Multi-Variável*, é utilizado num vasto leque de domínios, como os da *Medicina* ou da *Economia* [Brockwell & Davis, 1996]. Por exemplo, a evolução da cotação de uma dada acção da bolsa poderá ser melhorada caso se incluam também outros factores no sistema, tais como o valor de taxas de juro e/ou índices bolsistas.

Assim, os modelos desenvolvidos poderão ser extendidos para incorporar entradas de múltiplas variáveis, ficando a selecção da relevância destas a cargo do AGE de nível superior.

Apêndice A

Estatística

A disciplina de *Estatística* lida com valores numéricos de variáveis, tendo como objectivos fundamentais a recolha, a compilação, a análise e a interpretação de dados [Aczel, 1996]. Exemplos da sua aplicação surgem nos mais variados domínios, nomeadamente nos estudos de mercado, de medicina ou de controlo de qualidade.

Uma variável aleatória X pode ser vista como o nome de uma experiência com um resultado probabilístico. A recolha de dados de uma variável aleatória, designada de *amostra,* refere-se a um subconjunto de medições $X_1, ..., X_n$, escolhidas sobre um dado universo, chamado de *população.* O *valor esperado* (E), ou *média* $(\overline{X})$ de uma amostra é dado por:

$$E[X] = \overline{X} = \frac{\sum_{i=1}^{n} X_i}{n} \tag{A.1}$$

A *variância* (s^2) e o *desvio padrão* (s) são duas medidas de dispersão dos elementos de uma amostra, sendo obtidas a partir das equações:

$$s^2 = \frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}$$
$$s = \sqrt{s^2}$$

Um importante aspecto sobre as propriedades de uma variável aleatória é a forma da sua *distribuição*, que indica a frequência de valores para diferentes intervalos da variável em estudo. Em particular tem-se a distribuição *normal* (ou *gaussiana*), cuja curva, na forma de um sino, depende de dois parâmetros: a média e o desvio padrão (Figura A.1) [WEB, 2001]. Trata-se de uma distribuição presente em inúmeros fenómenos reais (daí o termo "normal"), onde cerca de 68% das observações estão contidas num intervalo de $\pm$ um (1) desvio padrão a partir da média.

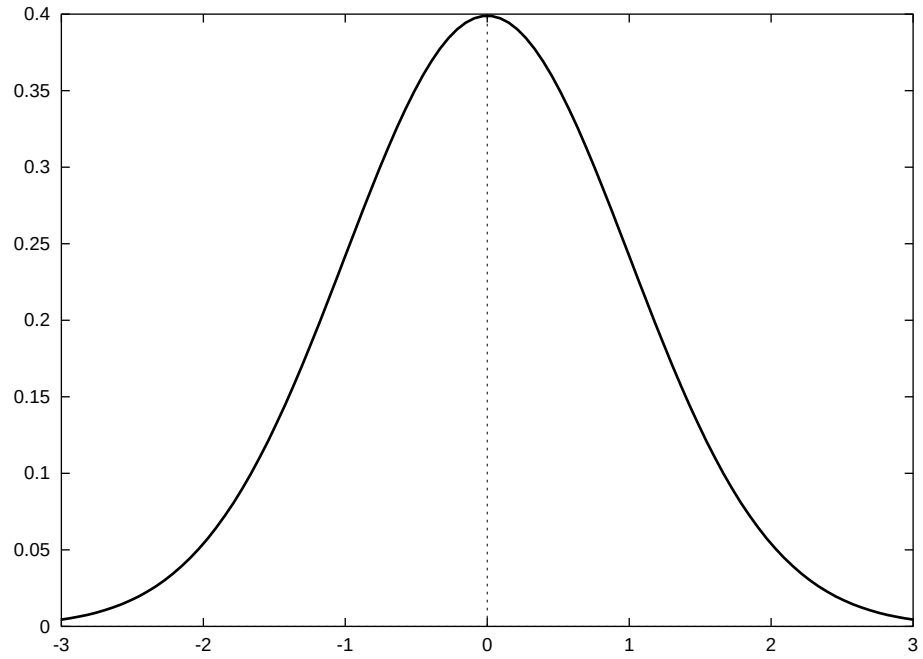


Figura A.1: Distribuição normal com uma média 0 e desvio padrão 1.

O *teorema do limite central* estipula que quando uma amostra de eventos independentes, retirada de uma população com uma média de μ e um desvio padrão σ (finito), contém um número elevado de elementos ($n \geq 30$), então a distribuição da amostra, com média $\overline{X}$, tenderá para uma distribuição normal (N) com média μ e desvio padrão $\sigma/\sqrt{n}$, ou seja:

$$\overline{X} \quad \acute{e} \quad N(\mu, \tfrac{\sigma}{\sqrt{n}}) \tag{A.2}$$

Um *intervalo de confiança* denota uma gama de valores que sustentam um parâmetro desconhecido para a população. Associado a este intervalo existe uma *medida de confiança* sobre se o intervalo de confiança sustenta ou não tal parâmetro. Para uma distribuição normal, o verdadeiro valor da média está contido dentro do intervalo:

$$\overline{X} \pm z_N \times \frac{\sigma}{\sqrt{n}} \tag{A.3}$$

onde z_N denota uma constante correspondente ao nível de significância (Tabela A.1) [Mitchell, 1997]. Em geral, utiliza-se um nível de significância de 95%, para o qual o valor de z_N é 1.96 .

Tabela A.1: Valores de z_N para diferentes níveis de significância.

Nível de significância	z_N
50%	0.67
68%	1.00
80%	1.28
90%	1.64
95%	1.96
98%	2.33
99%	2.58

Apêndice B

Caso Prático

No que se segue será descrito como se processa a criação dos modelos de previsão inspirados nos processos de evolução próprios dos seres vivos, considerando a série **passageiros** (Secção 3.4). Esta série contém um total de 144 observações (Tabela B.1), que serão divididas em dois conjuntos:

- *de treino*, utilizados para a construção do modelo (130 valores); e
- *de teste*, para avaliar a qualidade (ou eficácia) do modelo de previsão (últimos 14 valores, em *itálico* na tabela).

Tabela B.1: Número mensal passageiros de uma companhia aérea internacional (em milhares).

	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
1949	112	118	132	129	121	135	148	148	136	119	104	118
1950	115	126	141	135	125	149	170	170	158	133	114	140
1951	145	150	178	163	172	178	199	199	184	162	146	166
1952	171	180	193	181	183	218	230	242	209	191	172	194
1953	196	196	236	235	229	243	264	272	237	211	180	201
1954	204	188	235	227	234	264	302	293	259	229	203	229
1955	242	233	267	269	270	315	364	347	312	274	237	278
1956	284	277	317	313	318	374	413	405	355	306	271	306
1957	315	301	356	348	355	422	465	467	404	347	305	336
1958	340	318	362	348	363	435	491	505	404	359	310	337
1959	360	342	406	396	420	472	548	559	463	407	*362*	*405*
1960	*417*	*391*	*419*	*461*	*472*	*535*	*622*	*606*	*508*	*461*	*390*	*432*

Por uma questão de simplificação, optar-se-á pela escolha de modelos de previsão de menor cardinalidade, utilizando-se a *Janela Temporal Deslizante* $JTD =< 1, 12, 13 >$. A Figura B.1 exemplifica como são gerados os casos de treino a partir desta janela.

$$\begin{array}{ccc} < 112, 118, 115 > & \rightarrow & < 126 > \\ < 118, 132, 126 > & \rightarrow & < 141 > \\ \ldots & \ldots & \ldots \\ < 404, 359, 463 > & \rightarrow & < 407 > \end{array}$$

Figura B.1: Casos de treino para a série passageiros.

B.1 Redes Neuronais Unidireccionais

Em seguida, será demonstrado como se processa a aprendizagem com os modelos de *RNUs*. Para facilitar a exposição, apenas será utilizado o primeiro caso de treino ($x =< 112, 118, 115 >$, $y =< 126 >$), escolhendo-se uma topologia com apenas um nodo intermédio e seis conexões, conforme descrito na Figura B.2. Os valores iniciais dos pesos da rede são gerados de forma aleatória, segundo a fórmula proposta por Gallant [1993] (Secção 2.1.4). Na mesma figura também se apresentam os valores de integração (u_i, a *itálico*) e activação de cada nodo (s_i, a **negrito**):

$$\begin{array}{c} u_4 = x_1 \times w_{4,1} + x_2 \times w_{4,2} + x_3 \times w_{4,3} \\ u_4 = 112 \times -0.31 + 118 \times 0.24 + 115 * 0.31 = -42.2 \\ s_4 = 1/(1 + e^{-u_4}) = 4.7 \times 10^{-19} \\ u_5 = 112 \times -0.49 + 115 \times -0.01 + s_4 \times -0.27 = -56.2 \\ s_5 = u_5 = -56.2 \end{array} \quad \text{(B.1)}$$

A partir deste momento, pode iniciar-se a aprendizagem da rede. Aqui será utilizado o algoritmo *RPROP*, que depende da computação do respectivo gradiente, de acordo os cálculos que a seguir se apresentam (Figura B.3):

