

Sistemas Inteligentes



Paulo Salgado

2016

Paulo Salgado

CAPÍTULO I.....	4
1. INTRODUÇÃO.....	4
2. MODELO MATEMÁTICO DO NEURÓNIO.....	6
3. REDES NEURONAIAS ARTIFICIAIS.....	13
4. UMA VISÃO GERAL DAS RNAs.....	14
3.3.1. <i>Tipo de RNAs</i>	14
5. ESTRUTURA DA RNA.....	16
6. FUNÇÕES DE INTEGRAÇÃO E ATIVAÇÃO.....	19
7. APRENDIZAGEM	29
8. ALGORITMOS DE TREINO	29
9. ALGORITMOS DE RETROPOGAÇÃO	31
10. RADIAL BASIS FUNCTION – RBF.....	38
11. AMOSTRA DE TREINO E AMOSTRA DE TESTE.....	49
12. MEDIDAS DE DESEMPENHO.....	50
13. CONCLUSÃO	51
CAPÍTULO II.....	54
1. INTRODUÇÃO.....	54
2. LÓGICA DIFUSA.....	54
2.1.1 <i>Conceitos básicos</i>	55
3. OPERAÇÕES DIFUSAS.....	60
3.1. <i>Funções complemento difuso</i>	61
3.2. <i>Intersecção de conjuntos difusos</i>	62
3.3. <i>Reunião de conjuntos difusos</i>	65
4. PRINCÍPIO DA EXTENSÃO	69
5. RELAÇÃO E COMPOSIÇÃO	70
6. INFERÊNCIA DIFUSA.....	75
6.1. <i>Implicação difusa</i>	79
6.2. <i>Interpretação da regra difusa IF-THEN</i>	80
7. SUMÁRIO.....	82
CAPÍTULO III.....	84
1. INTRODUÇÃO.....	84
2. SISTEMAS DE LÓGICA DIFUSA.....	84
3. SLD UTILIZADOS EM ENGENHARIA.....	88
4. SISTEMAS DE IDENTIFICAÇÃO DIFUSOS	92
4.1.1. <i>O método “Back-propagation”</i>	93
4.1.2. <i>Método de mínimos quadrados ortogonais (“Orthogonal least squares”)</i>	96

4.1.3. <i>Mínimos quadráticos recursivos</i>	96
4.1.4. <i>Aprendizagem estrutural</i>	99
4.1.5. <i>Desenho do sistema difuso usando o método de “Table-Lookup”</i>	100
5. CONTROLADORES DE LÓGICA DIFUSA – CLD.....	104
5.1.1. <i>Controlador Linguístico</i>	105
5.1.2. <i>Controlo por modelo invertido</i>	105
5.1.3. <i>Gain Scheduling</i>	107
6. SUMÁRIO.....	108
CAPÍTULO IV	110
1. ALGORITMOS GENÉTICOS	110
2. PRINCIPIOS BÁSICOS	111
3. REPRESENTAÇÃO	113
4. FUNÇÃO DE DESEMPENHO.....	114
5. REPRODUÇÃO.....	114
6. SELEÇÃO.....	114
7. CRUZAMENTO.....	118
8. MUTAÇÃO	122
9. APLICAÇÕES DOS ALGORITMOS GENÉTICOS	124
10. OTIMIZAÇÃO POR EXAME DE PARTÍCULAS	131
11. OTIMIZAÇÃO POR COLÔNIAS DE FORMIGAS	135
12. SUMÁRIO	140
CAPÍTULO V APÊNDICE	143

Capítulo I

REDES NEURONIAIS

1. Introdução

A ideia de transplantar a mente humana ou a alma para outros sistemas é um velho sonho. Desde que se deu início a esta nova era tecnológica, com o advento novos métodos computacionais e de novas tecnologias de hardware poderosas, logo se perspetivaram, nas mentes mais acérrimas dos proveitos da ciência, a ambição de se construírem mecanismos cognitivos artificiais. Criar um humanoide com um cérebro eletrônico passou a ser um objetivo com um horizonte atingível. Uma das vias exploradas visou transpor o mecanismo elementar dos neurônios e a estrutura cerebral, através do estudo e modelação matemática e funcional do seu comportamento, para os sistemas computacionais. Deste esforço resultaram modelos matemáticos que descrevem o comportamento complexo dos neurônios, que embora podendo ser caracterizados como simples no seu detalhe, a sua interligação na estrutura complexa em rede e os ainda insuficientes mecanismos de aprendizagem fazem que estes sistemas ainda estejam longe dos objetivos finais. Estes são caminhos difíceis de construir, havendo dificuldades na sua arquitetura e gestão, embora com progressivos sucessos que lhe conferem hoje uma utilidade cada vez mais importante nas mais variadas aplicações da Inteligência Artificial.

Os neurônios são células básicas que transmitem impulsos elétricos e são fundamentais na determinação do comportamento do corpo humano e no funcionamento do raciocínio. A sua estrutura básica está ilustrada na Figura 1. É constituída por um corpo celular, dendrites, axônio e sinapses. O corpo celular contém o núcleo, onde se processam as transformações bioquímicas necessárias à síntese de certas moléculas enquanto as dendrites são responsáveis pela captação dos sinais que afluem ao neurônio. Cada neurônio possui um único axônio, que nasce do **cerne de implantação**, e termina com ramificações, que recebe o nome de **telodendro**. O axônio, ou fibra nervosa, é associada à transmissão de sinais emitidos para outros neurônios ou células musculares. A conexão de um axônio a uma dendrite de outro neurônio é feita pelas sinapses. Numa sinapse o terminal do axônio libera, para o espaço sináptico (20-40 nm), mensageiros químicos que interagem com receptores, induzindo pequenas correntes elétricas dentro da espinha dendrítica. Normalmente, estas correntes entram na célula excitando o neurônio ou, como outros casos, saem da célula inibindo a ativação do neurônio. Estes sinais elétricos

evoluem no tempo e propagam-se através dos neurônios pelas ligações existentes, razão pela qual os designaremos também por “ondas” neurológicas. Estas ondas de correntes (tanto as positivas como negativas) acumulam-se nas espinhas dendríticas e daqui viajam até ao corpo celular.

Tanto neurônios sensoriais como neurônios motores, todos têm em comum uma base funcional elétrica e química. Na regulação do sistema nervoso, os neurônios tanto competem como cooperam uns com os outros na tomada de decisão. Os sinais químicos que passam dos axônios para as dendritas são transformados em sinais elétricos. Estes sinais são integrados (com caráter de reforço ou de inibição), considerando todos os sinais elétricos recebidos por todas as sinapses que ligam ao neurônio, e deste processo resulta a decisão final de enviar, ou não, o sinal através do axônio para o neurônio(s) seguinte(s). Quando estas ondas não possuem muita atividade, depressa perdem força e extinguem-se. Porém se num determinado neurônio o potencial acumulado exceder o limiar de disparo, o neurônio responde com um fluxo nervoso (potencial) que será transmitido pelos seus axônios aos neurônios a ele ligados.

As diferentes partes do neurônio estão envoltas em processos biológicos dinâmicos de rearranjo que reflete a atividade celular e da ação do ambiente envolvente. As dendritas sofrem alterações de forma, criam novas ligações e eliminam outras. À medida que os neurônios lutam por terem um papel mais ou menos ativo na rede neuronal, os axônios desenvolvem novas ligações e novos terminais nervosos. Normalmente, as ligações sinápticas entre neurônios que exibem maior atividade são reforçadas.

O cérebro humano possui um número de neurônios da ordem de 10^{11} , cada um com aproximadamente 10^4 sinapses por neurônio. A forma como estes neurônios são interligados uns aos outros, o número de ligações e seus pesos entre neurônios e a tipologia funcional de cada um deles permitem a formação de redes neuronais complexas.

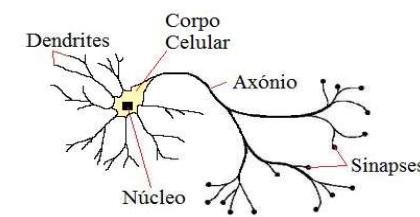


Figura 1. Neurônio Biológico

Os cientistas, inspirados pelos sistemas biológicos desenvolveram sistemas computacionais com capacidades de aprendizagem e generalização a partir de experiências. Assim surgiram as redes neurais artificiais (RNAs).

2. Modelo Matemático do neurónio

O sistema neuronal (SN), é composto por uma rede fortemente, interconectado de neurónios. A estrutura e mecanismos locais de um determinado neurónio biológico permite-lhe processar informação que lhe chegam pelas suas conexões de entrada, sendo por essa razão de carácter local. O comportamento inteligente do SN advém das múltiplas interações entre os neurónios, na forma como as ondas neuronais se propagam nessa rede. Do mesmo modo, uma rede neuronal artificial é composta por várias unidades de processamento (UP), cujo funcionamento elementar é de carácter simples. Essas unidades estão geralmente interligadas por canais (ramos) de comunicação com determinados pesos.

A operação de uma unidade de processamento, proposta por McCulloch e Pitts em 1943, pode ser resumida da seguinte maneira (ver Figura 2):

- Sinais são apresentados à entrada, x_i ($i=1,\dots,n$);
- Cada sinal de entrada é multiplicado por um peso w_i ($i=1,\dots,n$). Seu valor de magnitude maior traduz uma influência acrescida da respetiva entrada para a saída da unidade. Uma conexão entrada extra, de peso θ , é designada de *bias* ou *threshold*;
- No estágio de entrada da UP é realizada uma soma (integração) ponderada dos sinais, cujo resultado traduz o nível de atividade da UP. Frequentemente nesta operação de integração dos sinais de entrada é usado a função soma ponderada ($a = \sum_{i=1}^n w_i x_i - \theta$);
- Se o nível de atividade atingido no estágio de entrada da UP exceder um certo valor limite (*threshold*) a unidade gera um sinal de saída. Este mecanismo de ativação está associado a uma função f , designada de ativação, que pode ser linear ou não-linear. A função de ativação tem como características ser monótona, saturada entre dois limites mínimo e máximo, e frequentemente ser contínua. (Exemplo, a função degrau $f(a) = 0$ para $a < 0$ e $f(a) = 1$ para $a \geq 0$).
- Saída (s) representada (s) pela variável y .

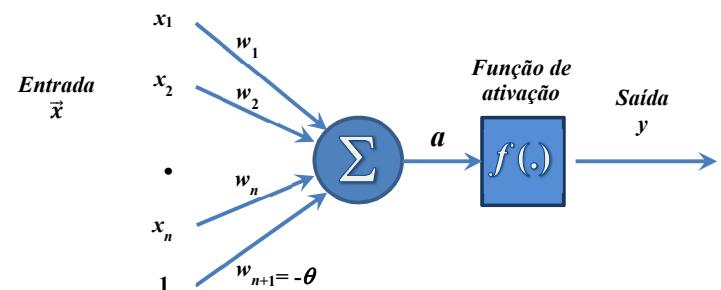


Figura 2 – Neurónio Artificial - UC

O modelo matemático do neurónio artificial (também designado por Unidade de Processamento) é:

$$y_k = f\left(\sum_{i=1}^n w_i x_i - \theta\right) \quad (1.1)$$

Sendo $\mathbf{x} = [x_1, x_2, \dots, x_n, 1]^T$ o vetor que agrupa as entradas da UP, e $\mathbf{w} = [w_1, w_2, \dots, w_n, -\theta]^T$ o vetor dos pesos, a função de transferência do neurónio é representada por:

$$y_k = f(\mathbf{w}^T \mathbf{x}) \quad (1.2)$$

Como se verificará, a função de integração terá impacto na forma como o espaço de entrada é repartido, efetivado pela função de ativação que gradua o nível da separação. Seja o exemplo apresentado da UP representado na figura 3. a), com função de integração linear, ou seja $a = \mathbf{w}^T \mathbf{x}$, e função de ativação degrau. O plano de pesos \mathbf{w} divide o espaço produto entrada/saída, $(\bar{x}, a) \in \mathbb{R}^{n+2}$, em duas regiões espaciais distintas, correspondente a, de um lado com valores positivo de $\mathbf{w}^T \mathbf{x} > 0$ e do outro com valor negativos, $\mathbf{w}^T \mathbf{x} < 0$ (ver figura 3. b). $\mathbf{w} = [w_1, w_2, w_3]^T$ é um vetor perpendicular a esse plano. Consequentemente, a reta de equação $\mathbf{w}^T \mathbf{x} = 0$ separa o espaço de entrada em duas sub-regiões abertas (ver figura 3.c). Para valores de entrada situadas em cada uma destas regiões, o neurónio reage atribuindo uma saída de valor distinto 0 ou 1, tarefa realizada pela função de ativação degrau, tal como mostra a figura 3.d).

Para valores da função de ativação, a , positivas ter-se-á como saída do neurónio o valor 1, enquanto para valores negativos de a terá um valor de saída 0. Deste modo a UP distingue as duas regiões do espaço de entrada, situadas de cada lado da reta separadora, sendo por isso de ser possível de ser usada como classificador um binário de classes disjuntas linearmente separáveis. A este neurónio é dado a designação de *Perceptron*.

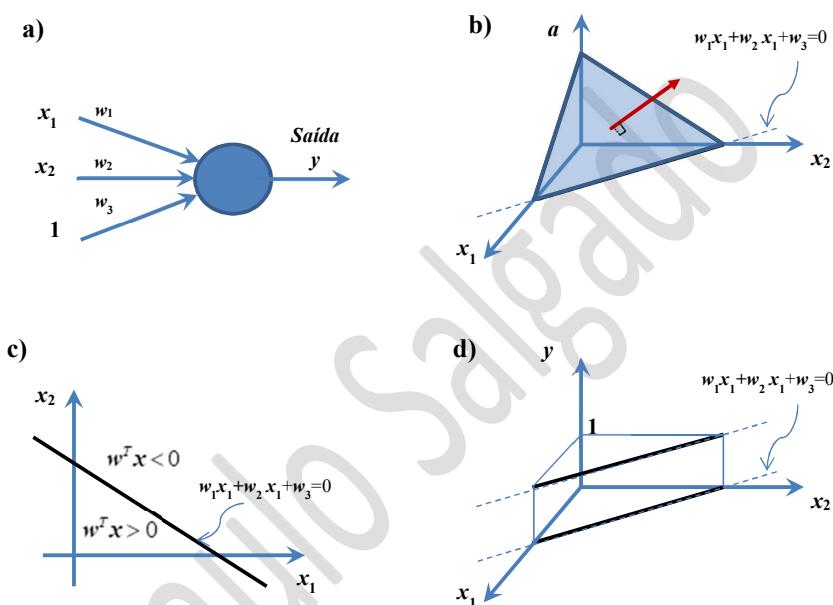


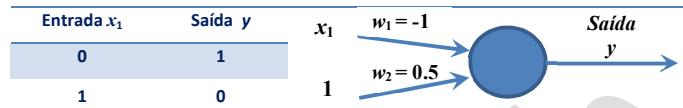
Figura 3 - (a) Neurónio com função de integração linear e ativação degrau; (b) linha separadora do espaço de entrada do neurónio; (c) Resposta da função de integração; (d) função de transferência do neurónio

O *Perceptron* é um classificador binário que mapa a entrada x (vector real) numa saída de valor binário:

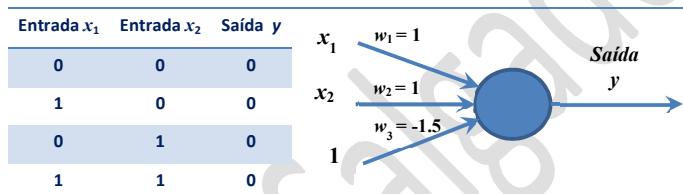
$$y = f(\mathbf{w}^T \mathbf{x}) = \begin{cases} 1 & \text{se } \mathbf{w}^T \mathbf{x} > 0 \\ 0 & \text{outros} \end{cases} \quad (1.3)$$

Exemplo 1.1: Para os neurónios representados nas figuras, com valores de pesos assinalados, verifique se implementam a função lógica assinalada, expressa pela tabela situada à sua esquerda. Considere que todos os neurónios possuem a função de transferência (1.1), com função de ativação, f , do tipo degrau (*Heaviside*).

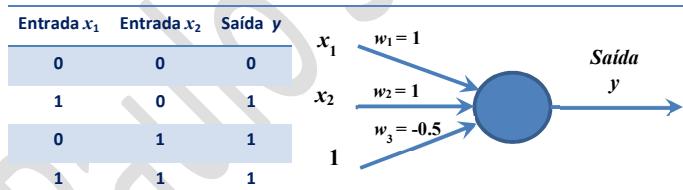
i) Operação Negação (NOT)



ii) Operação E (AND)



iii) Operação OU (OR)



Exercício 1.2: Para cada um dos neurónios do exercício anterior desenhe um gráfico onde represente a reta que divide o plano do espaço de entrada, assinalando em cada uma das partes do plano o valor de saída do neurónio. Assinale, em cada um dos gráficos, os valores da tabela lógica associada (com o símbolo “o” para valor 0 e ‘x’ para valor 1).

Exercício 1.3: Avalie a capacidade de um neurónio do tipo *Perceptron* ser capaz de implementar a função XOR.

Exemplo 1.1: Um estudo foi realizado para estabelecer a correlação entre o peso e o índice de massa corporal (IMC) de pacientes com três tipos de doenças associadas. Na Figura 4 estão representados todos os dados obtidos nesse estudo, sendo que as classes de doenças são separáveis linearmente uma das outras. Uma rede neuronal, com neurónios *perceptron*, será utilizada para classificar os três tipos de doença. Esta rede recebe como entrada o peso e o valor do IMC dos pacientes (2 entradas) e devolve como saída o tipo de doença, em representação Binária (com dois dígitos D_1 e D_0), número da doença em representação binária de acordo com a Tabela 1.

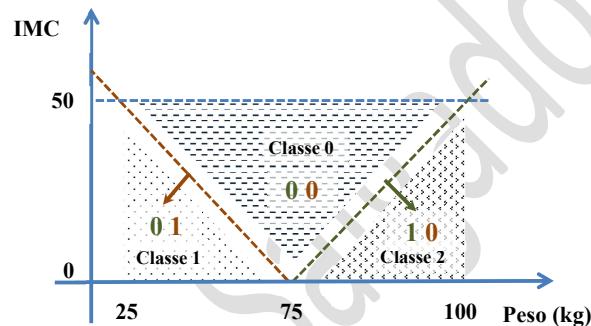


Figura 4- Gráfico da relação dos três tipos de doenças com as variáveis peso e IMC.

Tabela 1 – Tipos de Doenças

Decimal T	Binário $D_1 D_0$
0	0 0
1	0 1
2	1 0

Note-se que as classes são linearmente separáveis duas-a-duas. Mais se verifica que, o espaço associado a cada dígito de saída (D_0 e D_1) são linearmente separáveis, tal como se evidencia na figura 5, tarefa capaz de ser realizada de um *perceptron* por dígito, estrutura de rede neuronal representada na figura 6.

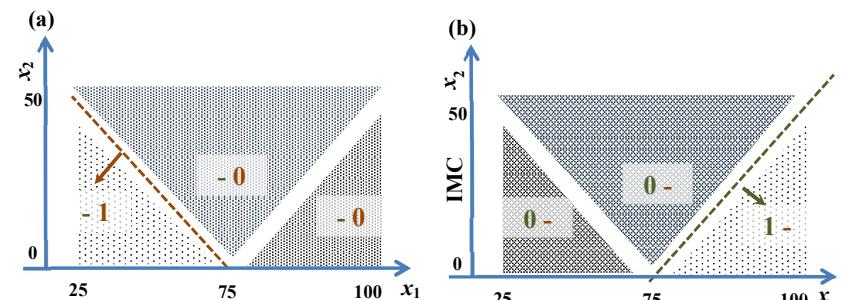


Figura 5: (a) Subespaços associado ao dígito D_0 ; (b) Subespaços associado ao dígito D_1 .

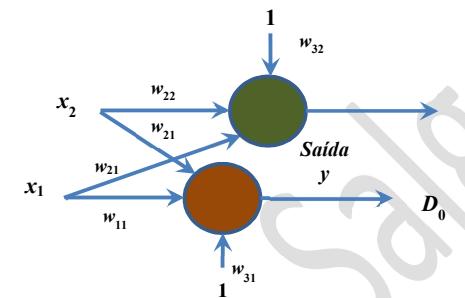


Figura 6- Rede Neuronal compostos por dois neurónios em paralelo, que conjugados fornecem o tipo de doença associado à entrada (x_1, x_2) , sendo D_0 e D_1 , respectivamente os dígitos binários da variável de saída.

Os pesos associados a cada um neurónio do dígito são os coeficientes das retas separadoras das regiões de valor 0 e 1. Assim ter-se-á para o dígito D_0 a reta separadora, $w_{11}x_1 + w_{21}x_2 + w_{31} = 0$, com vetor de pesos $\mathbf{w}_1 = [w_{11}, w_{21}, w_{31}] = [-2, -1, 100]$. Para o dígito D_1 tem-se $w_{12}x_1 + w_{22}x_2 + w_{32} = 0$, com pesos $\mathbf{w}_2 = [w_{12}, w_{22}, w_{32}] = [2, -1, -150]$.

Este último exemplo pode ser generalizado para um conjunto de c classes mutuamente exclusivas e linearmente separáveis, representadas por uma palavra digital com um número de dígitos, m , calculadas pela expressão $m=teto(\log_2(c))$, sendo que a função $teto(r)$ converte um número real r no menor número inteiro maior ou igual a r . Nesta circunstância ter-se-á para quaisquer par de classes, i e j , que $C_i \cap C_j = \emptyset$ e separáveis por um perceptron, cuja saída é será aqui designado por S_{ij} , com $i=1, \dots, c$ e $j > i$. O número máximo de neurónios exigível

para a separação completa será de $N = c(c-1)/2$, número de neurónios da camada intermédia, para uma camada de saída composta por m neurónios, um por cada dígito da palavra de saída. Assim, para um exemplo de $c=3$ classes ter-se-á uma rede constituída por 3 camadas. A camada de entrada terá dimensão igual ao do vetor de entrada, n , a segunda camada com $N=3$ neurónios e a camada de saída com $m = \text{teto}(\log_2(3)) = 2$ neurónios, tal como se esquematiza na figura 7. Para um número de $c=10$ classes ter-se-á no máximo $N=45$ neurónios na 2ª camada e $m=4$ neurónios na camada de saída, para um total número de pesos igual $nN + (N+1)m$.

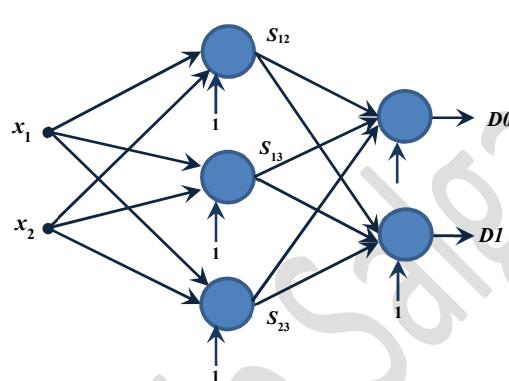


Figura 7: Exemplo da estrutura de um classificador de 3 classes, para um entrada de dimensão $n=2$.

(COLOCAR AQUI UM EXEMPLO E O POR O CODIGO MATLAB EM APENDICE).

3. Redes Neuronais Artificiais

As RNAs revelam grandes capacidades de reconhecimento e classificação de padrões. Presentemente são usadas em diversas tarefas em diferentes áreas da indústria e da ciência [].

As RNAs quando conjugadas com métodos de aprendizagem são capazes de partir de exemplos, aprendem e captam as relações funcionais entre os dados, mesmo que as relações subjacentes sejam desconhecidas ou difíceis de descrever. Assim, são indicadas para problemas onde as soluções requerem um conhecimento que é difícil precisar, mas para os quais existem dados suficientes [][], que incorporam as relações que se pretendem modelar. Esta abordagem de modelação juntamente com a capacidade de aprendizagem e experiência é muito útil para diversos problemas práticos, dado que muitas vezes é mais fácil obter dados de um sistema do que ter boas suposições teóricas sobre as leis que os regem.

Em segundo lugar, as RNAs podem generalizar. Depois identificarem, por meio de uma aprendizagem estrutural e paramétrica, as relações contidas nos dados de treino, as RNAs podem inferir respostas para dados de entrada nova.

Em terceiro lugar, as RNAs são aproximadores universais, pelo que possuem as capacidades, se dotadas de uma estrutura suficiente com número de neurónios adequado, de aproximar qualquer função contínua, para qualquer precisão desejada [][]. As RNAs têm formas funcionais mais flexíveis e eficazes do que os tradicionais métodos estatísticos.

Finalmente, as RNAs são modelos não-lineares. Qualquer modelo (de previsão) assume que existem relações entre as entradas e as saídas (previamente conhecidas ou desconhecida) e que estas podem ser descobertas e rescritas por relações matemáticas ou numéricos. Neste capo, os modelos estatísticos lineares têm dominado a área da modelação, englobando-se neste domínio o controlo clássico e as técnicas clássicas de processamento de sinal. A título de exemplo, para a previsão de séries temporais por modelos a Box-Jenkins ou método ARIMA [] assumem que a série temporal em estudo configure relações lineares entre os valores presentes das variáveis com os seus valores passados. Os modelos lineares, por possuírem uma estrutura matemática mais simples, têm a vantagem de poderem ser analisados por técnicas bem estabelecidas, que facilitam a sua generalização e utilização em variadíssimos géneros de problemas. No entanto, podem ser totalmente inadequados se o mecanismo subjacente apresentar relações não-lineares. Na verdade, a maioria dos sistemas do mundo real são do tipo não-lineares [], apresentando apenas por vezes zonas do seu espaço de trabalho onde as relações entre as suas variáveis são lineares ou aproximadamente lineares.

Na verdade, a formulação de um modelo não-linear para um conjunto de dados em particular é uma tarefa muito difícil, porquanto podem existir muitos padrões não-lineares e um modelo não-linear pré-definido pode não ser suficiente para capturar todas as características importantes. As RNAs, pela sua estrutura e funções, apresentam uma relação de transferência não-lineares, permite ser adaptada por processos de identificação orientadas a dados de treino, sem que tenha *à priori* conhecimento sobre as relações entre as variáveis de entrada e saída.

O algoritmo *Backpropagation* surge em 1986 [1] com um primeiro método eficaz permite identificar Redes Neuronais multicamada por aprendizagem dos dados de treino. Werbos [2] formula o primeiro *Backpropagation* e descobre que as RNAs com este algoritmo de treino superaram os tradicionais métodos estatísticos, como a regressão e abordagens Box-Jenkins. Outras variantes deste método são hoje utilizadas na aprendizagem das redes neuronais, permitindo a utilização das RNAs em diferentes domínios e aplicações reais.

4. Uma visão geral das RNAs

As RNAs são compostas por uma série de elementos básicos de processamento interligados, chamados de neurónios ou nós. Cada nó recebe um sinal de entrada constituído pela "informação" total de outros nós ou estímulos externos, processa-os localmente com recurso a uma função de ativação ou transferência e produz um sinal de saída, encaminhado para outros nós ou saídas externas. Apesar de individualmente o neurónio (elemento de processamento) por si ter fraco poder de cálculo ou processamento, coletivamente uma rede, composta pela interligação destes elementos, pode executar eficientemente um número surpreendente de tarefas. Essa característica faz com que o processamento da informação das RNAs seja uma poderosa ferramenta computacional capaz de aprender com exemplos, reproduzir essa informação e generaliza-la a novas situações.

3.3.1. Tipo de RNAs

Desde 1980, têm sido propostos muitos géneros de estruturas de RNAs. Do ponto de vista da topologia são basicamente implementadas em dois tipos:

- As RNAs sem realimentação (Feedforward Neural Network) são constituídas habitualmente por camadas de neurónios, que estão ligados para que o sinal se encaminhe da entrada para a saída sem ligações laterais ou de realimentação.

- As RNAs com realimentação (Feedback Neural Network) incluem, pelo menos, uma ligação entre neurónios da mesma camada, ou uma ligação de uma camada mais próxima da saída para uma camada precedente.

A Figura 8 representa alguns exemplos de topologias de RNAs. As redes *Hopfield* [3], funcionam como uma memória associativa, podendo recordar um exemplo de uma versão parcial ou distorcida. Redes de *Hopfield* não são compostas em camadas com a interconexão completa entre nós. As saídas da rede não são necessariamente as funções de entrada, contrariamente, são estados estáveis de um processo iterativo.

O recurso a mapas de *Kohonen* é inspirado pelo comportamento de auto-organização do cérebro humano [4]. Este modelo utiliza uma estratégia de aprendizagem de *clustering* não supervisionada. A rede acede a cada registo e atribui-o a um determinado grupo através da similaridade existente com outros registos desse mesmo grupo.

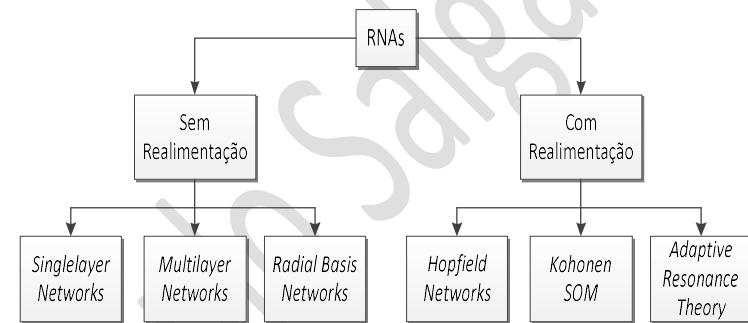


Figura 8 - Tipos de RNAs

As redes *Multi-Layer Perceptron* (MLP) são usadas em vários problemas, especialmente na previsão devido à inerente capacidade de mapeamento arbitrário de entrada-saída. Os leitores devem estar cientes de que outros tipos de RNAs, tais como redes de função de base radial (RBF) [5][6], redes *wavelet* [7], entre outras, também são muito úteis em algumas aplicações, devido à sua função de aproximação.

Uma MLP é geralmente composta por uma camada de entrada, uma camada de saída e por um número variável de camadas intermédias, também conhecidas por camadas escondidas. As camadas escondidas apresentam um número diferenciado de neurónios, enquanto o número de neurónios a incluir nas camadas de entrada e de saída estão diretamente relacionadas com a natureza do sistema dinâmico. A Figura 9 dá um exemplo de uma MLP totalmente conectada com uma camada escondida.

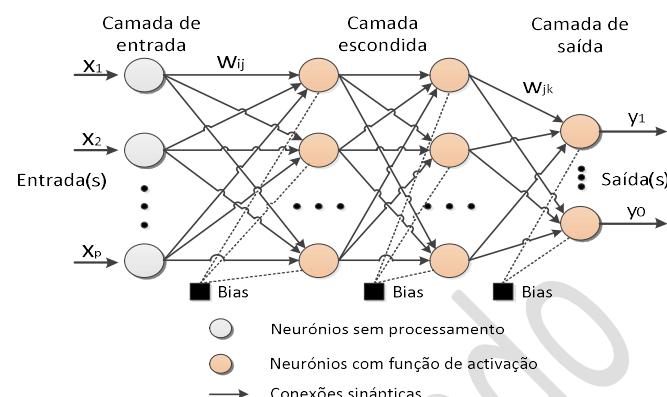


Figura 9- Rede Neuronal Multicamadas

Para um problema de modelação causal, as entradas de uma RNA são geralmente as variáveis independentes. A relação funcional estimada pela RNA pode ser escrita como:

$$y = f(x_1, x_2, \dots, x_p), \quad (3.2)$$

Onde x_1, x_2, \dots, x_p são p variáveis independentes e y é uma variável dependente. Nesse sentido, a rede neuronal é funcionalmente equivalente a um modelo não-linear. Por outro lado, para um problema extrapolativo ou previsão de séries temporais, as entradas são tipicamente as observações passadas da série de dados e a saída é o valor futuro. A RNA executa o mapeamento da seguinte função:

$$y_{t+1} = f(y_t, y_{t-1}, \dots, y_{t-p}), \quad (3.3)$$

onde y_t é a observação no tempo t . Assim, a RNA é equivalente ao modelo auto-regressivo não-linear para problemas de previsão de séries temporais.

5. Estrutura da RNA

O projeto da estrutura e dos mecanismos de aprendizagem de uma rede neuronal para um problema particular de previsão não é uma tarefa trivial. Existem vários factores que afetam o desempenho de uma RNA devem ser cuidadosamente considerados. Uma decisão crucial é determinar a estrutura apropriada, ou seja, o número de camadas, o número de neurônios em cada camada, bem como o número de conexões que interligam os neurônios. Outras decisões

de projeto da rede incluem a seleção de funções de ativação dos neurônios na camada escondida e de saída, o algoritmo de treino da transformação de dados ou métodos de normalização, conjuntos de treinos e teste, e medidas de desempenho. Ao projetar uma MLP, é necessário determinar as seguintes variáveis:

- Número de neurônios de entrada.
- Número de camadas escondidas e neurônios.
- Número de neurônios de saída.

A seleção destes parâmetros é basicamente dependente do problema. Embora existam várias abordagens diferentes, nenhum método pode garantir a solução ideal para todos os problemas de previsão real, nem existe nenhum método que indique o valor estritamente necessário desses parâmetros. Em termos práticos estas respostas são obtidas por heurísticas ou baseadas nos resultados de simulações de experiências realizadas.

O número de neurônios da entrada corresponde ao número de variáveis utilizadas no vector de entrada para previsão dos valores futuros. Para a previsão causal, o número de entradas é imposto pelo problema. Num problema de previsão de séries temporais, o número de neurônios da entrada corresponde ao número de observações desfasadas usadas para descobrir o padrão subjacente numa série temporal e para fazer previsões dos valores futuros, isto é a ordem da série. Como tal não é fácil determinar esse número. Porém um valor apropriado é essencial para o modelo ser capaz de incorporar as características originais dos dados e ou a dinâmica da série. Muito ou pouco número de neurônios de entrada pode afetar a aprendizagem ou a capacidade da previsão da rede. Para problema de previsão de séries temporais a sua determinação é essencial para que o modelo possa ter acesso a informações importantes sobre a estrutura complexa das relações (linear e/ou não-linear) dos dados.

O número de camadas escondidas e o número dos neurônios são parâmetros quase sempre de livre escolha, mas com impacto no desempenho da rede neuronal. Cada camada da rede mapeia o espaço de entrada num outro espaço de saída, com o detalhe que o número de neurônios utilizados lhe confere. Tal confere às redes neuronais a capacidade de capturar o padrão dos dados e realizar complicados processos de mapeamento não-linear entre as variáveis de entrada e saída. A maioria dos trabalhos mostram que uma única camada escondida é suficiente para as RNAs aproximar qualquer função complexa não-linear com a precisão desejada [1][2]. No entanto, uma camada escondida pode exigir um número muito elevado de neurônios, o que não é desejável, porque faz com que a rede necessite de um elevado tempo de treino e métodos de aprendizagem mais robustos. O número excessivo de camadas e neurônios pode ainda conduzir a uma perca capacidade de generalização da rede neuronal.

A questão da determinação do número apropriado de neurónios na camada escondida é crucial, mas de difícil determinação. Em geral, redes com poucos neurónios são preferíveis a redes com muitos neurónios uma vez que geralmente apresentam melhor capacidade de generalização e menor problema de *overfitting*. A Figura 10 mostra, recorrendo a três exemplos, este problema. Redes com um número reduzido de neurónios podem não ser capazes de aprender adequadamente o comportamento da série de dados (*underfitting*).

A maneira mais comum para determinar o melhor número de neurónios é através da realização de experiências, frequentemente no modo tentativa-erro. O nosso conhecimento sobre a complexidade das relações a modelizar ou o número de padrões de entrada e a sua separabilidade são sempre bons indicadores sobre o número de neurónios da camada escondida necessários. Para evitar o problema de *overfitting*, alguns investigadores têm fornecido regras empíricas para restringir o número de neurónios.

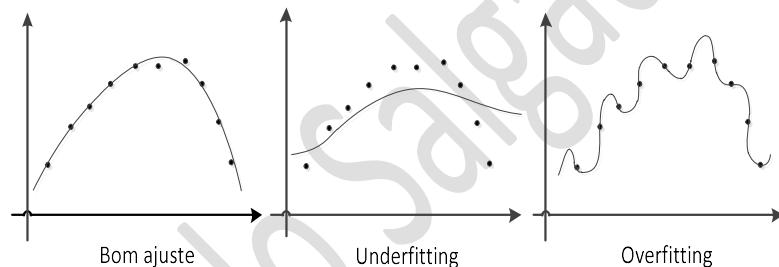


Figura 10 – Resultados de ajuste

O número de neurónios da saída é imposto pelo problema em estudo, sendo igual à dimensão do espaço de saída. No problema da previsão de séries temporais, o número de neurónios da saída está relacionado com o horizonte da previsão.

A arquitetura de rede só fica completa quando se estabelecem as interconexões entre neurónios das camadas. Estas determinam o comportamento fundamental da rede. Na maioria das aplicações, as redes são totalmente conectadas, estando todos os neurónios de uma camada conectados a todos os neurónios da camada seguinte, com à exceção da camada da saída. Porém é possível ter redes de baixa conexão [] ou incluir ligações diretas dos neurónios da entrada para os neurónios da saída []. Adicionar as ligações diretas entre a camada da entrada e a camada da saída pode ser vantajosa para determinados problemas, uma vez que pode enriquecer a capacidade de modelação da rede.

6. Funções de Integração e Ativação

A função de integração tem como tarefa agrregar todos os estímulos que chegam à entrada do neurónio. Esta tarefa é geralmente precedida pela multiplicação de pesos com cada uma das entradas. Estes pesos permitem atribuir diferentes graus de contribuição das entradas, podendo este ser do tipo excitador (para pesos positivos) ou inibidor (para pesos negativos).

Esta função de integração bem como os pesos associados assume um papel fundamental na transformação não linear do neurónio. De um modo geral esta função permitirá à função de ativação, que será introduzida em seguida, dividir o espaço de entrada em duas regiões distintas, tomando em conta regiões de valores positivos e negativos da função de integração. A informação localmente retida por um neurónio está fortemente associada aos valores dos pesos e da forma da função de ativação escolhida. Uma parte substancial da aprendizagem estrutural e paramétrica do neurónio concentra-se na seleção adequada da função de Integração e dos valores dos pesos a ela associada. Na Tabela 2 são apresentadas as principais funções de integração usadas em aplicações de redes neurais artificiais, com destaque, pela sua simplicidade, a função linear. A tarefa da função terminal do neurónio, convencionalmente referida como função de ativação, será fazer uso do valor fornecido pela função de integração, amplificando-o ou inibi-lo, gerando um sinal de saída geralmente compreendido entre dois patamares de saturação extremos.

Tabela 2 - Funções de Integração

Nome	Função
Linear	$a = \sum_{i=1}^n w_i x_i - \theta$
Esférica	$a = \rho^{-2} \sum_{i=1}^n (x_i - w_i)^2 - \theta^2$
Quadrática	$a = \sum_{i=1}^n w_i x_i^2 - \theta$
Polinomial ou sigma-pi ($\Sigma \pi$)	$a = \sum_{i=1}^n \sum_j w_j x_i x_j + x_i^{\alpha_i} + x_j^{\alpha_j} - \theta$

A função de ativação é também chamada de função de transferência. Determina a relação entre entradas e saídas de um neurónio. As funções de ativação de um neurónio artificial são funções não-lineares e saturáveis. Dentro do conjunto de funções candidatas apenas um pequeno número de funções são escolhidas para a generalidades dos casos, destacando-se pela preferência a função transferência sigmoide e tangente hiperbólica pelo facto de serem funções

deriváveis. A Tabela 3 resume as principais funções de ativação, com representações gráficas representadas na Figura 11. Com exceção da função de ativação linear, geralmente utilizada nos neurónios da camada de saída das redes neuronais, todas as outras funções são fortemente não lineares e limitadas a valores imagem no intervalo unipolar $[0, 1]$ ou bipolar $[-1, 1]$. Dentro destas, destacam-se ainda funções com características de continuidade e derivabilidade, como sejam as funções sigmoide ou Tangente Hiperbólica. Estas funções são dependentes do parâmetro λ , cujo valor modifica a forma como se processa a transição entre os limites extremos saturados. Acresce-se ainda o facto de o cálculo da derivada dessas funções ser facilmente determinado, recorrendo ao valor de saída do neurónio.

Tabela 3 – Funções de Ativação $y = f(\cdot)$

Nome	Função	Nome	Função
Heaviside	$\begin{cases} 1 & ; x \geq 0 \\ 0 & ; x < 0 \end{cases}$	Sigmoide	$y = \frac{1}{1 + e^{-\lambda x}}$
Heaviside Simétrico ou sinal	$\begin{cases} 1 & ; x \geq 0 \\ -1 & ; x < 0 \end{cases}$	Sigmoide bipolar	$y = \frac{1 - e^{-\lambda x}}{1 + e^{-\lambda x}}$
Linear	$y = x$	Tangente Hiperbólica	$y = \frac{e^{\lambda x} - e^{-\lambda x}}{e^{\lambda x} + e^{-\lambda x}}$
Linear com Saturação	$\begin{cases} 0 & ; x < 0 \\ x & ; 0 \leq x \leq 1 \\ 1 & ; x > 1 \end{cases}$	Gaussiana	$y = e^{-\lambda x^2}$
Linear simétrico Com Saturação	$\begin{cases} -1 & ; x < -1 \\ x & ; -1 \leq x \leq 1 \\ 1 & ; x > 1 \end{cases}$		

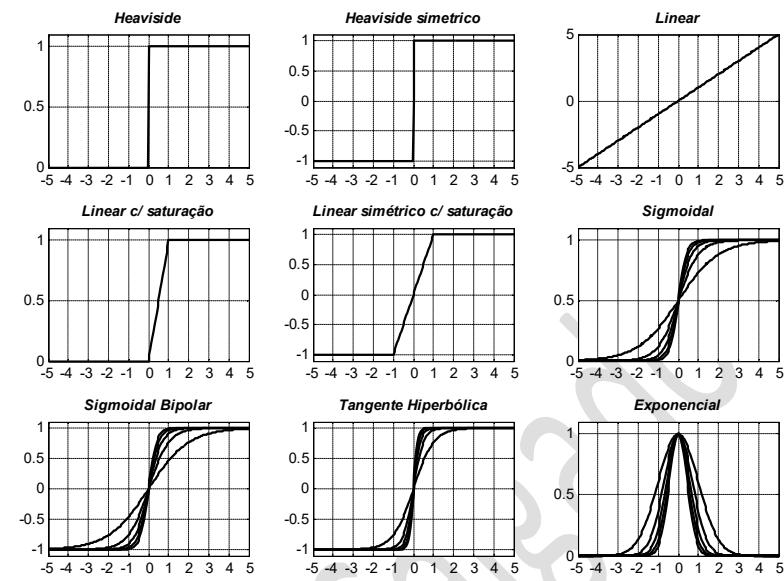


Figura 11: Funções de ativação: (a) Heaviside ou degrau; (b) Heaviside simétrico ou “signal”; (c) linear; (d) Linear com saturação; (e) Linear simétrico com saturação; (f) Sigmoidal; (g) Sigmoidal bipolar; (h) Tangente hiperbólica; (i) Exponencial.

Exemplo: A derivada da função sigmoide $f(a) = 1/(1 + e^{-\lambda a})$ será $f'(a) = \lambda e^{-\lambda a} / (1 + e^{-\lambda a})^2$ ou ainda $f'(a) = \lambda((1 + e^{-\lambda a}) - 1) / (1 + e^{-\lambda a})^2$. Manipulando este último resultado ter-se-á que: $f'(a) = \lambda(1/(1 + e^{-\lambda a}) - 1/(1 + e^{-\lambda a})^2)$. Finalmente temos que: $f'(a) = \lambda f(a)(1 - f(a))$. Com este resultado torna-se fácil determinar a derivada da função de ativação sigmoide uma vez que é conhecido o valor dessa função sigmoide.

Geralmente, uma rede pode ter neurónios com diferentes funções de ativação dentro de uma camada ou camadas diferentes [1]. Contudo, quase todas as redes utilizam as mesmas funções de ativação particularmente para os neurónios da mesma camada. Frequentemente, a camada de saída da rede pode ter funções de ativação lineares.

Exemplo 1.2.: Neurónio com função de integração esférica

Neste exemplo, contrariamente aos exemplos anteriores, o neurónio utiliza uma função de integração de integração esférica.

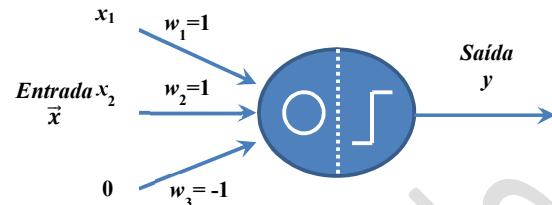


Figura 12. Neurónio com função de integração circular e função de activação heaviside.

A forma de integração dos sinais de entrada é agora realizada de modo diferente. A cada variável de entrada x_i é-lhe subtraido o valor do peso da ligação, w_i . A integração desses valores efetua-se pela soma dos quadrados dos seus valores, retirado da quantidade do valor do *thresholder* ao quadrado, isto é $a = \sum_{i=1}^n (x_i - w_i)^2 - \theta^2$. Em termos geométricos esta função corresponde a uma superfície parabólica, com cavidade voltada para cima e mínimo no ponto de (w_1, w_2) , cuja intersecção com o plano (x_1, x_2) é uma linha circular de raio θ . A região do espaço interior a este círculo o valor da função de integração é negativo e fora assume valores positivos (isto é, $a > 0$). Porém se houver uma troca de sinal na função integradora a região interior ao círculo será positiva e a exterior negativa. Deste modo a função utilizada neste neurónio foi $a = \text{sign}(\theta) \cdot \sum_{i=1}^n (x_i - w_i)^2 - \theta^2$, onde θ é uma quantidade negativa.

A função de transferência deste neurónio é: $y = \text{Heaviside}\left(\text{sign}(\theta) \cdot \sum_{i=1}^n (x_i - w_i)^2 - \theta^2\right)$, cuja representação gráfica está representada na Figura 13.

Se houver mudança da função de ativação *Heaviside* para *Sigmoidal*, cuja função apresenta características de continuidade e derivabilidade, a resposta do neurónio terá um comportamento mais suave, dado lugar a uma região de transição gradual em redor da linha circular. A Figura 14 mostra a função de transferência deste neurónio, sendo que agora existe uma transição contínua e suavizada entre as regiões de nível 0 (exterior ao círculo) e nível 1 (interior ao círculo).

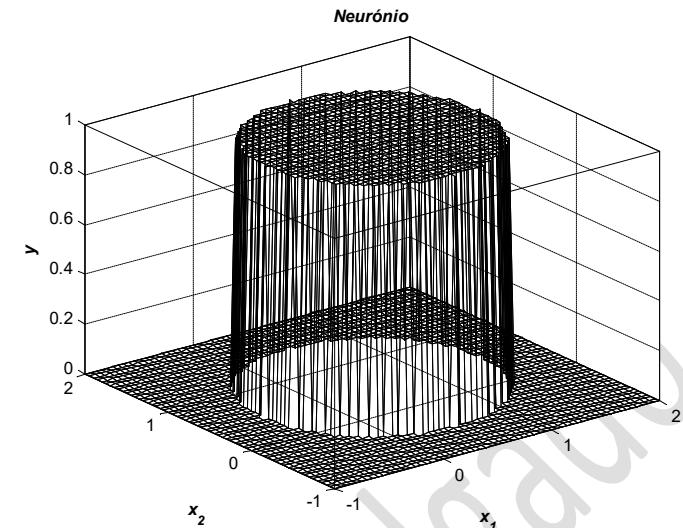


Figura 13. Resposta do neurónio, com função de integração esférica e função de ativação Heaviside.

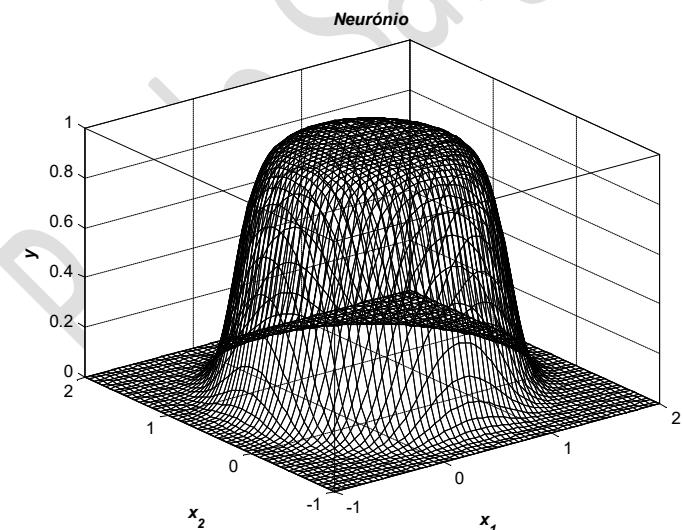


Figura 14. Resposta do neurónio, com função de integração esférica e função de ativação Sigmoidal, com $\lambda=5$.

Exemplo 1.3: (Exemplo dado na aula, com implementação em MATLAB)

Rotina: ex2_aula_28Mar2017.m

Na Figura 15 está representado uma rede neuronal de três camadas, compostas por neurónios com função de integração linear e funções de activação Heaviside. Os pesos de interligação que ligam os vários neurónios estão igualmente assinalados na figura.

Como visto na secção 1, cada neurónio da camada intermédia, divide o espaço de entrada em dois semi-planos. Para valores de entrada situado num dos semi-plano ter-se-á um valor de saída de +1 e para o outro semi-plano o valor de 0. Na figura 14 estão representadas as linhas divisoras dos semiplano em \mathbb{R}^2 , apontando as setas para as regiões de saída positivas (i.e., +1), e nas figuras 15 a) a c) a função de transferência de cada um desses neurónios. Os parâmetros destas rectas separadoras estão, deste modo, relacionados com os pesos de entrada de cada neurónio. O valores de saída destes neurónios, agora agrupados no vector $\vec{S} = [S_1, S_2, S_3]^T$, são as variáveis de entrada do neurónio da camada de saída, que divide o espaço da variável $\vec{S} \in \mathbb{R}^3$ em duas regiões linearmente separáveis. Em face aos pesos utilizados, a função integradora soma com iguais pesos as variáveis S_1 a S_3 . Na figura 15 d) está representada a superfície que relaciona as variáveis de entrada da rede e os valores da função de ativação deste neurónio. Para valores superiores ao valor de *thresholder* de 2,5 a saída do neurónio (e portanto da rede), será +1 e nos outros casos de 0. Como tal, neste exemplo, somente dentro do triângulo definido pelas três rectas da figura 13 se terá um valor de integração positivo ao qual corresponderá uma saída +1. A função de transferência desta rede neuronal pode ser vista na figura 16, na forma de um prisma triangular.

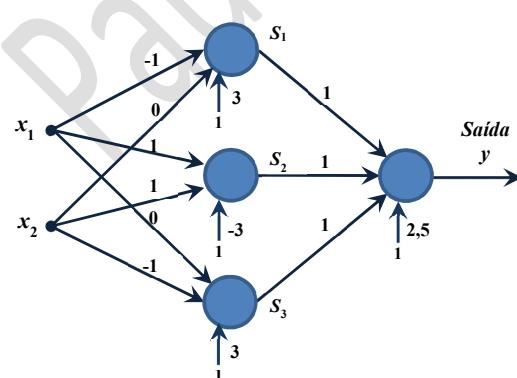


Figura 15. Estrutura da rede neuronal com três camadas e respectivos pesos.

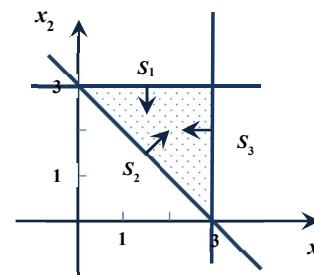


Figura 16. Linhas separadoras realizadas pelos três neurónios da camada escondida. Cada uma das setas associadas indica o lado positivo dessa divisão do espaço.

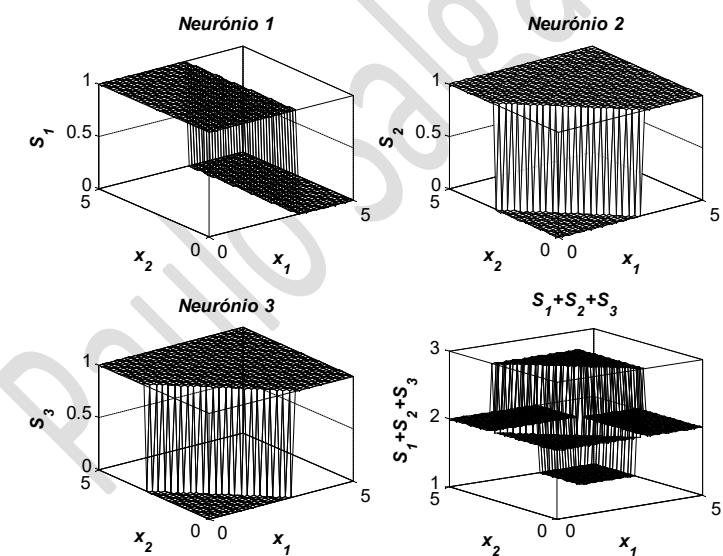


Figura 17. a) a c) Função de transferência dos três neurónios da camada escondida. d) Soma das três saídas dos neurónios da camada escondida, realizada com pesos iguais e unitários.

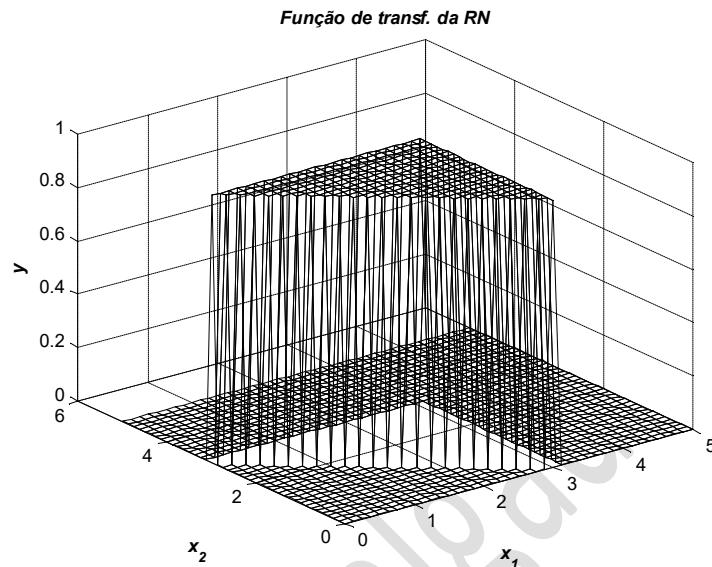


Figura 18. Função de transferência da rede neuronal

Se as funções de ativação da rede neuronal da figura 13 fossem alteradas para funções sigmoídes, as superfícies de separação entre duas regiões, por cada um dos neurónios seria agora não abrupta, derivável e monótona. Nestas circunstâncias todas as superfícies das figuras 15 e 16, serão agora mais suaves, regulados pelo parâmetro λ da função sigmoide (ver figuras 17). O resultado final da rede neuronal passará agora a ser um superfície continua e derivável, na forma de um prisma triangular mas com as zonas da arestas arredondadas, tal como se observa na figura 18, para um valor de $\lambda=5$. Para valores maiores de λ a superfície de transferência da rede aproxima-se da que era obtida por funções de ativação Heaviside, tal como se comprava da observação da figura 19, para um valor $\lambda=10$, mas ainda com condições de continuidade e derivabilidade das funções da rede.

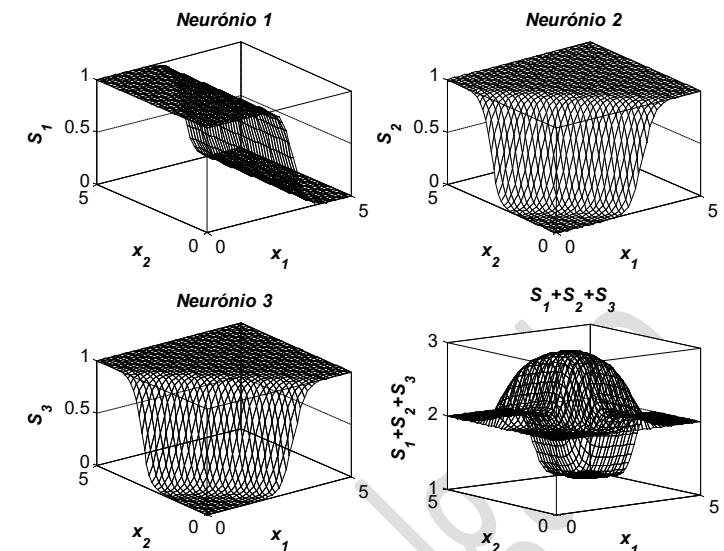
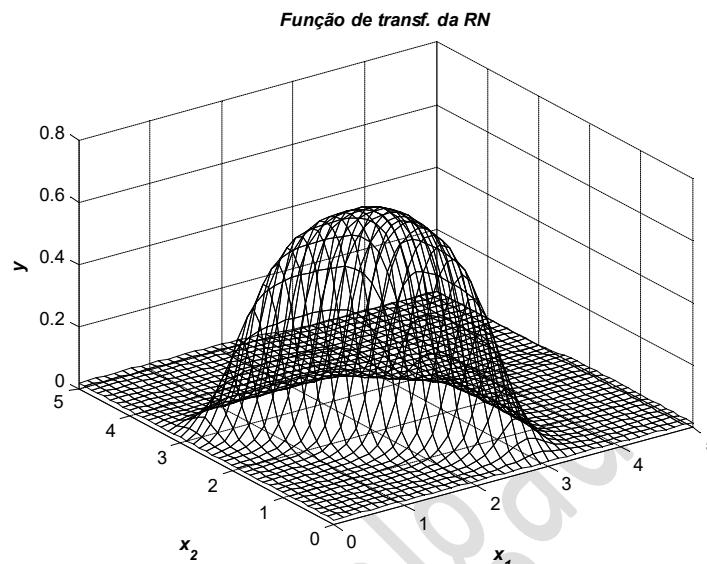
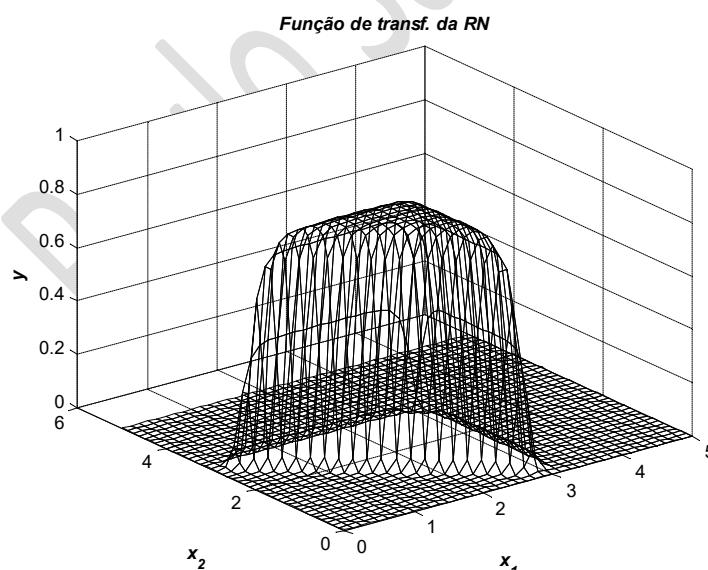


Figura 19. a) a c) Função de transferência dos três neurónios da camada escondida, com funções de ativação sigmoide. d) Soma das três saídas dos neurónios da camada escondida, realizada com pesos iguais e unitários.

Figura 20. Função de transferência da rede neuronal com funções de activação sigmoide (para $\lambda=5$).Figura 21. Função de transferência da rede neuronal com funções de activação sigmoide (para $\lambda=10$).

7. Aprendizagem

Antes de uma RNA poder ser usada para executar uma qualquer tarefa desejada, deve ser treinada. Basicamente, o treino é um processo iterativo de determinação dos pesos das conexões dos neurónios que compõe a RNA, de forma a obter-se a resposta adequada para um dado problema. Uma possibilidade é atribuir pesos aleatórios iniciais às conexões usando um conhecimento prévio, outra é treinar a rede, usando uma regra de aprendizagem, de maneira que assimile um determinado comportamento através de exemplos. Entre as principais regras de aprendizagem estão:

- Aprendizagem supervisionada, em que a rede é treinada recebendo um conjunto padrão de entradas e respetivas saídas, fornecidas por um “professor”. Os pesos sinápticos são ajustados iterativamente, até que o erro entre os padrões de saída gerados pela rede seja o desejado (ver [1]).
- Aprendizagem não supervisionada, onde a rede é capaz de descobrir aspectos importantes nos dados de treino. Não necessita de uma supervisão do “professor”. Este tipo é semelhante às técnicas de *Clustering* utilizadas para classificação das “curvas de carga”.

Uma formação MLP é supervisionada, em que a resposta desejada da rede (valor alvo) para cada padrão de entrada (por exemplo) à partida é sempre conhecida.

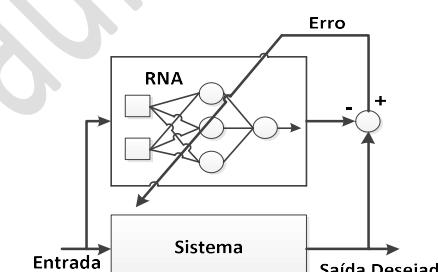


Figura 22 – Aprendizagem supervisionada

8. Algoritmos de treino

Os dados de entrada dos algoritmos de treino têm a forma de vectores, constituídos por variáveis de entrada ou de padrões de treino. Cada elemento de um vector de entrada

corresponde a um nó de entrada na camada de entrada da rede. Portanto, o número de nós de entrada é igual à dimensão dos vectores de entrada. Para um problema de previsão causal, o número de nós de entrada está bem definido e é o número de variáveis independentes associadas ao problema. No entanto, para um problema de previsão de séries temporais, o número adequado de nós de entrada não é fácil de determinar. Seja qual for a dimensão, o vector de entrada para um problema de previsão de séries será quase sempre composto por uma janela móvel de tamanho fixo ao longo da série. O total de dados disponíveis é normalmente dividido num conjunto de treino (dados dentro da amostra) e um conjunto de teste (fora da amostra, ou amostra em espera). O conjunto de treino é usado para estimar os pesos enquanto o conjunto de teste é utilizado para medir a capacidade de generalização das redes.

O processo de treino funciona geralmente da seguinte forma. Inicialmente, os exemplos do conjunto de treino são inseridos nos nós de entrada. Os valores de ativação dos nós de entrada são ponderados e acumulados em cada nó na primeira camada escondida. O total é então transformado por uma função de ativação no valor do nó de ativação. Por sua vez torna-se um contributo para nós na próxima camada, até que os valores de ativação de saída são encontrados. O algoritmo de treino é usado para encontrar os pesos de modo a minimizar uma medida do erro global como a soma dos erros quadrados (SSE) ou erro quadrático médio (MSE). Assim, o treino da rede é realmente um problema de minimização sem restrições não-lineares.

Para um problema de previsão de séries temporais, o treino padrão consiste num número fixo de observações desfasadas da série. Suponha que temos N observações y_1, y_2, \dots, y_n no conjunto de treino e temos um passo à frente de previsão, em seguida, usando uma RNA com n nós de entrada, temos $N-n$ padrões de treino. O primeiro padrão de treino será composto de y_1, y_2, \dots, y_n como entradas e y_{n+1} como a saída de destino. O segundo padrão de treino conterá y_2, y_3, \dots, y_{n+1} como entradas e y_{n+2} como a saída desejada. Finalmente o último padrão de treino será $y_{N-n}, y_{N-n+1}, \dots, y_{N-1}$ para entradas e y_N para o destino. Tipicamente, uma função objetivo baseada em SSE ou função de custo a ser minimizada durante o processo de formação é:

$$E = \frac{1}{2} \sum_{i=n+1}^N (o_i - y_i)^2 \quad (3.4)$$

Onde o_i é a saída real da rede e y_i a saída, obtida a cada iteração de treino.

Dada o problema de identificação paramétrica de uma RNA envolver a pesquisa de um número elevado de parâmetros e este ser ainda de carácter não linear não há garantia o método

escolhido dê como garantido uma solução global, pois frequentemente estes ficam presos em mínimos locais. Como tal, todos os algoritmos de otimização, na prática, inevitavelmente, sofrem de problemas de ótimos locais e, o máximo que se pode fazer é usar o método de otimização disponível que possa encontrar os melhores ótimos locais, se a verdadeira solução global não está disponível.

O método de treino mais popular é o algoritmo *Backpropagation*.

9. Algoritmos de retropropagação

O algoritmo retropropagação (designado em literatura científica de língua Inglesa como Backpropagation Learning Algorithm) treina uma rede neuronal multicamada com recurso a um conjunto de valores de entrada para os quais são conhecidos os seus valores imagens. Primeiramente, são determinadas a resposta da rede neuronal para cada um dos valores de entrada. A resposta de saída é então comparada com a saída desejada e um valor de erro é calculado. Por fim, com a informação do erro os pesos da ligação da rede serão ajustados no sentido de reduzir o erro global obtido. O conjunto destes padrões de amostra de treino é repetidamente apresentado à rede e todo o processo anterior executado, até o valor de erro ser minimizado.

Na Figura 23 está representado a estrutura de uma rede multicamada com M camadas, com N_0 neurónios de entrada e N_M na camada de saída. N_j representa o número do neurónio da camada j . Um conjunto de dados de treino com valores de entrada $X_i = \{x_1, x_2, \dots, x_k, \dots, x_n\}$, sendo cada elemento um vector de dimensão $N_0 \times 1$, e os valores de saída conhecidos como $D_o = \{d_1, d_2, \dots, d_k, \dots, d_n\}$ são apresentados em cada iteração do processo de treino. A resposta da rede neuronal é $Y_o = \{y_1, y_2, \dots, y_k, \dots, y_n\}$, onde y_k é o vector, de dimensão $N_M \times 1$, resposta da rede neuronal para o padrão de entrada x_k . Seja $Y_{ij}(k)$ a saída do neurónio i da camada j para o padrão k ; W_{jih} o peso de ligação do neurónio h da camada $(j-1)$ para o neurónio i da camada j e δ_{ji} o valor de erro associado ao i^{esimo} neurónio na camada j .

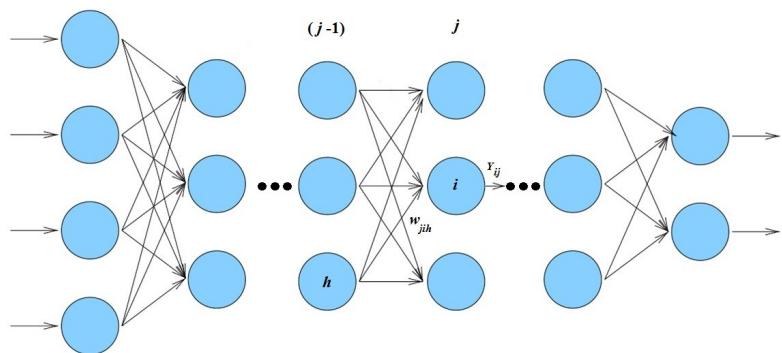


Figura 23. Rede Neuronal Multicamada (Feedforward).

Em resumo, este algoritmo compreende os seguintes passos [BJ91]:

1. Atribuir aos pesos de ligação valores baixos e aleatórios.
2. Apresentar à rede o vector amostra de treino k : à entrada ter-se-á o vector $\mathbf{x}_k = [x_{k1}, x_{k2}, \dots, x_{kN_0}]^T$ e à saída o correspondente vector $\mathbf{d}_k = [d_{k1}, d_{k2}, \dots, d_{kN_M}]^T$.
3. Passar os valores de entrada para a 1ª camada. Assim ao nó i de entrada ser-lhe-á atribuído o valor x_{ki} :

$$Y_{0i} = x_{ki}, \text{ para } i=1, \dots, N_0$$

4. Calcular a saída cada neurónio i da camada $j = 1, 2, \dots, M$, processando da camada de entrada para a de saída:

$$Y_{ji} = a \left(\sum_h^{N_{j-1}} W_{jih} Y_{(j-1)h} \right)$$

onde a é a função de ativação, sendo que frequentemente se utiliza a função sigmoide por esta apresentar propriedades interessantes, tais como de continuidade e de derivabilidade, $a(x) = 1/(1 + \exp(-\lambda x))$.

5. Obter o vector de saída da rede: $\mathbf{y}_k = [Y_{M1}, \dots, Y_{Mi}, \dots, Y_{MN_M}]^T$
6. Calcular o valor de erro δ_{ji} , para cada neurónio i da camada j , começando no sentido da última camada para a camada de entrada, isto é $j = M, M-1, \dots, 2, 1$. Para a camada de saída o erro será dado por:

$\delta_{Mi} = a'_{Mi}(\cdot)(d_{ki} - Y_{Mi})$, onde a'_{Mi} é a função derivada da função de ativação do neurónio i . Se a é uma função sigmoide então $\delta_{Mi} = Y_{Mi}(1 - Y_{Mi})(d_{ki} - Y_{Mi})$.

e para as camadas escondidas tem-se:

$$\delta_{ji} = a'_{ji}(\cdot) \sum_{h=1}^{N_{j+1}} W_{(j+1)ih} \delta_{(j+1)h}$$

ou simplesmente $\delta_{ji} = Y_{ji}(1 - Y_{ji}) \sum_{h=1}^{N_{j+1}} W_{(j+1)ih} \delta_{(j+1)h}$, para funções de ativação sigmoides.

7. Adaptar os pesos que ligam os neurónios entre sucessivas camadas:

$$W_{jih} \leftarrow W_{jih} + \beta \delta_{ji} Y_{ji}, \text{ para } j = M, \dots, 1; i = 1, \dots, N_j; h = 1, \dots, N_{j-1}$$

onde β é um coeficiente de aprendizagem, com valores entre 0 e 1.

As ações dos passos 2 a 6 devem ser repetidas para cada amostra de aprendizagem $k=1, \dots, n$, e de novo repetido o treino de todo o conjunto de treino até que o erro médio quadrático da saída E seja minimizado.

$$E = \frac{1}{2} \sum_{k=1}^N E_k \quad (*)$$

sendo $E_k = (\mathbf{d}_k - \mathbf{y}_k)^T (\mathbf{d}_k - \mathbf{y}_k)$ o erro quadrático referente à amostra k .

Alternativamente à otimização da função de erro Eq. (*), o processo iterativo de otimização pode ser conduzido a cada resultado da aplicação de uma amostra de treino. Neste caso, a função de erro a otimizar será, para a amostra de treino k :

$$E_k = \frac{1}{2} (\mathbf{d}_k - \mathbf{y}_k)^T (\mathbf{d}_k - \mathbf{y}_k)$$

No método *generalized delta rule*, o erro associado ao neurónio i da camada j é a taxa de variação do erro RMS, E_k , relativamente ao termo da função integradora ($a_{ji} = \sum_h^{N_{j-1}} W_{jih} Y_{(j-1)h}$), isto é a soma-do-produto do neurónio)

$$\delta_{ji} = -\frac{\partial E_k}{\partial a_{ji}}$$

Esta derivada pode ser determinada pelo produto da derivada parcial do erro E_k em relação à saída do neurónio i , da camada j , com a derivada da saída em relação a a_{ji} :

$$\delta_{ji} = -\frac{\partial E_k}{\partial Y_{ji}} \frac{\partial Y_{ji}}{\partial a_{ji}} \quad (**)$$

Este valor corresponde ao valor do gradiente da função de erro relativamente ao valor de ativação do neurónio i da camada j , importante no processo iterativo de otimização local para esse neurónio.