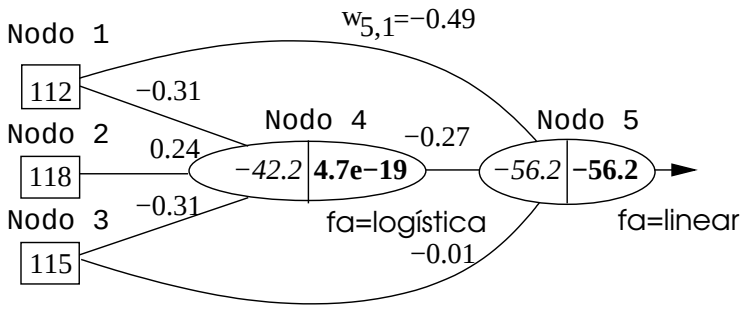


Figura B.2: Topologia inicial para a *RNU*.

$$
\begin{gathered}
\delta\xi/\delta s_5 = -(y_1 - s_5) = -(126 - (-56.2)) = -182.2 \\
\delta s_5/\delta w_{5,1} = fa'(u_5) \times s_1 = 1 \times 112 = 112 \\
\delta E/\delta w_{5,1} = \delta\xi/\delta s_5 \times \delta s_5/\delta w_{5,1} = -182.2 \times 112 = -2.0 \times 10^4 \\
\delta\xi/\delta s_4 = \delta\xi/\delta s_5 \times fa'(u_5) \times w5,4 = -182.2 \times 1 \times -0.27 = 49.2 \\
\delta s_4/\delta w_{4,1} = fa'(u_4) \times s_1 = fa'(-42.2) \times 112 \\
\delta s_4/\delta w_{4,1} = (sigmoid(-42.2) \times (1 - sigmoid(-42.2))) \times 112 = 5.2 \times 10^{-17} \\
\delta E/\delta w_{4,1} = \delta\xi/\delta s_4 \times \delta s_4/\delta w_{4,1} = 49.2 \times 5.2 \times 10^{-17} = 2.6 \times 10^{-15} \\
\ldots
\end{gathered}
\tag{B.2}
$$

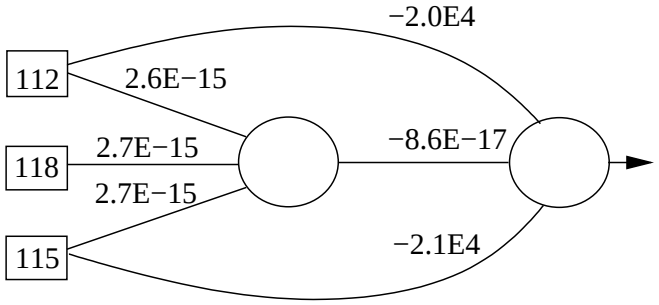


Figura B.3: Valores do gradiente para a *RNU*.

Após o cálculo do gradiente, pode executar-se a primeira iteração do algoritmo *RPROP*, o que gera uma nova matriz de pesos, em que se nota uma redução do erro de treino (Figura B.4).

O erro de treino final, considerando todos os casos de treino, é obtido ao fim de 1000 iterações, com um valor de $RMQE_{tr} = 32.3$ (Figura B.5). Esta rede produz as seguintes

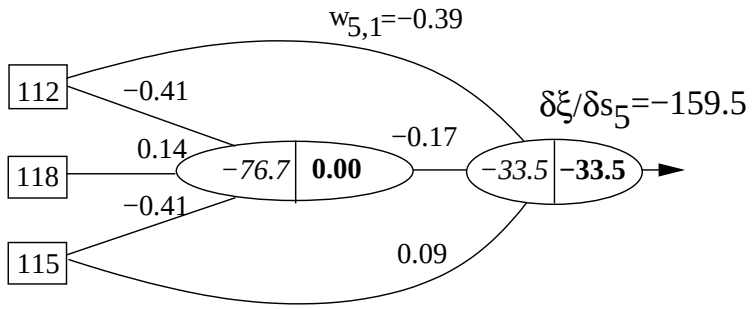


Figura B.4: Topologia da *RNU* após uma iteração *RPROP*.

previsões:

$$< 361, 373, 403, 390, 439, 437, 467, 515, 594, 624, 538, 467, 418, 438 >$$

que correspondem a um erro de previsão de $RMQE_{pr} = 19.8$.

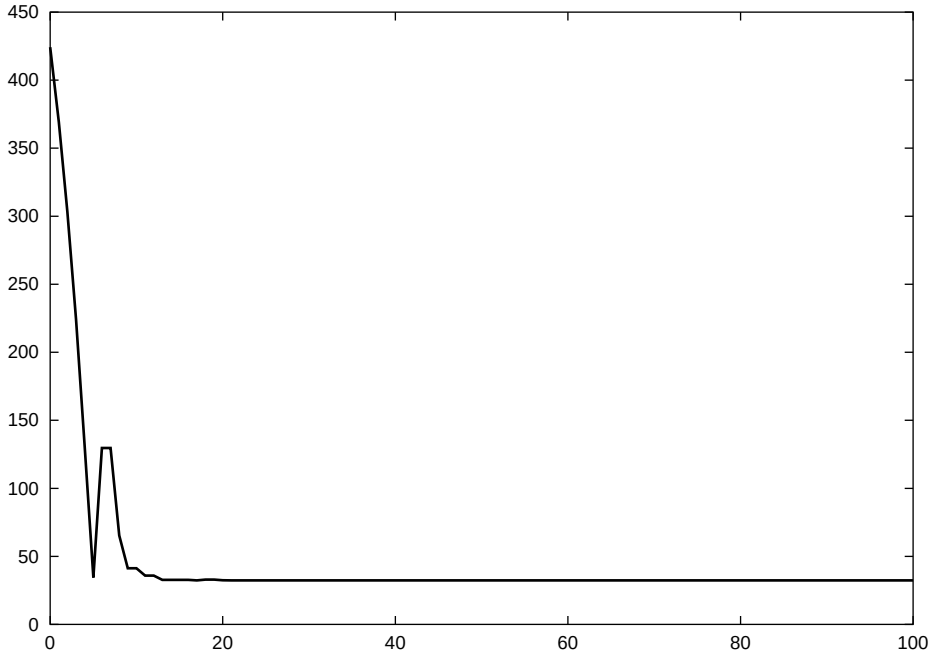


Figura B.5: Evolução do erro de treino ($RMQE$) ao longo de 100 iterações.

B.2 Algoritmos Genéticos e Evolucionários

Para esta exposição foi escolhida uma pequena população, com somente quatro indivíduos, sendo o modelo $ARMA$ constituído pelas janelas:

$$AR =< 1, 12, 13 >$$
$$MA =< 1, 12, 13 >$$

A Tabela B.2 exibe a configuração da população inicial, onde os genes foram aleatoriamente e tomaram valores no intervalo $[-1, 1]$. Na tabela, a coluna I_p denota o índice do indivíduo na população, enquanto que a coluna Apt denota o respectivo valor de aptidão ($RMQE_{tr}$). Esta população apresenta-se com uma aptidão mínima de 384 ($I_p = 4$) e média de 862.

Tabela B.2: População inicial do AGE.

I_p	μ	AR	MA	Apt
1	-0.55	$< 0.84_1, -0.59_{12}, 0.45_{13} >$	$< 0.46_1, 0.81_{12}, -0.30_{13} >$	2012
2	0.72	$< -0.17_1, -0.61_{12}, 0.37_{13} >$	$< 0.05_1, -0.68_{12}, 0.91_{13} >$	595
3	-0.75	$< -0.99_1, -0.23_{12}, -0.50_{13} >$	$< 0.31_1, 0.07_{12}, 0.22_{13} >$	456
4	-0.42	$< 0.81_1, 0.47_{12}, -0.76_{13} >$	$< 0.23_1, -0.47_{12}, 0.89_{13} >$	384

De seguida, é aplicada a primeira iteração do AGE, sendo gerados três novos indivíduos, de acordo com os cálculos que a seguir se apresentam:

$$\begin{gathered} g_3'^1 = g_3^3 + N(0.0, 0.5) = -0.23 + 0.18 = -0.05 \\ g_0'^2 = \lambda g_0^4 + (1-\lambda) g_0^2 = 0.50 \times -0.42 + 0.50 \times 0.72 = 0.15 \\ g_1'^2 = 0.50 \times 0.81 + 0.50 \times -0.17 = 0.32 \\ \ldots \end{gathered} \tag{B.3}$$

em que g_i^j denota o gene na posição i do indivíduo j, g' a população constituída pelos novos indivíduos (Tabela B.3), $N(m, d)$ uma distribuição *gaussiana* (ou *normal*) com média m e desvio padrão d, e λ é um valor gerado aleatório e pertencendo ao intervalo $[0, 1]$.

O primeiro indivíduo (g'^1) é construído através de uma mutação sobre o terceiro gene do indivíduo 3 da população inicial (g_3^3). Os restantes indivíduos (g'^2 e g'^3) são criados através do cruzamento aritmético entre os progenitores 4 e 2 da população original, sendo idênticos devido ao facto de $\lambda = 0.5$.

Para completar a iteração, resta a selecção dos sobreviventes. A aplicação dos valores

Tabela B.3: População com os novos indivíduos (g').