Recorrendo à regra da cadeia das derivadas de funções, pode-se relacionar a derivada de E_k em relação ao peso w_{jih} será:

$$\frac{\partial E_k}{\partial w_{jih}} = \frac{\partial E_k}{\partial a_{ji}} \frac{\partial a_{ji}}{\partial w_{jih}} \quad (***)$$

o que equivale a:

$$\frac{\partial E_k}{\partial w_{jih}} = -\delta_{ji} Y_{(j-1)h}$$

As mudanças dos pesos serão realizadas de modo proporcional a este valor anterior, ou seja:

$$\Delta W_{jih} = \beta \delta_{ji} Y_{(j-1)h} \quad (****)$$

onde β é o coeficiente de aprendizagem. Assim, a adaptação dos pesos da rede são realizados de acordo com:

$$W_{jih}^+ = W_{jih} + \Delta W_{jih}$$

Para a camada de saída $j=M$ e para $Y_{Mi}=y_{ki}$, ter-se-á

$$\delta_{Mi} = -\frac{\partial E_k}{\partial Y_{Mi}} \frac{\partial Y_{Mi}}{\partial a_{ji}}$$

Ou seja:

$$\delta_{Mi} = -\frac{\partial E_k}{\partial Y_{Mi}} (d_{ki} - y_{ki}) \frac{\partial f(a_{Mi})}{\partial a_{Mi}}$$

Se a função de ativação $f(\cdot)$, for do tipo sigmoide, ter-se-á, ainda que:

$$\delta_{Mi} = -\frac{\partial E_k}{\partial Y_{Mi}} (d_{ki} - y_{ki}) f(a_{Mi}) (1 - f(a_{Mi}))$$

Este procedimento não pode ser aplicável diretamente para determinar o erro das camadas escondidas, cujos valores desejados não são conhecidos, contrariamente ao que acontece para a camada de saída onde o vector d é dado. Por esta razão, a parte $\partial E_k / \partial Y_{ji}$ da equação (**) precisa ser determinada de modo diferente, utilizando a cadeia das derivadas, de produtos de derivadas sucessivas até à camada $(j+1)$

precisa ser determinada de modo diferente, utilizando a cadeia das derivadas, de produtos de derivadas sucessivas até à camada $(j+1)$

$$\frac{\partial E_k}{\partial Y_{ji}} = \frac{\partial E_k}{\partial a_{(j+1)1}} \frac{\partial a_{(j+1)1}}{\partial Y_{ji}} + \frac{\partial E_k}{\partial a_{(j+1)2}} \frac{\partial a_{(j+1)2}}{\partial Y_{ji}} + \dots + \frac{\partial E_k}{\partial a_{(j+1)N_{j+1}}} \frac{\partial a_{(j+1)N_{j+1}}}{\partial Y_{ji}}$$

ou $\frac{\partial E_k}{\partial Y_{ji}} = \sum_{h=1}^{N_{j+1}} \frac{\partial E_k}{\partial a_{(j+1)h}} \frac{\partial a_{(j+1)h}}{\partial Y_{ji}}$

Uma vez que $\partial E_k / \partial a_{(j+1)h} = -\delta_{(j+1)h}$ e $\partial a_{(j+1)h} / \partial Y_{ji} = W_{(j+1)hi}$, tem-se que:

$$\frac{\partial E_k}{\partial Y_{ji}} = -\sum_{h=1}^{N_{j+1}} \delta_{(j+1)h} W_{(j+1)hi}$$

Finalmente, combinando com $\partial Y_{ji} / \partial a_{ji}$ tem-se que:

$$\delta_{ji} = -\sum_{h=1}^{N_{j+1}} [\delta_{(j+1)h} W_{(j+1)hi}] \frac{\partial Y_{ji}}{\partial a_{ji}}$$

O que para neurónios com funções de ativação sigmoide dará:

$$\delta_{ji} = Y_{ji} (1 - Y_{ji}) \sum_{h=1}^{N_{j+1}} [\delta_{(j+1)h} W_{(j+1)hi}]$$

Por fim, a adaptação dos parâmetros pode ser de novo realizada com (****), isto é:

$$W_{jih}^+ = W_{jih} + \beta \delta_{ji} Y_{(j-1)h}$$

Exemplo: Uma rede neuronal de 3 camadas será construída para identificar caracteres numéricos 0 e 1, representados na Figura 24. Estes constituem 35 imagens binárias, cor preto e branco, com uma resolução 4x3. Que serão usado no processo de treino de uma rede neuronal com três camadas, com 10 neurónios na camada escondida, ajustando os seus pesos pelo método “Backpropagation”, procurando dar uma resposta 0 ou 1 para imagens dos caracteres 0 e 1. Todos os neurónios da camada escondida e de saída usam funções de ativação sigmoide unipolar.

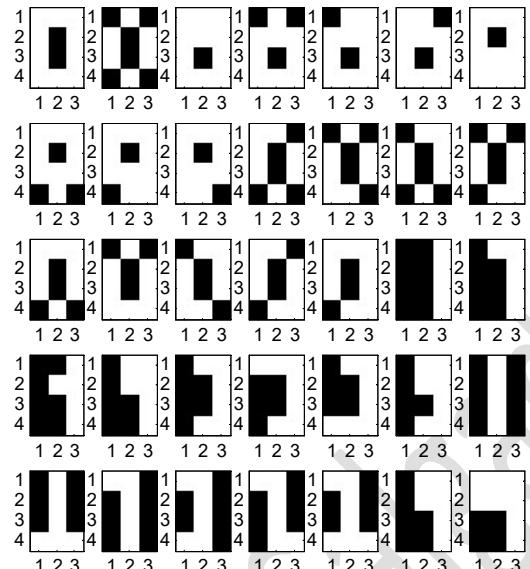


Figura 24. Imagens binárias 4×3 de caracteres dos números 0 e 1, usando no treino e teste de uma rede neuronal com 3 camadas.

Uma matriz de dados de treino de entrada, X, é construída a partir das imagens, onde cada uma das suas colunas é uma imagem da Figura 24, onde 1 e 0 são, respectivamente, pixéis branco e preto. Estes vetores são sucessivamente apresentados à rede neuronal e o seu valor de saída determinado. O vetor Y contém os valores desejados, sendo os primeiros 19 valores zeros e os restantes 16 valores uns. Os erros entre a resposta da rede e o valor desejado Y são então utilizados pelo método *backpropagation* para ajustar os pesos da rede neuronal, fazendo diminuir progressivamente o erro.

$\Sigma = \{0, 1\}$

Ao fim de 7 iterações de treino a rede neuronal responde com um erro extremamente reduzido: Erro RMS= 4.7599e-07; Erro MAD=4.7937e-04.

O erro (em parte por milhar) da resposta da Rede Neuronal, para cada uma das 35 imagens é:

Erro da rede neuronal para cada imagem de treino (%)

-1.2; -0.8; -2.3; -1.7; -0.8; -1.0; -1.3; -0.6; -1.0; -0.8; -0.7; -1.5; -0.7; -0.7; -0.7; -0.5; -1.3; -1.4; -0.3; -0.8; 0.5 0.7; 1.0;
0.5 0.7; 0.9 0.6; 4.0 1.3; 1.0 0.8; 0.9 0.4; 0.4 0.5; 1.4

Se o resultado da rede neuronal for arredondado para o número inteiro mais próximo a resposta da rede apresenta um erro nulo de identificação. Na Tabela seguinte estão representados os pesos das várias camadas da rede neuronal obtida pelo método de backpropagation.

Tabela 8 – Pesos da rede

Camada escondida

W≡

1.481	0.315	-0.278	0.454	0.907	-0.402	-0.895	0.272	0.279	-0.335	-0.007	0.172
0.546	0.318	0.091	-0.697	0.326	0.985	0.813	0.938	0.926	-1.304	1.152	0.878
-0.148	0.347	1.965	-0.011	1.302	-0.364	-0.127	-0.854	-0.264	-0.911	-0.690	0.222
0.360	-0.221	0.637	-0.924	-0.626	0.118	0.334	0.001	0.092	-0.301	-0.466	0.489
-0.387	0.290	-1.126	0.090	0.310	-1.233	0.283	-0.369	0.196	-0.105	-0.704	0.249
-0.237	-0.030	0.013	0.117	0.119	1.105	-0.451	-0.758	-0.839	-0.251	0.135	-0.143
0.261	-0.496	0.784	-0.341	-0.474	-0.277	0.098	-0.250	0.804	0.214	-0.341	0.256
1.516	-0.487	0.034	0.223	-1.130	-0.701	-0.015	-0.942	0.048	0.578	1.030	-0.998
0.700	0.070	-0.475	-0.485	0.145	0.065	-0.361	0.177	-0.042	-0.195	0.024	0.422
0.585	0.196	-0.516	0.044	-0.544	0.321	-0.045	-0.131	-0.551	1.210	-0.928	0.187

onde $w(i,j)$ é o peso que liga a entrada i ao neurónio j da 2^a camada

Vector Bias

-2.2271 -1.1303 0.5734 -0.7422 0.5723 -0.4150 -1.2606 1.1363 0.6174 1.5440

onde cada elemento i deste vector é o valor Bias do neurónio i dessa camada.

Camada de saída

w=

-1.2618 0.4733 -1.5611 0.1157 -0.1799 0.2044 -0.5805 -1.5002 -1.5893 -0.5508

onde $w_{i,j}$ é o peso que liga o neurónio i da camada escondida ao neurónio $j=1$ da camada de saída.

Valor Bias:

1.4441

Valor Bias do neurónio de saída.

10. Radial Basis function – RBF

As Funções de Base Radial (RBF- Radial Basis Function) emergiram na década de 80 como uma variante das redes neurais artificiais. A sua raiz tem origem nas técnicas de reconhecimento de padrões tais como funções potenciais, técnicas de agrupamento, aproximação funcional, interpolação por splines e mistura de modelos [¹]. A RBF possui uma estrutura de rede com três camadas onde na camada escondida estão neurónios com funções de ativação radial. Na primeira, a entrada é mapeada para cada neurónio RBF enquanto o neurónio de saída (última camada) agrupa, com pesos adequados, as respostas dos neurónios da camada escondida. Deste modo a RBF é não-linear na camada escondida e linear na saída, o que lhe confere a capacidade de se constituir como um approximador universal [²][³], permitindo-lhe modelar relações complexas entre variáveis. Ela pode aproximar qualquer função multivariável contínua num domínio compacto com um arbitrário grau exatidão, bastando para tal ter um número suficiente de unidades e pesos de valores adequados. Tal tarefa é igualmente possível de realizar por uma rede neuronal multicamada, com múltiplas camadas intermédias, mas as RBFs são uma alternativa eficaz por serem geralmente mais fáceis e rápidas de treinar. As redes RBF foram aplicadas com sucesso a uma grande diversidade de aplicações [⁴], incluindo-se a interpolação, a modelação de séries temporais caóticas, sistema de identificação, controle de processos, equalização, reconhecimento de fala, restauração de imagens, shapefrom-shading, fusão de dados, etc.

A estrutura da rede RBF fica definida pela especificação do número de neurónios da camada escondida, o género da função de ativação radiais, o seu posicionamento (centros) no espaço de trabalho e a sua híper-superfície de ativação [⁵]. Para a camada de saída é necessário também encontrar os pesos. Estas tarefas são realizadas por um algoritmo de treino (estrutural ou paramétrico), com recurso a um conjunto de treino constituído por pares de entrada-saída, que optimiza a estrutura ou os parâmetros de rede a fim de melhorar o desempenho da rede.

A estrutura da rede RBF network está representada na Figura 25. Cada nó da camada escondida usam funções de base radial, designada por $\phi(r)$, como função de ativação não-linear. A camada escondida mapeia a variável de entrada com uma transformação não-linear, com a camada de saída combinado linearmente estas transformações num novo espaço.

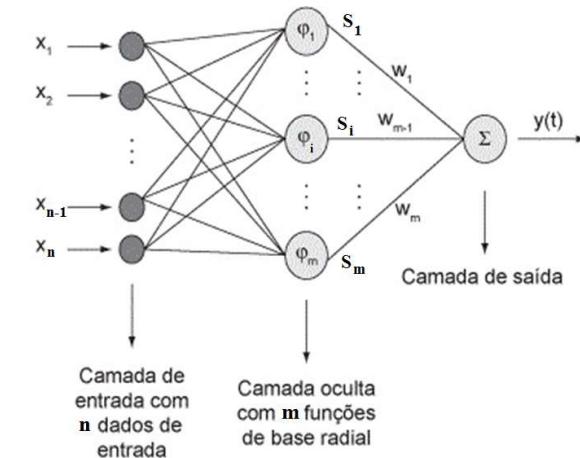


Figura 25. Estrutura de uma rede RBF

Frequentemente a mesma RBF é usada em todos os nós, onde $\phi(r) = \phi(\bar{x} - \bar{c}_i)$, $i = 1, \dots, m$, onde c é o centro da função radial do nó i .

O procedimento de aprendizagem de uma RBF é sintetizado em três etapas: (i) selecionar o número de centros; (ii) escolher os valores dos centros; e (iii) ajustar os pesos. Existe também uma variedade de algoritmos à configuração deste tipo de rede neuronal. Os algoritmos podem ser agrupados em duas categorias: (i) métodos construtivos e (ii) métodos de otimização não-linear [⁶].

No método construtivo, denominado *k-médias*, os parâmetros não-lineares (centros e variâncias) da RBF são fixos e técnicas de estimativa lineares determinam os pesos da camada de saída. Nestes métodos construtivos o processo de identificação inicia-se pela determinação dos centros e pela escolha da função base, por exemplo, recorrendo a métodos de agrupamento. Os pesos da RBF são ajustados posteriormente para que a resposta da RBF seja o mais próxima possível da desejada, através de um critério de minimização do erro. Uma vez que existe uma relação linear entre os pesos W s e a saída da RBF, um esquema de estimativa otimização linear pode ser utilizado para determinação dos valores ótimos dos pesos, como seja o algoritmo de mínimos quadrados [⁷].

- Vantagens

As redes RBF têm a vantagem de não sofrer, no processo de aprendizagem, do problema da retenção no mínimo local da mesma forma que sucede no Multi-Layer Perceptrons. Isso ocorre porque os únicos parâmetros que são ajustados no processo de aprendizagem são o mapeamento linear da camada escondida para a camada de saída. A linearidade assegura que a superfície de erro é quadrática e, portanto, tem um único mínimo facilmente encontrado. Em problemas de regressão isto pode ser encontrado numa operação de uma matriz. Em problemas de classificação, a não-linearidade introduzida pela função de saída sigmoide é tratada de modo mais eficiente usando iterativamente mínimos quadrados reponderados.

- Desvantagens

As redes RBF têm a desvantagem de exigir uma boa cobertura do espaço de entrada em funções de base radial. Os centros são determinados com referência à distribuição dos dados de entrada, mas sem referência à tarefa de previsão. Como resultado, os recursos de representação podem ser desperdiçados em áreas do espaço de entrada que são irrelevantes para a tarefa de aprendizagem. Uma solução comum é associar os dados de cada ponto com o seu próprio centro, embora isso possa tornar o sistema linear a ser resolvido na camada final, e requer técnicas de encolhimento para evitar overfitting.

Associando cada dado de entrada com uma RBF conduz naturalmente aos métodos de kernel como Support Vector Machines (SVM) e Processos Gaussianos (o RBF é a função kernel). Estas três abordagens usam uma função kernel não-linear para projetar os dados de entrada num espaço de aprendizagem onde o problema pode ser resolvido através de um modelo linear [1].

Funções Radiais:

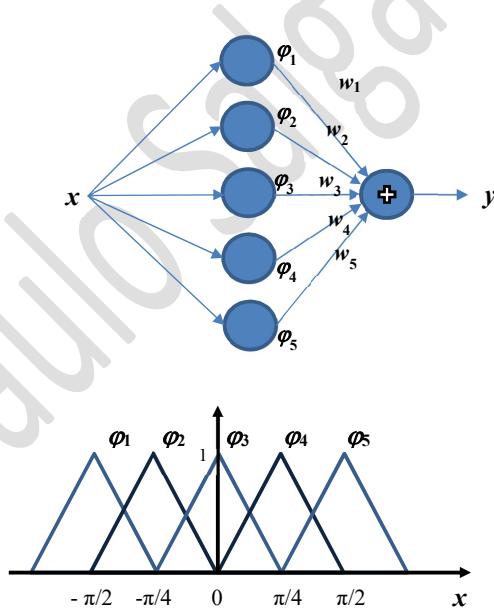
Um conjunto diversos de estudos empíricos e teóricos revelaram que muitas das propriedades de interpolação são relativamente insensíveis à forma precisa das funções de base $\phi(r)$. Algumas das mais usadas funções radiais estão representadas na Tabela 3.

Tabela 4 – Funções Radiais

Nome	Função	Nome	Função
Gaussiana	$\phi(r) = e^{-r^2/2\sigma^2}$	Spline –poli-harmonica	$\phi(r) = r^k, \quad k=1, 3, 5, \dots$ $\phi(r) = r^k \ln(r), \quad k=2, 4, 6, \dots$
Inverso multiquadrado	$\phi(r) = \frac{1}{(\sigma^2 + r^2)^\alpha}, \quad \alpha > 0$	Sigmoide bipolar	$\phi(r) = r^2 \ln(r)$
Multiquadrado	$\phi(r) = (\sigma^2 + r^2)^\alpha, \quad 0 < \alpha < 1$	Linear	$\phi(r) = r$
Função Logística	$\phi(r) = \frac{1}{1 + e^{(r/\sigma)-\theta}}$		

Exemplo:

1. Na figura seguinte está representada uma estrutura de uma rede RBF, onde $x \in \mathbb{R}$ é a variável de entrada e $y \in \mathbb{R}$ a variável de saída.



Os nós da camada escondida são constituídos por funções de base radial (*RBF - Radial Basis Function*), designada por $\phi(r)$, como função de ativação não-linear, que neste exemplo-questão são em número de 5 e de funções triangulares (Ver figura). Cada uma destas funções

tem centros em c_i , igualmente espaçados entre $-\pi/2$ e $\pi/2$ (isto é $-\pi/2; -\pi/4; 0; \pi/4$ e $\pi/2$), e extremos ligados aos centros dos seus vizinhos. Esta rede RBF será usada para aproximar a

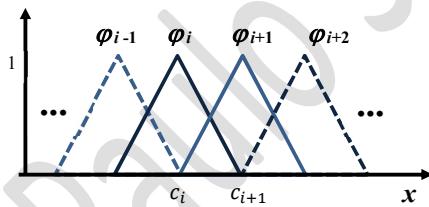
função trigonométrica de $f(x) = \sin(x)$, no intervalo $x \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$

- Indique a função de transferência desta rede, considerando a utilização de funções triangulares, considerando que os pesos $w_i = \sin(c_i)$, com $i = 1, \dots, 5$.
- Determine o erro máximo da aproximação da rede RBF à função $f(x)$ para o intervalo considerado.

Resolução:

Pela observação da estrutura da rede RBF, representada na figura ?, a variável de saída y é imagem de x com um valor que resulta da soma pesada das funções triangulares, para o valor da variável de entrada x , pelos coeficientes w 's.

$$y = \sum_{i=1}^5 w_i \varphi_i(x)$$



O domínio da função $f(x) = \sin(x)$ está completamente coberto por funções triangulares. Para um determinado valor de x situado no intervalo $[c_i, c_{i+1}]$ apenas duas das funções referidas assumem valores superiores ou igual a zero, sendo elas as correspondentes às funções de centros vizinhos, isto é;

Para $x \in [c_i, c_{i+1}]$, $\varphi_i(x) \geq 0$ e $\varphi_{i+1}(x) \geq 0$, assumindo as restantes funções valores nulos. Deste modo, ter-se-á, para $x \in [c_i, c_{i+1}]$

$$y = w_i \varphi_i(x) + w_{i+1} \varphi_{i+1}(x)$$

$$\varphi_i(x) = 1 - \frac{x - c_i}{c_{i+1} - c_i}$$

$$\varphi_{i+1}(x) = \frac{x - c_i}{c_{i+1} - c_i}$$

$$y = w_i \varphi_i(x) + w_{i+1} \varphi_{i+1}(x) = w_i \left(1 - \frac{x - c_i}{c_{i+1} - c_i}\right) + w_{i+1} \left(\frac{x - c_i}{c_{i+1} - c_i}\right)$$

3 Aprendizagem

Para uma entrada \bar{x} , a rede RBF apresenta a saída dado por:

$$y_i(\bar{x}) = \sum_{l=1}^m w_{li} \phi(\|\bar{x} - \bar{c}_l\|), \quad i = 1, \dots, n_o \quad (?)$$

em que $y_i(\bar{x})$ é a i^{esima} saída, w_{li} é o peso de ligação do l^{esimo} neurónio da camada escondida para a unidade de saída i . A dimensão de entrada da RBF é igual à dimensão do vector \bar{x} , enquanto o número de saída são n_o .

Analisemos, daqui em diante, a rede RBF com funções radiais gaussianas.

Para um conjunto de dados de treino, compostos por pares de dados $\{(\bar{x}_k, \bar{d}_k), k = 1, \dots, N\}$, a equação (?), na forma vectorial assume a seguinte representação:

$$Y = W^T \Phi$$

onde $W = [\bar{w}_1, \dots, \bar{w}_{n_o}]$ é uma matriz $m \times n_o$, $\Phi = [\bar{\phi}_1, \dots, \bar{\phi}_m]$ á uma matriz $m \times N$ com o vector $\bar{\phi}_k = [\phi(\|\bar{x}_k - \bar{c}_1\|), \dots, \phi(\|\bar{x}_k - \bar{c}_m\|)]^T$, $Y = [\bar{y}_1, \dots, \bar{y}_N]$ uma matriz $n_o \times N$ com $\bar{y}_k = [y_1(k), \dots, y_{n_o}(k)]^T$, e $D = [\bar{d}_1, \dots, \bar{d}_N]$.

A aprendizagem da rede RBF pode ser formulada como um problema de minimização da média dos erros quadráticos, isto é:

$$E = \frac{1}{N} \sum_{k=1}^N \sum_{i=1}^{n_o} (d_i(k) - y_i(k))^2 \quad ??$$

onde $\| \cdot \|_F^2$ é a norma Frobenius definida como $\|A\|_F^2 = \text{tr}(A^T A)$, sendo tr é a função traço da matriz.

A aprendizagem da rede RBF requer a determinação dos centros das RBFs e os pesos do neurónio de saída. A seleção dos centros é a tarefa mais crítica e complexa, geralmente escolhidas como subconjunto aleatório dos dados de treino, de modo a cobrirem o espaço de entrada, ou como resultado de técnicas de agrupamento.

Em contrapartida, a determinação dos pesos pode ser realizada por métodos lineares, como sejam o dos mínimos quadrados ou por técnicas de ortogonalização.

Pelo método dos mínimos quadrados, os pesos são determinados de acordo com:

$$W = (\Phi^T \Phi)^{-1} \Phi D$$

Os centros das funções radiais podem ser iterativamente ajustados, de modo a reduzirem o erro (??), fazendo variar os seus valores na direcção oposta ao gradiente da função de erro em relação ao parâmetro. Deste modo:

$$\vec{c}_l^+ = \vec{c}_l - \eta \cdot \nabla_{\vec{c}} E$$

onde η é um parâmetro de aprendizagem.

Exemplo: (meter aqui os exemplos dados no inicio do semestre)

Erro RMS= 8.4410e-05

Erro MAD= 5.8096e-03

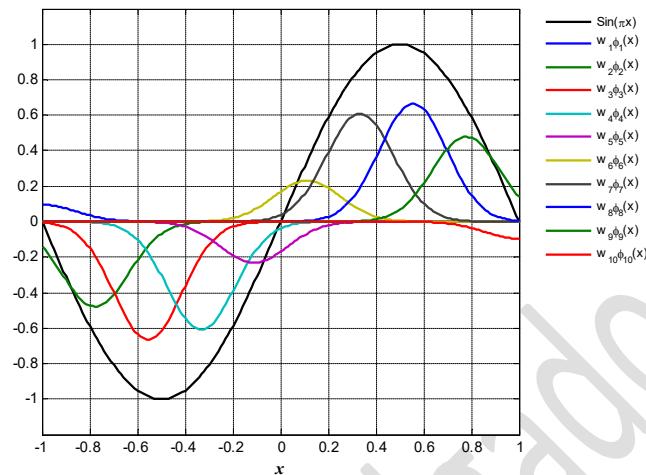


Figura 26:

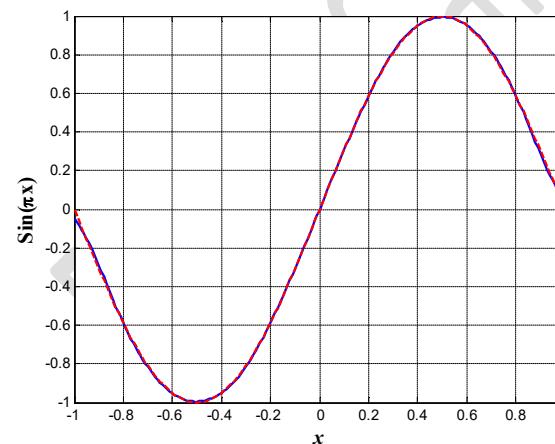


Figura 27:

M=10 ; S=[0.4 0.4 0.4 0.4 0.4]
 $x_0 = [-1.0000 -0.5000 0 0.5000 1.0000]$
 $W=[0.1803 -1.0948 0.0000 1.0948 -0.1803]$
 Erro_MLS = 5.0142e-03 Erro_RAD = 5.2380e-02

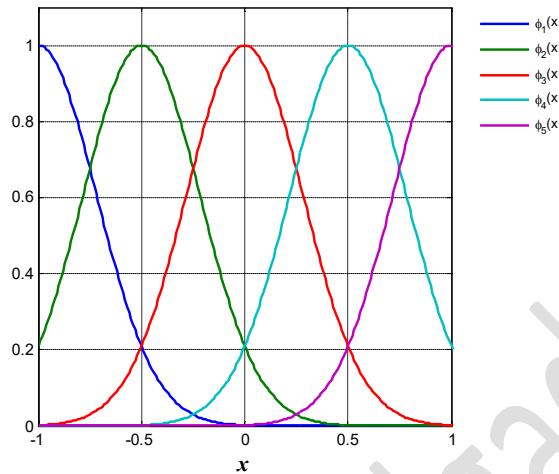


Figura 28

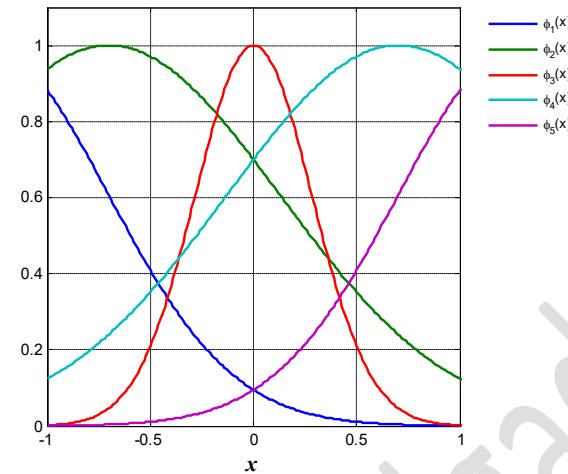


Figura 30

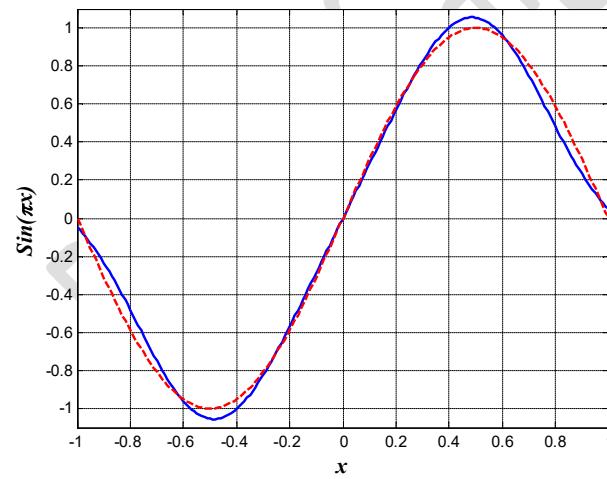


Figura 29

Erro_MLS = 2.7618e-08 Erro_RAD = 1.4062e-04

W=[3.6837 -3.9845 0.0000 3.9845 -3.6837]

X0 =[-1.3002 -0.7018 0.0000 0.7018 1.3002]

S=[0.8469 1.1752 0.4000 1.1752 0.8469]

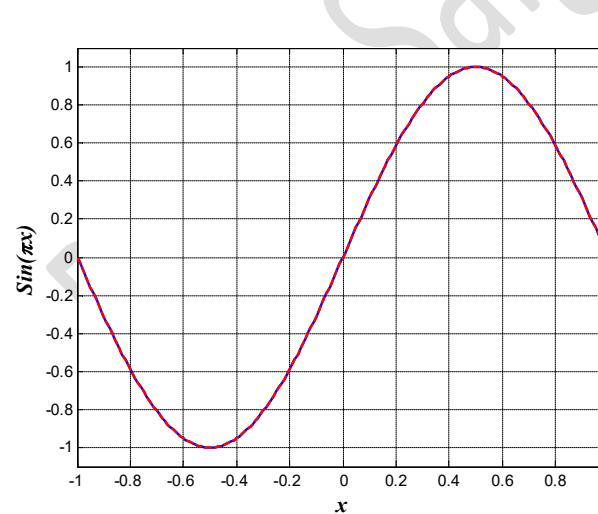
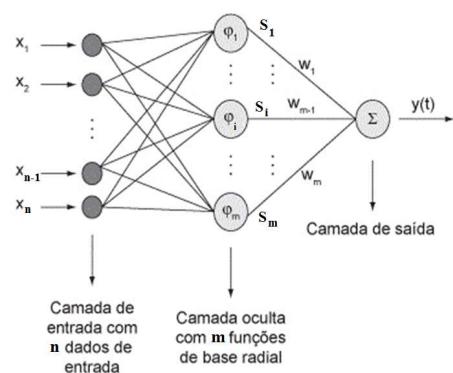


Figura 31

Exercício: Na figura seguinte está representada uma estrutura de uma rede RBF, onde $\bar{x} = [x_1, \dots, x_j, \dots, x_n]^T \in \mathbb{R}^n$ é o vector de entrada e $y \in \mathbb{R}$ a variável de saída.



Os nós da camada escondida são constituídos por funções de base radial (RBF - Radial Basis Function), designada por $\phi(r)$, como função de ativação não-linear. Neste exemplo-questão, a mesma função RBF é usada em todos os nós, sendo que $\varphi_i(\bar{x}) = \min_{j \leq n} \varphi_i^{(j)}(x_j)$, $i = 1, \dots, m$. Esta função é composta por funções de cada componente da variável \bar{x} , na forma triangular:

$$\varphi_i^{(j)}(x) = \begin{cases} \frac{|x(j) - c_i(j)|}{\sigma}, & \text{para } |x(j) - c_i(j)| < \sigma \\ 0 & \text{outros} \end{cases}, \quad (\text{função triangular})$$

onde \bar{c}_i é o centro da função radial do nó i , $c_i(j)$ a sua coordenada j e σ um parâmetro correspondente à metade do comprimento da base da “função triangular”.

A camada escondida mapeia a variável de entrada com uma transformação não-linear. Por fim a camada de saída agrupa linearmente estas transformações num novo espaço, apresentando uma resposta na saída.

O procedimento de aprendizagem de uma RBF é sintetizado em três etapas: (i) selecionar o número de funções radiais; (ii) escolher os valores dos seus centros; e (iii) ajustar os pesos.

a) Indique a função de transferência desta rede, considerando a utilização de funções triangulares.

- b) Considere realizar a aproximação da função $f(x_1, x_2) = \sin(x_1) \cdot \cos(x_2)$, através de um conjunto valores de $\{(x_1, x_2)\}$ e de $f(x_1, x_2)$, com as variáveis independentes compreendidas entre $-\pi$ e π . As m funções RBF's (em numero de 100) estão igualmente espaçadas no espaço domínio considerado. Apresente um método de otimização para encontrar os valores ótimos dos pesos da rede, W , que minimize a soma dos erros quadráticos da aproximação realizada.

11. Amostra de treino e amostra de teste

Como referido anteriormente, um treino e uma amostra de teste são normalmente necessários para a construção de uma RNA. A amostra de treino é usada para o desenvolvimento do modelo RNA e a amostra de teste é adoptada para avaliar a capacidade de previsão do modelo. Ocionalmente, uma terceira amostra chamada de validação é utilizada para evitar o problema de *overfitting* ou para determinar o ponto de paragem do processo de treino. É comum usar um conjunto de testes, para finalidades de validação e de testes particularmente com pequenas séries de dados. A seleção da amostra de treino e teste pode afectar o desempenho das RNAs.

A primeira questão a considerar nesta fase é a divisão dos dados para os conjuntos de treino e testes. Embora não haja nenhuma solução geral para este problema, diversos fatores tais como as características do problema, o tipo de dados e o tamanho dos dados disponíveis devem ser considerados. A separação imprópria dos conjuntos de treino e de teste afetará a seleção ideal da estrutura e a avaliação do desempenho da previsão da RNA.

Um outro factor intimamente relacionado é o tamanho da amostra. Não existe nenhuma regra definida para a exigência do tamanho da amostra para um dado problema. A quantidade de dados para o treino da rede depende da estrutura, do método de treino, e da complexidade do problema em particular ou da quantidade de ruído nos dados. Geralmente, em qualquer abordagem estatística, o tamanho da amostra está estreitamente relacionado com a precisão requerida do problema. Quanto maior o tamanho da amostra, mais precisos serão os resultados. Dado um determinado nível de precisão, torna-se necessário uma amostra maior, enquanto o relacionamento subjacente entre saídas e entradas torna-se mais complexo ou o ruído nos dados aumenta. No entanto, na realidade, o tamanho da amostra é limitado pela disponibilidade dos dados. A precisão de um problema particular de previsão pode igualmente ser afetada pelo tamanho da amostra usado no treino e/ou conjunto de teste.

De notar, que cada modelo tem limites na precisão que pode conseguir para problemas reais. Por exemplo, se considerarmos apenas dois factores: o ruído nos dados e no modelo subjacente, o limite de precisão de um modelo linear, tal como o *Box-Jenkins* é determinado então pelo ruído nos dados e no grau a que o formulário funcional subjacente é não-linear. Com mais observações, a precisão do modelo não pode melhorar se existir uma estrutura não-linear nos dados. Em RNAs, o ruído por si só determina o limite de precisão devido à capacidade da aproximação da função geral. Com uma amostra suficientemente grande, RNAs podem modelar qualquer estrutura complexa dos dados. Assim, as RNAs podem tirar melhor proveito das grandes amostras do que os modelos estatísticos lineares. É interessante notar que RNAs não exigem necessariamente uma amostra maior do que os modelos lineares, para obter uma melhor previsão.

12. Medidas de desempenho

Embora possam existir muitas medidas de desempenho para uma RNA de previsão, como o tempo de modelação e o tempo de treino, a última e mais importante medida de desempenho é a precisão da previsão, que pode ser obtida além dos dados de treino. No entanto, uma medida apropriada de precisão para um determinado problema não é universalmente aceite pelos académicos e profissionais de previsão. Uma medida de precisão é frequentemente definida em termos do erro de previsão, correspondendo à diferença entre o real (desejado) e o valor previsto. Na bibliografia da especialidade, há uma série de medidas de precisão da previsão e cada uma tem vantagens e limitações. Os mais utilizados são:

- Desvio médio absoluto (MAD) = $\frac{\sum |e_t|}{N}$;
- Soma dos erros quadráticos (SSE) = $\sum e_t^2$;
- Erro médio quadrático (MSE) = $\frac{\sum e_t^2}{N}$;
- Erro de raiz quadrático ($RMSE$) = \sqrt{MSE} ;
- Erro médio absoluto percentual ($MAPE$) = $\frac{1}{N} \sum \left| \frac{e_t}{y_t} \right| \times 100$;

Onde e_t é o erro de previsão individual; o y_t é o valor real; e N é o número de termos do erro.

Devido às limitações associadas a cada medida, pode-se usar múltiplas medidas de desempenho num problema particular.

É importante salientar que as primeiras quatro medidas de desempenho mais utilizadas são medidas absolutas e de valor limitado quando usadas para comparar séries temporais diferentes. O MSE é a medida de precisão mais utilizada na literatura.

13. Conclusão

Foi revisto o estado atual do uso de redes neurais artificiais para a previsão. Os resultados mais importantes são resumidos de seguida:

- As características originais das RNAs - adaptabilidade, não-linearidade, capacidade de mapeamento arbitrário da função – torna-as apropriadas e úteis para tarefas de previsão. Em geral, as RNAs têm um desempenho satisfatório na previsão.
- Pesquisas consideráveis têm sido feitas nesta área. Os resultados não são conclusivos se e quando as RNAs são melhores do que os métodos clássicos.
- Há muitos fatores que afetam o seu desempenho. No entanto, não há nenhuma investigação sistemática destas questões. O método “tentativa e erro” é normalmente o adoptado pela maioria de investigadores.

As RNAs oferecem uma aproximação alternativa aos tradicionais métodos lineares, no entanto, também incorporam um elevado grau de incerteza. Como os modelos estatísticos, as RNAs têm pontos fracos, assim como pontos fortes, que as tornam completamente apropriadas para uma variedade de problemas. As RNAs podem ser mais indicadas para as seguintes situações:

- Grandes séries de dados;
- Problemas com estrutura não-linear;
- Problemas da previsão de séries temporais múltiplas.

Para melhor utilizar RNAs para problemas de previsão assim como outras tarefas, é importante compreender as suas limitações, o que podem ou não realizar. Diversos pontos devem de ser destacados, a saber:

- São métodos não-lineares. Para processos lineares estáticos com pouca perturbação, não podem ser melhores do que métodos estatísticos lineares.
- São métodos de caixa-preta (black-box). Não existe uma forma explícita para explicar e analisar a relação entre entradas e saídas. Isto causa dificuldade em interpretar os

resultados das redes. Igualmente nenhum método de teste estatístico formal pode ser usado.