I_p	μ	AR	MA	Apt
1	-0.75	$< -0.99_1, -0.05_{12}, -0.50_{13} >$	$< 0.31_1, 0.07_{12}, 0.22_{13} >$	427
2	0.15	$< 0.32_1, -0.07_{12}, -0.19_{13} >$	$< 0.14_1, -0.57_{12}, 0.90_{13} >$	486
3	0.15	$< 0.32_1, -0.07_{12}, -0.19_{13} >$	$< 0.14_1, -0.57_{12}, 0.90_{13} >$	486

padrão utilizados nesta tese ($VE = 1$, $TS = 60\%$) faz com que sejam seleccionados 2 indivíduos de cada uma das populações (g e g'): g^4, g^2, g'^1 e g'^2 (Tabela B.4). De notar que embora se mantenha o valor de aptidão mínimo (384), o valor médio diminuiu (473) .

Tabela B.4: População após a primeira geração.

I_p	μ	AR	MA	Apt
1	-0.42	$< -0.81_1, 0.47_{12}, -0.76_{13} >$	$< 0.23_1, -0.47_{12}, 0.89_{13} >$	384
2	0.72	$< -0.17_1, -0.61_{12}, 0.37_{13} >$	$< 0.05_1, -0.68_{12}, 0.91_{13} >$	595
3	-0.75	$< -0.99_1, -0.05_{12}, -0.50_{13} >$	$< 0.14_1, -0.57_{12}, 0.90_{13} >$	427
4	0.15	$< 0.32_1, -0.07_{12}, -0.19_{13} >$	$< 0.14_1, -0.57_{12}, 0.90_{13} >$	486

O modelo optimizado, considerando uma população padrão (50 indivíduos), é obtido ao fim da geração 1000 (Figura B.6), correspondendo a um erro de $RMQE_{tr} = 17.7$, em que $\mu = 10.26$, $AR =< 0.77_1, 0.22_{12}, 0.00_{13} >$ e $MA =< 0.30_1, 0.41_{12}, 0.21_{13} >$. Com este modelo, e em termos de previsões, tem-se que:

$$< 375, 360, 422, 403, 416, 427, 477, 490, 580, 634, 555, 465, 436, 402 >$$

valores estes que corporizam um elevado erro de previsão ($RMQE_{pr} = 31$), devido ao fenómeno de *sobre-ajustamento*, que aqui ocorre.

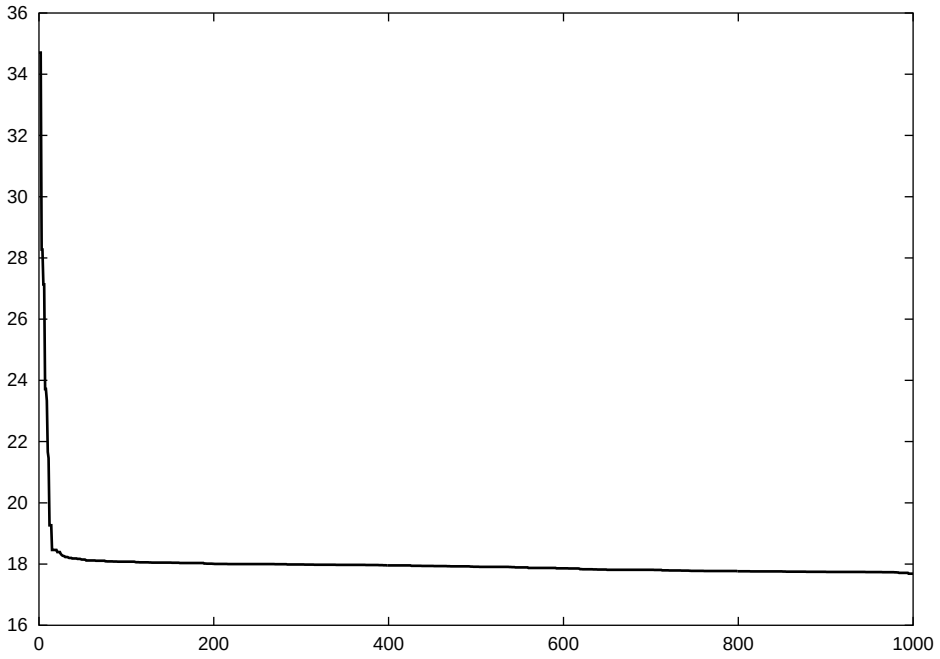


Figura B.6: Evolução do erro de treino ($RMQE$) ao longo de 1000 gerações.

Apêndice C

Ambientes de Programação

No Capítulo 2 foram apresentados, de um modo sucinto, os ambientes de programação utilizados nesta tese. Optou-se por esta forma, devido ao facto da ênfase que se pôs neste trabalho estar no uso dos modelos de previsão baseados nos processos de evolução próprios dos seres vivos, e não a defesa de uma arquitectura dos ambientes de programação que os sustentam. Todavia, neste apêndice procurar-se-á dar uma visão diferente, sob um ponto de vista do utilizador ou programador do sistema.

C.1 Redes Neuronais Artificiais

O *Ambiente de Programação de Redes Neuronais Artificiais (APRNA)* possui todo um conjunto de funcionalidades para lidar com *RNAs*, nomeadamente com redes supervisionadas, do tipo unidireccional ou recorrente. Em seguida são descritas as classes de objectos fundamentais para a aplicação de *RNUs*:

Classe *Cases*

Descrição: Classe que lida com o conjunto de casos de treino.

Métodos Principais

Cases(Ficheiro)

Descrição: Cria os casos de treino a partir de um ficheiro com o nome **"Ficheiro"**.

Exemplo: `Cases xor("xor.dat");`

ostream& operator<<(PS, C)

Descrição: Imprime o conteúdo dos casos de treino **C** para o periférico de saída **PS**.

Exemplo: `cout << "Casos de Treino:\n" << xor;`

void save(Ficheiro)

Descrição: Grava os casos de treino para um ficheiro com o nome **"Ficheiro"**.

Exemplo: `xor.save("xor2.dat");`

int test()

Descrição: Retorna o número de casos de teste.

Exemplo: `int test=xor.test();`

int training()

Descrição: Retorna o número de casos de treino.

Exemplo: `int training=xor.training();`

Exemplo do formato do ficheiro que contém os casos de treino:

```
# 3 Training cases
[ 112 118 115 ][ 126 ]
[ 118 132 126 ][ 141 ]
[ 132 129 141 ][ 135 ]
# 2 Test cases
[ 129 121 135 ][ 125 ]
[ 121 135 125 ][ 149 ]
```

Classe *FF*

Descrição: Trata-se da classe central para a criação e manipulação de *RNUs*.

Métodos Principais

FF(E , I, S, F, B, A)

Descrição: Cria uma *RNU* com uma camada intermédia, com a topologia completamente interligada **E-I-S**, onde **E**, **I** e **S** denotam valores do tipo inteiro (**int**). Este construtor adopta uma mesma função de activação para todos os nodos, definida pelo valor da variável **F** (e.g., logística). As entradas **B** e **A** constituem variáveis do tipo booleano, definindo a existência das ligações de *bias* e de atalhos, caso o seu valor seja verdadeiro (**true**).

Exemplo: `FF rede(2,2,1,SIGMOID,true,false);`

FF(N , T, M)

Descrição: Construtor genérico. Cria uma *RNU* a partir da definição de **N,** objecto que representa o conjunto dos nodos (i.e., funções de activação); **T**, vector de inteiros, contendo o número de nodos por camada; e **M**, matriz de pesos do tipo unidireccional (classe **FFWM**).

Exemplo:

```
Vint[3]; t[0]=2; t[1]=2; t[2]=1; // Topologia
FunctionType f1=SIGMOID; FunctionType f2=LINEAR;
Vector<Function> f(2); f[0]=f1; f[1]=f2; // Funções
Layers l(t); // Camadas
Nodes n(l,f); // Nodos
FFWM m(l,true,false) // Matriz de pesos
FF rede(n,t,m);
```

FF(Ficheiro)

Descrição: Cria uma *RNU* a partir de um ficheiro com o nome "**Ficheiro**".

Exemplo: `FF rede("exemplo.ann");`

void output(VE, VS)

Descrição: Calcula o vector de saída (**VS**) da rede, a partir do vector de entrada (**VE**).

Exemplo:

```
VReal e(2), s(1);
e[0]=1.0; e[1]=0.0; // Valores de entrada
rede.output(e,s);
```

Métodos Herdados da Classe *SANN*

Real earlystopping(PS, CP, C, PC, AT, PA, TE)

Descrição: Executa o treino da rede, com critérios de paragem, para o conjunto de casos de treino **C**. As restantes variáveis referem-se a: **PS** - periférico de saída para os resultados de erro; **CP** - vector de reais que contém os parâmetros dos critérios de paragem; **PC** - vector de inteiros sobre diversos parâmetros dos casos de treino (e.g., número de casos de teste); **AT** - algoritmo de treino; **PA** - vector de reais que denota os parâmetros do algoritmo de treino; e **TE** - o tipo de erro. Entre outros, estão disponíveis os algoritmos de treino: *RP, RP* com *termo de momentum, QuickProp* e *RPROP*.

Exemplo:

```
// Critérios de Paragem
const int MINI=0; // Número mínimo de iterações
const int MAXI=1000; // Número máximo de iterações
int FT=5; // Faixa de treino
Real ME=0.1; // Medida de erro estacionário
VReal cp(3); cp[0]=MINI; cp[1]=MAXI; cp[2]=FT;

// Casos de treino
Cases c(“xor.dat”);

// Parâmetros dos casos de treino
int test=0; // Número de casos de teste
int Est=Alltr; // Usar todos casos para treino
Vint pc(2); pc[0]=Est; pc[1]=test;

// Tipo de Erro
Error te(RMSE); // RMSE - ROOT MEAN SQUARED ERROR (= RMQE)
```

```
// Algoritmo de Treino
int at=Rprop; // Algoritmo de treino RPROP
VReal pa(2); pa[0]=0.1; pa[1]=50.0; // Parâmetros RPROP

// Treino da rede
rede.earlystopping(cout,cp,c,pc,at,pa,te);
```

Real error(TE, C)

Descrição: Calcula o erro, do tipo **TE**, da rede, sobre o conjunto de casos de treino **C**.

Exemplo:

```
Cases c("xor.dat");
Error te(RMSE); // RMSE - ROOT MEAN SQUARED ERROR (= RMQE)
Real e=rede.error(te,c);
```

void gallant_weights()

Descrição: Inicializa os pesos da rede, gerados de modo aleatório, segundo a fórmula sugerida por Gallant [1993].

Exemplo: `rede.gallant_weights();`

ostream& operator<<(PS, rede)

Descrição: Imprime o conteúdo da **rede** para o periférico de saída **PS**.

Exemplo: `cout << "Rede:\n" << rede;`

void save(Ficheiro)

Descrição: Grava o conteúdo de uma *RNU* para um ficheiro com o nome **"Ficheiro**".

Exemplo: `rede.save("exemplo.ann");`

int weights()

Descrição: Retorna o número de pesos da rede.

Exemplo: `int p=rede.weights();`

Exemplo do formato do ficheiro cujo conteúdo é uma *RNU* com a topologia 2-2-1:

```
WeightMatrix:
[ 1 1 1 X X X ]
[ 1 1 1 X X X ]
[ 1 X X 1 1 X ].
Nodes:
[ sigmoid,1 sigmoid,1 linear,1 ]
Layers:
[ 0 1 ]
[ 2 3 ]
[ 4 ].
```

Classe *FFWM*

Descrição: Classe que manipula uma matriz de pesos unidireccionais.

Métodos Principais

FFWM(TR, B, A)

Descrição: Construtor que cria uma matriz de pesos para uma dada topologia da rede (**TR**) completamente interligada. As entradas **B** e **A** constituem variáveis do tipo booleano, definindo a existência das ligações de *bias* e de atalhos, caso o seu valor seja verdadeiro (**true**).

Exemplo:

```
Vint[3]; t[0]=2; t[1]=2; t[2]=1; // Camadas
Layers l(t); // Topologia da rede
FFWM matriz(l,true,false);
```

void gradient(VE, VS, TR, F, G)

Descrição: Calcula o gradiente (**G**) a partir dos vectores de entrada e saída (**VE** e **VS**), da topologia da rede (**TR**) e do conjunto de funções de activação (**F**).

Exemplo:

```
VReal e(2), s(1);
```

```
e[0]=1.0; e[1]=0.0; // Entradas da rede
s[0]=0.5; // Valor desejado na saída
Vint[3]; t[0]=2; t[1]=2; t[2]=1; // Camadas
Layers l(t); // Topologia da rede
FunctionType f1=SIGMOID; FunctionType f2=LINEAR;
Vector<Function> f(2); f[0]=f1; f[1]=f2;
Nodes n(l,f); // Funções de activação
g=0.0; // Inicialização do gradiente
FFWM g=matriz.gradient(e,s,l,n,g);
```

FFWM &operator=(X)

Descrição: Estabelece todos os pesos da matriz com o valor de **X**.

Exemplo: `matriz=0.0;`

void output(VE, TR, F, VS)

Descrição: Calcula o vector de saída (**VS**) a partir de um vector de entrada (**VE**), da topologia da rede (**TR**) e do conjunto de funções de activação (**F**).

Exemplo:

```
VReal e(2), s(1);
e[0]=1.0; e[1]=0.0; // Valores de entrada
Vint[3]; t[0]=2; t[1]=2; t[2]=1; // Camadas
Layers l(t); // Topologia da rede
FunctionType f1=SIGMOID; FunctionType f2=LINEAR;
Vector<Function> f(2); f[0]=f1; f[1]=f2;
Nodes n(l,f); // Funções de activação
matriz.output(e,l,n,s);
```

Métodos Herdados da Classe *WeightMatrix*

void gallant_weights()

Descrição: Inicializa os pesos da matriz, gerados de modo aleatório, segundo a fórmula sugerida por Gallant [1993].

Exemplo: `matriz.gallant_weights();`

ostream& operator<<(PS, M)

Descrição: Imprime o conteúdo da matriz de pesos (**M**) para o periférico de saída **PS**.

Exemplo: `cout << "Matriz de pesos:\n" << matriz;`

int weights()

Descrição: Retorna o número de pesos da matriz.

Exemplo: `int p=matriz.weights();`

Classe *Function*

Descrição: Classe que lida com funções de activação.

Métodos Principais

Function(TF, K)

Descrição: Cria uma função de activação do tipo **TF** (e.g., *linear* ou *logística*) e com a inclinação **K**.

Exemplo:

```
// Funções de activação disponíveis:
// LINEAR (linear)
// PWISE (por troços)
// SIGMOID (logística)
// TANH (tangente hiperbólica)
// SIN (seno)
// COS (coseno)
// GAUSSIAN (gaussiana)
// SQUARE (quadrada)
FunctionType TF=SIGMOID;
Real K=1.0;
Function f(TF,K);
```

Real derive(X)

Descrição: Calcula a derivada da função de activação para a entrada **X**.

Exemplo: `Real d=f.derive(0.955);`

Real operator()(X)

Descrição: Calcula o valor da função de activação para a entrada **X**.

Exemplo: `Real v=f(0.995);`

ostream& operator<<(PS, F)

Descrição: Imprime o conteúdo da função de activação **F** para o periférico de saída **PS**.

Exemplo: `cout << "Função de Activação:\n" << F;`

void read(SC)

Descrição: Lê uma função de activação a partir da sequência de caracteres **SC**.

Exemplo: `f.read("sigmoid,1.0");`

Classe *Layers*

Descrição: Classe que estabelece a topologia de uma *RNA*.

Métodos Principais

Layers(T)

Descrição: Construtor que utiliza um vector de inteiros (**T**).

Exemplo:

```
Vint[3]; t[0]=2; t[1]=2; t[2]=1; // Topologia
Layers l(t);
```

int inputs()

Descrição: Devolve o número de nodos de entrada da rede.

Exemplo: `int inputs=l.inputs();`

int layer(N)

Descrição: Devolve o número da camada que contém o nodo **N**.

Exemplo: `int nodelayer=l.layer(3);`

int nodes()

Descrição: Devolve o número total de nodos da rede.

Exemplo: `int n=l.nodes();`

Real outputs()

Descrição: Devolve o número de nodos de saída da rede.

Exemplo: `int outputs=l.outputs();`

ostream& operator<<(PS, TR)

Descrição: Imprime o conteúdo da topologia da rede **TR** para o periférico de saída **PS**.

Exemplo: `cout << "Topologia:\n" << l;`

void remove(N)

Descrição: Remove o nodo **N** da topologia.

Exemplo: `l.remove(3);`

int size()

Descrição: Devolve o número de camadas da topologia.

Exemplo: `int s=l.size();`

int size(C)

Descrição: Devolve o número de nodos da camada **C**.

Exemplo: `int inputs=l.size(0);`

Classe *Nodes*

Descrição: Classe que lida com um conjunto de funções de activação. Nota: o termo "Nodes" foi escolhido devido ao facto de esta classe manipular a informação presente ao nível do nodo de uma rede. Assim, para outras arquitecturas de *RNAs*, esta classe poderia incluir mais elementos para além da função de activação.

Métodos Principais

Nodes(TR, VF)

Descrição: Construtor que utiliza a topologia de rede **TR** e o vector **VF**, que contém apenas uma função de activação por cada tipo de camada activa (intermédia ou de saída) da rede.

Exemplo:

```
Vint[3]; t[0]=2; t[1]=2; t[2]=1; // Topologia
Layers l(t);
FunctionType f1=SIGMOID; FunctionType f2=LINEAR;
Vector<Function> f(2); f[0]=f1; f[1]=f2;
Nodes nodos(l,f);
```

void add(P, F)

Descrição: Acrescenta a função de activação **F** na posição **P**.

Exemplo:

```
Function f(SIGMOID,2.0);
nodos.add(3,f);
```

int size()

Descrição: Devolve o número total de nodos da rede.

Exemplo: `int n=nodos.nodes();`

ostream& operator<<(PS, N)

Descrição: Imprime o conteúdo dos nodos (**N**) para o periférico de saída **PS**.

Exemplo: `cout << "Nodos:\n" << nodos;`

void remove(N)

Descrição: Remove o nodo **N**.

Exemplo: `nodos.remove(3);`

C.2 Ambiente de Programação de Algoritmos Genéticos e Evolucionários

Conforme descrito anteriormente (Secção 2.16), o *APAGE* assenta em quatro blocos fundamentais: os indivíduos (classe **Indiv**), a população (classe **Popul**), o *AGE* (classe **GA**) e a avaliação (classe **Eval**).