- São propensas a ter problemas de *overfitting* devido a um modelo excessivamente complexo para ser estimado.
- Não há atualmente nenhum método estruturado para identificar qual a estrutura de rede que melhor pode aproximar a função, mapeamento de entradas e saídas. Assim, as cansativas experiências e procedimentos de “tentativa-e-erro” são usados frequentemente.
- Geralmente requerem mais dados e tempo de computação para o treino.

De um modo geral, pode haver um limite no que RNAs podem aprender a partir dos dados e fazer previsões. Esta limitação pode vir da sua propriedade não paramétrica. O futuro da previsão com RNA será ainda mais brilhante com os esforços de pesquisa cada vez mais dedicados nesta área.

*Capítulo II.***LÓGICA DIFUSA****1. Introdução**

Neste capítulo são introduzidos conceitos fundamentais da teoria de conjuntos difusos. É ainda apresentada a notação matemática utilizada.

Os conjuntos difusos são considerados como uma generalização dos conjuntos bivalentes, pela generalização dos valores da função de pertença (característica) do conjunto $\{0,1\}$ para o intervalo de números reais $[0,1]$. São revistos vários conceitos dos conjuntos difusos, como sejam a representação, suporte, α -cuts, convexidade, etc.. É apresentado o princípio da resolução, que pode ser usado para expandir um conjunto difuso nos seus termos α -cuts, e o princípio da extensão, que permite a generalização de uma qualquer função f , no mapeamento de subconjuntos difusos. Várias operações sobre conjuntos difusos, como *T-norms*, *T-conorms* e outros operadores de agregação, são também consideradas.

As relações difusas são, da mesma forma, vistas como a generalização das relações binárias. São introduzidos alguns conceitos básicos e operações em relações difusas, e em composições relações-relações e conjuntos-relações. São abordados os principais mecanismos de inferência difusa na interpretação de regras difusas, do tipo IF-THEN, analisadas as várias operações de inferência, bem como a interpretação das suas propriedades. São, ainda, referidos os controladores de lógica difusa, a sua arquitetura básica e metodologia do desenho. Por último, são revistos os SLD's e os principais métodos de identificação utilizados em engenharia.

2. Lógica Difusa

Um conjunto é definido de tal forma que dicotomiza um ponto individual de um dado universo em dois grupos: os membros, que são aqueles que seguramente lhe pertencem, e os não-membros, que são aqueles que de certeza não lhe pertencem, respetivamente com graus de pertença 1 e 0. Desta forma existem fronteiras rígidas afetas a cada conjunto. Porém, em categorias empregues com frequência na linguagem natural, como classes de pessoas altas, carros caros, dias solarengos, etc., não existem estas características, pois as fronteiras não são rígidas e a transição de membro para não membro aparece de uma forma gradual, em vez de abrupta [6]. Seja o exemplo de um conjunto de alunos { Paulo (18 anos); José (22 anos); Maria (45 anos)}, do universo que frequenta a universidade. Uma forma possível de caracterizar o

grau de “juventude” no Universo do discurso, seria $J = \{\text{Maria}/0.1; \text{José}/0.8; \text{Paulo}/0.9\}$.

Assim, os conjuntos difusos introduzem o conceito de transição contínua ou gradual entre membro e não membro de um conjunto, com vista a aumentar a abrangência destes conceitos. O conjunto difuso pode ser matematicamente definido pela atribuição, a cada ponto individual do universo do discurso, de um valor representativo do seu grau de pertença a um conjunto difuso.

Lógica é o estudo do método da “razão” em todas as suas formas possíveis. Como é sabido, a lógica clássica utiliza *proposições* que assumem apenas dois valores: *verdadeiro* ou *falso*. Uma área da lógica, referida como *lógica proposicional*, usa combinações de variáveis (lógicas) de acordo com proposições arbitrárias. Cada proposição pode ser dividida em duas partes: *sujeito* e *predicado*. Por outras palavras, uma simples proposição pode ser expressa em geral, na forma canónica:

$$x \text{ is } P \quad (2.1)$$

em que x designa o *sujeito* e P designa o *predicado*. Seja agora x , um padrão de qualquer *sujeito* de um designado universo do discurso X , e P o *predicado*, que desempenha o papel de uma função em X . Esta função é geralmente chamada de *função predicado* e é representada por $P(x)$, assumindo apenas dois valores, *verdadeiro* ou *falso*, quando um dado *sujeito* em X é substituído em x .

Pode estabelecer-se que a lógica proposicional é isomórfica com respeito à teoria dos conjuntos, e estabelecer correspondências entre componentes dos dois sistemas matemáticos. Por sua vez, ambos os sistemas são isomórficos com a álgebra Booleana. O isomorfismo entre a álgebra Booleana, a teoria dos conjuntos e a lógica proposicional, garante que para cada teorema, em qualquer um destes paradigmas, existe um teorema correspondente nos outros paradigmas.

2.1.1 Conceitos básicos.

O conceito de conjunto “difuso” designa um conjunto em que a cada membro está associado um grau de pertença; o grau de pertença é uma variável contínua, em que valores elevados evidenciam um alto grau de pertença ao conjunto, enquanto valores baixos denotam um diminuto grau de pertença.

Esta função é chamada *função de pertença* e o conjunto por ela definido é chamado *conjunto difuso*.

Seja A um conjunto difuso definido em X . Então a sua função de pertença μ_A é usualmente

definida como:

$$\mu_A : X \rightarrow [0,1] \quad (2.2)$$

Por exemplo, uma possível função pertença de um conjunto difuso de números reais próximos de 0 é $\mu_A(x) = \exp(-x^2)$, representada na Figura 32.

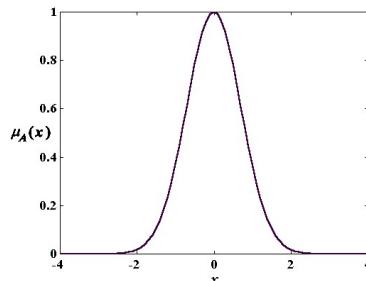


Figura 32- Exemplo de uma função pertença "near to zero"

É notório que podemos ter uma outra função de pertença para o conjunto difuso; por exemplo $\mu_A(x) = 1/(1+10x^2)$. Estas duas e diferentes funções de pertença mostram que a sua formulação é de natureza subjetiva. Todavia, não pode ser formulado arbitrariamente. Uma estimativa qualitativa que reflecta uma dada ordem dos elementos em A pode ser suficiente. Para outros casos, a sua estimação pode ser complicada, e uma melhor aproximação passa pela utilização das poderosas técnicas de aprendizagem usadas nas redes neurais.

Em acréscimo à terminologia, símbolos e definições usados na lógica bivalente, existe um conjunto próprio da lógica difusa, que será referido em seguida de forma breve.

Um conjunto difuso A em U pode ser representado como um conjunto de pares ordenados, em que o elemento genérico x é graduado pela função de pertença :

$$A = \{ (x, \mu_A(x)) \mid x \in U \} \quad (2.3)$$

Uma variável linguística é caracterizada pelo quinteto $(x, T(x), U, G, M)$ [7], em que x é o nome da variável; $T(x)$ é o conjunto de termos (valores) da variável x , isto é, o conjunto de nomes linguísticos da variável x em que cada elemento (valor) define um número difuso no universo do discurso U ; G são as regras sintáticas da geração dos vários nomes dos termos (valores) da variável x , e M são as regras semânticas para a associação de cada valor com o seu significado.

Exemplo 2.1: A temperatura atmosférica pode ser interpretada como uma variável linguística, cujo

conjunto dos seus termos $T(\text{temperatura})$ pode ser: $T(\text{Temperatura}) = \{\text{"baixo"}, \text{"moderado"}, \text{"alto"}\}$ em que, cada termo T é caracterizado por um conjunto difuso no universo do discurso $U = [-10^\circ, 40^\circ]$. Podemos interpretar "baixa" a temperatura abaixo dos 5°C , "moderada" em torno dos 15°C , e "alta" para temperaturas superiores a 25°C . Podemos associar a esses termos os conjuntos difusos cujas funções de pertença estão representado na Figura 33.

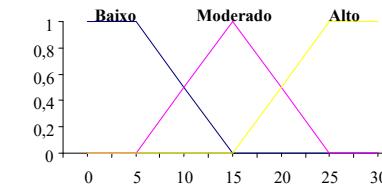


Figura 33 - Funções pertença dos conjuntos "Baixo", "Moderado", "Alto" associado à variável Temperatura

Assim, podemos definir as variáveis linguísticas (de uma forma intuitiva): se uma variável pode tomar, como seus valores, palavras na linguagem natural (por exemplo: "small", "fast", e assim sucessivamente), esta variável é definida como uma variável linguística. Estas palavras são usualmente utilizadas como nomes de conjuntos difusos. A variável linguística pode assim tomar para seus valores palavras ou números. Outros exemplos sobre variáveis podem ser encontrados em [8].

Refere-se em seguida a noção de *hedge* relacionada com a forma mais ou menos abrupta da fronteira de um conjunto.

Seja F um conjunto difuso em X . Então, "muito F " é definido como um conjunto difuso em X com a função de pertença na forma: $\mu_{\text{muito } F}(x) = (\mu_F(x))^2$. De forma similar, "mais ou menos F " é um conjunto difuso em X com a função de pertença na forma: $\mu_{\text{mais ou menos } F}(x) = (\mu_F(x))^{\frac{1}{2}}$. No primeiro caso, o conjunto difuso $F^2 = \text{"muito } F"$ constitui um sub-conjunto de F , mas com menor ambiguidade de limites na sua fronteira. A mesma ilação se retira para o conjunto difuso $F^{\frac{1}{2}} = \text{"mais ou menos } F"$, em que o conjunto "mais ou menos F " contém o conjunto F , com uma maior ambiguidade de limites na sua fronteira.

O suporte do conjunto difuso A no universo do discurso é o conjunto bivalente que contém todos os elementos de X que têm um valor não nulo da função de pertença.

$$\text{supp } A = \{ x \in X \mid \mu_A(x) > 0 \} \quad (2.4)$$

O conjunto difuso vazio tem como conjunto suporte o conjunto vazio. O conjunto difuso

A , cujo suporte é um único ponto em U com $\mu_A(x) = 1$, é referido como um conjunto singular (“fuzzy singleton”). O elemento $x \in X$ ao qual $\mu_A(x) = 0.5$ é chamado de ponto de *cruzamento* (“crossover”). O *núcleo (kernel)* do conjunto A consiste nos elementos x cujo valor de pertença é 1, ou seja,

$$Ker(A) = \{x \mid \mu_A(x) = 1\} \quad (2.5)$$

Usando o suporte do conjunto difuso A , este pode representar-se como:

$$A = \mu_1(x_1)/x_1 + \dots + \mu_n(x_n)/x_n = \sum_{i=1}^n \mu_i(x_i)/x_i$$

em que $+$ representa a operação união dos elementos e μ_i o seu grau de pertença. Se X não for discreto, mas contínuo num intervalo de números reais, pode usar-se a notação $A = \int_U \mu_F(x)/x$, em que \int representa a união dos elementos de A .

A *altura (height)*, H , de um conjunto difuso A é o maior dos valores da função de pertença, para todos os elementos do conjunto, ou seja, o supremo de $\mu_A(x)$ em U : $H(A) = \sup_x \mu_A(x)$. Um conjunto diz-se normalizado quando pelo menos um dos seus elementos tem valor 1, valor máximo, da função de pertença. De outra forma, o conjunto diz-se sub-normal.

Um “ α -cut” (ou *conjunto de nível α*) de um conjunto difuso A é um conjunto bivalente A_α que contém todos os elementos do conjunto universo X com um valor da função de pertença de A maior ou igual a α , ou seja:

$$A_\alpha = \{x \in X \mid \mu_A(x) \geq \alpha\} \quad (2.6)$$

O conjunto de todos os níveis $\alpha \in (0,1]$, de um dado conjunto A , que representam conjuntos distintos “ α -cut”, é designado como o conjunto de nível de A , ou seja:

$$A_\lambda = \{\alpha \mid \mu_A(x) = \alpha \text{ para alguns } x \in X\} \quad (2.7)$$

Com o conceito de “ α -cut” é possível introduzir uma propriedade importante dos conjuntos difusos, denominada *princípio de resolução*.

Teorema 2.1: Seja A um conjunto difuso no universo do discurso X . Então, a função de pertença de A pode ser expandida em termos dos seus conjuntos “ α -cut”, de acordo com

$$\mu_A(x) = \sup_{\alpha \in (0,1]} [\alpha \wedge \mu_{A_\alpha}(x)] \quad \forall x \in X \quad (2.8)$$

em que \wedge representa a operação mínimo e $\mu_{A_\alpha}(x)$ é a função característica do conjunto

bivalente A_α ,

$$\mu_{A_\alpha}(x) = \begin{cases} 1 & \text{sse } x \in A_\alpha \\ 0 & \text{outros} \end{cases}$$

Seja A um conjunto difuso no universo do discurso X . Seja αA_α o conjunto difuso com a função de pertença

$$\mu_{\alpha A_\alpha}(x) = [\alpha \wedge \mu_{A_\alpha}(x)] \quad \forall x \in X \quad (2.9)$$

Então, com base no *princípio da resolução*, o conjunto A pode ser expresso na forma

$$A = \bigcup_{\alpha \in \Lambda_A} \alpha \cdot A_\alpha \quad \text{ou} \quad A = \int_0^1 \alpha \cdot A_\alpha \quad (2.10)$$

ou seja, o princípio da resolução indica que o conjunto A pode ser decomposto em αA_α conjuntos, $\alpha \in (0,1]$, ou inversamente, o conjunto A pode ser definido como a união dos seus αA_α (teorema da representação).

Um conjunto difuso é *convexo* se e só se (*sse*) todos os seus conjuntos “ α -cut” forem convexos. Equivalentemente, pode dizer-se que um conjunto difuso A é convexo se e só se (*sse*) verificar a inequação:

$$\mu_A(\lambda x_1 + (1-\lambda)x_2) \geq \min[\mu_A(x_1), \mu_A(x_2)] \quad \forall x_1, x_2 \in X \text{ e } \lambda \in [0,1] \quad (2.11)$$

A Figura 34 a) ilustra um conjunto difuso convexo, enquanto que a Figura 34 b) ilustra um conjunto difuso não convexo.

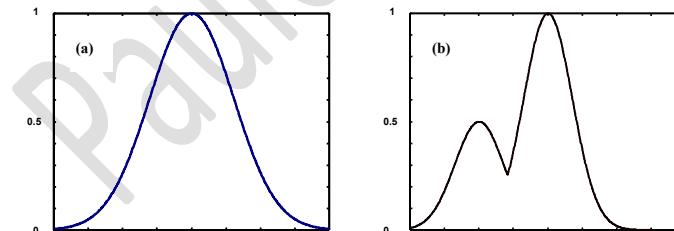


Figura 34 - Exemplo de um conjunto convexo (a) e não convexo (b)

Similarmente à cardinalidade de um conjunto bivalente, que é definida como o número de elementos do conjunto, a cardinalidade do conjunto difuso A é o somatório dos graus de pertença de todos os elementos de x em A . Isto é,

$$|A| = \sum_{x \in X} \mu_A(x) \quad (2.12)$$

3. Operações difusas

No caso da lógica bivalente, existe um conjunto de operações básicas: *not* (*negação*); *and* (*e*) e *or* (*ou*). Qualquer outra operação lógica pode ser expressa apenas com base em combinações destas operações. No caso da lógica difusa existe um número ilimitado de operações. No entanto, nem todas podem ser expressas em termos de operações básicas. Serão abordadas em seguida, as mais importantes.

Como casos especiais de operações difusas complemento, intersecção, e união, temos:

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \quad (2.13)$$

$$\mu_{A \cap B}(x) = \min[\mu_A(x), \mu_B(x)] \quad (2.14)$$

$$\mu_{A \cup B}(x) = \max[\mu_A(x), \mu_B(x)] \quad (2.15)$$

para todo o $x \in X$. Estas operações são designadas como *operações difusas standard*.

Como seria de esperar, estas operações correspondem a operações de conjuntos disjuntos, quando os valores de pertença estão confinados ao conjunto {0,1}, como facilmente se verifica para as *operações difusas standard*. As operações difusas deverão constituir uma generalização das correspondentes operações de conjuntos disjuntos.

Os correspondentes difusos das operações bivalentes *not*, *and* e *or* são as operações *negação difusa*, *T norm*, e *S norm*. Contrariamente à lógica bivalente, na lógica difusa não existe uma forma única de as implementar. Diferentes funções podem ser apropriadas para representar estas operações em contextos diferentes. Qualquer que seja a função escolhida, ela deverá respeitar um conjunto de axiomas [9], que lhe conferem um conjunto de propriedades.

A intersecção difusa (*T norm*) e a união difusa (*S norm*) constituem operações de agregação possíveis de conjuntos difusos. Elas são, porém, as únicas formas de agregação associativa de conjuntos difusos. Devido à falta de associatividade, as restantes operações de agregação devem ser definidas como funções de n argumentos (para $n > 2$). Operações de agregação que, para qualquer valores de pertença dados a_1, a_2, \dots, a_n , produzam um grau de pertença que está compreendido entre os valores de $\min(a_1, a_2, \dots, a_n)$ e $\max(a_1, a_2, \dots, a_n)$ são chamados de *operações de media*. O resultado desta operação é um conjunto difuso que é maior que qualquer conjunto intersecção e menor que qualquer conjunto união.

A capacidade de determinar as funções de pertença e operações difusas apropriadas, no contexto de cada uma particular aplicação, é crucial na utilização prática da teoria dos conjuntos difusos.

3.1. Funções complemento difuso

Seja A um conjunto difuso em X . Então, por definição, $\mu_A(x)$ é interpretado como o grau de pertença de x em A . \bar{A} representa o complemento difuso de A , em que $\mu_{\bar{A}}(x)$ pode ser interpretado não só como o grau de x pertencer a \bar{A} , mas também como o grau x não pertencer a A . Similarmente, $\mu_A(x)$ também pode ser interpretado como o grau de pertença de x não pertencer a \bar{A} .

Definição 2.1: A função complemento do conjunto A , cA ou \bar{A} , pode ser definida pela função

$$c: [0,1] \rightarrow [0,1]$$

que assinala um valor $c(\mu_A(x))$ cada valor de pertença $\mu_A(x)$ de um qualquer conjunto difuso A .

Para que uma função complemento c ou $(\bar{\cdot})$ corresponda ao significado de complemento difuso deverá satisfazer pelo menos os axiomas $N1$ e $N2$, sendo desejável que, igualmente, cumpra os axiomas $N3$ e $N4$.

A. Axiomas das funções complemento NOT:

$\bar{\cdot} : [0,1] \rightarrow [0,1];$
$(N1) : \bar{0} = 1;$
$(N2) : a_1 < a_2 \Rightarrow \bar{a}_1 \geq \bar{a}_2$
$(N3) : A operação \bar{\cdot} é uma função contínua$
$(N4) : (\bar{\bar{a}}) = a \text{ para } \forall a \in [0,1];$

(2.16)

A tabela 2.1 apresenta exemplos típicos de operações *complemento difusos*.

$\bar{a} = 1 - a$	Complemento de 1
$\bar{a} = \frac{1-a}{1+\lambda a}, \quad -1 < \lambda < \infty$	Complemento λ (Classe Sugeno)
$\bar{a} = (1-a^w)^{\frac{1}{w}}, \quad 0 < w < \infty$	Complemento w (Classe de Yager)

Tabela 2.1 - Algumas das funções *Complemento difuso*.

Quando os parâmetros sejam $\lambda=0$, no complemento λ , e $w=1$, no complemento w , ambas as funções ficam na forma de *complemento de 1*, ou seja, o complemento difuso padronizado. Na Figura 35 estão representadas várias funções complemento w de Yager.

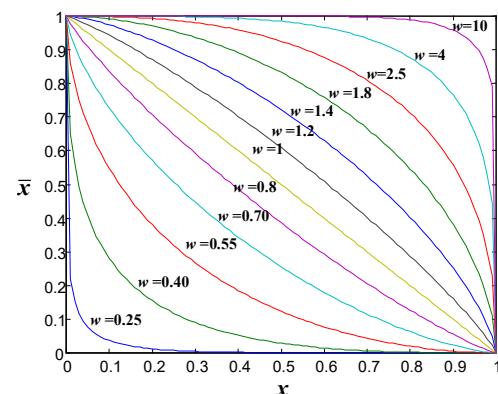


Figura 35 – Funções Complemento w (Classe de Yager) para vários valores de w .

Definem-se de seguida as operações de intersecção e união de conjuntos difusos, que são muitas vezes referidas como *normas triangulares* (“*Triangular norms – T-norms*”) e *co-normas triangulares* (“*triangulares conorms – T-conorms ou S-norms*”) [10][11].

3.2. Intersecção de conjuntos difusos

Sejam A e B dois conjuntos difusos em X . $\mu_A(x)$ e $\mu_B(x)$ representam o grau de pertença de x em A e B respectivamente.

Definição 2.2: A intersecção de dois conjuntos difusos A e B é especificada por uma operação binária no intervalo unitário

$$*: [0,1] \times [0,1] \rightarrow [0,1]$$

ou seja, para cada elemento x do conjunto Universo X , esta função toma como argumentos os par de pertença de x em A e B . Resulta da aplicação desta função um grau de pertença de x no conjunto de intersecção de A com B , dado por:

$$\mu_{A \cap B}(x) = \mu_A(x) * \mu_B(x)$$

A intersecção difusa ou *t-norm* é uma operação binária no intervalo unitário que satisfaz pelo menos os seguintes axiomas:

3.2.1. Axiomas das funções T- norm:

$$\begin{aligned} * &: [0,1] \times [0,1] \rightarrow [0,1]; \\ (T1) &: a * 1 = a, \quad a * 0 = 0 \text{ para } \forall a \in [0,1]; \\ (T2) &: a_2 \leq a_3 \Rightarrow a_1 * a_2 \leq a_1 * a_3 \\ (T3) &: a_1 * a_2 = a_2 * a_1; \\ (T4) &: a_1 * (a_2 * a_3) = (a_1 * a_2) * a_3; \end{aligned} \quad (2.17)$$

em que T1 representa as condições de fronteira, T2 de monotocidade, T3 de comutatividade e T4 de associatividade.

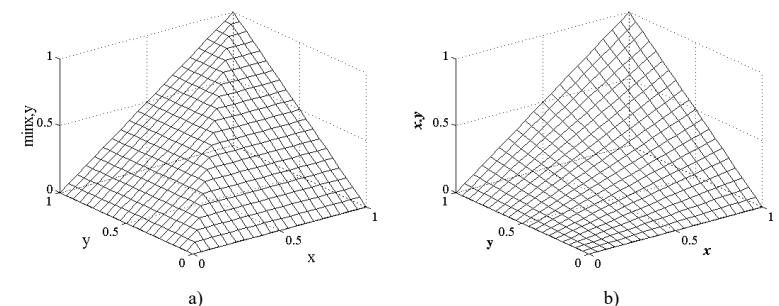
Adicionalmente, três novos requerimentos de continuidade (T5), de subpotenciação e de monotocidade restrita (T5) poderão ser expressos:

$$\begin{aligned} (T5) &: * \text{ é uma função contínua} \\ (T6) &: a_1 \leq a_3 \text{ e } a_2 \leq a_4 \Rightarrow a_1 * a_2 \leq a_3 * a_4 \\ (T7) &: a * a < a \end{aligned}$$

Na tabela 2.2 estão algumas das funções *T-norm* mais utilizadas, e na Figura 36 a sua representação gráfica.

$a \wedge b = \min\{a,b\}$	Produto Lógico ou Intersecção
$A \bullet b = a \cdot b$	Produto Algébrico
$a \odot b = \max\{0, a+b-1\}$	Produto Limitado
$a \wedge b = \begin{cases} a & \text{se } b=1 \\ b & \text{se } a=1 \\ 0 & \text{se } a,b<1 \end{cases}$	Produto Drástico
Com: $0 \leq a \wedge b \leq a \odot b \leq a \bullet b \leq a \wedge b \leq 1$	

Tabela 2.2 - Algumas das funções *T-norm* mais usadas.



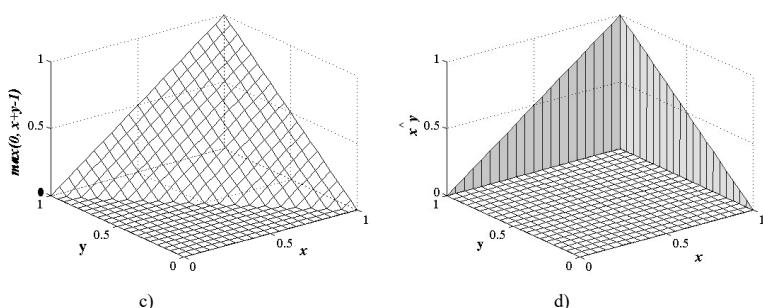


Figura 36 – Funções T -norm: a) Intersecção; b) Produto algébrico; c) Produto limitado d) Produto drástico.

Muitas outras funções T -norm podem ser definidas. Por exemplo Yager [12], Dubois e Prade [10], Frank, Weber, Schweizer, e outros [13] sugerem um número infinito de funções T -norm, usando um único parâmetro real. Algumas destas funções estão representadas na tabela 2.3. Na Figura 37 estão representados graficamente os resultados da operação T -norm da classe de Yager, para vários parâmetros w . Todavia, qualquer uma destas funções está situada entre dois limites: o *produto drástico* e o *produto lógico*.

$\frac{a \cdot b}{\max \{a, b, \alpha\}}, \quad 0 < \alpha < 1$	Dubois e Prade [1980]
$(a \cdot b)^{(1-\lambda)} [1 - (1-a)(1-b)]^\lambda, \quad 0 \leq \lambda \leq 1$	Zimmermann e Zysno [1980]
$1 - \min \left\{ 1, \left((1-a)^w + (1-b)^w \right)^{\frac{1}{w}} \right\}, \quad 0 < w < \infty$	Yager [1980]

Tabela 2.3 - Algumas funções T -norm..

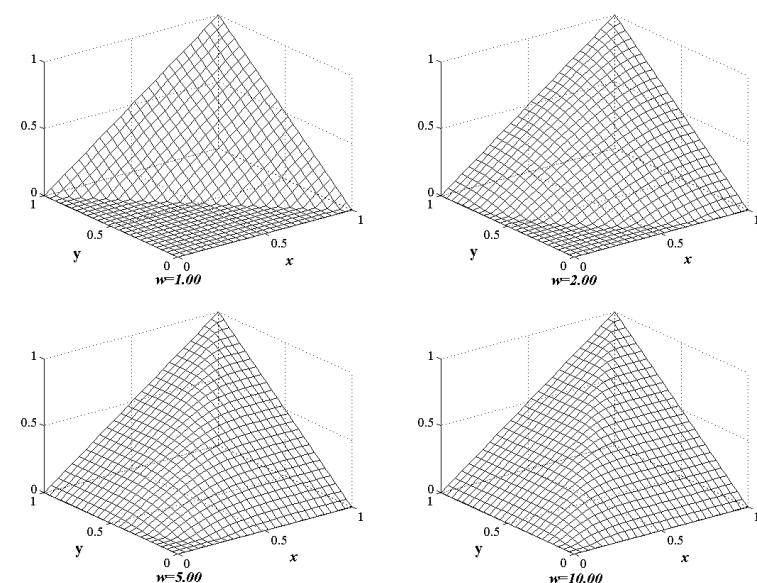
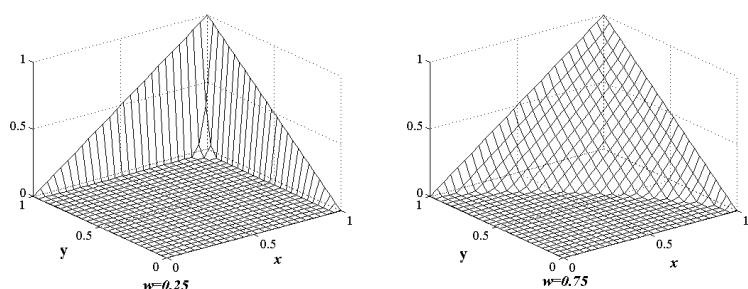


Figura 37 – Exemplos da operação de intersecção difusa, segundo a classe de Yager.

3.3. Reunião de conjuntos difusos

Tal como na intersecção difusa, a união difusa pode ser especificada por qualquer função que obedeça à seguinte definição.

Definição 2.3: A união de dois conjuntos difusos A e B é especificado por uma operação binária no intervalo unitário

$$*: [0,1] \times [0,1] \rightarrow [0,1]$$

ou seja, para cada elemento x do conjunto Universo X , esta função toma como argumentos os pares de pertença de x em A e B . Resulta da aplicação desta função um grau de pertença de x no conjunto de união de A com B , dado por:

$$\mu_{A \cup B}(x) = \mu_A(x) + \mu_B(x)$$

A união difusa ou S-norm é uma operação binária no intervalo unitário que satisfaz pelo menos os seguintes axiomas:

3.3.1.1. Axiomas das funções *T-conorm* ou *S-norm*:

$\dot{+} : [0,1] \times [0,1] \rightarrow [0,1]$;
(S1) : $a + 1 = 1$, $a + 0 = a$ para $\forall a \in [0,1]$;
(S2) : $b \leq d \Rightarrow a + b \leq a + d$
(S3) : $a + b = b + a$;
(S4) : $a + (\dot{b} + c) = (\dot{a} + b) + c$;

(2.18)

Este conjunto de axiomas assegura as condições de fronteira (T1), de monotonicidade (T2), de comutatividade (T3) e de associatividade (T4). Adicionalmente, três novos requerimentos, de continuidade (T5), de subpotenciação e de monotonicidade restrita (T5), poderão ser assegurados pelos seguintes axiomas:

- (S5) : * é uma função contínua
- (S4) : $a \leq c$ e $b \leq d \Rightarrow a + b \leq c + d$
- (S7) : $a + a > a$

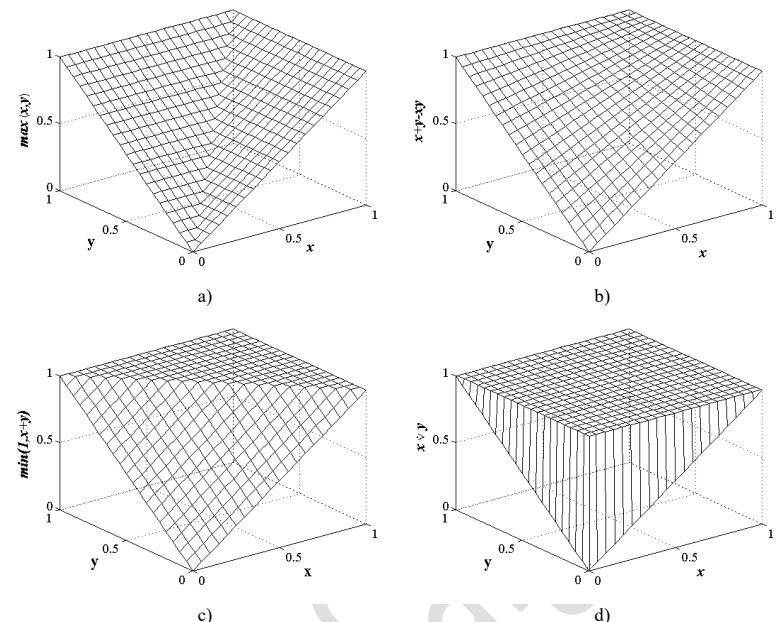
Na tabela 2.4 indicam-se algumas das funções *T-conorm* mais utilizadas e na Figura 38 a sua representação gráfica.

$x \vee y = \max\{x,y\}$	Soma Lógica ou União
$x \dot{+} y = x + y - xy$	Soma Algébrica
$x \oplus y = \min\{1, x+y\}$	Soma Limitada
$x \dot{\vee} y = \begin{cases} x & \text{se } y=0 \\ y & \text{se } x=0 \\ 1 & \text{se } x,y>0 \end{cases}$	Soma Drástica
Com: $0 \leq x \vee y \leq x \dot{+} y \leq x \oplus y \leq x \dot{\vee} y \leq 1$	

Tabela 2.4 - Algumas das funções *T-conorm* mais usadas

Os limites inferior e superior das operações *S-norm* são respetivamente a *soma lógica* e a *soma drástica*.

Na tabela 2.5 estão representadas mais algumas funções *S-norm*, todas elas com a particularidade de serem também função de um parâmetro. Na Figura 39 estão representadas as superfícies, para diferentes valores do parâmetro w, da operação S-norm da classe de Yager.

Figura 38 – Funções *S-norm*: a) União; b) Soma algébrica; c) Soma limitada d) Soma drástica.

$\frac{x + y - x \cdot y - \min\{x, y, 1 - \alpha\}}{\max\{1 - x, 1 - y, \alpha\}}$, $0 < \alpha < 1$	Dubois e Prade [1980]
$\min\left\{1, \left(x^w + y^w\right)^{\frac{1}{w}}\right\}$, $0 < w < \infty$	Yager [1980]

Tabela 2.5 - Algumas funções *S-norm*..

Várias funções de negação *difusa*, *T-norm* e *S-norm* podem ser usadas. Como é compreensível, é conveniente empregar apenas aquelas que satisfaçam as leis de Morgan *difusas* [14]:

$$\overline{(a + b)} = \bar{a} * \bar{b} \quad (2.19)$$

$$\overline{(a * b)} = \bar{a} + \bar{b} \quad (2.20)$$

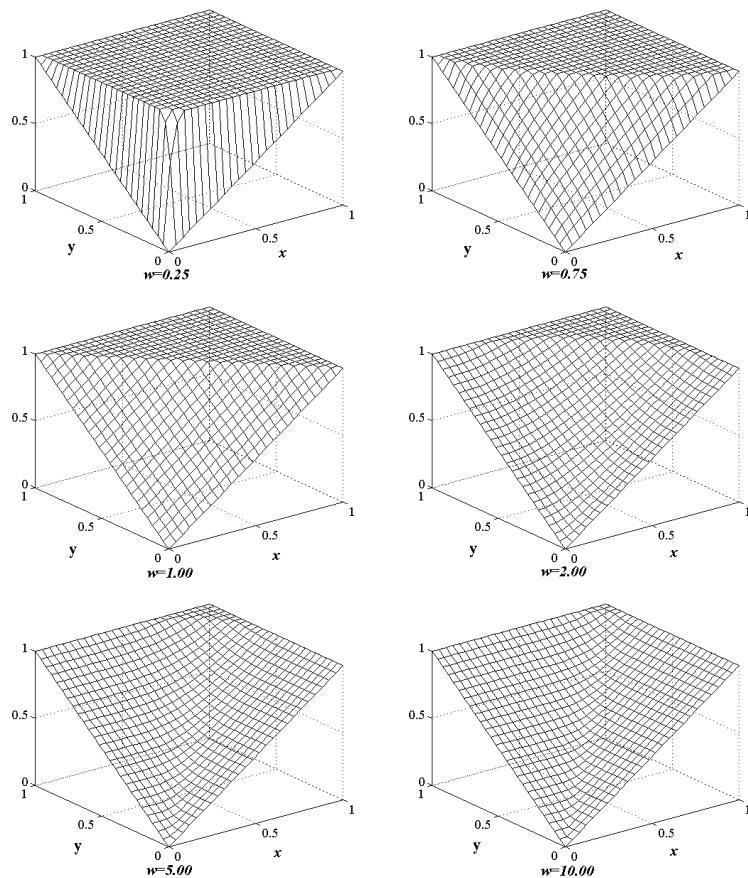


Figura 39 – Exemplos da operação de reunião difusa, segundo a classe de Yager.

Quando as equações (2.19) e (2.20) são verificáveis, as operações *T-norm* e *S-norm* são duais em relação à negação difusa. São exemplos os pares de funções *produto lógico* e *soma lógica*, *produto algébrico* e *soma algébrica*, *produto limitado* e *soma limitada*, *produto drástico* e *soma drástica*, em relação à negação difusa *complementar de 1*.