Ao todo, este ambiente de programação contempla mais de vinte classes, divididas pelos blocos já referidos, que podem ser ajustadas às necessidades do utilizador (encontra-se fora do âmbito deste trabalho uma descrição detalhada de todas estas [Rocha & Neves, 2000]). Assim, optar-se-á por uma exposição do código principal do *AGE*, que optimiza modelos *ARMA*:

```
// Definição do Problema
TS serie("passengers.ts"); // Série temporal
VReal AR(2); VReal MA(2); Real U; // Coeficientes ARMA
int TI=AR.size()+MA.size()+1; // Tamanho do indivíduo
int NP=14; // Número de previsões

// Operadores genéticos
int NO=2; // Número de operadores
OperationPar op1(AVG,0.67,2,2,0,1.0); // Cruzamento aritmético
OperationPar op2(FMUT,0.33,1,1,0,1.0); // Mutação gaussiana
OperationPar op[NO]; op[0]=op1; op[1]=op2;

// Selecção
int TP=50; // Cardinalidade da população
Real TS=0.6*TP; // Taxa de substituição (60% da população)
SelectionPar sp(MIN,TS,TS,ROUL_WHEEL,RANKING,DESC_RANK,1,TS,1);
```

```
// População com valores tomados do domínio dos números reais
Popul<RBRIndiv> pop(PS,TI,op,NO,sp);
pop.initRandom(); // Inicialização aleatória

// Função de avaliação
EvalARMA *eval = new EvalARMA(serie,U,AR,MA,NP);

// Parâmetros do AGE
int GMAX=1000; // Número máximo de gerações
GAPar<Popul<RBRIndiv,Real>> gapar(op, NO, sp, eval, GMAX, TP,
TI, 0, 0, STATS_ALL);

// AGE
GA ga(gapar, pop);
ga.run(); // Executar o algoritmo
```

C.3 Previsão de Séries Temporais

A aplicação para a *PST*, designada **tsf**, possibilita o uso dos diferentes tipos de previsão mencionados na tese, nomeadamente *AE*, *ARMA*, *RNUs*, *AGEs*, *RNEs* e *Meta-AGEs*. Para tal, recorre aos ambientes de programação *APRNA* e *APAGE*, e à classe **TS**.

Classe *TS*

Descrição: Classe que permite criar e manipular séries.

Métodos Principais

TS(Ficheiro)

Descrição: Cria uma série temporal a partir de um ficheiro com o nome **"Ficheiro"**.

Exemplo: `TS serie("passengers.ts");`

ostream& operator<<(PS, ST)

Descrição: Imprime o conteúdo da série **ST** para o periférico de saída **PS**.

Exemplo: `cout << "Série Temporal:\n" << serie;`

Real arma(AR, MA, U, TE, EI, EF, VP);

Descrição: Efectua a previsão de curto prazo, via modelo *ARMA*, desde o elemento **EI** até ao elemento **EF**, para os coeficientes **AR**, **MA** e **U**. O vector **VP** contém os valores das previsões, sendo devolvido o erro, do tipo **TE**, de ajuste dos dados.

Exemplo:

```
VReal ar(2); ar[0]=1; ar[1]=-0.5; // Componente AR
VReal ma(1); ma[0]=0.09; // Componente MA
VReal vp(12); // Vector com as previsões
Real error=serie.arma(ar,ma,1.2,RMSE,132,144,vp);
```

Real avg()

Descrição: Retorna a média dos elementos da série.

Exemplo: `Real avg=serie.avg();`

Real holtwinters(Alpha, Beta, Gamma, K, TE, EI, EF, VP)

Descrição: Efectua uma previsão de curto prazo, via *AE*, desde o elemento **EI** até ao elemento **EF**, para os parâmetros **Alpha**, **Beta**, **Gamma** e **K**. O vector **VP** contém os valores das previsões, sendo devolvido o erro, do tipo **TE**, de ajuste dos dados.

Exemplo:

```
VReal vp(12);
Real error=serie.holtwinters(0.2,0.2,0.6,12,RMSE,132,144,vp);
```

void save(Ficheiro)

Descrição: Grava a série para um ficheiro com o nome "**Ficheiro**".

Exemplo: `serie.save("passengers2.dat");`

int size()

Descrição: Retorna o número de elementos da série.

Exemplo: `int size=serie.size();`

Cases toCases(EI, EF, JE, JS)

Descrição: Cria um conjunto de casos de treino a partir de uma série, começando desde o elemento **EI** até o elemento **EF**, utilizando as janelas temporais deslizantes de entrada (**JE**) e saída (**JS**). Trata-se de um método que permite fazer a interacção da série com as *RNUs*.

Exemplo:

```
Vint je(3); je[0]=1; je[1]=12; je[2]=13;
Vint js(1); js[0]=1;
Cases c=serie.toCases(0,ts.size(),je,js);
```

Exemplo do formato do ficheiro cujo conteúdo é uma série temporal:

```
112
118
132
129
121
135
148
136
```

Exemplos da utilização da aplicação tsf

Alisamento Exponencial

Comando: `tsf h passengers 14 12`

Resultado:

```
Time Series:passengers
Forecasts:14
Holt-Winters >> Sazonality:  12
Grid 0.01 search:  RMSE=10.5266 Alpha=0.29 Beta=0.03 Gamma=0.95
F TS CP
1 362 348.863
2 405 387.376
```

```
...
14 432 433.888
SRMSE= 16.5
...
Total time:  324 s.
```

Redes Neuronais Unidireccionais

Comando: `tsf n passengers 14 1 0 13 1,12,13 2 1 1 0`

Resultado:

```
Time Series:passengers
Forecasts:14
Neural Networks >> Randseed:  968238686
Runs:  1 Emethod:  0 w= [ 1 12 13 ]
Hidden Nodes:  2
WeightMatrix:
[ 1 1 1 1 X X X ]
[ 1 1 1 1 X X X ]
[ 1 1 1 1 1 1 X ].
Nodes:
[ sigmoid,1 sigmoid,1 linear,1 ]
Layers:
[ 0 1 2 ]
[ 3 4 ]
[ 5 ].
Ep:  0 t:  0 RMSEtr=202.529 RMSEts=324.468
...
Stopping Criterium:  Training Progress:  pk= 0.0181572
Best Training:  Epoch 1000 Time 14 RMSE Tr 9.99094 Ts 18.7969
...
```

Apêndice D

Referencial

Neste apêndice é realizada uma compilação sobre um conjunto de referências relativas às áreas relevantes da tese: *Previsão*, *Redes Neuronais Artificiais* e *Computação Genética e Evolucionária*. As referências encontram-se estruturadas em quatro grupos distintos: conferências, revistas, livros editados, e guia de páginas *Internet*.

D.1 Previsão

D.1.1 Conferências

- *International Symposium on Forecasting (ISF)*;
- *Forecasting Financial Methods (FFM)*;
- *Federal Forecasters Conference (FFC)*; e
- *Nostradamus International Conference*.

D.1.2 Livros

[Box & Jenkins, 1976][Brockwell & Davis, 1996][Chatfield, 1989][Hanke & Reitsh, 1989][Makridakis & Wheelwright, 1989][Weigend & Gershenfeld, 1994]

D.1.3 Revistas

- *International Journal of Forecasting*, Elsevier Science Publishers;

- *Journal of Forecasting*, John Wiley & Sons, Ltd; e
- *Journal of Business Forecasting*, Institute of Business Forecasting (IBF).

D.1.4 Recursos Electrónicos

Time Series Data Library

URL: http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/

Descrição: Base de dados, sob a alçada de Rob Hyndman, com mais de 500 séries temporais.

Principles of Forecasting

URL: http://hops.wharton.upenn.edu/forecast/

Descrição: Portal, criado por J. Armstrong, onde se encontram resumidos os princípios básicos da previsão.

International Institute of Forecasters (IIJ)

URL: http://forecasting.cwru.edu/Institute/

Descrição: Página oficial da *IIJ*, uma instituição não lucrativa que tenta desenvolver e estimular o campo da previsão.

Forecast Pro Home Page

URL: http://www.forecastpro.com/

Descrição: Página sobre o *Forecast Pro*, um programa de previsão bastante popular, de fácil uso.

Time Series Analysis and Forecasting Techniques

URL: http://ubmail.ubalt.edu/~harsham/stat-data/opre330Forecast.htm

Descrição: Sumário sobre métodos de análise e de previsão de séries temporais, sob a direcção de Hossein Arsham.

D.2 Redes Neuronais Artificiais

D.2.1 Conferências

- *International Joint Conference on Neural Networks (IJCNN)*;
- *European Symposium on Artificial Neural Networks (ESANN)*;
- *International Work-Conference on Artificial and Natural Neural Networks (IWANN)*; e
- *International Conference on Engineering Applications of Neural Networks (EANN).*

D.2.2 Livros

[Bose & Liang, 1996][Haykin, 1999][Patterson, 1996][Rojas, 1996]

D.2.3 Revistas

- *IEEE Transactions on Neural Networks*, IEEE;
- *Neural Computation*, MIT Press;
- *International Journal of Neurocomputing*, Elselvier Science Publishers;
- *Journal of Artificial Neural Networks (JANN),* ABLEX; e
- *Neural Processing Letters*, Kluwer Academic Publishers.

D.2.4 Recursos Electrónicos

Neural Networks Frequently Asked Questions

URL: ftp://ftp.sas.com/pub/neural/FAQ.html

Descrição: Repositório das perguntas mais frequentes sobre *RNAs*, sob a direcção de Warren Sarle.

Benchmarking of learning algorithms

URL: http://www.cs.stir.ac.uk/~lss/NNIntro/

Descrição: Página sobre a avaliação de algoritmos de aprendizagem, com um forte destaque para as *RNAs*.

Neural Networks at PNNL

URL: http://www.emsl.pnl.gov:2080/proj/neuron/neural/

Descrição: Portal da *Pacific Northwest National Laboratory (PNNL)*, contendo variada informação sobre *RNAs*: conferências, laboratórios, cursos *on-line*, *software*, etc.

International Neural Network Society

URL: http://www.inns.org/

Descrição: Página oficial da *International Neural Network Society (INNS)*, uma associação internacional que inclui cientistas, engenheiros e estudantes, com o objectivo de divulgar e melhorar o domínio da *computação neuronal*.

Grupo de Redes Neuronais

URL: http://ilusion.inesc.pt/

Descrição: Servidor WWW do grupo de redes neuronais do *Instituto de Engenharia de Sistemas e Computadores (INESC)*, Lisboa.

D.3 Computação Genética e Evolucionária

D.3.1 Conferências

- *Genetic and Evolutionary Computation Conference (GECCO)*;
- *Congress on Evolutionary Computation (CEC)*;
- *International Symposium on Adaptive Systems: Evolutionary Computation and Probabilistic Graphical Models (ISAS);* e
- *European Conference on Genetic Programming (EUROGP).*

D.3.2 Livros

[Banzhaf et al., 1998][Goldberg, 1989][Michalewicz, 1996][Schwefel, 1981]

D.3.3 Revistas

- *IEEE Transactions on Evolutionary Computation*, IEEE;
- *Evolutionary Computation Journal,* MIT Press;
- *Artificial Life*, MIT Press;
- *Biological Cybernetics*, Springer-Verlag; e
- *BioSystems Journal of Biological and Information Processing Sciences*, Elselvier Science Publishers.

D.3.4 Recursos Electrónicos

ENCORE

URL: http://www.cs.bham.ac.uk/Mirrors/ftp.de.uu.net/EC/clife/

Descrição: Portal sobre a *CGE*, incluindo, entre outros: perguntas mais frequentes, descrição dos métodos da *CGE*, guia de recursos electrónicos, grupos de investigação e personalidades ligadas ao ramo.

GA Arquives

URL: http://www.aic.nrl.navy.mil/galist/

Descrição: Repositório sobre *AGs*.

Illinois Genetic Algorithms Laboratory

URL: http://gal4.ge.uiuc.edu/illigal.home.html

Descrição: Portal do laboratório de *AGs* de Illinois, EUA, sob a direcção de David E. Goldberg.

Evolutionary Computation at PNNL

URL: http://www.emsl.pnl.gov:2080/proj/neuron/evolve/

Descrição: Portal da *PNNL*, contendo variada informação sobre a *CGE*: conferências, laboratórios, cursos *on-line*, *software*, etc.

The Hitch-Hiker's Guide to Evolutionary Computation

URL: http://www.cs.bham.ac.uk/Mirrors/ftp.de.uu.net/EC/clife/www/

Descrição: Um vasto guia electrónico sobre a *CGE*, sobre a direcção de Jörg Heitkötter e David Beasley.

Apêndice E

Créditos

O software de previsão *Forecast Pro* é uma marca registada da Business Forecast Systems, Inc., EUA *(http://www.forecastpro.com/).*

Intel e *Pentium* são marcas registadas da Intel (*http://www.intel.com*).

Esta tese foi escrita com *LyX* (*http://www.lyx.org*), versão 1.1.6fix2, o primeiro processador de texto do tipo *WYSIWYM*[1]. O *LyX* facilita a geração de documentos *LaTeX* (http://www.latex-project.org/), um sistema de produção de documentação científica e técnica, de elevada qualidade.

A maior parte das figuras foram desenhadas via *xfig* (*http://www.xfig.org*), um programa de desenho vectorial para o ambiente gráfico *XWindows*. Quanto aos gráficos bidimensionais, este foram obtidos via *gnuplot* (*http://www.gnuplot.info*), um programa interactivo para o desenho de curvas a partir de pontos, linhas e superfícies.

[1] Acrónimo, do inglês *What You See Is What You Mean.*

Bibliografia

[Ackley & Littman, 1994] Ackley, D. H. and Littman, M. L. *A case for Lamarckian evolution*, pages 3–10. Addison-Wesley, Reading, MA.

[Aczel, 1996] Aczel, A. *Statistics - Concepts and Applications*. IRWIN, USA.

[Agapie & Agapie, 1997] Agapie, A. and Agapie, A. Forecasting the Economic Cycles Based on an Extension of the Holt-Winters Model. A Genetic Algorithms Approach. In *Proceedings of the IEEE Computational Intelligence for Financial Forecasting Engineering*, pages 96–99.

[Akaike, 1973] Akaike, H. Information Theory and an Extension of the Maximum Likehood Principle. In Petrov, B. N. and Csaki, F., editors, *2nd International Symposium on Information Theory*, pages 267–281, Budapest.

[Alves, 2001] Alves, V. *Resolução de Problemas em Ambientes Distribuídos - Uma Contribuição nas Áreas da Inteligência Artificial e da Saúde*. PhD thesis, Departamento de Informática, Escola de Engenharia, Universidade do Minho.

[Angeline et al., 1994] Angeline, P., Sauders, G., and Pollack, J. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Trans. on Neural Networks*, 5(1):54–65.

[Arnow & Weiss, 1998] Arnow, D. and Weiss, G. *Introduction to Programming using Java: An Object-Oriented Approach*. Addison-Wesley.

[Azoff, 1994] Azoff, E. *Neural Network Time Series Forecasting of Financial Markets*. John Wiley & Sons.

[Baker, 1987] Baker, J. Reducing Bias and Inefficiency in the Selection Algorithm. In Grenfenstette, J., editor, *Second International Conference on Genetic Algorithms and their Applications*. Lawrence Erlbaum Associates.

[Banzhaf et al., 1998] Banzhaf, W., Nordin, P., Keller, R., and Francone, F. *Genetic Programming, An Introduction.* Morgan Kaufmann Publishers, Inc, USA.

[Barnett, 2001] Barnett, M. Adaptive Neural Networks for Intelligent Operation of the Activated Sludge Process. White paper, available from http://www.gensym.com/expert_operations/papers/paper.htm.

[Beasley et al., 1993] Beasley, D., Bull, D., and Martin, R. An overview of genetic algorithms: Part 2, research topics. *University Computing*, 15(4):170–181.

[Bonabeau et al., 1999] Bonabeau, E., Dorigo, M., and Théraulaz, G. *Swarm Intelligence: from natural to artificial systems.* Oxford University Press.

[Bose & Liang, 1996] Bose, N. and Liang, P. *Neural Network Fundamentals with Graphs, Algorithms and Applications.* McGraw-Hill, USA.

[Boser et al., 1992] Boser, B., Guyon, I., and Vapnik, V. A training algoritm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. San Mateo, CA: Morgan Kaufmann.

[Box & Jenkins, 1976] Box, G. and Jenkins, G. *Time Series Analysis: Forecasting and Control.* Holden Day, San Francisco, USA.

[Bramlette & Bouchard, 1991] Bramlette, M. and Bouchard, E. Genetic Algorithms in Parametric Design of an Aircraft. In Davis, L., editor, *Handbook of Genetic Algorithms.* Van Nostrand.

[Branke, 1995] Branke, J. Genetic algorithms for neural network design and training. In *Proceedings of the First Nordic Workshop on Genetic Algorithms*, pages 145–163, University of Vaasa, Finland.

[Breedam, 1996] Breedam, A. V. An analysis of the effect of local improvement operators in genetic algorithms and simulated annealing for the vehicle routing problem. RUCA Working Paper 96/14, University of Antwerp.

[Brockwell & Davis, 1996] Brockwell, P. and Davis, R. *Introduction to Time Series Forecasting.* Springer-Verlag, New York, USA.

[Broomhead & Lowe, 1988] Broomhead, D. and Lowe, D. Multivariable functional interpolation and adaptative networks. *Complex Systems*, 2:321–355.

[Burgess, 1995] Burgess, C. A Genetic Algorithm for the Optimization of a Multiprocessor Computer Architecture. Technical Report CSTR-95-024, Dep. Computer Science, University of Bristol.

[Chai et al., 1997] Chai, C., Chuek, C., DP, M., and Huat, T. Time Series Modelling and Forecasting using Genetic Algorithms. In *Proceedings of the First International Conference on Knowledge-Based Intelligent Electronic Systems*, volume 1, pages 260–268, Adelaide, Australia.

[Charytoniuk & Chen, 2000] Charytoniuk, W. and Chen, M. Very Short-Term Load Forecasting Using Artificial Neural Networks. *IEEE Transactions on Power Systems*, 15(1):263–268.

[Chatfield, 1989] Chatfield, C. *The Analysis of Time Series - An Introduction*. Chapman and Hall, UK.

[Chen & Lu, 1999] Chen, S. and Lu, C. Would Evolutionary Computation Help in Designs of ANNs in Forecasting Foreign Exchange Rates? In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 1, pages 267–274.