Na maior parte das aplicações da teoria dos conjuntos difusos são usadas as operações:

- 1) *T-norm* para intersecção: *min* ou *produto algébrico*;
- 2) *S-norm* para a união: *max* ou *soma algébrica*.
- 3) Negação difusa: $\mu(\bar{x}) = 1 - \mu(x)$

4. Princípio da extensão

O princípio da extensão, introduzido por Zadeh [15], é uma ferramenta importante da teoria dos conjuntos difusos. Este princípio permite a generalização dos conceitos da matemática bivalente para o ambiente de trabalho dos conjuntos difusos e estende o mapeamento de pontos-para-pontos em mapeamentos entre conjuntos difusos. Ou seja, permite que qualquer função f que mapeia pontos $\vec{x} = (x_1, \dots, x_n)$ do conjunto U num ponto do conjunto V , seja generalizado no mapeamento dos subconjuntos difusos em U num subconjunto difuso em V .

Seja a função $f: U \rightarrow V$ e o conjunto difuso A em U , em que $A = \mu_1/x_1 + \dots + \mu_n/x_n$. O princípio da extensão estabelece que:

$$\begin{aligned} f(A) &= f(\mu_1/x_1 + \dots + \mu_n/x_n) \\ &= \mu_1/f(x_1) + \dots + \mu_n/f(x_n) \end{aligned} \quad (2.17)$$

Se mais de um elemento de U é mapeado por f no mesmo elemento y de V , então é tomado o valor de máximo grau de pertença, isto é:

$$\mu_{f(A)}(y) = \max_{\substack{x_i \in U \\ f(x_i) = y}} [\mu_A(x_i)] \quad (2.18)$$

em que x_i são os elementos que são mapeados no mesmo y . Muitas vezes, a função f é tal que mapeia o ponto $\vec{x} = (x_1, \dots, x_n)$ em U num ponto em V .

Seja U o produto Cartesiano do universo $U = U_1 \times \dots \times U_n$ e A_1, A_2, \dots, A_n sejam n conjuntos difusos em U_1, U_2, \dots, U_n , respectivamente. A função f mapeia um ponto $\vec{x} = (x_1, \dots, x_n)$ do conjunto U num ponto y do conjunto do conjunto V , isto é, $y = f(x_1, x_2, \dots, x_n)$. O princípio da extensão permite que a função $f(x_1, x_2, \dots, x_n)$ seja generalizada para atuar sobre n subconjuntos difusos em U, A_1, A_2, \dots, A_n , tal que:

$$B = f(A) \quad (2.19)$$

em que B é a imagem difusa (conjunto difuso) de A_1, A_2, \dots, A_n , através de $f(\cdot)$.

O conjunto difuso B é definido por:

$$B = \{(y, \mu_B(y)) \mid y = f(x_1, x_2, \dots, x_n), (x_1, x_2, \dots, x_n) \in U\} \quad (2.20)$$

em que:

$$\mu_B(y) = \sup_{\substack{(x_1, x_2, \dots, x_n) \in U \\ y = f(x_1, x_2, \dots, x_n)}} \min[\mu_{A_1}(x_1), \mu_{A_2}(x_2), \dots, \mu_{A_n}(x_n)] \quad (2.21)$$

5. Relação e composição

O conceito de *relação difusa* [16][17] surge como uma extensão natural aos conjuntos difusos e tem um papel importante na teoria de tais conjuntos e suas aplicações.

As relações difusas são uma generalização das relações binárias e permitem a existência de vários graus de associação entre elementos. Refletem o grau de associação expresso, do mesmo modo que o grau de pertença dos elementos num conjunto difuso, através da respetiva função de pertença. Assim, as relações difusas são, de facto, conjuntos difusos.

Seja o espaço produto cartesiano para a família dos conjuntos bivalentes $\{x_i \mid i \in N_n\}$, representado por $X_1 \times X_2 \times \dots \times X_n$, ou simplesmente por $\prod_{i \in N} X_i$. Os elementos do espaço produto Cartesiano $x = (x_1, x_2, \dots, x_n)$ de n conjuntos bivalentes são tais que: $x_i \in X_i$ para todos os $i \in N_n$. Assim: $\prod_{i \in N} X_i = \{(x_1, x_2, \dots, x_n) \mid x_i \in X_i \forall i \in N_n\}$. É possível que todos os conjuntos X_i sejam iguais (é vulgar ser $X_i \equiv \mathcal{U}$ para todo o $i \in N_n$).

A relação dos conjuntos bivalentes X_1, X_2, \dots, X_n é um subconjunto do espaço produto cartesiano, referido como $R(X_1, X_2, \dots, X_n) \subseteq X_1 \times X_2 \times \dots \times X_n$. Pode interpretar-se que a relação R existe no espaço produto $X_1 \times X_2 \times \dots \times X_n$ se um ponto $\vec{x} = (x_1, x_2, \dots, x_n)$ pertence ao conjunto $R(X_1, X_2, \dots, X_n)$; doutra forma a relação R não existe no espaço produto. Como a relação é ela mesma um conjunto, são-lhe aplicáveis os mesmos conceitos.

Se a relação R é binária, pode ser definida por uma função característica, que assume apenas os valores 0 ou 1.

$$R(x_1, x_2, \dots, x_n) = \begin{cases} 1 & \text{se } (x_1, x_2, \dots, x_n) \in R \\ 0 & \text{outros} \end{cases} \quad (2.22)$$

Exemplo 2.2: Seja R uma relação dos seguintes conjuntos: $X = \{\text{Portugal (P), EUA, França (Fr), Canadá (C)}\}$; $Y = \{\text{escudo, dólar, euro}\}$; $Z = \{\text{Europa, América}\}$. A relação R associa o país à moeda e ao continente. Assim, a relação R pode ser representada pela seguinte matriz tridimensional função de pertença:

	P	EUA	Fr	C
escudo	1	0	0	0
dolar	0	0	0	0
EURO	1	0	1	0

	Europa	America	
escudo	0	0	0
dolar	0	1	0
EURO	0	0	0

Este conceito pode ser generalizado para permitir vários graus, ou pesos, nas relações. Assim, uma relação difusa é definida como um conjunto difuso no espaço produto dos

conjuntos X_1, X_2, \dots, X_n , em que cada elemento $\vec{x} \in X_1 \times X_2 \times \dots \times X_n$ pode ter vários graus de pertença, geralmente representados por um valor real no intervalo $[0,1]$:

$$\mu_R : \prod_{i \in N} X_i \rightarrow [0,1] \quad (2.23)$$

Exemplo 2.3: Seja R uma relação difusa que relaciona a distância entre dois conjuntos de cidades: $X = \{\text{Coimbra, Lisboa, Porto}\}$; $Y = \{\text{Porto, Vila Real}\}$ e representada pela seguinte matriz tridimensional função de pertença:

$$\begin{matrix} & P & V \\ C & \left[\begin{array}{cc} 0.3 & 0.6 \\ 0.8 & 1 \end{array} \right] \\ L & \left[\begin{array}{cc} 0 & 0.2 \end{array} \right] \\ P & \left[\begin{array}{cc} 0 & 0.2 \end{array} \right] \end{matrix}$$

A relação difusa, $R(X_1, X_2, \dots, X_n)$ é um conjunto difuso no espaço produto $X_1 \times X_2 \times \dots \times X_n$ e é caracterizado pela função de pertença $\mu_R(x_1, x_2, \dots, x_n)$, em que $x_i \in X_i$ ($i=1, \dots, n$), isto é:

$$R(x_1, x_2, \dots, x_n) = \{(x_1, x_2, \dots, x_n), \mu_R(x_1, x_2, \dots, x_n) \mid (x_1, x_2, \dots, x_n) \in \prod_{i \in N} X_i\}$$

ou

$$R(x_1, x_2, \dots, x_n) = \bigcup_{(x_1, x_2, \dots, x_n) \in \prod_{i \in N} X_i} \mu_R(x_1, x_2, \dots, x_n) / (x_1, x_2, \dots, x_n), \quad (x_1, x_2, \dots, x_n) \in \prod_{i \in N} X_i$$

O domínio da relação binária difusa $R(X, Y)$ é o conjunto difuso “Dom $R(X, Y)$ ” com a função de pertença:

$$\mu_{\text{dom}R}(x) = \max_{y \in Y} \mu_R(x, y) \quad \text{para cada } x \in X \quad (2.25)$$

A gama (“range”) da relação binária difusa $R(X, Y)$ é o conjunto difuso “ran $R(X, Y)$ ” com a função de pertença:

$$\mu_{\text{ran}R}(y) = \max_{x \in X} \mu_R(x, y) \quad \text{para cada } y \in Y. \quad (2.26)$$

Analogamente à altura do conjunto difuso, a altura da relação difusa R é um valor $H(R)$ definido por:

$$H(R) = \sup_{x \in X} \sup_{y \in Y} \mu_R(x, y) \quad (2.27)$$

Se $H(R)=1$, então R é uma relação normal; de outra forma é uma relação difusa sub-normal.

Similarmente ao princípio de resolução dos conjuntos difusos, cada relação binária difusa

$R(X, Y)$ pode ser representada por:

$$R = \bigcup_{\alpha \in \Lambda_R} \alpha R_\alpha \quad (2.28)$$

em que Λ_R é o conjunto nível de R e αR_α é definido como na equação 2.8. Assim, a relação difusa pode ser representada como a série de relações bivalentes que correspondem aos seus “ α - cuts”.

Como a relação difusa é ela mesma um conjunto definido no espaço produto, continuam válidas a teoria dos conjuntos e as operações algébricas anteriormente definidas (T -norm, S -norm, complemento difusa, ...), como se realça no exemplo 2.4.

Exemplo 2.4: Seja S uma relação difusa que relaciona a qualidade das estradas (em desconforto) entre dois conjuntos de cidades: $X=\{\text{Coimbra}, \text{Lisboa}, \text{Porto}\}$; $Y=\{\text{Porto}, \text{Vila Real}\}$ e representada pela seguinte matriz tridimensional função de pertença:

$$\begin{matrix} & P & V \\ C & [0.4 \quad 0.9] \\ L & [0.5 \quad 0.8] \\ P & [1 \quad 0.6] \end{matrix}$$

Pretende-se obter a relação composta que defina “a menor distância e o melhor conforto entre as cidades x e y ”. A função da função pertença composta pode ser obtida usando uma apropriada operação t -norm (p. ex. min): $\mu_{R \circ S}(x,y) = \min[\mu_R(x,y), \mu_S(x,y)]$. Então, a função pertença da relação composta (sobre o mesmo Universo) será obtida por:

Por exemplo, $\mu_{R \circ S}(L,P) = \min[\mu_R(L,P), \mu_S(L,P)] = \min(0.8, 0.5) = 0.5$

$$\mu_{R \circ S}(x,z) = \begin{matrix} & P & V \\ C & [0.3 \quad 0.6] \\ L & [0.8 \quad 0.8] \\ P & [0 \quad 0.6] \end{matrix}$$

Uma importante operação sobre relações difusas é a composição de relações. Basicamente, existem dois tipos de operações de composição: *max-min* e *min-max*. Estas composições podem ser aplicadas em composições *relação-relação* e *conjunto-relação*.

Seja a composição entre duas relações $P(X,Y)$ e $Q(Y,Z)$, com apenas um conjunto Y em comum. A composição destas duas relações pode ser dada por [16]:

$$R(X,Z) = P(X,Y) \circ Q(Y,Z) \quad (2.29)$$

e é definida como um subconjunto em $U \times W$ tal que $(x,z) \in R$ se e só se existir pelo menos um

$y \in Y$ tal que $(x,y) \in P$ e $(y,z) \in Q$.

Duas das composições muito comuns são:

$$\mu_{P \circ Q}(x,z) = \max_{y \in Y} \min[\mu_P(x,y), \mu_Q(y,z)] \quad \text{Composição max-min} \quad (2.30)$$

$$\mu_{P \oplus Q}(x,z) = \min_{y \in Y} \max[\mu_P(x,y), \mu_Q(y,z)] \quad \text{Composição min-max} \quad (2.31)$$

Para todo o $x \in X$ e $z \in Z$.

A composição dual da composição *max-min* é a composição *min-max*:

$$\overline{P(X,Y) \square Q(Y,Z)} = \overline{P(X,Y)} \circ \overline{Q(Y,Z)}$$

A composição *max-min* é geralmente mais usada. A equação 2.30 indica que a composição *max-min* da relação difusa pode ser interpretada como indicando a maior ligação da cadeia de relações existentes entre os elementos X e Z .

A composição *max-min* pode ser generalizada pela substituição do operador *min* por qualquer outra operação T -norm. Em particular, quando é usado o produto algébrico, resulta na composição *max-produto*, representada como $P(X,Y) \odot Q(Y,X)$, e definida por:

$$\mu_{P \odot Q}(x,z) = \max_{y \in Y} [\mu_P(x,y) \cdot \mu_Q(y,z)] \quad \text{Composição max-produto} \quad (2.32)$$

Para todo o $x \in X$ e $z \in Z$.

Pode provar-se que as composições (2.30) e (2.32) são idênticas, quando as relações P e Q são bivalentes. A resolução de (2.30) e (2.32) pode seguir a analogia do produto entre duas matrizes em que, em vez da multiplicação, ter-se-á no caso da equação (2.30) a operação *min* e, em ambos os casos, em vez da operação soma a operação *max*.

A composição de duas relações difusas é análoga à composição bivalente, diferindo apenas no facto de as funções pertença de P e Q assumirem valores no intervalo $[0,1]$. No entanto, a fórmula usualmente empregue para a função de pertença composta, designada como composição *Sup-star*, com base nas equações (2.30) e (2.32), é a que se segue:

$$\mu_{P \circ Q}(x,z) = \sup_{y \in Y} [\mu_P(x,y) * \mu_Q(y,z)] \quad (2.33)$$

Comparando com as equações (2.30) e (2.32) pode observar-se que as operações *min* e *produto* têm como extensão natural a operação T -norm e a operação *max* o supremo, e usada no exemplo 2.5.

Exemplo 2.5: Seja a relação difusa S do exemplo 2.3 e U uma relação difusa que relaciona a qualidade das estradas entre dois conjuntos de cidades: $Y=\{\text{Porto, Vila Real}\}$ e $Z=\{\text{Braga, Chaves}\}$ representada pela seguinte matriz bi-dimensional função de pertença:

$$\begin{array}{c} \text{B} \quad \text{Ch} \\ \text{P} \quad \begin{bmatrix} 0.8 & 0.4 \\ 0.2 & 0.6 \end{bmatrix} \\ \text{V} \end{array}$$

Pretende-se obter a relação composta que defina "a menor distância entre as cidades x e z , passando obrigatoriamente numa cidade $y \in Y$ ". Como o Universo dos discursos são discretos, o supremo da eq. (1.24) reduz-se a um máximo, e assim podemos efectuar a composição através das duas mais conhecidas composições: max-min ou max-produto, com resultados diferentes uma da outra.

Por exemplo a função de pertença composta pode ser obtida usando a composição max-produto:

$$\mu_{S \circ U}(x,z) = \max[\mu_S(x,y) \cdot \mu_U(y,z)]. \text{ Então:}$$

$$\mu_{S \circ U}(x,z) = \begin{bmatrix} 0.4 & 0.9 \\ 0.5 & 0.8 \\ 1 & 0.6 \end{bmatrix} \circ \begin{bmatrix} 0.8 & 0.4 \\ 0.2 & 0.6 \end{bmatrix} = \begin{bmatrix} 0.32 & 0.54 \\ 0.40 & 0.48 \\ 0.80 & 0.40 \end{bmatrix}$$

Qualquer tipo de composição deverá obedecer às três propriedades seguintes:

$$\begin{aligned} P \circ Q &\neq Q \circ R \\ (P \circ Q)^{-1} &= Q^{-1} \circ P^{-1} \\ (P \circ Q) \circ R &= P \circ (Q \circ R) \end{aligned} \quad (2.34)$$

A última relação sugere uma forma de compor relações complexas, associando-as duas a duas.

(METER AQUI OS EXEMPLOS E CODIGO QUE FIZ NA AULA – 2019)

Seja a composição de duas relações S e R , respetivamente entre os universos $U \times V$ e $V \times W$. É possível que S seja uma relação de U em U , com $V=U$, em que $x=y$: $S(x,x)=A(x)$. Assim, a relação S reduz-se a um conjunto difuso em U , isto é, $\mu_S(x,y)$ reduz-se a $\mu_A(x)$. Nesta situação a Figura 40 a), representativa da composição de S com R , reduz-se à Figura 40b).

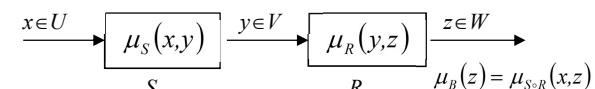
A composição do conjunto-relação de A com R , $A \circ R$, resulta num conjunto difuso em W . Chamando ao conjunto difuso resultante B , tem-se

$$A \circ R = B \quad (2.35)$$

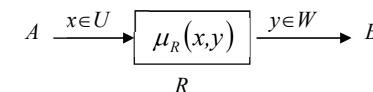
cuja função de pertença, aplicando a equação 2.33, resulta:

$$\mu_{A \circ R}(z) = \sup_{x \in U} [\mu_A(x) * \mu_R(x,z)] \quad (2.36)$$

que corresponde a uma função da variável de saída z .



a) Diagrama de Blocos interpretando a composição *Sup-star*.



b) Diagrama de blocos interpretando a composição *Sup-star* quando a primeira relação é um conjunto difuso.

Figura 40 - Diagrama de Blocos interpretando a composição *Sup-star*.

A equação 2.36 é assim chamada de *equação da relação difusa*. Vendo R como um sistema difuso, A como a entrada difusa, e B como a saída difusa, pode considerar-se a equação 2.36 como descrevendo a característica do sistema difuso através das suas relações difusas entrada-saída. Este conceito está representado na Figura 40 b).

6. Inferência difusa

A *Inferência Difusa* assume especial importância nas aplicações da lógica difusa, uma vez que constitui um conceito nuclear para os sistemas de lógica difusa.

As regras, expressões do tipo *IF-THEN* (*SE u é A, ENTÃO v é B*, em que $u \in U$ e $v \in V$), representam um caso especial de relação entre A e B , representada pela função de pertença $\mu_{A \rightarrow B}(x,y)$, e estão intimamente ligadas com a implicação lógica.

Felizmente, como foi referido na secção 2.2, existe um isomorfismo entre a lógica proposicional, a teoria dos conjuntos e a álgebra Booleana, garantindo que qualquer teorema aplicável a uma dessas áreas tem o seu dual nas restantes. Esta situação torna possível desenvolver determinado raciocínio numa das áreas e no final transpor o resultado para outra.

Em álgebra booleana, qualquer função lógica pode ser expressa apenas pela combinação de três operações fundamentais [18]: conjunção, disjunção e implicação. Pelo isomorfismo o mesmo acontece na lógica proposicional, podendo as proposições ser combinadas com outras através das operações de conjunção (\wedge , *and*), disjunção (\vee , *or*) e implicação (\rightarrow , *IF-THEN*). A tabela 2.6 indica os resultados, bem conhecidos, de algumas operações muito usadas, sobre proposições A e B .

A	B	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$	\bar{A}
V	V	V	V	V	V	F
V	F	F	V	F	F	F
F	V	F	V	V	F	V
F	F	F	F	V	V	V

V- verdadeiro; F- falso

Tabela 2.6 - Tabela de verdade de funções lógicas

É de notar o facto de a implicação não traduzir em completo a relação causa e efeito, muito usada em engenharia, como é o caso de um antecedente verdadeiro implicar uma consequência falsa.

A função característica da função implicação do conjunto A em B pode ser representada de várias formas. As mais comuns são:

$$(A \rightarrow B) \Leftrightarrow \overline{[A \wedge \bar{B}]} \Leftrightarrow \bar{A} \vee B \quad (2.37)$$

Se para a conjunção for usada a operação *T-norm min* e *produto*, resultam da equação 2.37 várias funções características, que traduzem a implicação entre as proposições A e B :

$$\begin{aligned} \mu_{A \rightarrow B}(x,y) &= 1 - \mu_{A \cap \bar{B}}(x,y) \\ &= 1 - \min[\mu_A(x), \mu_{\bar{B}}(y)] \end{aligned} \quad (2.38)$$

$$\begin{aligned} \mu_{A \rightarrow B}(x,y) &= \mu_{\bar{A} \cup B}(x,y) \\ &= \max[1 - \mu_A(x), \mu_B(y)] \end{aligned} \quad (2.39)$$

$$\mu_{A \rightarrow B}(x,y) = \mu_A(x)(1 - \mu_B(y)) \quad (2.40)$$

$$\mu_{A \rightarrow B}(x,y) = \min[1, 1 - \mu_A(x) + \mu_B(y)] \quad (2.41)$$

Várias formas de tautologia podem ser usadas para realizar inferência. Elas são referidas como regras de inferência. Duas das mais importantes regras de inferência na lógica proposicional tradicional são *Modus Ponens* e *Modus Tollens*, cujo procedimento de inferência é a seguir descrito.

- *Modus Ponens* (MP):

premissa 1: x is A

premissa 2: IF x is A , THEN y is B (2.42)

consequência: y is B

em que A e B são proposições.

Modus Ponens pode ser traduzido pelo seguinte conjunto de operações:

$$(A \wedge (A \rightarrow B)) \rightarrow B \quad (2.43)$$

- *Modus Tollens* (MT):

premissa 1: y is not B

premissa 2: IF x is A , THEN y is B (2.44)

consequência: x is not A

em que A e B são proposições.

Modus Tollens pode ser traduzido pelo seguinte conjunto de operações:

$$(\bar{B} \wedge (A \rightarrow B)) \rightarrow \bar{A} \quad (2.45)$$

Da mesma forma que foi possível estender o conceito de conjunto e relações a conjuntos e relações difusas, o mesmo sucede com o conceito de implicação difusa e das regras de inferência “*Modus Ponens*” generalizada (*GMP*) e “*Modus Tollens*” generalizada (*GMT*):

- “*Modus Ponens*” Generalizada (*GMP*):

premissa 1: x is A

premissa 2: IF x is A , THEN y is B (2.46)

consequência: y is B'

em que A' , A , B e B' são conjuntos difusos e x e y são variáveis linguísticas.

- “*Modus Tollens*” Generalizado (*GMT*):

premissa 1: y is B'

premissa 2: IF x is A , THEN y is B (2.47)

consequência: x is A'

em que A' , A , B e B' são conjuntos difusos e x e y são variáveis linguísticas.

Fukami, Mizumoto e Tanaka [19] propuseram um conjunto de critérios intuitivos para a escolha da função implicação. Esse critério intuitivo relaciona a premissa 1 e a consequência, dada pela premissa 2, de acordo com a Tabela 2.7 para a generalização *Modus Ponens* e a Tabela 2.8 para a generalização *Modus Tollens*.

Critério	$x \text{ is } A'$ (premissa 1)	$y \text{ is } B'$ (consequência)
1	$x \text{ is } A$	$y \text{ is } B$
2-1	$x \text{ is very } A$	$y \text{ is very } B$
2-2	$x \text{ is very } A$	$y \text{ is } B$
3-1	$x \text{ is more or less } A$	$y \text{ is more or less } B$
3-2	$x \text{ is more or less } A$	$y \text{ is } B$
4-1	$x \text{ is not } A$	$y \text{ is unknown}$
4-2	$x \text{ is not } A$	$y \text{ is not } B$

Tabela 2.7 - Relação intuitiva da aplicação do GMP

A satisfação dos critérios 2-2 e 3-2 é permitida porque a relação causal entre “ $x \text{ is } A$ ” e “ $y \text{ is } B$ ” não é uma premissa forte na implicação difusa. O critério 4-2, que não constitui uma relação válida na lógica formal, pode ser interpretado como: “*IF* $x \text{ is } A$ *THEN* $y \text{ is } B$, *ELSE* $y \text{ is not } B$ ”. Com efeito, esta relação é muitas vezes, assim interpretada na nossa atividade de raciocínio diário.

Critério	$y \text{ is } B'$ (premissa 1)	$x \text{ is } A'$ (consequência)
5	$y \text{ is not } B$	$x \text{ is not } A$
6	$y \text{ is not very } B$	$x \text{ is not very } A$
7	$y \text{ is not more or less } B$	$x \text{ is not more or less } A$
8-1	$y \text{ is } B$	$x \text{ is unknown}$
8-2	$y \text{ is } B$	$x \text{ is } A$

Tabela 2.8 - Relação intuitiva da aplicação do GMT

A tabela 2.8 mostra alguns dos critérios intuitivos relacionando a premissa 1 e a consequência na Generalização “modus tollens”. Similarmente aos critérios da tabela 2.7, alguns critérios na tabela 2.8 não são válidos na lógica clássica, mas são por vezes usados no nosso dia-a-dia.

De notar que os conjuntos A' e B' não são necessariamente os mesmos que os conjuntos A e B , respetivamente. Contrariamente, na lógica bivalente as regras são ativas se e só se a primeira premissa for exatamente igual ao antecedente da regra ($A' = A$), resultando uma consequência igual ao consequente da regra ($B' = B$). As regras difusas serão sempre ativas enquanto houver um grau de similaridade não-nulo entre a primeira premissa e o antecedente (podendo ser $A' \neq A$), resultando numa consequência com grau de similaridade não-nulo com a consequência da regra (podendo ser $B' \neq B$). De uma forma intuitiva, pode dizer-se que a inferência deverá produzir um valor tanto quanto possível próximo da função verdade de entrada, em lugar de um valor igual.

Resumindo, a premissa 2 da regra da equação 2.46 é uma relação que pode ser expressa como implicação $R = A \rightarrow B$. De acordo com a regra de composição de inferência, a conclusão

B' pode ser obtida pela composição do conjunto A' e a relação (aqui a relação é a implicação) $A \rightarrow B$: $B' = A' \circ R = A' \circ (A \rightarrow B)$.

Uma regra difusa é uma relação expressa por implicação difusa. Todavia, existem diferentes maneiras de definir a implicação difusa, tal como sucede com as operações AND e OR (secção 2.2.2). A determinação da função de pertença da regra difusa implica uma nova análise da operação implicação, e que não será única como a da tabela 2.6.

6.1. Implicação difusa

Seja A um conjunto difuso em U e B um conjunto difuso em V . A implicação difusa, denotada por $A \rightarrow B$, é uma relação difusa no espaço produto $U \times V: [0,1] \times [0,1] \rightarrow [0,1]$. Estas relações podem ser classificadas como se segue:

$$\bullet \text{ Conjunção Difusa: } \mu_{A \rightarrow B}(u,v) = \mu_A(u) * \mu_B(v) \quad (2.48)$$

$$\bullet \text{ Disjunção Difusa: } \mu_{A \rightarrow B}(u,v) = \mu_A(u) + \mu_B(v) \quad (2.49)$$

• Implicação Difusa:

$$\text{- Implicação Material: } \mu_{A \rightarrow B}(u,v) = \mu_{\bar{A}}(u) + \mu_B(v) \quad (2.50)$$

$$\text{- Cálculo Proposicional: } \mu_{A \rightarrow B}(u,v) = \mu_{\bar{A}}(u) + \mu_{A \circ B}(v) \quad (2.51)$$

$$\text{- Cálculo Proposicional Estendido: } \mu_{A \rightarrow B}(u,v) = \mu_{\bar{A} \times \bar{B}}(u,v) + \mu_B(v) \quad (2.52)$$

- Generalização “modus ponens”:

$$\mu_{A \rightarrow B}(u,v) = \text{Sup} \{ c \in [0,1] \mid \mu_A(u) * c > \mu_B(v) \} \quad (2.53)$$

- Generalização “modus tollens”:

$$\mu_{A \rightarrow B}(u,v) = \text{Inf} \{ c \in [0,1] \mid \mu_B(v) + c < \mu_A(u) \} \quad (2.54)$$

para qualquer $u \in U$ e $v \in V$.

Com base nestas definições, podem ser geradas várias funções de implicação pelo uso das várias operações *T-norm* (*) e *T-conorm* (+).

A implicação difusa $A \rightarrow B$ pode ser interpretada como a regra difusa *IF-THEN* : *IF* $x \text{ is } A$, *THEN* $y \text{ is } B$, em que $x \in U$ e $y \in V$ são variáveis linguísticas. As equações (2.48)-(2.54) correspondem a diferentes interpretações das regras *IF-THEN*, baseadas em critérios intuitivos ou na generalização da lógica clássica.

6.2. Interpretação da regra difusa IF-THEN

Um Sistema de Lógica Difusa – SLD, Figura 41, mapeia os conjuntos difusos de entrada nos conjuntos difusos de saída. Seja um conjunto difuso F definido no universo de entrada U (por exemplo: $U = R^n$) e G no universo de saída. O conjunto F (ou G) em U (V) é caracterizado pela função de pertença $\mu_F(\mu_G) : U(V) \rightarrow [0,1]$, com $\mu_F(u)$ ($\mu_G(v)$) representando o grau de pertença de u (v) $\in U(V)$ no conjunto difuso F (G).

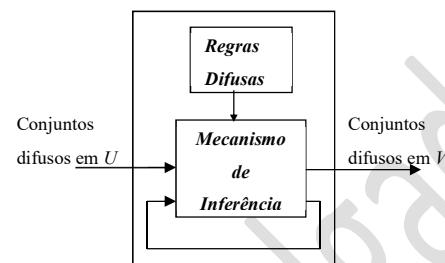


Figura 41 - Sistema de Lógica Difusa puro

As regras difusas do sistema consistem numa coleção de regras do tipo IF-THEN:

$$R^{(i)}: \text{IF } x_1 \text{ is } F_1^i \text{ and } \dots \text{ and } x_n \text{ is } F_n^i \text{ THEN } y \text{ is } G^i \quad (2.55)$$

O mecanismo de inferência difusa usa as regras IF-THEN e determina a correspondência entre os conjuntos difusos do universo do discurso de entrada U e os conjuntos difusos do universo de saída V , com base nos princípios da lógica difusa.

Cada regra i , IF-THEN, define um conjunto difuso $F_1^i \times F_2^i \times \dots \times F_n^i \rightarrow G^i$ no espaço produto $U \times V$. Uma regra da forma (2.55) é interpretada como uma implicação difusa $F_1^i \times F_2^i \times \dots \times F_n^i \rightarrow G^i$ em $U \times V$.

Pela interpretação da Generalização Modus Ponens dada na Figura 42 e pela comparação com a Figura 40 b), verifica-se que a Generalização Modus Ponens constitui uma composição difusa em que a primeira relação é meramente o conjunto difuso F' .

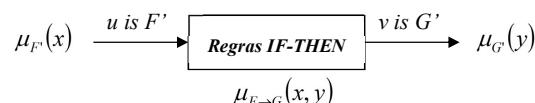


Figura 42 - Interpretação da Generalização Modus Ponens

Resulta da equação (2.36), que para cada regra i da equação (2.55), a função de pertença do conjunto difuso de saída G^i será dada por:

$$\mu_{G^{(i)}}(y) = \text{Sup}_{\vec{x} \in U} \left[\mu_{F_1^i \times \dots \times F_n^i \rightarrow G^i}(x, y) * \mu_G(x) \right] \quad (2.56)$$

em que $*$ representa qualquer operação T -norm (por exemplo *min*, *produto*, *produto limitado*, *produto drástico*).

Seja o conjunto difuso A' em U , uma entrada do mecanismo de inferência difusa. Então cada regra difusa (i) IF-THEN determina um conjunto difuso B^i em V definido pela composição Sup-star:

$$\mu_{B^{(i)}}(y) = \text{Sup}_{\vec{x} \in U} \left| \mu_{F_1^i \times \dots \times F_n^i \rightarrow G^i}(x, y) * \mu_A(x) \right| \quad (2.57)$$

Sem perda de generalidade pode assumir-se que:

$$- \vec{x} = (x_1, \dots, x_n) \in U_1 \times \dots \times U_n$$

$$- y \in V \text{ (espaço unidimensional)}$$

$$- F_i^i \text{ e } G^i \text{ são conjuntos difusos em } U_i \subset R \text{ e } V \subset R, \text{ respectivamente}$$

Baseados nos vários tipos de implicação referidas na secção 2.5.1, diversos autores, utilizando as operações T -norm e T -conorm, propuseram várias funções de implicação. De seguida é referido um leque das mais usadas em aplicações de engenharia.

Seja $F_1^i \times \dots \times F_n^i = A$ e $G^i = B$. Então a implicação de A para B ($A \rightarrow B$) pode ser conseguida com uma das seguintes regras (Tabela 2.8):

em que $\mu_A(\vec{x}) = \mu_{F_1^i \times \dots \times F_n^i}(\vec{x})$ é definido de acordo com uma das seguintes regras:

$$- \text{Regra de operação mínima: } \mu_{F_1^i \times \dots \times F_n^i}(\vec{x}) = \min \{ \mu_{F_1^i}(x_1), \dots, \mu_{F_n^i}(x_n) \} \quad (2.68)$$

$$- \text{Regra de operação produto: } \mu_{F_1^i \times \dots \times F_n^i}(\vec{x}) = \mu_{F_1^i}(x_1) \times \dots \times \mu_{F_n^i}(x_n) \quad (2.69)$$

Operação - min - R_c [Mandani [20]]:

$$\mu_{A \rightarrow B}(x, y) = \min \{ \mu_A(x), \mu_B(y) \} \quad (2.58)$$

Operação - produto - R_p [Larsen [21]]:

$$\mu_{A \rightarrow B}(x, y) = \mu_A(x) \cdot \mu_B(y) \quad (2.59)$$

Produto limitado - R_{lp}

$$\mu_{A \rightarrow B}(x,y) = \max \{0, \mu_A(x) + \mu_B(y) - 1\} \quad (2.60)$$

Produto drástico - R_d

$$\mu_{A \rightarrow B}(x,y) = \begin{cases} \mu_A(x), & \mu_B(y) = 1 \\ \mu_B(y), & \mu_A(x) = 1 \\ 0, & \mu_A(x), \mu_B(y) < 1 \end{cases} \quad (2.61)$$

Aritmética - R_a [Zadeh [22]]:

$$\mu_{A \rightarrow B}(x,y) = \min \{1, 1 - \mu_A(x) \cdot \mu_B(y)\} \quad (2.62)$$

Max-Min - R_m [Zadeh]

$$\mu_{A \rightarrow B}(x,y) = \max \{ \min [\mu_A(x), \mu_B(y)] 1 - \mu_A(x) \} \quad (2.63)$$

Sequência padrão - R_s :

$$\mu_{A \rightarrow B}(x,y) = \mu_A(x) > \mu_B(y)$$

em que $\mu_A(x) > \mu_B(y) = \begin{cases} 1 & \mu_A(x) \leq \mu_B(y) \\ 0 & \mu_A(x) > \mu_B(y) \end{cases}$ (2.64)

Booleana - R_b :

$$\mu_{A \rightarrow B}(x,y) = \max \{1 - \mu_A(x), \mu_B(y)\} \quad (2.65)$$

Lógica de Gödel

$$\mu_{A \rightarrow B}(x,y) = \begin{cases} 1 & \mu_A(x) \leq \mu_B(y) \\ \mu_B(y) & \mu_A(x) > \mu_B(y) \end{cases} \quad (2.66)$$

Goguen - R_Λ :

$$\mu_{A \rightarrow B}(x,y) = \begin{cases} 1 & \text{se } \mu_A(x) \leq \mu_B(y) \\ \frac{\mu_B(y)}{\mu_A(x)} & \text{se } \mu_A(x) > \mu_B(y) \end{cases} \quad (2.67)$$

Tabela 2.8 - Regras de implicação difusas

No caso da implicação difusa com a operação *min* ou *produto lógico*, a implicação bivalente ($x_1 \rightarrow x_2 \Leftrightarrow \bar{x}_1 + x_2$) não é preservada. Por outras palavras, quando x_1 é falso (0) a implicação bivalente é verdadeira (1). O mesmo não sucede com a implicação de operação *min* ou *produto lógico* das quais resulta um valor zero.

7. Sumário

Neste capítulo foram introduzidos conceitos fundamentais da teoria de conjuntos difusos, incluindo as notações matemáticas. Em seguida, foram introduzidos o princípio da resolução, que pode ser usado para expandir um conjunto difuso nos seus termos α -cuts, e o princípio da extensão, que permite a generalização de uma qualquer função f , no mapeamento de subconjuntos difusos. Foram abordadas várias operações sobre conjuntos difusos, como *T-*

norms, *T-conorms* e outros operadores de agregação. Depois de definido o conceito de relação, foram revistos alguns dos principais conceitos e operações em relações difusas, em composições relações-relações e conjuntos-relações. Foram revistos os principais mecanismos de inferência difusa, para interpretação de regras difusas do tipo IF-THEN. Analisaram-se as várias operações de inferência, bem como a interpretação das suas propriedades, na satisfação de um conjunto de critérios intuitivos das abordagens Generalizada *Modus Ponens* e Generalizada *Modus Tollens*. Este último estudo permitiu tomar sensíveis, do ponto de vista de engenharia, as melhores regras de implicação.