[Chiraphadhanakul et al., 1997] Chiraphadhanakul, S., Dangprasert, P., and Avatchanakorn, V. Genetic Forecasting Algorithm with Financial Applications. In *Proceedings of Intelligent Information Systems - IIS'97*, pages 174–178.

[Cortez et al., 1996] Cortez, P., Machado, J., and Neves, J. An Evolutionary Artificial Neural Network Time Series Forecasting System. In Hamza, M. H., editor, *IASTED International Conference on Artificial Intelligence, Expert Systems and Neural Networks*, pages 278–281, Honolulu, Havai. IASTED ACTA Press.

[Cortez et al., 1995] Cortez, P., Rocha, M., Machado, J., and Neves., J. A Neural Network Based Forecasting System. In *ICNN'95 - IEEE International Conference on Neural Networks Proceedings*, volume 5, pages 2689–2693, Perth, Western Australia. IEEE Computer Society.

[Cortez et al., 1999] Cortez, P., Rocha, M., and Neves., J. An Evolutionary and Connectionist Approach for Time Series Forecasting. In *Proceedings of International Conference on Systems Engineering - ICSE'99*, pages 19–24, Las Vegas, USA. INCOSE.

[Cortez et al., 2002] Cortez, P., Rocha, M., and Neves., J. A Lamarckian Approach for Neural Network Training. *Neural Processing Letters*, 15(2):105–116.

[Cramer, 1985] Cramer, N. A representation for the adaptive generation of simple sequential programs. In Grefenstette, J. J., editor, *International Conference on Genetic Algorithms and Applications*, pages 183–187.

[Darwin, 1859] Darwin, C. *The Origin of Species by means of natural selection, or the preservation of favoured races in the struggle for life.* J. Murray, London, UK.

[Denoeux & Rizand, 1995] Denoeux, T. and Rizand, P. Analysis of radar images for rainfall forecasting using neural networks. *Neural Computing and Applications*, 3.

[Ding et al., 1995] Ding, X., Canu, S., and Denoeux, T. Neural network based models for forecasting. In *Proceedings of ADT'95.*

[Efron & Tibshirani, 1993] Efron, B. and Tibshirani, R. *An Introduction to the Bootstrap.* Chapman & Hall, USA.

[Fahlman, 1988] Fahlman, S. Faster Learning Variations on Back-Propagation: An Empirical Study. In D. Touretzky, G. H. and Sejnowski, T., editors, *Proceedings of Connectionist Models Summer School*, pages 38–51, Los Altos CA, USA. Morgan Kaufmann Publishers.

[Falco et al., 1998] Falco, I., Cioppa, A., Iazzetta, A., Natale, P., and Tar, E. Optimizing neural networks for time series prediction. In *Proceedings of the Third World Conference on Soft Computing (WSC3).*

[Falkenauer, 1994] Falkenauer, E. Setting New Limits in Bin Packing with a Grouping GA Using Reduction. Technical Report RO108, CRIF-Research Center for Belgian Metalworking Industry.

[Faraday & Chatfield, 1998] Faraday, J. and Chatfield, C. Times Series Forecasting with Neural Networks: A Case Study. *Applied Statistics*, 47:231–250.

[Flexer, 1996] Flexer, A. Statistical evaluation of neural networks experiments: Minimum requirements and current practice. In *Proceedings of the 13th European Meeting on Cybernetics and Systems Research*, volume 2, pages 1005–1008, Vienna, Austria.

[Fogel et al., 1966] Fogel, L., Owens, A., and Walsh, M. *Artificial Intelligence Through Simulated Evolution.* John Wiley, New York.

[Freisleben & K. Ripper, 1995] Freisleben, B. and K. Ripper, K. Economic Forecasting Using Neural Networks. In *ICNN'95 - IEEE International Conference on Neural Networks Proceedings*, Perth, Western Australia. IEEE Computer Society.

[Gallant, 1993] Gallant, S. *Neural Network Learning and Expert Systems.* MIT Press, Cambridge, USA.

[Goldberg, 1989] Goldberg, D. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Publishing Company, Inc., NY, USA.

[Gong & Denceux, 1996] Gong, N. and Denceux, T. Urban storm water pollution forecasting using recurrent neural networks. *Journal of Systems Engineering*, 6:137–147.

[Grenfenstette, 1986] Grenfenstette, J. Optimization of control parameters for Genetic Algorithms. *IEEE Transaction on Systems, Man, and Cybernetics*, 16(1):122–128.

[Gruau & Whitley, 1993] Gruau, F. and Whitley, D. Adding learning to the cellular development of neural networks: Evolution and the baldwin effect. *Evolutionary Computation*, 3(1):213–233. MIT Press.

[Hallas & Dorffner, 1998] Hallas, M. and Dorffner, G. A comparative study on feedforward and recurrent neural networks in time series prediction using gradient descent learning. In *Proceedings of the European Meetings on Cybernetics and Systems Research (EMCSR'98)*, Vienna, Austria. Lawrence Erlbaum Associates.

[Hanke & Reitsch, 1989] Hanke, J. and Reitsch, A. *Business Forecasting.* Allyn and Bancon Publishing Company Inc., Massachussetts, USA.

[Haykin, 1999] Haykin, S. *Neural Networks - A Compreensive Foundation.* Prentice-Hall, New Jersey, 2nd edition.

[Holland, 1975] Holland, J. *Adaptation in Natural and Artificial Systems.* PhD thesis, University of Michigan, Ann Arbor.

[Hopfield, 1982] Hopfield, J. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. In *the National Academy of Science*, volume 79, pages 2554–8.

[Hyndman, 2001] Hyndman, R. *Time Series Data Library.* Available from http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/.

[Jordan, 1995] Jordan, M. Why the logistic function? a tutorial discussion on probabilites and neural networks. 9503, Massachusetts Institute of Technology, USA.

[Kaboudan, 1999] Kaboudan, M. A. Genetic Evolution of Regression Models for Business and Economic Forecasting. In *Proceedings of the 1999 Congress on Evolutionary Computation - CEC 99*, volume 2, pages 1217–1223.

[Kemsley & Martinez, 1992] Kemsley, D. and Martinez, T. A Survey Of Neural Network Research And Fielded Applications. *International Journal of Neural Networks: Research and Applications*, 2:123–133.

[Kennedy & Eberhart, 1995] Kennedy, J. and Eberhart, R. Particle Swarm Optimization. In *ICNN'95 - IEEE International Conference on Neural Networks Proceedings*, pages 1942–1948, Perth, Western Australia. IEEE Computer Society.

[Kernigham & Ritchie, 1988] Kernigham, B. and Ritchie, D. *The C Programming Language.* Prentice Hall, second edition.

[Khotanzad et al., 1995] Khotanzad, A., Abaye, A., and Maratukulam, D. An Adaptive and Modular Recurrent Neural Network Based Power System Load Forecaster. In *Proceedings of ICNN'95, 1995 IEEE International Conference on Neural Networks*, Perth, Western Australia.

[Koehn, 1994] Koehn, P. Combining Genetic Algorithms and Neural Networks. Thesis for Master Science Degree, University of Tennessee, Knoxville, USA.

[Kohavi, 1995] Kohavi, R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Montreal, Quebec, Canada.

[Kohonen, 1982] Kohonen, T. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69.

[Korf, 1985] Korf, R. Depth-first iterative deepening: An optimal admissible tree search. *Artificial Intelligence*, 27:97–109.

[Koza, 1989] Koza, J. Hierarchical genetic algorithms operating on populations of computer programs. In Sridharan, N. S., editor, *Proceedings of Eleventh International Joint Conference on Artificial Intelligence IJCAI-89*, volume 1, pages 768–774. Morgan Kaufmann.

[Kwok & Yeung, 1999] Kwok, T. and Yeung, D. Constructive algorithms for structure learning in feedforward neural networks for regression problems problems: A survey. *IEEE Transactions on Neural Networks*, 8(3):630–645.

[Lapedes & Farber, 1987] Lapedes, A. and Farber, R. Non-Linear Signal Processing Using Neural Networks: Prediction and System Modelling. Technical Report LA-UR-87-2662, Los Alamos National Laboratory, USA.

[LeBaron, 1995] LeBaron, B. Experiments in evolutionary finance. Working paper, University of Winsconsin, Madison, USA.

[Luger & Stubblefield, 1998] Luger, G. and Stubblefield, W. *Artificial Intelligence, Structures and Strategies for Complex Problem Solving.* Addison Wesley Longman, Inc., USA.

[Makridakis, 1982] Makridakis, S. The accuracy of extrapolation (times series) methods: Results of a forecasting competition. *Journal of Forecasting*, 1:111–153.

[Makridakis & Wheelwright, 1989] Makridakis, S. and Wheelwright, S. *Forecasting Methods for Management.* John Wiley & Sons, New York, fifth edition.

[Mathias et al., 1998] Mathias, L., Bouchard, K., and Bisson, J. A Real-Time Operational Inflow Forecasting System. In *1998 International Water Resources Engineering Conference*, Memphis, USA.

[Mayer, 2001] Mayer, U. Linux/unix nbench. Available from http://www.tux.org/ mayer/lynux/bmark.html,Tux.Org.

[McCluskey, 1993] McCluskey, P. Feedforward and recurrent neural networks and genetic programs for stock market and time series forecasting. Thesis for master science degree, Brown University, Rhode Island, USA.

[McCulloch & Pits, 1943] McCulloch, W. and Pits, W. A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

[Memmi, 1989] Memmi, D. Connectionism and artificial intelligence. In *Neuro-Nimes'89 International Workshop on Neural Networks and their Applications*, pages 17–34, Nimes, France.

[Michalewicz, 1996] Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, USA, third edition.

[Miller et al., 1989] Miller, G., Todd, P., and Hedge, S. Designing Neural Networks using Genetic Algorithm. In Schaffer, D. J., editor, *3rd International Conference on Genetic Algorithms*. Morgan Kaufmann.

[Mitchell, 1997] Mitchell, T. *Machine Learning*. McGraw Hill.

[Musser & Saini, 1996] Musser, D. and Saini, A. *STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library*. Addison-Wesley.

[Neves & Cortez, 1998] Neves, J. and Cortez, P. Combining Genetic Algorithms, Neural Networks and Data Filtering for Times Series Forecasting. In Mastorakis, N. E., editor, *2nd IMACS International Conference on: Circuits, Systems and Computers - IMACS-CSC'98*, volume 2, pages 933–939, Piraeus, Greece. IMACS.

[Neves et al., 1999] Neves, J., Rocha, M., Rodrigues, H., Biscaia, M., and Alves, J. Adaptive Strategies and the Design of Evolutionary Applications. In W.Banzhaf, J.Daida, A.Eiben, M.Garzon, V.Honavar, M.Jakiela, and R.Smith, editors, *Genetic and Evolutionary Computation Conference (GECCO99)*, pages 473–479, Orlando, Florida, USA. Morgan Kaufmann.

[Papadourakis et al., 1993] Papadourakis, G., Spanoudakis, G., and Gotsias, A. Application of Neural Networks in Short-Term Stock Price Forecasting. In *Proceedings of the First International Workshop on Neural Networks in the Capital Markets*, London, UK.

[Patterson, 1996] Patterson, D. *Artificial Neural Networks - Theory and Applications*. Prentice Hall, Singapore.

[Peitgen et al., 1992] Peitgen, H., Jürgens, H., and Saupe, D. *Chaos and Fractals - New Frontiers of Science*. Springer-Verlag, New York.

[Petrovski & McCall, 1997] Petrovski, A. and McCall, J. Comparison of a statistical approach to ga parameter selection with the meta-gas method. In *Proceedings of the Third Joint Conference on Information Sciences*.

[Pohlheim, 1996] Pohlheim, H. Genetic algorithms: Principles, methods and algorithms. Technical report, Technical University Ilmenau.

[Prechelt, 1994] Prechelt, L. PROBEN1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules. Research Report, Fakultät für Informatik, Universität Karlsruhe Germany.

[Prechelt, 1998] Prechelt, L. *Early Stopping – but when?* In: *Neural Networks: Tricks of the trade*, Springer Verlag, Heidelberg.

[Rechenberg, 1973] Rechenberg, I. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Holzboog Verlag, Stuttgart.

[Riedmiller, 1994] Riedmiller, M. Supervised Learning in Multilayer Perceptrons - from Backpropagation to Adaptive Learning Techniques. *Computer Standards and Interfaces*, 16.

[Riedmiller & Braun, 1993] Riedmiller, M. and Braun, H. A Direct Adaptative Method for Faster Backpropagation Learning: The RPROP Algorithm. In *Proceeedings of the IEEE International Conference on Neural Networks*, San Francisco, CA, USA.

[Rocha, 1997] Rocha, M. Uma aproximação à resolução do caixeiro viajante via programação genética. Thesis for Master Science Degree, Departamento de Informática, Universidade do Minho, Braga, Portugal.

[Rocha et al., 2001] Rocha, M., Mendes, R., Cortez, P., and Neves, J. Sitting Guest at a Wedding Party: Experiments on Genetic and Evolutionary Constrained Optimization. In *Proceedings of The 2001 Congress on Evolutionary Computation (CEC2001)*, volume 1, pages 671–678, Seoul, Korea. IEEE Computer Society.

[Rocha & Neves, 2000] Rocha, M. and Neves, J. Computação Genética e Evolucionária. *Apontamentos de apoio à disciplina de Sistemas Inteligentes*, Unidade de Ensino, Departamento de Informática, Escola de Engenharia, Braga, Portugal.

[Rocha et al., 2000] Rocha, M., Vilela, C., Cortez, P., and Neves, J. Viewing Scheduling Problems through Genetic and Evolutionary Algorithms. In et al., J. R., editor, *Parallel and Distributed Processing, IPDPS 2000 Workshops Proceedings, Lecture Notes in Computer Science 1800*, pages 612–619. Springer.

[Rojas, 1996] Rojas, R. *Neural Networks - A Systematic Introduction.* Springer-Verlag, Germany.

[Rumelhart et al., 1986] Rumelhart, D., Hinton, G., and Williams, R. Learning Internal Representations by Error Propagation. In Rulmelhart, D. and McClelland, J., editors, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, volume 1, pages 318–362, MIT Press, Cambridge MA.

[Russell & Norvig, 1995] Russell, S. and Norvig, P. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, New Jersey, USA.

[Rycroft, 1993] Rycroft, R. Microcomputer software of interest to forecasters in comparative review. *International Journal of Forecasting*, pages 531–575.

[Santos, 1999] Santos, M. *Sistemas de Classificação em Ambientes Distribuídos.* PhD thesis, Universidade do Minho.

[Sarle, 1995] Sarle, W. Stopped Training and Other Remedies for Overfitting. In *Proceedings of the 27th Symposium on the Interface of Computer Science and Statistics*, pages 352–360.

[Sarle, 1999] Sarle, W. Neural Network Frequently Asked Questions. Available from ftp://ftp.sas.com/pub/neural/FAQ.html.

[Schaffer et al., 1992] Schaffer, J., Whitley, D., and Eshelman, L. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In Whitley and Schaffer, editors, *Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pages 1–37.

[Schwarz, 1978] Schwarz, G. Estimating the Dimension of a Model. *The Annals of Statistics*, 6:461–4.

[Schwefel, 1981] Schwefel, H.-P. *Numerical Optimization of Computer Models.* Wiley.

[Sharda & Rampal, 1996] Sharda, R. and Rampal, R. Neural Networks and Management Science/Operations Research: A Bibliographic Essay. *Encyclopedia of Library and Information Science*, 61:247–259.

[Shi et al., 1999] Shi, Z., Tamura, Y., and Ozaki, T. Nonlinear time series modelling with the radial basis function-based state-dependent autoregressive model. *International Journal of Systems Science*, 30(7):717–727.

[Shoneburg, 1990] Shoneburg, E. Price Prediction using Neural Networks: A Project Report. *Neurocomputing*, 2:17–27.

[Stroustrup, 1991] Stroustrup, B. *The C++ Programming Language*. Addison Wesley, 2nd edition.

[Tang & Fishwick, 1993] Tang, Z. and Fishwick, F. Feed-forward Neural Nets as Models for Time Series Forecasting. *ORSA Journal of Computing*, 5(4):374–386.

[Thimm & Fiesler, 1997] Thimm, G. and Fiesler, E. Pruning of Neural Networks. Research report idiap-rr 97-03, IDIAP.

[Ulbricht, 1994] Ulbricht, C. Multi-recurrent Networks for Traffic Forecasting. In *Proceedings of AAAI'94 Conference*, Seatle, Washington, USA.

[WEB, 2001] WEB. Engineering statistics handbook. Available from http://www.itl.nist.gov/div898/handbook, NIST SEMATECH.

[Weigend & Gershenfeld, 1994] Weigend, A. and Gershenfeld, N. *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley, USA.

[Whitley, 1989] Whitley, D. The GENITOR Algorithm and Selection Pressure: Why Rank-based Allocation of Reproductive Trials is Best. In Schafer, J. D., editor, *Third International Conference on Genetic Algorithms*. George-Mason University, Morgan Kaufman.

[Whitley, 1995] Whitley, D. Genetic Algorithms and Neural Networks. In Perioux, J., Winter, G., Galan, M., and Cuesta, P., editors, *Genetic Algorithms in Engineering and Computer Science*. John Willey & Sons Ltd.

[Whitley et al., 1990] Whitley, D., Starkweatherand, T., and Bogart, C. Genetic Algorithms and Neural Networks: Optimizing Connections and Connectivity. *Parallel Computing*, 14:347–361.

[y Cajal, 1990] y Cajal, S. R. *New Ideas on the Structure of the Nervous System in Man and Vertebrates*. MIT Press, Cambridge, MA, translation of the French edition of 1894.

[Yao & Poh, 1995] Yao, J. and Poh, H. Forecasting the KLSE Index Using Neural Networks. In *ICNN'95 - IEEE International Conference on Neural Networks Proceedings*, Perth, Western Australia. IEEE Computer Society.

[Yooni et al., 1994] Yooni, B., Holmes, D., Langholz, G., and Kandel, A. Efficient genetic algorithms for training layered feedforward neural networks. *Information Sciences*, 76:67–85.

[Yule, 1927] Yule, G. U. On a method of investigating periodicities in disturbed series with special reference to wolfer's sunspot numbers. *Philos. Trans. Roy. Soc. London A*, 226:267–298.

Índice