*Capítulo III.***SISTEMAS DE LÓGICA DIFUSA****1. Introdução**

A teoria difusa, cujos pilares essenciais foram abordados no capítulo II, coloca ao nosso dispor um respeitável aparato de ferramentas teóricas para lidar com conceitos expressos na linguagem natural. Estas ferramentas permitem-nos representar conceitos linguísticos, a maioria dos quais são inherentemente vagos, por conjuntos difusos, de vários tipos, e a sua manipulação numa grande variedade de formas, segundo o fim em vista. Permite-nos, ainda, expressar e manusear com várias relações, funções, e equações que envolvam conceitos linguísticos; e possibilitam fuzificar qualquer área da matemática clássica, para facilitar o seu uso em novas aplicações emergentes.

Um sistema difuso é qualquer sistema cujas variáveis de estado (ou, pelo menos algumas delas) tenham gama de valores que sejam conjuntos difusos. A cada variável, o seu conjunto difuso associado é definido como sendo um subconjunto do conjunto universo, sendo representativo de um intervalo de números reais. Nestas circunstâncias, os conjuntos difusos são números difusos, e as variáveis associadas são variáveis linguísticas.

De um modo geral, os sistemas de lógica difusa são casos especiais de “*expert systems*”. Cada um deles emprega o conhecimento de base, expresso num conjunto de regras difusas, e um apropriado mecanismo de inferência para resolver um dado problema de identificação; controlo; etc. Este capítulo debruça-se sobre este particular no que concerne a aplicações de engenharia.

2. Sistemas de lógica difusa

A arquitetura típica de um SLD está representada na Figura 43. É constituído por quatro componentes principais: um *fuzificador*, uma *base de regras difusas*, um *mecanismo de inferência* e um *desfuzificador* [23].

O *fuzificador* tem a tarefa de transformar dados numéricos em variáveis linguísticas. A *base de regras difusas* armazena o conhecimento empírico das operações do processo, em termos do domínio do conhecimento. O mecanismo de inferência é o “núcleo” do SLD, pois é aí que são interpretadas as regras e tomadas as decisões. O *desfuzificador* é usado para converter as decisões (ou acções de controlo) difusas, provenientes do mecanismo de inferência, em

decisões não difusas.

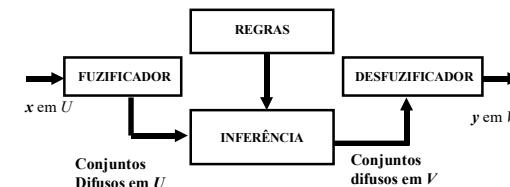


Figura 43 - Configuração base de um sistema difuso com *fuzificador* e *desfuzificador*.

A base de regras difusas e o mecanismo de inferência difuso foram já apresentados anteriormente. O mecanismo de inferência difusa pode apresentar, para uma entrada A' (um conjunto difuso em U), uma saída em duas formas distintas: uma coleção de conjuntos difusos de saída ou a união dos conjuntos difusos, ou seja:

- M conjuntos difusos B^l ($l = 1, 2, \dots, M$) obtidos a partir de cada uma das M regras *IF-THEN*, obtidos a partir da equação 2.55.
- Um conjunto difuso B' dado pela união dos M conjuntos difusos B^l :

$$\mu_{B'}(y) = \mu_{B^1}(y) + \dots + \mu_{B^M}(y) \quad (0.21)$$

Kosko [24][25] utiliza a operação adição em vez da operação *T-conorm* para a agregação dos referidos conjuntos, a que dá o nome de *Sistema Difuso Aditivo*.

No entanto, como na maior parte das aplicações de engenharia, apenas estamos interessados em mapear números (variáveis de entrada) em números (variáveis de saída). Nesta perspectiva existem ainda mais dois blocos, um de entrada chamado *fuzificador*, que faz corresponder os números reais de entrada em conjuntos difusos (níveis difusos), e um outro de saída, *desfuzificador* que realiza a operação inversa.

A operação de fuzificação tem como efeito a transformação de dados “ordinários” num (nos) conjunto(s) difuso(s). A experiência na implementação dos SLD sugere dois principais tipos de fuzificadores [26]: o fuzificador do tipo singular (*singleton fuzzifier*) e o fuzificador do tipo não singular (*nonsingleton fuzzifier*). O fuzificador do tipo *singular* mapeia um ponto de valor real $x \in U$ num conjunto difuso A' em U , que tem valor de pertença unitário em x' e zero em todos os outros pontos de U :

$$\mu_{A'}(\vec{x}') = \begin{cases} 1 & \text{para } \vec{x}' = \vec{x} \\ 0 & \text{outros} \end{cases} \quad (0.22)$$

Basicamente um conjunto difuso tipo *singleton* é um *fuzificador* preciso e não introduz nenhum grau de fuzificação. Esta estratégia é largamente usada em aplicações de controlo difuso por ser de fácil implementação.

O Fuzificador do tipo não-singular (*nonsingleton fuzzyfier*) atribui o valor de $\mu_A(\vec{x})=1$ para $\vec{x}'=\vec{x}$, enquanto $\mu_A(\vec{x})$ decresce de 1 à medida que \vec{x}' se afasta de \vec{x} [27]. Um exemplo é o da equação (0.23), uma função Gaussiana

$$\mu_{A'}(\vec{x}') = \exp \left[-\frac{(\vec{x}' - \vec{x})^T (\vec{x}' - \vec{x})}{\sigma^2} \right] \quad (0.23)$$

em que σ é o parâmetro que determina a forma de $\mu_A(\vec{x})$.

Neste sentido, a operação de fuzificação traduz a crença ou a plausibilidade de os dados de entrada pertencerem ao conjunto difuso. Torna-se assim numa forma eficiente de representar a incerteza.

Este método de fuzificação é aconselhado quando os dados observados estão corrompidos por ruído aleatório [28].

O "desfuzificador" efectua a transposição (mapeamento) do(s) conjunto(s) difuso(s) B' em V no ponto "ordinário" definido, $y^* \in V$. Conceptualmente, a tarefa de desfuzificação consiste em especificar um ponto em V que melhor represente o(s) conjunto(s) difuso(s) B' . Isto é similar ao valor médio de uma variável aleatória. Como não existem critérios matemáticos definidos para a desfuzificação, as considerações a ter em conta são os seguintes critérios intuitivos:

- ◆ *Plausibilidade*: o ponto y^* deve representar B' de um ponto de vista intuitivo; por exemplo, ele deve estar ligado aproximadamente ao meio do *suporte de B'* ou ao valor de maior grau de pertença de B' ;

- ◆ *Simplicidade computacional*: este critério é particularmente importante no controlo difuso porque os controladores funcionam em tempo real;

- ◆ *Continuidade*: uma pequena mudança em B' não deve resultar numa grande variação de y^* .

Três das técnicas de desfuzificação [29][30][31] mais usadas são: desfuzificador de centro de gravidade; desfuzificador de centro médio e desfuzificador de máximo.

O *desfuzificador de centro de gravidade*, especifica y^* como o centro da área coberta pela função de pertença B' , isto é:

$$y^* = \frac{\int_V y \cdot \mu_{B'}(y) dy}{\int_V \mu_{B'}(y) dy} \quad (0.24)$$

A Figura 44 mostra esta operação graficamente.

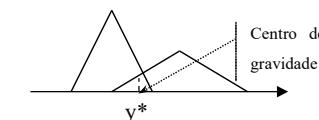


Figura 44 - Representação gráfica da desfuzificação de centro de gravidade.

O sistema difuso representado em (0.24), realiza a esperança condicional $E[Y|X]$ e desta forma realiza uma estimativa média quadrática ótima não-linear (*mean-square optimal nonlinear estimator*) [24].

O *desfuzificador de centro médio* é realizado com base na ideia de que, como o conjunto difuso B' resulta da união ou intersecção de M conjuntos bivalentes, uma boa aproximação da equação (0.24) é considerar a média pesada dos centros dos M conjuntos difusos, com pesos de valor igual à altura dos mesmos.

Seja, \bar{y}^l o centro do conjunto difuso B^l e w^l o seu peso, para $l=1,\dots,M$. O desfuzificador de centro médio calcula y^* como se segue:

$$y^* = \frac{\sum_{l=1}^M \bar{y}^l \cdot W^l}{\sum_{l=1}^M W^l} \quad (0.25)$$

A Figura 45 ilustra esta operação graficamente, para o exemplo de $M=2$.

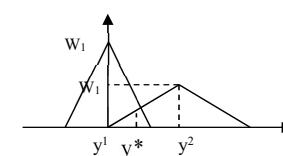


Figura 45 - Representação gráfica do desfuzificação de centro médio

O desfuzificador de centro médio é aquele que é mais usado nos sistemas difusos. É computacionalmente simples, plausível e contínuo (para pequenas mudanças em \bar{y}^l e W^l resultam pequenas mudanças de y^*).

O desfuzificador de máximo, como o nome indica, escolhe y^* como o ponto em V no qual a função $\mu_B(y)$ atinge o valor máximo.

Define-se o conjunto

$$h(B') = \left\{ y^* \in V \mid \mu_{B'}(y^*) = \sup_{y \in V} \mu_{B'}(y) \right\} \quad (0.26)$$

em que $h(B')$ é o conjunto de todos os pontos em V , no qual $\mu_{B'}(y)$ atinge um valor máximo.

O valor de y^* pode ser escolhido arbitrariamente em $h(B')$, o menor dos máximos, o maior dos máximos, ou o valor médio dos máximos.

O desfuzificador de máximo é plausível e computacionalmente simples, mas não é contínuo (pequenas variações em B' podem resultar em grandes mudanças de y^*).

A Tabela 3.1 compara os três desfuzificadores abordados de acordo com os critérios propostos.

	Centro de gravidade	Centro médio	Máximo
Plausibilidade	Sim	Sim	Sim
Simplicidade computacional	Não	Sim	Sim
Continuidade	Sim	Sim	Não

Tabela 3.1 - Comparação dos métodos de desfuzificação em relação ao centro de gravidade, centro médio e máximo.

3. SLD utilizados em engenharia.

Vão abordar-se em seguida os sistemas de lógica difusa (SLD) mais usados em aplicações de engenharia e que são diferentes combinações das regras de inferência-mínima ("min-inference"), de inferência-produto ("product inference"), de fuzificação do tipo singular ("singleton fuzzyfier"), de desfuzificação de centro médio, de desfuzificação de centro de média modificado, e de funções de pertença Gaussianas e triangulares.

Lema 3.1: Se o sistema de lógica difusa usa um desfuzificador de centro médio, a regra da inferência-produto ($\mu_{A \rightarrow B}(x, y) = \mu_A(x) \cdot \mu_B(y)$), com $\mu_A(\bar{x}) = \mu_{F_1^l}(x_1) \times \dots \times \mu_{F_n^l}(x_n)$, e um fuzificador do tipo singular, tem a forma de:

$$f(\vec{x}) = \frac{\sum_{l=1}^M \bar{y}^l \left(\prod_{i=1}^n \mu_{F_i^l}(x_i) \right)}{\sum_{l=1}^M \left(\prod_{i=1}^n \mu_{F_i^l}(x_i) \right)} \quad (0.27)$$

em que \bar{y}^l é o ponto em que a função de pertença μ_{G^l} atinge o seu valor máximo, que se assume como $\mu_{G^l}(\bar{y}^l) = 1$.

Lema 3.2: Se o sistema de lógica difusa usa um desfuzificador de centro médio, com a regra de "mini-inferência" e um fuzificador do tipo singleton, resulta:

$$f(\vec{x}) = \frac{\sum_{l=1}^M \bar{y}^l \left[\min(\mu_{F_1^l}(x_1), \dots, \mu_{F_n^l}(x_1)) \right]}{\sum_{l=1}^M \left[\min(\mu_{F_1^l}(x_1), \dots, \mu_{F_n^l}(x_1)) \right]} \quad (0.28)$$

em que \bar{y}^l é o ponto em que μ_{G^l} atinge o seu valor máximo. Assume-se que: $\mu_{G^l}(\bar{y}^l) = 1$. Se as funções de pertença do espaço de entrada são da forma Gaussiana, temos:

$$\mu_{F_i^l}(x_i) = a_i^l \cdot \exp \left[-\left((x_i - \bar{x}_i^l) / \sigma_i^l \right)^2 \right] \quad (0.29)$$

em que a_i^l, \bar{x}_i^l e σ_i^l são parâmetros ajustáveis.

Lema 3.3: Se o sistema difuso usa um desfuzificador de centro de média, regra inferência-produto, fuzificador do tipo singular e função de pertença do espaço de entrada Gaussiana, tem a forma:

$$f(\vec{x}) = \frac{\sum_{l=1}^M \bar{y}^l \left(\prod_{i=1}^n a_i^l \cdot e^{-\left(\frac{(x_i - \bar{x}_i^l)^2}{\sigma_i^l} \right)} \right)}{\sum_{l=1}^M \left(\prod_{i=1}^n a_i^l \cdot e^{-\left(\frac{(x_i - \bar{x}_i^l)^2}{\sigma_i^l} \right)} \right)} \quad (0.30)$$

com $\bar{y}^l \in V$, $a_i^l \in (0,1)$, $\bar{x}_i^l \in U_i$ e $\sigma_i^l > 0$.

Se se usar a função de pertença em forma triangular, resulta:

$$\mu_{F_i^l} = \begin{cases} 1 - \frac{|x_i - c_i^l|}{b_i^l} & \text{se } x_i \in [c_i^l - b_i^l, c_i^l + b_i^l] \\ 0 & \text{outros} \end{cases} \quad (0.31)$$

em que c_i^l e b_i^l são parâmetros ajustáveis.

Lema 3.4: Sejam μ_{G^l} funções Gaussianas; então μ_{G^l} toma a forma:

$$\mu_{G^l}(y) = \exp \left[-\left(\frac{y - \bar{y}^l}{\sigma^l} \right)^2 \right] \quad (0.32)$$

A função do sistema lógico do Lema 3.3, mas com um desfuzificador do centro médio modificado, proposto por Wang [32], tem a forma de:

$$f(x) = \frac{\sum_{l=1}^M \left(\bar{y}^l \left[\prod_{i=1}^n a_i^l \cdot \exp \left[-\left(\frac{x_i - \bar{x}_i^l}{\sigma_i^l} \right)^2 \right] \right] \right) / c_l}{\sum_{l=1}^M \left(\prod_{i=1}^n a_i^l \cdot \exp \left[-\left(\frac{x_i - \bar{x}_i^l}{\sigma_i^l} \right)^2 \right] \right)} \quad (0.33)$$

De seguida analisa-se resumidamente o *sistema difuso aditivo (SDA)*, proposto por Kosko [24], que por constituir um tipo diferente de agregação dos conjuntos B' , merece uma análise distinta.

Seja um sistema difuso aditivo $F: R^n \rightarrow R^p$ que contém M regras *IF-THEN* do tipo “*IF $x=A^l$ THEN $y=B^l$* ”. Esta declaração pode ser interpretada, usando “*approximate reasoning*” [33][34][35][36], como a proposição “ $(x,y) \text{ is } S'$ ”, em que S' é o conjunto produto cartesiano de A^l por B^l , $A^l \times B^l \subset R^n \times R^p$. O conjunto difuso de saída B é a soma ou adição dos conjuntos activados $B' \subset R^p$:

$$B = \sum_{l=1}^M w_l \cdot B'_l \quad (0.34)$$

para regras pesadas pelo escalar $w_l \in R$. Os pesos das regras podem refletir a importância, a credibilidade da fonte de conhecimento, ou a frequência da regra. Diferentemente, a maior parte dos sistemas difusos, não-aditivos, combina os conjuntos difusos ativados B' de acordo com a equação (0.34).

Seja a operação *max* a operação *T-conorm* escolhida. Então, o conjunto difuso B obtido ignora o consenso ou a sobreposição dos conjuntos B' e corresponde à envolvente dos conjuntos difusos ativados. Este tipo de combinação, *max*, é consequência do *princípio da extensão*, descrito na secção 2.3.

Teorema 3.1: Considere-se o sistema difuso $F: R^n \rightarrow R^p$, de um modelo aditivo padrão.

Seja $F(x) = \text{centróide} \left(\sum_{l=1}^M \mu^l(x) \cdot B^l \right)$, em que $\mu^l(x)$ constitui a função de pertença da agregação do antecedente da regra l . Então, $F(x)$ é a soma convexa dos centróides, c_l , dos m conjuntos consequêncial:

$$F(\vec{x}) = \frac{\sum_{l=1}^M \mu_{F_l}(\vec{x}) \cdot V_l \cdot c_l}{\sum_{l=1}^M \mu_{F_l}(\vec{x}) \cdot V_l} = \sum_{l=1}^M p_l(\vec{x}) \cdot c_l \quad (0.35)$$

Os coeficientes convexos e os pesos da probabilidade discreta $p_1(\vec{x}), \dots, p_M(\vec{x})$ dependem da entrada \vec{x} através da expressão:

$$p_l(\vec{x}) = \frac{\mu_{F_l}(\vec{x}) \cdot V}{\sum_{l=1}^M \mu_{F_l}(\vec{x}) \cdot V_l} \quad (0.36)$$

em que V_l é o volume (ou área, se $p=1$) e c_l é o centróide do conjunto consequência B_l .

Os pesos convexos $p_1(\vec{x}), \dots, p_M(\vec{x})$ definem a função discreta de densidade de probabilidade, para cada entrada \vec{x} , pois os m termos são não negativos e a sua soma é um. Assim, para cada entrada \vec{x} , o sistema difuso F define um valor esperado dos m centróides de saída c_l , relativamente a $p(x)$.

Se os valores dos centróides c_l dos conjuntos B_l coincidirem com o ponto de máximo, \bar{y}^l , e se os volumes forem todos iguais, o *sistema difuso aditivo* da equação (0.35) com desfuzificação de centro de gravidade coincide com a desfuzificação de centro médio e é descrito pelas equações (0.27), (0.28) e (0.30). A equação (0.33) corresponde assim a um caso particular de um sistema aditivo.

Existem três razões principais para o uso do sistema de lógica difusa representada na equação (0.30):

- Primeiro, foi provado por Wang e Mendel, 1992 [37], Kosko, 1996, e por Hão Ying, 1994 [38], que este sistema de lógica difusa constitui um aproximador universal;

- Em segundo lugar, o sistema de lógica difusa da equação (0.30), é construído a partir de regras difusas *IF-THEN*, tornando assim possível incorporar informação proveniente de especialistas humanos;

- Por fim, pela observação da sua forma funcional, verifica-se que pode ser representado numa estrutura em rede neuronal, do tipo “*feedforward*” de três camadas, como mostra a Figura 46, o que torna possível a aplicação do método de “*back-propagation*” para ajustar os

parâmetros das funções de pertença, $(\bar{y}^l, \bar{x}_i^l \text{ and } \sigma_i^l)$, de cada regra l .

Um outro modelo difuso muito importante é o modelo difuso de Sugeno (*TSK-fuzzy model*), proposto por Takagi, Sugeno e Kang [39][40]. As regras típicas do modelo de Sugeno têm a forma:

$$R^l : \text{IF } x_1 \text{ is } A_1^l \text{ and } \dots \text{ and } x_n \text{ is } A_n^l \text{ THEN } y = f_l(x_1, \dots, x_n) \quad (0.37)$$

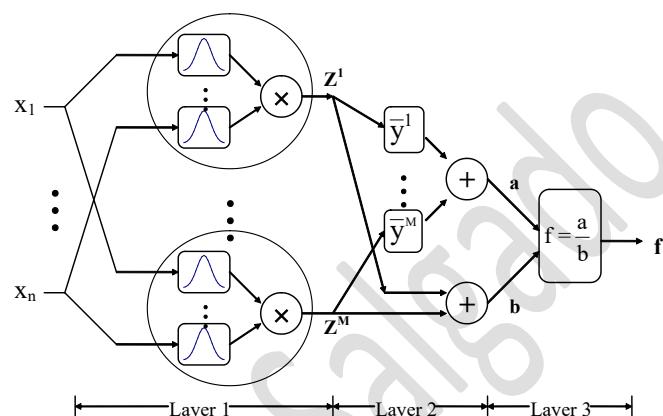


Figura 46 - Representação em rede do sistema de lógica difusa.

em que A_i^l são os conjuntos antecedentes, enquanto $y = f_l(x_1, \dots, x_n)$ é uma função analítica, consequente da regra. Usualmente $f_l(x_1, \dots, x_n)$ é uma função polinómio das variáveis de entrada x_i , mas pode ser qualquer outro tipo de função que descreva apropriadamente as saídas do sistema, dentro das regiões difusas especificadas pela premissa da regra.

Quando $f_l(x_1, \dots, x_n)$ é um polinómio de 1ª ordem, o sistema de inferência difuso resultante é denominado modelo difuso de 1ª ordem de Sugeno. Quando f é uma constante, resulta um modelo de ordem 0 de Sugeno, que constitui o sistema difuso da equação (0.30).

4. Sistemas de identificação difusos

Os sistemas de identificação difusos (*SID*) requerem dois tipos distintos de aprendizagem: a estrutural e a paramétrica.

A aprendizagem estrutural envolve a formulação da estrutura das regras difusas tal como o número de variáveis, e para cada variável de entrada ou saída a partição do universo do

discurso, o número de regras e o tipo de conjunção que agregue as regras e/ou os seus antecedentes, e assim por diante.

Assim que seja obtida uma estrutura satisfatória, o *SID* necessita da realização de ajustes nos seus parâmetros. Nesta fase de aprendizagem paramétrica, são passíveis de sintonização os parâmetros associados às funções de pertença, como os centros, largura e declives, os parâmetros das conexões difusas parametrizadas e os pesos das regras difusas.

Nesta secção são revistas algumas técnicas de aprendizagem estrutural, paramétrica ou híbrida.

De acordo com a equação (0.30), pode definir-se a *função básica difusa*, FBD (ou *fuzzy basis function*) como:

$$p_l(\vec{x}) = \frac{\prod_{i=1}^n \mu_{F_i^l}(x_i)}{\sum_{l=1}^M \left(\prod_{i=1}^n \mu_{F_i^l}(x_i) \right)}, \quad l = 1, 2, \dots, M. \quad (0.38)$$

Então, o sistema difuso descrito pela equação (0.30) é equivalente a uma expansão em série das funções FBD ou de uma rede FBD (RFBD):

$$f(\vec{x}) = \sum_{l=1}^M p_l(\vec{x}) \cdot \theta^l \quad (0.39)$$

em que $\theta^l = \bar{y}^l \in R$.

Existem duas formas distintas de treino das RFBD. Na primeira, todos os parâmetros a_i^l, \bar{x}_i^l e σ_i^l em $p_l(\vec{x})$ e θ^l são parâmetros livres do desenho, logo a RFBD é não linear nos seus parâmetros. Neste caso, é necessário recorrer a técnicas de aprendizagem não linear (por ex.: “*back-propagation*”). Por outro lado, se todos os parâmetros de $p_l(\vec{x})$ são fixados no início do desenho então os únicos parâmetros livres são os θ^l . Neste caso, $f(\vec{x})$ é linear nos seus parâmetros e alguns algoritmos eficientes de estimação de parâmetros lineares [41] podem ser usados para treinar os RFBD.

4.1.1. O método “*Back-propagation*”

O algoritmo de aprendizagem através de “*back-propagation*” é um dos desenvolvimentos mais importantes das redes neurais [42] e, por arrastamento, dos sistemas difusos. Este algoritmo de aprendizagem é aplicado a redes multi-camada do tipo “*feedforward*”, para sintonia de parâmetros de funções contínuas e diferenciáveis.

Para um dado par entrada-saída $(\vec{x}^{(k)}, \vec{d}^{(k)})$, o algoritmo “*back-propagation*” processa-se em duas fases. Na primeira os dados de entrada $\vec{x}^{(k)}$ são propagados da camada de entrada para

a camada de saída. Como resultado é produzida uma saída $f(\bar{x}^{(k)})$. Então, por exemplo, o sinal de erro resultante da diferença entre $\bar{d}^{(k)}$ e $f(\bar{x}^{(k)})$ é realimentado da camada de saída para as camadas anteriores, modificando nesta passagem os diversos parâmetros. Assim, esta aprendizagem instantânea das regras é formulada pela minimização instantânea de uma função de erro, na maior parte dos casos o erro quadrático médio, e os parâmetros são adaptados usando a regra de descida de gradiente.

O objetivo do processo de aprendizagem é encontrar os parâmetros do sistema, que no caso da sua aplicação a sistemas difusos, minimizam a função de erro:

$$E^k = 1/2 [f(\bar{x}^k) - d^k]^2 \quad (0.40)$$

para cada instante k , em que f e d^k são respetivamente a saída obtida e a saída desejada do sistema difuso, para o vetor de entrada \bar{x} .

O método de “back-propagation” para o sistema difuso da figura 3.4 é aqui descrito. O objetivo consiste na sintonização dos parâmetros \bar{y}^l , \bar{x}_i^l e σ_i^l que minimize a função de erro E (equação (0.40)) :

$$\bar{y}^l(k+1) = \bar{y}^l(k) - \alpha_y \frac{\partial E}{\partial \bar{y}^l} = \bar{y}^l(k) - \alpha_y \frac{f-d}{b} z^l \quad (0.41)$$

$$\bar{x}_i^l(k+1) = \bar{x}_i^l(k) - \alpha_x \frac{\partial E}{\partial \bar{x}_i^l} = \bar{x}_i^l(k) - \alpha_x \frac{f-d}{b} \left(\bar{y}^l - f \right) \cdot z^l \cdot \frac{2(x_i^k - \bar{x}_i^l(k))}{(\sigma_i^l(k))^2} \quad (0.42)$$

$$\bar{\sigma}_i^l(k+1) = \bar{\sigma}_i^l(k) - \alpha_\sigma \frac{\partial E}{\partial \bar{\sigma}_i^l} = \bar{\sigma}_i^l(k) - \alpha_\sigma \frac{f-d}{b} \left(\bar{y}^l - f \right) \cdot z^l \cdot \frac{2(x_i^k - \bar{x}_i^l(k))^2}{(\sigma_i^l(k))^3} \quad (0.43)$$

em que: $f = a/b$, $a = \sum_{l=1}^M \bar{y}^l z^l$, $z^l = \prod_{i=1}^n \exp \left(-\left(\frac{x_i^k - \bar{x}_i^l(k)}{\sigma_i^l(k)} \right)^2 \right)$ e α_y , α_x , α_σ são as taxas de

aprendizagem.

O algoritmo acima descrito adota uma aprendizagem “instantânea” ou “incremental” dos parâmetros [43], isto é, os parâmetros são modificados sempre que um par de dados de treino é apresentado. Em alternativa, no treino em “batch-mode”, os parâmetros são apenas modificados depois de todos os pares de dados de treino serem apresentados. As diferenças e semelhanças dos dois métodos podem ser encontradas em B. Widrow e Michael A. L. [42]. A relativa eficiência dos dois métodos depende do problema em consideração, mas o treino por “batch-mode” requer maior capacidade de armazenamento de dados e dos parâmetros a sintonizar.

Exemplo 3.1: Neste exemplo, pretende-se mostrar a habilidade do uso de um sistema difuso na modelização de um sistema de várias entradas e várias saídas (MIMO, Multi-Input-Multi-Output), descrito pelo seguinte sistemas de equações às diferenças:

$$\begin{bmatrix} y_1(k+1) \\ y_2(k+1) \end{bmatrix} = \begin{bmatrix} \frac{y_1(k)}{1+y_2^2(k)} \\ \frac{y_1(k)y_2(k)}{1+y_2^2(k)} \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}$$

Os dois modelos a identificar são os dois sistemas difusos \hat{f}_1 e \hat{f}_2 , tal que:

$$\begin{bmatrix} \hat{y}_1(k+1) \\ \hat{y}_2(k+1) \end{bmatrix} = \begin{bmatrix} \hat{f}_1(y_1(k), y_2(k)) \\ \hat{f}_2(y_1(k), y_2(k)) \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}$$

Ambos os modelos \hat{f}_1 e \hat{f}_2 são da forma (0.30) com $M=100$ (regras). O processo de identificação foi realizado em 5000 iterações de treino (uma iteração para cada ponto), usando valores aleatórios para $y_1(k)$ e $y_2(k)$, com distribuição uniforme no intervalo [-1,1]. As taxas de aprendizagem escolhidas foram $\alpha_x = \alpha_y = \alpha_\sigma = 0.5$. Para valor inicial dos parâmetros das funções de pertença tomaram-se $\bar{y}^l = 0$, $\sigma_i^l = 2$, e para \bar{x}_i^l valores aleatórios no intervalo [-5,5], para $i=1, \dots, M$, e $i=1, 2$.

O comportamento do modelo difuso e do modelo real, para um vector de entrada $[u_1(k), u_2(k)] = [\sin(2\pi k/25), \cos(2\pi k/25)]$, estão representadas nas Figura 47 a) e 3.5 b), respectivamente para as variáveis $y_1(k)$, $\hat{y}_1(k)$ e $y_2(k)$, $\hat{y}_2(k)$. É visível o bom comportamento do modelo difuso identificado que segue quase perfeitamente a trajectória do modelo real.

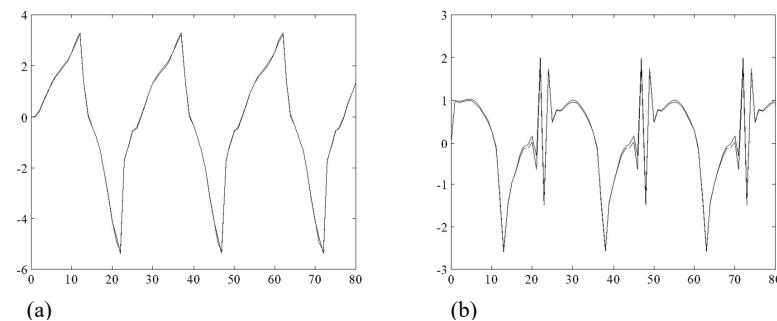


Figura 47 - a) Saidas real $y_1(k)$ e do modelo difuso $\hat{y}_1(k)$, b) Saidas real $y_2(k)$ e do modelo difuso $\hat{y}_2(k)$, ambos após 5000 iterações de treino

Muitos outros algoritmos de treino utilizam o método de “Back-propagation”, para sintonizar todos ou apenas alguns dos seus parâmetros. São exemplos, Lin e Lee [1991][44] que

o usam no seu “fuzzy adaptive learning control network (FALCON)”, para o refinamento dos parâmetros das funções de pertença (depois da aprendizagem estrutural), Nomura (1992)^[45], e muitos outros^[46].

4.1.2. Método de mínimos quadrados ortogonais (“Orthogonal least squares”)

Se todos os parâmetros de $p_i(\bar{x})$ são fixados no princípio do desenho do RFBD, então os únicos parâmetros livres são os θ^l . Neste caso, a função $f(\bar{x})$ (equação (0.39)) é linear em relação aos parâmetros livres, existindo vários estimadores lineares de parâmetros eficientes que podem ser usados no treino ou “desenho” dos RFBD. Um destes métodos, a ortogonalização quadrática de Gram-Schmidt^[47] [48] é utilizada com este objetivo. Esta estratégia de aprendizagem determina os números próprios ou significativos, de FBD, automaticamente [32]^[49]. Constitui, por isso, uma espécie híbrida de aprendizagem, estrutural e paramétrica. É capaz de encontrar FBD significativos e as θ^l associadas para o RFBD.

O sistema difuso descrito pela equação (0.39) pode ser visto como a expansão em série das funções FBD, ou seja, como um caso especial de um modelo de regressões lineares:

$$d(t) = \sum_{i=1}^M p_i(t) \cdot \theta_i + e(t) \quad (0.44)$$

em que $d(t)$ é a saída do sistema, θ_i são parâmetros reais, $p_i(t) = p_i(\bar{x}(t))$ são as funções de regressão, que são conhecidas e fixas, e $e(t)$ é um sinal de erro que é assumido ser aleatório.

4.1.3. Mínimos quadráticos recursivos

Como descrito anteriormente, nos sistemas difusos do tipo “singleton”, a função $f(\bar{x})$ é linear nos parâmetros θ^l . Assim, fixos os valores dos parâmetros das funções de pertença de entrada (centros e larguras) e sendo q o número de pontos de treino $(\bar{x}^k, d^k), k = 1, 2, \dots, q$, está-se diante de um sistema linear de q equações a M incógnitas, respeitantes aos parâmetros θ^l . Este problema pode ser representado na seguinte forma matricial:

$$\mathbf{d} = \mathbf{P} \cdot \boldsymbol{\theta} \quad (0.45)$$

Sendo geralmente $M > q$ o problema não tem solução, a menos que se queira resolvê-lo segundo um critério de minimização da soma dos erros quadráticos. Desta forma, a melhor solução para $\boldsymbol{\theta}$, que minimiza $\|\mathbf{P} \cdot \boldsymbol{\theta} - \mathbf{d}\|^2$, é um estimador de mínimos quadráticos $\boldsymbol{\theta}^*$:

$$\boldsymbol{\theta}^* = (\mathbf{P}^T \mathbf{P})^{-1} \mathbf{P}^T \mathbf{d} \quad (0.46)$$

em que \mathbf{P}^T é a matriz transposta de \mathbf{P} e $(\mathbf{P}^T \mathbf{P})^{-1}$ é a matriz pseudo-inversa de \mathbf{P} .

Este método pode ainda ser empregue de uma forma recursiva, ao qual se dá o nome de método dos mínimos quadrados recursivos (*Recursive Least Square*, RLS). Desta forma, os parâmetros óptimos de $\boldsymbol{\theta}$ são determinados usando recursivamente as seguintes expressões^[50]:

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \mathbf{S}_{i+1} \mathbf{p}_{i+1}^T (\mathbf{d}^{(i+1)} - \mathbf{p}_{i+1} \boldsymbol{\theta}_i) \quad (0.47)$$

$$\mathbf{S}_{i+1} = \mathbf{S}_i - \frac{\mathbf{S}_i \mathbf{p}_{i+1}^T \mathbf{p}_{i+1} \mathbf{S}_i}{1 + \mathbf{p}_{i+1} \mathbf{S}_i \mathbf{p}_{i+1}^T}, \quad i = 0, 1, \dots, q-1 \quad (0.48)$$

em que $\boldsymbol{\theta}_0 = \mathbf{0}$ e $\mathbf{S}_0 = \gamma \mathbf{I}$, são os valores iniciais de $\boldsymbol{\theta}$ e \mathbf{S} , γ é um número grande positivo, $\mathbf{0}$ a matriz nula e \mathbf{I} a matriz identidade^[51].

Esta estratégia de aprendizagem é ainda importante na identificação de parâmetros, em tempo real, em sistemas que mudam de características ao longo do tempo. Para o efeito, as formulas recursivas de minimização quadrática (0.47) e (0.48) deverão ter em conta as variações temporais dos dados recebidos, dando um menor contributo ao dados mais antigos face aos mais recentes. Este problema está bem estudado na literatura de identificação de sistemas e de controlo adaptativo, existindo numerosas soluções^[52].

O problema pode ser formulado como a minimização da seguinte função

$$E_W(\boldsymbol{\theta}) = (\mathbf{d} - \mathbf{P} \cdot \boldsymbol{\theta})^T \cdot \mathbf{W} \cdot (\mathbf{d} - \mathbf{P} \cdot \boldsymbol{\theta}) \quad (0.49)$$

em que a matriz \mathbf{W} terá como responsabilidade diferenciar os diferentes dados temporais.

A solução que minimiza E_W é em tudo semelhante à equação (0.46), com

$$\boldsymbol{\theta}^* = (\mathbf{P}^T \mathbf{W} \mathbf{P})^{-1} \mathbf{P}^T \mathbf{W} \mathbf{d} \quad (0.50)$$

Um método muito usado consiste em formular a medida de erro quadrática pesada, que atribui um maior peso aos dados mais recentes e um menor aos mais antigos, pela adição de um fator de esquecimento:

Se considerarmos um *fator de esquecimento* λ tal que a matriz \mathbf{W} seja,

$$\mathbf{W} = \begin{bmatrix} \lambda^{m-1} & 0 & \cdots & 0 \\ 0 & \lambda^{m-2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}$$

as formulas recursivas originais darão lugar a:

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \mathbf{S}_{i+1} \mathbf{p}_{i+1}^T (\mathbf{d}^{(i+1)} - \mathbf{p}_{i+1} \boldsymbol{\theta}_i) \quad (0.51)$$

$$S_{i+1} = \frac{1}{\lambda} \left[S_i - \frac{S_i p_{i+1}^T p_{i+1} S_i}{\lambda + p_{i+1}^T S_i p_{i+1}^T} \right] \quad (0.52)$$

em que λ assume valores práticos entre 0.9 e 1. Para pequenos valores de λ , maior será o efeito de esquecimento. Se $\lambda=1$ ter-se-á um processo sem esquecimento, e a equação (0.52) tornar-se-á igual à equação (0.48).

Jang, [1992][⁵³] propôs o método “*adaptive network-based fuzzy inference system (ANFIS)*”. Nesta arquitetura os parâmetros são adaptados de acordo com o algoritmo de “*back-propagation*”, de forma similar ao anteriormente exposto. Todavia, o algoritmo RLS introduzido pode ser usado para encontrar os parâmetros consequência do sistema ANFIS.

Exemplo 3.2: Neste exemplo, pretende-se mostrar a habilidade do uso desta técnica de identificação na escolha dos parâmetros ótimos θ^l , segundo um critério de minimização da soma dos erros quadráticos, na identificação de uma imagem de um peixe (150×100 pixels, 8 bits em cores de cinzentos) representada na Figura 48 a).

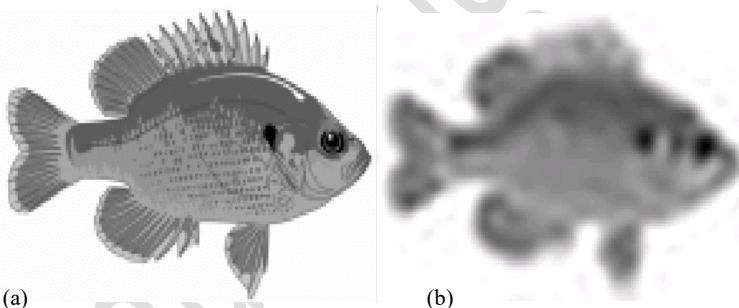


Figura 48 - Imagem a) real, b) do sistema difuso com 520 regras

Para o efeito, um conjunto de $M=520$ regras foi posicionado, igualmente espaçado, de forma a cobrir a totalidade da imagem. As funções de pertença escolhidas são as da equação (0.53), na forma de Gaussianos simétricos, com $\sigma_1^l = \sigma_2^l = 5$ pixels, $l = 1, \dots, M$. O sistema difuso da equação (0.54), e as suas regras, é aqui usado para descrever a imagem, segundo o qual, dando-lhe as coordenadas (x,y) de um ponto da imagem, ele indica o seu valor de intensidade I , de valores normalizados no intervalo $[0,1]$. As equações (0.47) e (0.48) foram recursivamente usadas, para cada pixel da imagem, para identificar os parâmetros ótimos de θ^l , com valores iniciais $\theta_0 = \mathbf{0}$ e $S_0 = 10000 \cdot I$. A Figura 48 b) mostra o resultado da descrição do sistema difuso.

4.1.4. Aprendizagem estrutural

A aprendizagem estrutural está relacionada com a extração das regras difusas dos dados de treino numéricos e a sintonização da partição difusa dos espaços de entrada e de saída. Muitos algoritmos realizam apenas a partição do espaço de entrada e/ou saída com as funções de pertença associadas, e a extração das regras do conjunto de dados de treino. Nesta secção serão introduzidos alguns destes métodos.

Kosko [1992][²⁴][⁵⁴] propôs um procedimento geométrico para a extração das regras, chamado “*product-space clustering*”. O projetista especifica as funções de pertença dos conjuntos difusos A_i ($i=1, \dots, r$) e B_j ($j=1, \dots, s$) das entradas e saídas, respetivamente. Os conjuntos difusos $\{A_i\}$ e $\{B_j\}$ definem $r \times s$ (rs) malhas difusas (*fuzzy grids*) no espaço entrada-saída $X \times Y$. Cada grelha difusa define uma possível regra difusa. A geração automática das regras a partir dos dados de treino (x_i, y_i) , $i=1, 2, \dots$, é realizada pela utilização de um algoritmo de quantificação vetorial que encontra e associa um vetor de quantificação aos dados de treino e aos gradeamentos difusos. De seguida, determina o peso de cada regra de acordo com o número de vetores de quantificação que surgem dentro do gradeamento correspondente. Dos algoritmo de quantificação vetorial destaca-se a regra de aprendizagem de Kohonen[⁵⁵][⁵⁶], nomeadamente a regra de aprendizagem diferencial competitiva (*differential competitive learning, DCL*) [⁵⁷].

Sejam $\vec{t}_1, \vec{t}_2, \dots, \vec{t}_m$, os m vetores de quantificação, com $m \geq r \cdot s$ de modo que os m vetores de quantificação possam descrever a distribuição uniforme da trajetória dos dados no espaço produto. Os vetores de quantificação \vec{t}_i pesam a estimativa de uma regra difusa, isto é, quanto maior for o número de vetores de quantificação agrupados numa regra (isto é grelha difusa) maior é a possibilidade de essa regra existir. Supondo que existem k_i vetores de quantificação agrupados no “ $i^{\text{ésimo}}$ ” gradeamento difuso, então o peso da regra “ i ” é: $w_i = k_i/m$, com $k = m$. Na prática devemos escolher as regras com o peso maior ou aquelas que tenham um valor maior que o peso mínimo w_{min} .

Dickserson e Kosko, 1993 [⁵⁸] propõem um método não supervisionado de aprendizagem competitivo, “*the ellipsoidal covariance learning algorithm*”, para estimar os centróides e as covariâncias das partições difusas. Este algoritmo de aprendizagem usa as propriedades de estatísticas de 1^a e 2^a ordem dos dados para estimar as regras difusas e os conjuntos difusos da entrada-saída. É usado um primeiro método de aprendizagem competitivo (Kohonen), em que os \vec{t}_j vetores de quantificação convergem exponencialmente para os centróides, \hat{x}_j . Estes

centróides fornecem uma estimativa de 1^a ordem de como a função de densidade de probabilidade $p(x)$, desconhecida, se comporta na região D_j . Um segundo método de aprendizagem do tipo competitivo é usado para estimar a matriz de covariância condicional local K_j :

$$K_j = E[(\bar{x} - \hat{x}_j)(\bar{x} - \hat{x}_j)^T | D_j] \quad (0.55)$$

O desenho do sistema difuso pode ser obtido usando o método da vizinhança. Este pode ser tão simples como estabelecer um raio de influência r . A cada um dos agrupamentos criados é associada uma regra difusa. A equação (0.30) pode ser vista como [59]:

$$f(\bar{x}) = \frac{\sum_{l=1}^M A^l e^{-\frac{|\bar{x}-\bar{x}^l|}{\delta}}}{\sum_{l=1}^M B^l e^{-\frac{|\bar{x}-\bar{x}^l|}{\delta}}} \quad (0.56)$$

$$A^i(k) = A^i(k-1) + y^i \quad B^i(k) = B^i(k-1) + 1 \quad (0.57)$$

Começando com um primeiro ponto (\bar{x}, y) , estabelece-se um primeiro agrupamento de centro $\bar{x}^1 = \bar{x}$ em X . Para todas as amostras futuras cuja distância $|\bar{x} - \bar{x}^i|$ seja menor que a distância a qualquer outro agrupamento e também menor ou igual ao raio r , deve atualizar-se o agrupamento i de acordo com a equação (0.57). Numa amostra j cuja distância à vizinhança mais próxima seja maior do que r devemos criar uma nova vizinhança com o seu centro situado no ponto \bar{x}^j , e assim mais uma regra é gerada. Os coeficientes dos numeradores e denominadores do sistema difuso (equação (0.56)) são perfeitamente determinados numa única passagem dos dados, não havendo a necessidade de um processo iterativo.

Muitos outros métodos usam os Algoritmos Genéticos (“*Genetic algorithms – GA*”) para a partição do espaço de entrada-saída, e para a determinação das funções de pertença das entradas-saiidas e das regras [60][61][62]. Neste método, cada cromossoma pode representar um conjunto particular de todas as funções de pertença empregues, do tipo de funções de pertença e de regras. Com base na ideia da evolução das espécies é assim possível determinar as funções, o número de regras e os parâmetros consequência das regras, usando algoritmos genéticos.

4.1.5. Desenho do sistema difuso usando o método de “Table-Lookup”

Para ilustrar este método, proposto por Li Xin Wang e J. M. Mendel [63], vamos supor que temos um conjunto de pares (entrada-saída) de dados de treino:

$(x_1^{(1)}, x_2^{(1)}, y^{(1)}), (x_1^{(2)}, x_2^{(2)}, y^{(2)}) \dots$, em que x_1 e x_2 são as entradas e y a saída. A extensão deste método ao caso multidimensional da entrada-saída é óbvia. O objetivo é gerar um conjunto de regras a partir dos pares de dados entrada-saída e usar estas regras para determinar a função de mapeamento $f: (x_1, x_2) \rightarrow y$. Este método consiste nos cinco passos que passam a descrever-se, com auxílio de um exemplo ilustrativo:

Passo 1: Dividir o espaço de entrada e saída em regiões distintas: Assumir que o intervalo do domínio de x_1 , x_2 e y é $[x_1^-, x_1^+]$, $[x_2^-, x_2^+]$ e $[y^-, y^+]$, respectivamente. Dividir cada domínio em $2N + 1$ (o valor de N pode ser diferente para cada variável) regiões e assinalar para cada uma delas uma função de pertença. A Figura 49 mostra um exemplo em que o domínio de x_1 é dividido em cinco regiões, x_2 em sete regiões e o domínio de y em cinco regiões. A forma de cada função de pertença é triangular, o seu vértice ligado ao centro da região e na qual a função de pertença tem o valor unitário, e os outros dois vértices ligados aos centros das duas regiões vizinhas e na qual a função de pertença tem valor nulo. É claro que são possíveis outros tipos de divisão das regiões, e outras formas para as funções de pertença.

Passo 2: Gerar regras difusas para os pontos dados: Primeiro, determinar o grau de pertença dos dados $x_1^{(i)}, x_2^{(i)}$ e $y^{(i)}$ nas diferentes regiões. Por exemplo $x_1^{(i)}$, na Figura 49, tem um grau de pertença de 0,8 em B_1 , 0,2 em B_2 e zero nas restantes regiões. Similarmente $x_2^{(i)}$, na figura 3.6, tem o grau de 1 na região CE e zero nas demais.

Em segundo lugar assinalar $x_1^{(i)}, x_2^{(i)}$ e $y^{(i)}$ na região de máximo grau de pertença. Por exemplo $x_1^{(i)}$, na Figura 49, é assinalado a B_1 , $x_2^{(2)}$ a CE e $y^{(1)}$ a CE . Finalmente, são obtidas as regras do par entrada-saída:

$$(x_1^{(1)}, x_2^{(1)}, y^{(1)}) \Rightarrow [x_1^{(1)}(0.8 \text{ em } B_1), x_2^{(1)}(0.7 \text{ em } S_1), y^{(1)}(0.9 \text{ em } CE)] \Rightarrow \text{regra 1}$$

R¹: IF x_1 is B_1 AND x_2 is S_1 THEN y is CE ;

$$(x_1^{(2)}, x_2^{(2)}, y^{(2)}) \Rightarrow [x_1^{(2)}(0.6 \text{ em } B_1), x_2^{(2)}(1.0 \text{ em } CE), y^{(2)}(0.7 \text{ em } B_1)] \Rightarrow \text{regra 2}$$

R²: IF x_1 is B_1 AND x_2 is CE THEN y is B_1 ;

Passo 3: Assinalar o peso de cada regra. Para resolver o problema do conflito de regras com o mesmo antecedente, mas diferentes consequentes, e para se reduzir o número de regras, devemos atribuir um peso a cada regra gerada dos pontos de dados, e aceitar a regra do grupo em conflito que tenha o maior peso.

A seguinte estratégia é usada para determinar o peso de cada regra: O peso da regra *IF* x_1 is A *AND* x_2 is B *THEN* y is C , chamado de D (*regra*), é definido como:

$$D(\text{regra}) = \mu_A(x_1) \cdot \mu_B(x_2) \cdot \mu_C(y)$$

Por exemplo, a regra 1 tem um peso de

$$\begin{aligned} D(R^1) &= \mu_{B_1}(x_1) \cdot \mu_{S_1}(x_2) \cdot \mu_C(y) \\ &= 0.8 \times 0.7 \times 0.9 = 0.504, \end{aligned}$$

e a regra 2 tem o peso de

$$\begin{aligned} D(R^2) &= \mu_{B_1}(x_1) \cdot \mu_{CE}(x_2) \cdot \mu_{B_1}(y) \\ &= 0.6 \times 1.0 \times 0.7 = 0.42. \end{aligned}$$

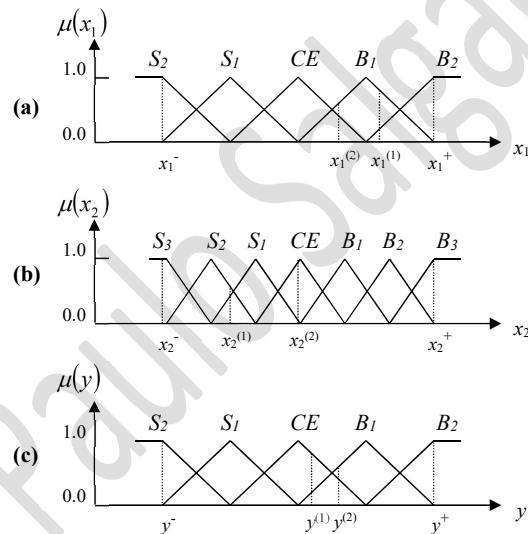


Figura 49 - Divisão do espaço de entrada e saída em regiões difusas e a correspondentes funções de pertença. a) $\mu(x_1)$, b) $\mu(x_2)$, c) $\mu(y)$

Na prática, existe informação *à priori* sobre os dados de treino. Por exemplo, se um especialista verificar os dados de treino pode sugerir que alguns são mais úteis e cruciais, mas que outros são de menor utilidade, eventualmente causados por erros de medida ou ruído do sinal. Pode, assim, atribuir-se um peso a cada par de dados, que representa para esse par o valor

crença na sua veracidade ou a utilidade do dado de treino:

$$D(\text{regra } l) = \mu_A(x_1) \cdot \mu_B(x_2) \cdot \mu_C(y) \cdot \mu^{(l)} \quad (0.58)$$

em que o peso da regra é definido entre o produto do peso dos seus componentes e o peso do par de treino que gera a regra l , $\mu^{(l)}$. No final da apresentação deste método, é sugerida a nova estratégia de atribuição de pesos às regras que, não conhecendo *à priori* a qualidade dos dados de treino, usa as suas propriedades estatísticas, para encontrar um peso.

Passo 4: Criar a combinação do conjunto das regras. A Figura 50 ilustra uma representação tabular (“table-lookup”) do conjunto das regras. Preenchem-se os quadros da tabela com as regras de acordo com a seguinte estratégia: a coleção das regras provém das regras geradas a partir dos dados de treino e/ou das regras linguísticas (assume-se que a regra linguística também tem um peso atribuído pelo especialista humano, que reflete a sua importância). Se existe mais do que uma regra para um quadro, deve escolher-se aquela que possui maior peso. Neste sentido, as regras, que contêm a informação numérica e linguística, são codificadas num mesmo patamar de trabalho, a coleção das regras difusas. Se a agregação dos antecedentes de uma regra é do tipo “and”, apenas um quadro da tabela é preenchido pela regra; se a agregação é do tipo “or”, são preenchidos todos os quadros de uma linha ou coluna, correspondentes à região do antecedente da regra. Os quadros selecionados são preenchidos com a função de pertença de saída correspondente.

	S ₂	S ₁	CE	B ₁	B ₂
S ₃					
S ₂					
S ₁					
CE					
B ₁					
B ₂					
B ₃					

Figura 50 - Ilustração da tabela de regras.

Passo 5: Determinação do mapeamento baseado nas regras da tabela. A estratégia de desfuzzificação para determinação da saída y correspondente a um ponto de entrada (x_1, x_2) , pode seguir os seguintes passos: primeiro, para a entrada (x_1, x_2) , são combinados os antecedentes das regras usando a operação *T-norm produto* para determinar o peso da região agregação do antecedente da regra i , $\mu_{O^i}^i$, isto é:

$$\mu_{O^i}^i = \mu_{f_1^i}(x_1) \cdot \mu_{f_2^i}(x_2) \quad (0.59)$$

em que O^i representa a região do antecedente da regra i , e I^i representa a região da regra i para a componente j .

Segundo, para uma desfuzzificação de centro médio, a saída do sistema difuso pode representar-se pela expressão:

$$y = \frac{\sum_{l=1}^M \bar{y}^l \mu_{O^l}^l}{\sum_{l=1}^M \mu_{O^l}^l} \quad (0.60)$$

cuja forma e significado são idênticas aos da equação (0.27).

5. Controladores de Lógica Difusa – CLD

Em geral, os controladores difusos são casos especiais de sistemas periciais. Ambos empregam uma base de conhecimento, expressa em termos de regras difusas, e um apropriado mecanismo de inferência para resolver um dado problema de controlo.

Na generalidade dos casos, o controlador de Lógica Difusa, CLD, (*Fuzzy Logic Controller - FLC*) consiste em quatro módulos: “fuzzy rule base”, mecanismo de inferência e os módulos de fuzzificação/desfuzzificação. A interligação destes módulos e o processo a controlar (*Planta*) está representado na Figura 51, no que se designa por sistema de controlo.

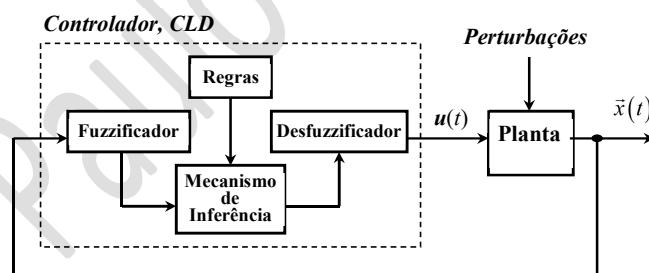


Figura 51 - Diagrama de blocos de um sistema de controlo em malha fechada.

Teoricamente o comportamento da planta pode ser descrito por um conjunto de equações diferenciais (ou equações às diferenças), geralmente não lineares e em outras também variáveis. Estas equações descrevem a evolução das variáveis de estado $x(t)$, que neste propósito são assumidas como mensuráveis. Em contraste, o controlador é usualmente uma função estática, representado por g . Ele mapeia as variáveis de estado $x(t)$ nas ações de controlo $u(t)$ por forma

a ser atingido um dado objetivo de controlo.

Assim de uma forma geral, para um sistema de controlo invariante no tempo, este pode ser representado pelo seguinte conjunto de equações:

$$\begin{aligned} \frac{\partial \bar{x}(t)}{\partial t} &= f(\bar{x}(t), u(t)) && (\text{dinâmica da planta}) \\ u(t) &= g(\bar{x}(t)) && (\text{controladores}) \end{aligned} \quad (0.61)$$

O problema principal na engenharia de controlo é encontrar as ações de controlo u por forma a que as saídas da planta x cumpram um dado objetivo. A cada método de desenho dos controladores de lógica difusa corresponde uma via para obter uma ação de controlo, algumas das quais serão abordadas em seguida.

Um grande número de Controladores de Lógica Difusa (CLD) apresenta-se estruturalmente sobre a forma de uma rede adaptativa. A estrutura resultante pode tomar como vantagem todas as técnicas de desenho dos controladores de Redes Neuronais, propostos na literatura. Nesta secção introduziremos, de forma resumida, as principais técnicas de desenho dos CLD (FLC).

5.1.1. Controlador Linguístico

A intenção original dos controladores difusos são imitar o especialista humano, tendo como último objetivo substituí-lo no controlo de sistemas complexos. Como exemplos temos o controlo climático das estufas, o controlo de travagem (ABS), o controlo eletrónico da ignição de motores de combustão, processos de reação químicos, etc.

Um operador humano experimentado pode usualmente sumariar as suas ações de controlo num conjunto de regras difusas IF-THEN bem como as inerentes funções de pertença. Nestas circunstâncias, a especificação das regras é, maior parte das vezes, obtida pelo método de tentativa erro [Mamdani, E. H., 1974 [64]] [Salgado, P. [65]].

A existência de informação numérica e o uso de algoritmos de aprendizagem proporciona acrescidamente a possibilidade de refinar as funções de pertença de uma forma sistemática.

5.1.2. Controlo por modelo invertido

O conhecimento do modelo matemático da planta a controlar, representada por uma função f , proporciona uma outra forma de construir o modelo do controlador. A existência da sua inversa, f^{-1} , é garantia para a realização do controlador. Neste caso a função do controlador, g , será igual à função inversa de f .

Em muitos outros casos não se conhece verdadeiramente a função f . Ela, porém, poderá ser estimada ou apreendida por um sistema de aprendizagem, que com vantagens óbvias identificará desde logo a sua inversa.

Um esquema desse processo está representado na Figura 52. Inicialmente processa-se uma fase de aprendizagem do modelo inverso, Figura 52 a), ao que se segue a fase da sua aplicação como controlador, Figura 52 b).

Na fase de aprendizagem, um conjunto de treino é obtido pela geração aleatória das atuações $u(k)$, e pela observação das correspondentes saídas $\bar{x}(k)$, produzidas pela planta. Um modelo de identificação difuso (SID ou FIS) é então usado para aprender o modelo inverso da planta.

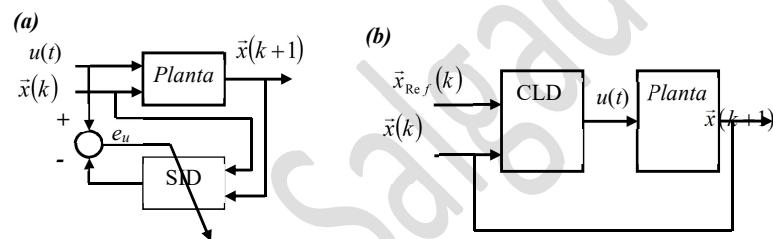


Figura 52 - Diagrama de blocos do método de controlo inverso: (a) fase de aprendizagem; (b) fase de aplicação.

Na fase de aplicação, o modelo SID é transportado para o controlador de Lógica difusa (CLD), para a geração de atuações que levem a saída da planta (trajetória das variáveis de estado, $\bar{x}(k)$) para os valores de referência (trajetória referência das variáveis de estado, $\bar{x}_{Ref}(k)$).

Este processo de construção do controlador apresenta algumas desvantagens. Em primeiro lugar tem como condição necessária a existência do modelo inverso da planta, o que nem sempre é verificável. Se satisfizer esta condição, carece de um processo de aprendizagem para se encontrar o modelo inverso da planta. Mais ainda, a minimização do erro quadrático das atuações, $\|e_u(k)\|^2$, por parte do sistema SID, na fase de aprendizagem, não garante a minimização do erro do sistema $\|\bar{x}_{Ref}(k) - \bar{x}(k)\|^2$, na fase de controlo.

Alguns exemplos deste tipo de controlo podem ser encontrados em numerosos artigos

[⁶⁶][⁶⁷][⁶⁸]. Outras técnicas [D. Psaltis et al.⁶⁹] evitam o problema maior do esquema de controlo inverso atrás referido. Em alternativa, deve minimizar-se o erro do sistema em vez do erro da rede na fase de aprendizagem. Nestas condições, é necessário propagar o sinal de erro através do modelo da planta, por uma linearização local do modelo da planta [P. Salgado, 1997] ou pelo conhecimento da matriz Jacobiana [J.-S. R. Jang, 1992^[70]].

5.1.3. Gain Scheduling

Nesta secção, consideramos a estrutura de controlo hierarquizado em dois níveis, como mostra a Figura 53. O primeiro nível consiste num controlador convencional (por exemplo um controlador Proporcional-Integrativo-Derivativo, PID) e um segundo nível compreendendo um controlador difuso, que supervisiona o funcionamento do controlador inferior.

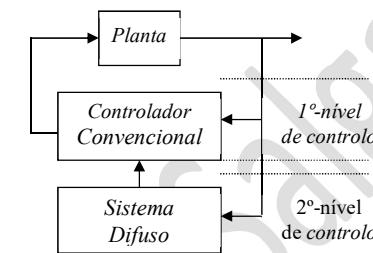


Figura 53 - Arquitetura do *Gain Scheduling*, em que o 1º nível é um controlador convencional e o 2º nível consiste num sistema difuso que supervisiona e modifica o funcionamento do controlador convencional.

Nesta linha, é proposto aqui mostrar, de forma introdutória, dois dos mais usados controladores difusos do tipo *Gain Scheduling*: o controlador PID com *gain scheduling* (*gain scheduling of PID controller*) e o sistema difuso TSK [Takagi and Sugeno (1983,^[71])].

Devido às suas características de estrutura simples e robusta, os controladores PID são os mais largamente usados no controlo de processos industriais. A função de transferência do controlador PID tem a seguinte forma:

$$u(t) = k_p \left(1 + \frac{1}{T_i} \int_0^t e(t) dt + T_d \frac{de(t)}{dt} \right) \quad (0.62)$$

em que $e(t)$ é o sinal de erro, $u(t)$ a saída do controlador, k_p o ganho proporcional, T_i ($k_i = 1/T_i$) a constante de tempo integrativa e T_d ($k_d = 1/T_d$) a constante de tempo derivativa (em que $T_d = \alpha T_i$).

Assume-se que as entradas dos sistemas difusos são os sinais de erro, $e(t)$, e derivada do

erro, $\dot{e}(t)$, e que as saídas são os valores normalizados dos parâmetros do PID. Assim, cada um dos três controladores supervisionador difuso têm por tarefa a sintonia de um dos parâmetros do PID, como mostra a Figura 54.

Sobre certas condições, o modelo difuso de 1^a ordem de Sugeno, sob o ponto de vista do controlo, constitui um *gain scheduler*, que comuta de modo difuso entre vários conjuntos de ganhos de realimentação ou de controladores. A consequência de uma regra é, neste caso, função das variáveis linguísticas de entrada. Por simplicidade, assume-se aqui existirem apenas duas regras de controlo:

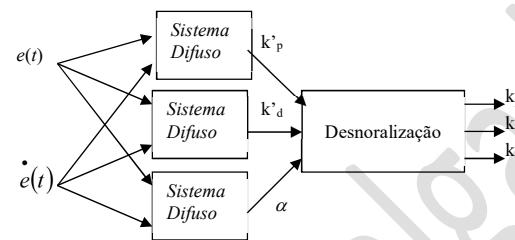


Figura 54 - Sintonização dos ganhos do PID por sistemas difusos.

$$R^1 : \text{IF } x_1 \text{ is } A_1^1 \text{ and } x_2 \text{ is } A_2^1 \text{ THEN } u \text{ is } f_1(x_1, x_2) \quad (0.63)$$

$$R^2 : \text{IF } x_1 \text{ is } A_1^2 \text{ and } x_2 \text{ is } A_2^2 \text{ THEN } u \text{ is } f_2(x_1, x_2) \quad (0.64)$$

Os valores das ações de controlo inferidos pela primeira e segunda regra são α_1 e α_2 , respectivamente. Consequentemente, a ação de controlo é dada por

$$u = \frac{\alpha_1 \cdot f_1(x_1, x_2) + \alpha_2 \cdot f_2(x_1, x_2)}{\alpha_1 + \alpha_2} \quad (0.65)$$

Da equação (0.65), podemos interpretar a saída de controlo, u , como a soma pesada dos valores de atuação dadas pelo controlador f_1 e f_2 .

6. Sumário

Neste capítulo foram referidos os sistemas de lógica difusa, a sua arquitetura básica e metodologias do desenho. Se bem que tal não tenha sido demonstrado, estes sistemas constituem bons aproximadores de funções não-lineares, a qualquer grau arbitrário de precisão. Foram revistos vários tipos de SLD e métodos de identificação utilizados em engenharia. Foram

estudados com maior detalhe dois métodos de aprendizagem difusa: o método de “back-propagation” e o método de “Table-Lookup”. Foram ainda revistos vários métodos de controlo difusos.

Capítulo IV

ALGORITMOS EVOLUTIVOS

1. Algoritmos genéticos

Os algoritmos genéticos são algoritmos inspirados nos mecanismos da seleção natural, da genética, e da evolução. Com os conhecimentos científicos atuais é consensual que a evolução da vida está intimamente ligada aos cromossomas, moléculas orgânicas que codificam a estrutura da vida. Ao longo de sucessivas gerações, as populações evoluem de acordo com os princípios da seleção natural e da “sobrevivência das espécies”, tal como expôs Charles Darwin (1859) no livro “*The Origin of Species*”. Assim, os elementos de uma população com melhores características de adaptação ao seu meio ambiente têm maior probabilidade de sobreviver e de procriarem, passando as suas melhores características aos seus descendentes. Geração após geração, a acumulação de pequenas variações pode, consequentemente, mudar as características da população, ou seja, em outras palavras a *evolução* da espécie.

A seleção natural é realizada pelo desempenho que um dado indivíduo apresenta num dado ambiente, estando intimamente ligado com o sucesso da descodificação e desempenho da estrutura cromossómatica. A seleção natural premeia os cromossomas que codificam a estrutura com sucesso, reproduzindo-a mais vezes que os congêneres que não apresentam igual desempenham. A *reprodução* é um mecanismo pelo qual os indivíduos (cromossomas) são copiados de acordo com o seu valor de desempenho, mediante operações de *cruzamento* e *mutação*.

O processo de *cruzamento* envolve o gerar dos cromossomas dos filhos pela combinação diferenciada do material genético dos pais, resultando, em princípio, cromossomas diferentes dos seus pais. Durante, ou no final deste processo, pode haver mutação genética, fazendo com que os cromossomas filhos sejam diferentes dos seus pais biológicos.

Os princípios básicos dos Algoritmos Genéticos (GA) foram inicialmente estabelecidos por Holland (1975 [72]), e estão bem descritos em vários textos científicos

[73] [74][75].

As potencialidades dos GA advêm do facto de esta técnica ser robusta, e poder ser utilizada num grande leque de problemas, incluindo-se aqueles em que as técnicas convencionais não existem ou apresentam grandes dificuldades. Não existem garantias que os GA encontrem a solução ótima global de um problema, mas são geralmente capazes de encontrar uma solução “aceitavelmente boa” de forma “aceitavelmente rápida”. A sua combinação com outras técnicas convencionais pode contribuir para o refinamento das soluções encontradas. Assim, os algoritmos genéticos têm sido usados para resolver problemas difíceis, em que a função objetivo não apresenta propriedades tais como continuidade, diferenciabilidade, que não satisfaça a Condição de Lipschitz, etc. [72-75].

2. Princípios Básicos

Na Figura 55 está representado o fluxograma do Algoritmo Genético (GA) standard.

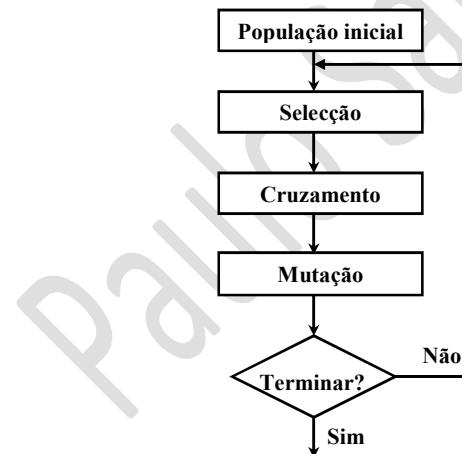


Figura 55: Fluxograma de um Algoritmo Genético Tradicional

Antes de os GA funcionarem é necessário codificar (ou representar) o problema eficazmente. Também é necessário construir e especificar a função de desempenho, que irá avaliar o mérito de cada solução codificada. Durante o funcionamento, os pais devem ser selecionados para a reprodução, e recombinados para gerar filhos.

A função de desempenho $f(i)$ assinala para cada indivíduo da população o seu

desempenho, em que altos valores representam um bom desempenho. A função pode ser não linear, não diferenciável, descontínua, e positiva.

Desta forma, os algoritmos genéticos podem ser usados na procura de uma solução de uma função através do uso da evolução, realizada de modo simulado no computador.

No contexto deste capítulo, dar-se-á maior ênfase ao uso dos algoritmos genéticos como método de otimização.

De modo geral, os problemas de otimização são formulados como se segue:

- i) Criar uma população P_0 de N indivíduos e respetivo valor de desempenho.
- ii) $i \leftarrow 1$
- iii) $P_i \leftarrow função_selecção(P_{i-1})$
- iv) $P_i \leftarrow função_reprodução(P_i)$
- v) avaliar(P_i)
- vi) $i \leftarrow i+1$
- vii) Se o critério de paragem não é satisfeito voltar ao passo 3.
- viii) Mostrar a melhor solução encontrada.

Os algoritmos genéticos providenciam uma população inicial. O método mais comum é gerar aleatoriamente soluções para a população inteira. Contudo os algoritmos genéticos podem melhorar iterativamente as soluções existentes, e consequentemente, a população inicial pode ser semeada com potenciais boas soluções. Os algoritmos genéticos movem-se de geração em geração selecionando e reproduzindo pais até um critério de paragem ser atingido. O critério de paragem mais usualmente usado é o de especificar um certo número de gerações. Outro envolve um critério de convergência da população. Geralmente os algoritmos genéticos irão forçar a população inteira a convergir para uma única solução. Quando a soma dos desvios entre indivíduos se tornar menor de uma especificada, o algoritmo pode ser terminado. O algoritmo pode também terminar devido à não existência do melhoramento da solução depois de um determinado número de gerações. Alternativamente, pode escolher-se um valor para a função de avaliação como aceitável. Diversas estratégias podem ser usadas em conjunção com outras.

3. Representação

A representação da solução de um problema pressupõe que esta possa ser apresentada sob a forma de um conjunto de parâmetros (por exemplo, a sequência das cidades a percorrer por um caixeiro viajante). Estes parâmetros, conhecidos como *genes*, são ligados entre si formando uma frase de valores, também designada de *cromossoma*. Por exemplo, se o nosso problema é maximizar a função $f(x,y,z)$ de três variáveis, pode representar-se cada variável por um número binário, conjunto de 10 bits (escalonado). Assim o nosso cromossoma contém três genes formados por 30 dígitos binários.

Em termos gerais, cada indivíduo ou cromossoma é realizado pela sequência de genes de um certo tipo de alfabeto. Um tipo de alfabeto pode consistir em:

- dígitos binários (0 e 1)
- números de vírgula flutuante
- inteiros
- símbolos (isto é, A, B, C, D, ...)
- matrizes ...

De todas as formas de representação acima descritas, a representação em números naturais é, de todas, aquela que apresenta maior eficácia de codificação enquanto que a representação em números reais (de vírgula flutuante) é a mais eficiente em termos de tempo de CPU.

Em termos genéticos, o conjunto de parâmetros representados por um cromossoma particular é referenciado como um *genótipo*. O *genótipo* contém a informação necessária à construção do organismo, que é referido como o *fenótipo*. Estes termos, com o mesmo significado, são usados nos GA. Por exemplo, na escolha do trajeto (cidades) de um caixeiro-viajante, um conjunto de parâmetros (código das cidades) que especifica um particular trajeto designa-se por genótipo. A especificação do trajeto corresponde ao fenótipo. O valor de desempenho de cada indivíduo depende do comportamento do seu fenótipo. Este pode ser inferido do genótipo, isto é, pode ser calculado do cromossoma, usando a função de desempenho.

4. Função de desempenho

A função de desempenho deve ser criada para cada problema a ser resolvido. Dado um cromossoma particular, o resultado da função de desempenho é um valor numérico de “desempenho” ou “figura de mérito”, que reflete proporcionalmente a “utilidade” ou “habilidade” do indivíduo que o cromossoma representa.

Para um grande leque de problemas, a função de desempenho monitoriza a função a otimizar, podendo retornar o seu valor. Noutros casos, pode envolver a solução de uma optimização combinatória.

A função de desempenho $f(i)$ assinala para cada indivíduo da população o seu desempenho, em que altos valores representam um bom desempenho. A função pode ser não linear, não diferenciável, descontínua e positiva.

5. Reprodução

Durante a fase de reprodução, os indivíduos são selecionados de uma população e recombinados, produzindo filhos que farão parte da próxima geração. Os pais são selecionados aleatoriamente da população usando-se um esquema que favorece os melhores indivíduos. Estes terão maior probabilidade em ser escolhidos (uma ou múltiplas vezes) numa geração, contrariamente ao que pode acontecer aos indivíduos piores.

Existindo dois pais, os seus cromossomas são recombinados, tipicamente usando o mecanismo de *cruzamento* (“crossover”) e *mutação*.

6. Seleção

A seleção de pais é uma tarefa que atribui oportunidades reprodutivas a cada indivíduo. Em princípio, os indivíduos da população são copiados para a “mating pool” (lagoa nupcial), podendo os indivíduos de alto valor de desempenho ser copiados mais que uma vez enquanto os indivíduos de fraco desempenho podem não ser copiados. Após este processo, pares de indivíduos da “mating pool” são selecionados ou não, aleatoriamente segundo uma probabilidade de cruzamento, p_c , e sujeitos ao mecanismo de cruzamento. Este processo é repetido para todos os indivíduos da “mating pool”. O tamanho desta pode ser igual ou inferior ao tamanho da população. No primeiro caso, a população original é totalmente substituída.

O desempenho dos GA é extremamente dependente da forma como os indivíduos

são selecionados para o “mating pool”. Este procedimento pode ser dividido em dois tipos de metodologia. O primeiro, toma o valor de desempenho de cada indivíduo, mapeia-o numa nova escala, e usa este último valor como o número de cópias do indivíduo a colocar no “mating pool”. O outro método é legatário do mesmo efeito, mas sem realizar o passo intermédio de calcular um valor de desempenho modificado, ou seja, utiliza diretamente o valor de desempenho original. Os que obedecem à primeira metodologia designa-se por remapeamento explícito de desempenho e aos que seguem a segunda metodologia tomam o nome de remapeamento implícito de desempenho.

Sem os caracterizar, apresentam-se em seguida um conjunto de métodos de seleção.

Método da Roleta

O método da roleta foi desenvolvido por Holland [76], e foi o primeiro método de seleção a ser usado.

A seleção de um pai, segundo o método da roleta, é realizada pela rotação de uma roleta simulada, cujas aberturas têm diferentes tamanhos proporcionais ao valor de desempenho de cada indivíduo. De cada vez que é necessário um filho, uma simples rotação da roleta, com diferentes divisórias (como mostra a figura 1.2, referente ao exemplo 1.1.), seleciona um candidato. Esta técnica pode ser implementada algoritmicamente através da execução dos seguintes passos:

- Somar os valores de desempenho de todos os membros da população e chamar a este resultado, desempenho total.
- Gerar um valor aleatório num número aleatório entre 0 e o desempenho total.
- Retornar o primeiro elemento da população cujo desempenho, somado com os desempenhos precedentes dos membros da população, seja maior ou igual a n .

Desta forma assegura-se que a probabilidade de um candidato ser selecionado, é dada pela seguinte expressão:

$$P[\text{escolha do individuo } i] = \frac{F_i}{\sum_{j=1}^{\text{PopSize}} F_j} \quad (1.1)$$

ou seja, proporcional ao seu desempenho.

O exemplo seguinte ilustra a técnica de seleção pelo método da roleta.

Exemplo 1.1: Considere-se uma população de seis cromossomas cujo valor total cumulativo do desempenho é 50, como mostra a tabela seguinte.

A técnica consiste em gerar números aleatórios no intervalo 0 a 50. Para cada número é selecionado um pai que apresente o primeiro valor de desempenho total acumulado maior ou igual que o número aleatório.

Nº	String (Cromossoma)	Desempenho	% do total	Desempenho total
1	01110	8	16	8
2	11000	15	30	23
3	00100	2	4	25
4	10010	5	10	30
5	01100	12	24	42
6	00011	8	16	50

O quadro seguinte apresenta, para um conjunto de 7 números gerados aleatoriamente, os índices dos cromossomas selecionados. Na Figura 56 está representado um esquema de uma roleta, onde cada uma faixa está associado a um indivíduo, enquanto a sua largura reflete percentualmente o seu desempenho. Indivíduos com maior desempenho têm associado faixas da roldana de maior largura do que sucede com os indivíduos de menor desempenho, aumentando a probabilidade de serem escolhidos para o processo de reprodução.

Número aleatório	26	2	49	15	40	36	9
Escolha do cromossoma nº	4	1	6	2	5	5	2

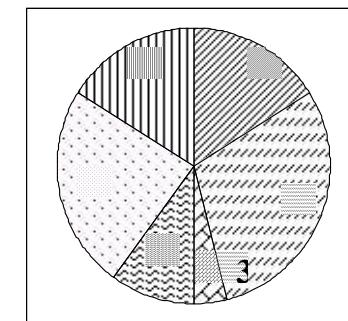


Figura 56 - Roleta pesada do exemplo 1.1.

Ranking (ordenação)

O processo de seleção por ordenação (ranking) é um procedimento não paramétrico. Neste método, a população é ordenada de acordo com o seu valor de desempenho. A seleção de um indivíduo é realizada unicamente em função da sua posição no “ranking”.

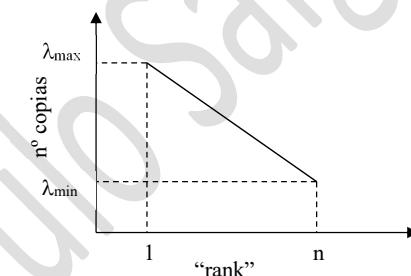


Figura 57 - Ranking segundo Baker.

Uma função proposta por Baker (1985) [77] está representada na Figura 57. Especificamente, o número esperado de cópias do indivíduo i é calculado de acordo com:

$$e_i = -\frac{2(\lambda_{\max} - 1)}{n-1} \text{rank}(i) + 1 + (\lambda_{\max} - 1) \frac{n+1}{n-1} \quad (1.2)$$

em que λ_{\max} é um valor bem definido, $1 \leq \lambda_{\max} \leq 2$, e n é o tamanho da população. A gama de valores e_i é o intervalo $[2 - \lambda_{\max}, \lambda_{\max}]$. A função da equação (1.2) é um caso especial da figura 1.3, considerando-se para este caso que $\lambda_{\min} = 2 - \lambda_{\max}$.

Os métodos de ordenação apenas requerem o conhecimento do valor da função de desempenho de todos os indivíduos para construir um conjunto parcialmente ordenado

dos indivíduos. O método de ranking assinala uma probabilidade P_i , do elemento i da lista ordenada de todos os elementos da população, ser selecionado.

A *ordenação geométrica normalizada* [78] define a probabilidade P_i para cada indivíduo dado por:

$$\begin{aligned} P[\text{escolha do individuo } i] &= q'(1-q)^{r-1} \\ q &= \text{probabilidade de escolha do melhor elemento} \\ r &= \text{rank do individuo (1 = o melhor)} \\ P &= \text{tamanho da população} \end{aligned} \quad (1.3)$$

$$q' = \frac{q}{1-(1-q)^P}$$

Torneio

A seleção por torneio (*Tournament*), tal como o método de ordenação, apenas requer os valores de desempenho de todos os indivíduos para construir uma lista parcialmente ordenada, mas sem atribuir uma probabilidade de seleção. Este método escolhe aleatoriamente j indivíduos da população, inserindo o melhor deste conjunto na nova população. Este procedimento é repetido até que sejam selecionados n indivíduos (para uma completa substituição da população).

7. Cruzamento

Este operador é o responsável pela mistura e recombinação de blocos de código genético. Constitui o mais poderoso operador e pode ser considerado como o principal responsável pelo mecanismo de pesquisa dentro dos GAs.

Uma forma simples de realizar o cruzamento está esquematizada na Figura 58. Selecionados dois indivíduos, os seus respetivos cromossomas são quebrados numa posição escolhida aleatoriamente, produzindo dois segmentos “cabeças” e dois segmentos de “cauda”. Os dois segmentos de “cauda” são então trocados entre si para produzir dois novos e completos cromossomas. Os dois filhos, assim criados, contêm genes de ambos os seus pais. Este processo é conhecido como um cruzamento simples.

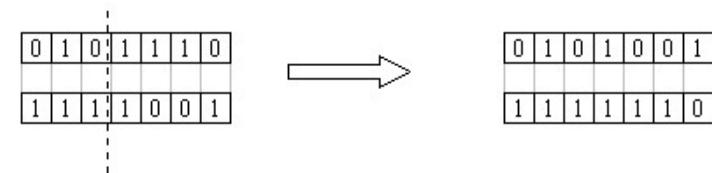


Figura 58 - Cruzamento simples

Sejam dois cromossomas X e Y de comprimento m , designando dois indivíduos (pais) da população. A operação de cruzamento simples resume-se aos dois seguintes passos:

- Gerar um número aleatório r no intervalo $[1, m]$, segundo uma distribuição de probabilidades uniforme.
- Construir dois cromossomas filhos, a partir da informação genética dos pais:

$$x'_i = \begin{cases} x_i & ; \text{se } i < r \\ y_i & ; \text{outros} \end{cases} \quad (1.4)$$

$$y'_i = \begin{cases} y_i & ; \text{se } i < r \\ x_i & ; \text{outros} \end{cases} \quad (1.5)$$

Outros operadores de cruzamento podem ser realizados, geralmente envolvendo mais que um ponto de quebra. DeJong [79] investigou a eficácia do cruzamento multi-quebra, e concluiu ([73], p119) que o cruzamento de dois pontos de quebra apresenta melhor desempenho, dado que mais pontos de quebra reduzem a performance do GA. O aumento do número de pontos de quebras faz crescer o número de quebras e de blocos de código, mas traz como vantagem a minuciosa pesquisa da solução.

No cruzamento com dois pontos de quebra (e em cruzamento de multi-quebras) os cromossomas são vistos de forma circular, com os seus extremos ligados. A troca de um segmento de um círculo para o outro círculo requer a seleção de dois pontos de quebra, tal como mostra a Figura 59.

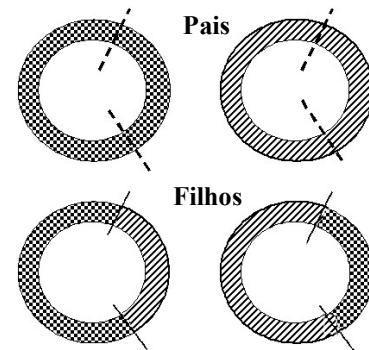


Figura 59 - Representação circular dos cromossomos e o cruzamento com dois pontos de quebra.

Nesta perspectiva, o cruzamento simples pode ser visto como o cruzamento de 2 pontos de quebra com os dois pontos de quebra coincidentes. Assim, o cruzamento com 2 pontos de quebra processa-se da mesma maneira que no caso simples (isto é, com a troca de um único segmento), mas de forma mais geral.

A operação de cruzamento para dois pontos de quebra, processa-se como descrito em seguida.

- Gerar dois números aleatórios, r e s , no intervalo $[1, m]$, segundo uma distribuição de probabilidades uniforme, com $r \leq s$.
- Construir dois cromossomos filhos, a partir da informação genética dos pais:

$$x'_i = \begin{cases} y_i & ; \text{ se } s \leq i \leq r \\ x_i & ; \text{ outros} \end{cases} \quad (1.6)$$

$$y'_i = \begin{cases} x_i & ; \text{ se } s \leq i \leq r \\ y_i & ; \text{ outros} \end{cases} \quad (1.7)$$

A generalização da operação cruzamento a mais que dois pontos é perfeitamente compreensível, pela alternância de troca de segmentos de código genético, como mostra a Figura 60.

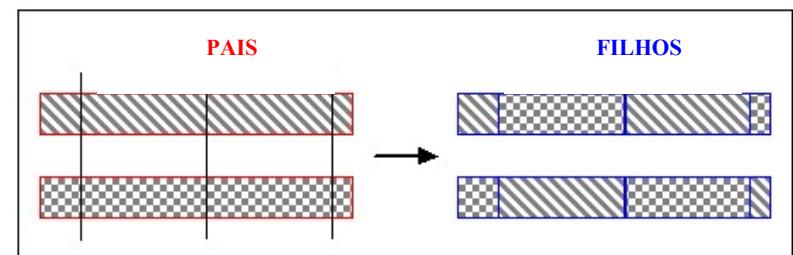


Figura 60 - Operação de cruzamento em múltiplos pontos de quebra.

Uma outra operação, substancialmente diferente das até aqui apresentadas, designa-se por *cruzamento uniforme*. Cada gene de um filho é criado pela cópia do correspondente gene de cada um dos seus pais, de acordo com o valor de uma *máscara de cruzamento*, gerada aleatoriamente. Assim, o gene do filho, a ser criado, é construído pela cópia do gene respetivo do 1º pai se o valor da máscara de cruzamento for unitário, se não (máscara 0) é copiado do gene do 2º pai, tal como mostra a Figura 61. O processo é repetido com a posição dos pais trocadas para a criação do segundo filho. Para diferentes pares de pais novas máscaras de cruzamento são geradas aleatoriamente.

Máscara de Cruz	1	0	0	1	1	0	1	1	0	1
1º Pai	1	0	1	0	0	0	1	1	1	0
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
1º Filho	1	1	1	0	0	1	1	1	0	0

2º Pai	0	1	0	0	1	1	0	0	0	1
↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
2º Filho	0	0	1	1	0	1	0	1	1	0

Figura 61 - Cruzamento uniforme.

O algoritmo do cruzamento uniforme pode ser implementado nos dois seguintes passos:

- Gerar a máscara de cruzamento, z , de números binários gerados aleatoriamente, segundo uma distribuição de probabilidades uniforme.
- Construir dois cromossomos filhos, a partir da informação genética dos pais, segundo os valores da máscara z :

$$x'_i = \begin{cases} x_i & ; \text{ se } z_i = 1 \\ y_i & ; \text{ se } z_i = 0 \end{cases} \quad (1.8)$$

$$y'_i = \begin{cases} y_i & ; \text{ se } z_i = 1 \\ x_i & ; \text{ se } z_i = 0 \end{cases} \quad (1.9)$$

A operação de cruzamento não é, geralmente, aplicada a todos os pares de indivíduos selecionados para o “casamento”. Uma escolha aleatória é realizada, em que a probabilidade de cruzamento, p_c , assume valores tipicamente entre 0,6 e 1,0. Se a operação de cruzamento não é aplicada, os filhos são produzidos, simplesmente, pela duplicação dos seus pais. Esta circunstância permite, a cada indivíduo, a oportunidade de passar a sua informação genética intacta, sem a disruptão do cruzamento, para a geração seguinte.

8. Mutação

A mutação é um operador de manipulação genética, que envolve alterações aleatórias dos genes, durante o processo de cópia de um cromossoma de uma geração para a seguinte. A probabilidade de um gene ser alterado, p_m , é, tipicamente baixa, na ordem de 0,001. A Figura 62 mostra que o 1º, 8º e 9º bits sofreram mutação.

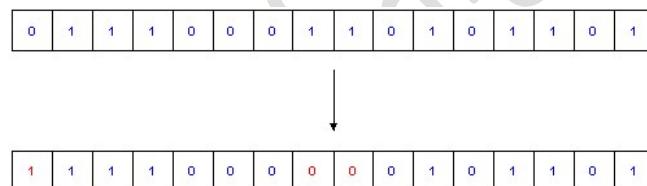


Figura 62 - Exemplo de Mutação binária

O cruzamento, como se viu, é muito importante como técnica para a rápida exploração no espaço de pesquisa da solução. A mutação providencia uma pesquisa aleatória, que ajuda a assegurar que qualquer ponto do espaço de pesquisa tenha uma probabilidade não nula de ser considerado.

A mutação é, adicionalmente, usada para evitar a convergência prematura, que constitui um problema frequente na utilização dos GAs. Por exemplo, quando o processo de seleção utilizado é proporcional ao desempenho, de geração em geração, todos os cromossomas da população se tornam muito similares, mesmo antes de se alcançar a solução ótima (ou sub-óptima). Em tais casos a mutação é essencial, pois atua contra este fenômeno, por constantemente gerar novos genes, e assim prevenir que a população convirja para um máximo local. Todavia, algumas vezes pode resultar na perda de um

bom indivíduo. Pelo exposto, é necessário fazer o balanço entre a convergência prematura, pela inserção de diferente informação genética, e a perda de eficiência, devido à danificação de bom material genético.

A operação de mutação é usada, pouco frequentemente, ou seja com uma baixa probabilidade de mutação, p_m .

A *mutação binária* inverte bits aleatoriamente escolhidos das “strings”, de acordo com a seguinte equação:

$$x'_i = \begin{cases} 1 - x_i & ; \text{ se } U(0,1) < p_m \\ x_i & ; \text{ outros} \end{cases} \quad (1.10)$$

Um operador de mutação para representação de valores reais, isto é, um alfabeto de “floats”, foi desenvolvido por Michalewicz [1974]. Para dois cromossomas em representação real X e Y , as seguintes operações são definidas: mutação uniforme, mutação não-uniforme, mutação multi-uniforme e mutação de fronteira. Sejam a_i e b_i os limites inferior e superior, respectivamente, de cada variável i .

A *mutação uniforme* seleciona aleatoriamente uma variável j , e substitui-a por um número aleatório, no intervalo a_i e b_i , com distribuição uniforme $U(a_i, b_i)$:

$$x'_i = \begin{cases} U(a_i, b_i) & ; \text{ se } i = j \\ x_i & ; \text{ outros} \end{cases} \quad (1.11)$$

A *mutação de fronteira* seleciona aleatoriamente uma variável j , e substitui-a por um dos seus extremos, em que $r = U(0,1)$:

$$x'_i = \begin{cases} a_i & ; \text{ se } i = j, r < 0.5 \\ b_i & ; \text{ se } i = j, r \geq 0.5 \\ x_i & ; \text{ outros} \end{cases} \quad (1.12)$$

A *mutação não-uniforme* seleciona aleatoriamente uma variável j , e substitui-a por um número aleatório não uniforme:

$$x'_i = \begin{cases} x_i + (b_i - x_i)f(G) & ; \text{ se } r < 0.5 \\ x_i - (x_i + a_i)f(G) & ; \text{ se } r \geq 0.5 \\ x_i & ; \text{ outros} \end{cases} \quad (1.13)$$

em que:

$$f(G) = \left(r_2 \left(1 - \frac{G}{G_{\max}} \right) \right)^{\alpha} \quad (1.14)$$

r_1, r_2 = número aleatório uniforme no intervalo $[0,1]$,

G = geração corrente,

G_{\max} = número de gerações máximo,

α = fator forma.

A mutação multi-uniforme aplica o operador de mutação não-uniforme a todas as variáveis que fazem parte do pai X .

As três operações básicas dos GS (seleção, cruzamento e mutação) são aplicadas repetidamente até que os filhos gerados constituam a nova população. Desta forma, a geração seguinte é constituída por três tipos de filhos resultado de: mutação cruzamento seguido de mutação; cruzamento e ausência de mutação e nem cruzamento nem mutação, mas selecionado.

Geralmente, para um algoritmo genético simples, é necessário especificar quatro parâmetros:

n = tamanho da população,

p_c = probabilidade de cruzamento,

p_m = probabilidade de mutação

G = sobreposição de gerações

A sobreposição de gerações foi introduzida por De Jong para permitir sobreposições de populações. O seu valor está compreendido entre 0 e 1, limites estes que correspondem à sobreposição total ou ausência de sobreposição, respetivamente.

9. Aplicações dos Algoritmos Genéticos

Para mostrar as potencialidades da utilização dos algoritmos genéticos, apresenta-se -ão nesta secção dois exemplos ilustrativos. No primeiro, o Algoritmo Genético é usado para encontrar o máximo de uma função não linear, problema no qual os métodos tradicionais de otimização não linear falham. O segundo visa encontrar um conjunto de regras difusas capazes de controlar eficazmente um sistema de pêndulo invertido. Neste exemplo, as melhores relações entre os conjuntos difusos das variáveis de entrada e de saída, previamente estabelecidas, são encontradas. Estas relações podem ser

representadas em forma tabelar. Por ultimo, os Algoritmos Genéticos são usados na construção de classificadores baseados em Redes Neuronais, tendo como tarefa encontrar os pesos dos neurónios que separam um conjunto de classes.

Exemplo 1.2: Otimização de uma função

O problema a resolver consiste em encontrar o ponto do espaço (x, y) que maximize a seguinte função:

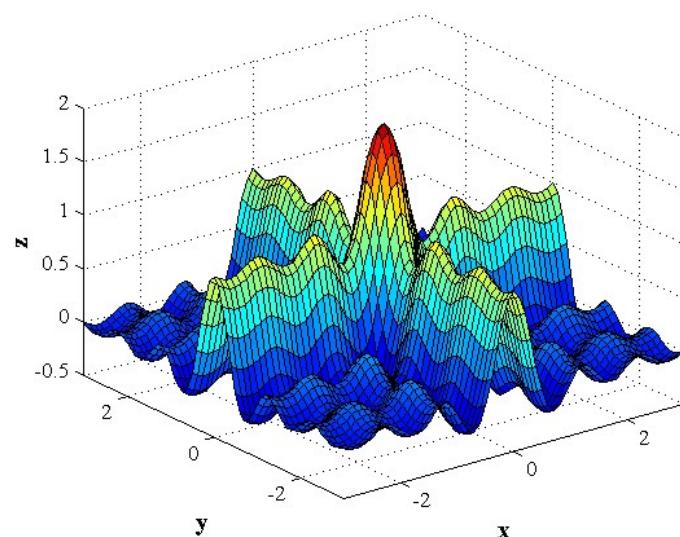
$$z = f(x, y) = \frac{\sin(x)}{x} + \frac{\sin(y)}{y} \quad (1.15)$$

O gráfico da função f está representado na Figura 63, para x no intervalo $[-3;3]$ e y no intervalo $[-3;3]$. Constata-se a existência de múltiplos máximos e mínimos locais, sendo o ponto $(0,0)$ aquele em que a função f assume o valor máximo, assumida como desconhecida antes da aplicação dos GA.

Os indivíduos, que fazem parte, em cada geração, da mesma população, constituem-se individualmente como solução do problema. A representação num cromossoma de um indivíduo, é neste caso uma cadeia de dois genes do tipo real (float), sendo o primeiro a variável x e o segundo a variável y . A população inicial é de 20 elementos gerados de forma aleatória, com os genes x e y no intervalo $[-4; 4]$.

A operação de seleção eleita para o presente problema foi a *ordenação geométrica normalizada*, sendo a probabilidade de se escolher o melhor elemento de 0,08.

As operações de cruzamento utilizadas são, simultaneamente, uma aritmética, uma heurística e uma simples. Quatro operações de mutação são usadas, a saber: de fronteira (2 vezes); multi-não-uniforme (3 vezes); não uniforme (2 vezes) e uniforme (2 vezes). O número de gerações máxima foi de 50, sendo este o critério de paragem.

Figura 63 - Gráfico da função $f(x,y) = \sin(x)/x + \sin(y)/y$

O melhor resultado aconteceu na geração 47, com o seguinte cromossoma e valor de desempenho:

<i>x</i>	<i>y</i>	Valor de desempenho
0.00331460109571	0.00190298616708	1.99999756534499

Como se observa na Figura 63 a solução apontada pelo algoritmo ótimo é um valor muito próximo do valor ótimo, ponto (0,0), com valor de desempenho de 2.

Foi realizado um exemplo parecido na aula. Construíram-se as rotinas para os operadores genéticos.

Exemplo 1.3: Algoritmos genéticos na construção de um controlador difuso

Os algoritmos genéticos apresentam-se como muito virtuosos na procura de soluções, que do ponto de vista matemático são extremamente difíceis de obter. O exemplo que se segue é prova disso. O problema consiste em construir um controlador de lógica difusa que indique a actuação óptima u para o sistema de pêndulo invertido (ou sistema carro-haste), representado na Figura 64. Após serem estabelecidas as variáveis difusas de cada uma das variáveis do processo, a tarefa do algoritmo GA será encontrar

as relações entre estas variáveis de estado (ver Figura 65) e a variável de atuação, garantindo um menor tempo no movimento que conduza a haste à posição vertical com velocidade angular nula. Estas relações podem ser registadas em forma de tabela como mostra a Figura 66.

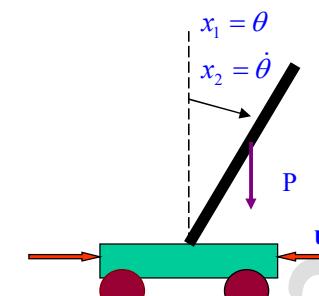


Figura 64 - Sistema do pêndulo invertido

Sejam $x_1 = \theta$ e $x_2 = \dot{\theta}$. As equações dinâmicas do sistema do pêndulo invertido são:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{g \sin x_1 - \frac{mlx_2^2 \cos x_1}{m_c + m}}{l\left(\frac{4}{3} - \frac{m \cos^2 x_1}{m_c + m}\right)} + \frac{\frac{\cos x_1}{m_c + m}}{l\left(\frac{4}{3} - \frac{m \cos^2 x_1}{m_c + m}\right)} u \end{aligned} \quad (1.16)$$

em que $g = 9,8 \text{ m/s}^2$ é a aceleração devido à gravidade, m_c é a massa do carro, m é a massa da haste, l metade do comprimento da haste, e u a força (de controlo) aplicada. Para o presente exemplo escolhemos $m_c = 1 \text{ kg}$, $m = 0,1 \text{ kg}$ e $l = 0,5 \text{ m}$.

A solução do presente problema resume-se ao preenchimento da tabela da Figura 66, com as designações dos conjuntos difusos associados à variável força, F . A codificação do problema passa pela numeração dos conjuntos (números) difusos com números inteiros (em detrimento das siglas S1, S2 e CE) e na sequenciação, num cromossoma, dos valores da tabela, segundo uma ordem previamente estabelecida (por ex., linha a linha). Assim, cada cromossoma de um indivíduo representa uma tabela candidata à tabela solução. A codificação dos genes poderá ser binária.

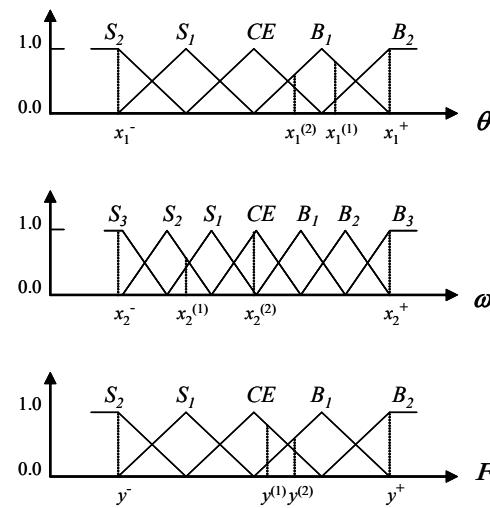


Figura 65 - Variáveis difusas da posição angular, velocidade angular e da força.

Uma outra forma, particular, de encarar a construção da tabela da Figura 66 consiste em associar a cada elemento da tabela um conjunto difuso próprio, representado pelo valor real do seu centro. Nestas circunstâncias o problema resume-se a encontrar um conjunto de variáveis reais, em número igual ao tamanho da tabela, da forma como foi abordada no exemplo anterior. O resultado do comportamento de um controlador difuso obtido desta forma está representado na Figura 71, onde é visível a convergência da posição e velocidade angular para valores próximos de zeros, em menos de um segundo.

		θ				
		S_2	S_1	CE	B_1	B_2
ω	S_3					
	S_2					
	S_1			CE		
	CE				B_1	
	B_1					
	B_2					
	B_3					

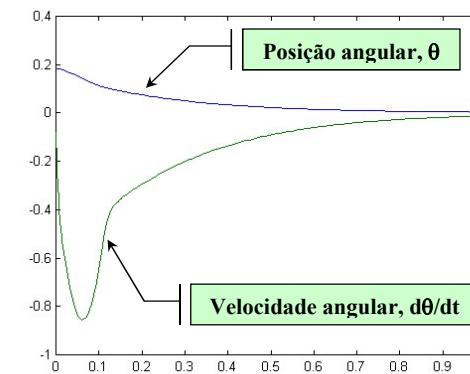
Figura 66 - Tabela-quadro onde estão especificadas as relações entre as variáveis difusas de entrada (θ , ω) e da saída (F).

Figura 67 - Resposta do sistema pêndulo invertido + controlador para uma condição de partida.

A Figura 68 mostra a evolução do desempenho do melhor cromossoma e da média de desempenho da população. A população é composta por 50 indivíduos. A operação de seleção e as operações de mutação e cruzamento usadas neste exemplo foram as mesmas que as referidas no exemplo 1.2.

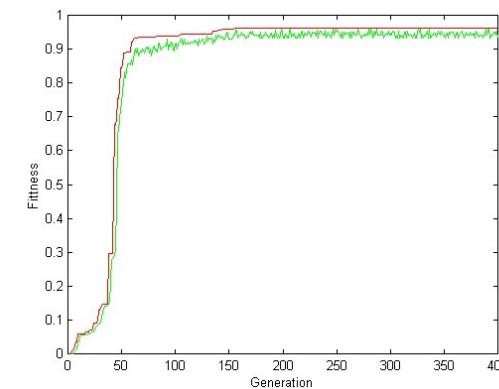


Figura 68 - Traçado do desempenho da população ao longo de várias gerações.

Exemplo dado na aula. Sugeri aos alunos a construção do algoritmo no MATLAB

Exemplo 1.4: Rede neuronal como classificador

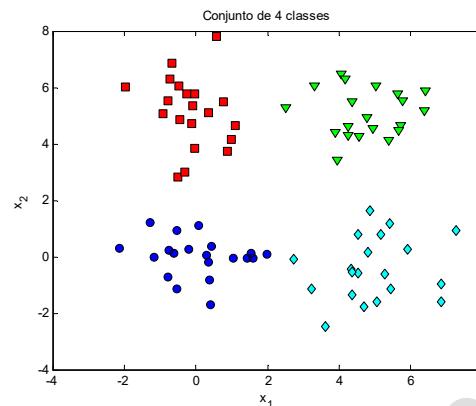


Figura 69 – Exemplo de um conjunto de 4 classes no espaço R^2 .

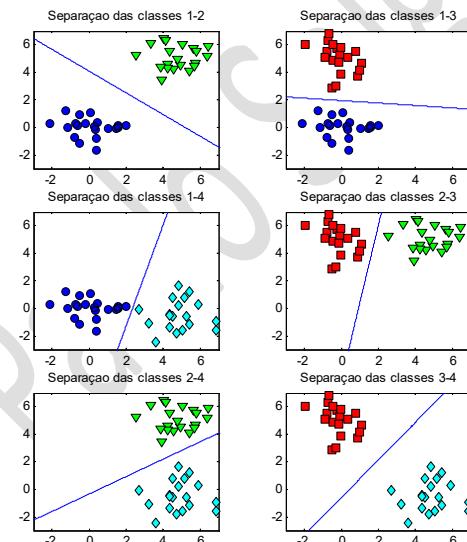


Figura 70 – Resultado do processo de classificação.

4.2. Programação genética

10. Otimização por exame de partículas

A otimização por exame de partículas (*Particule Swarm Optimization*, PSO) é um método popular baseado em técnicas de otimização estocástica desenvolvida pelo Dr. Eberhart e Dr. Kennedy em 1995, inspirada no comportamento social de bandos de pássaros, cardumes de peixes e de enxames de insetos. Neste método, as partículas viajam pelo espaço de soluções do problema, seguindo uma corrente de partículas ótimas. Cada partícula, fazendo parte de um exame, é um indivíduo com necessidades pessoais e sociais, que procura diretrizes para o seu comportamento recorrendo ao seu passado pessoal e da comunidade onde se integra.

O método PSO tem sido sucessivamente aplicado em diferentes áreas de pesquisa e na procura de soluções para problemas práticos, demonstrando ter um excelente desempenho, que resulta da rapidez e da sua simplicidade funcional, quando comparado com outros métodos. Contribui para o seu sucesso o facto de ser um método fácil de ser compreendido e possuir poucos parâmetros a ajustar.

PSO é uma forma de inteligência social e é inspirada em grupos espécies de animais, como os bandos de pássaros, os cardumes de peixe, etc. Neste grupo, se um dos elementos do enxame observa um caminho ótimo (por exemplo, encontrar comida, água, proteção) o resto do enxame logo tenderá em segui-lo. Paralelamente, cada indivíduo é dotado de um comportamento pessoalizado, tal como regressar a uma posição passada

mais marcante ou um ter a liberdade de realizar movimento errático ou exploratórios. Esta parcela estratégica do movimento melhora as capacidades exploratórias no espaço de pesquisa. Deste modo, os elementos de um enxame podem ser influenciados pelos outros elementos do enxame mas também são parcialmente independentes para explorar no espaço de trabalho de modo autónomo.

Cada partícula no espaço multidimensional pode ser caracterizada pela sua posição, velocidade e aceleração. Em cada posição ocupada a partícula sabe qual o valor de desempenho nesse local. Estas podem movimentar-se no espaço (usualmente em R^n) dotadas, geralmente, de duas capacidades cognitivas: possuírem memória sobre a sua melhor posição histórica e o conhecimento atualizado da melhor posição por onde o exame tenha passado.

Os elementos de um enxame comunicam as suas melhores posições aos restantes membros e ajustam a sua própria posição e velocidade baseada nessa informação. Tal é geralmente realizado o registo, de modo sempre atualizado, da seguinte informação:

- A melhor posição global de todos os elementos (*pbest*): Sempre que haja um elemento que encontre uma melhor posição global esta informação é comunicada e atualizada por todos.
- A melhor partícula da vizinhança (*lbest*): cada partícula comunica com as suas vizinhas (sub-conjunto do enxame) de modo a conhecerem a melhor posição entre elas. No limite a vizinhança de cada partícula é a própria partícula, pelo que pelo que será registado em memória a sua melhor posição histórica pessoal, *pbest*.

O processo de otimização por enxame de partículas consiste, em cada instante iterativo, mudar a velocidade de cada partícula, sujeitando-a a uma aceleração estatisticamente adequada. Neste processo decisório do comportamento cinemático são tidos em conta a vontade de a partícula se mover para as posições referenciadas por *lbest* e *gbest*.

Sejam $x_i(t)$ e $v_i(t)$, respetivamente, a posição e a velocidade da partícula i no instante t . A sua aceleração instantânea é o resultado da soma pesada de termos, com pesos de valores aleatórios, que conjugam direções que o conduziriam separadamente para as posições *pbest* e *lbest*: A adaptação do vetor velocidade da partícula é determinado por:

$$v_i(t+1) = \alpha v_i(t) + c_1 \times \text{rand} \times (pbest(t) - x_i(t)) + c_2 \times \text{rand} \times (gbest(t) - x_i(t)) \quad (?)$$

onde α é um peso de atrito, c_1 e c_2 são constantes de aceleração que intentam mudar a direção da partícula para as posições *pbest* e *lbest*, e rand é um número aleatório com distribuição uniforme no intervalo [0,1]. O primeiro termo desta equação ($\alpha v_i(t)$) impõe a velocidade anterior da partícula, correspondente à diversificação da pesquisa, enquanto os dois restantes são utilizados para mudar (a direção, sentido e amplitude) do vetor velocidade da partícula, se traduz numa intensificação da pesquisa. Sem o primeiro termo a velocidade é apenas determinada pela posição actual e os valores históricos *pbest* e *lbest*, contribuindo para que o método intensifique a pesquisa para essas posições. Doutro modo e apenas com o primeiro termo a partícula seguiria em frente (sem alterar a direção e sentido da velocidade) encaminhando-se para novas zonas do espaço (diversificação).

Em seguida a posição da partícula é adaptada:

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (??)$$

Em resumo, o método PSO pressupõe a execução iterativa dos seguintes passos:

Passo 1: Inicializar a população do enxame, geralmente espalhando-a pelo espaço de pesquisa.

Passo 2: Calcular o valor de desempenho de cada partícula individual;

Passo 3: Atualizar as posições *gbest* e *pbest*. Utilizar a eq. (?) para calcular a nova velocidade.

Passo 4: Mover a partícula para a nova posição, calculada pela eq. ??

Passo 5: Voltar ao passo 2, enquanto a condição ou critério de convergência esteja satisfeita.

Exemplo: O método PSO será utilizado na procura do ponto $\mathbf{x}^* = [x_1, x_2]^T \in \mathbb{R}^2$ onde a seguinte função: $\mathbf{x}^* = f(x_1, x_2) = \cos(2\pi r)e^{-r}$, com $r = \sqrt{x_1^2 + x_2^2}$, é máxima, para um domínio de pesquisa de $x_1, x_2 \in [-3, 3]$. Na Figura 71 está representado a superfície desta função, sendo observável que o seu valor máximo é alcançado no ponto origem $\mathbf{x}^* = [0, 0]^T$.

Um ensaio com a utilização do método PSO devolveu um resultado o ponto $[0.0068, 0.0004]^T$ (ponto assinalado na Figura 71) com um valor de desempenho de 1.9923, valor este resultado do cálculo da função f nesse ponto mais um. Neste exemplo $\alpha=0.4$ e $c_1=c_2=0.2$, com uma evolução de desempenho do $gBest$, ao longo do processo iterativo, tal como representado na Figura 72, onde a linha a preto dá-nos o desempenho médio das partículas do enxame. (EXEMPLO DADO NA AULA DE ROBOTICA)

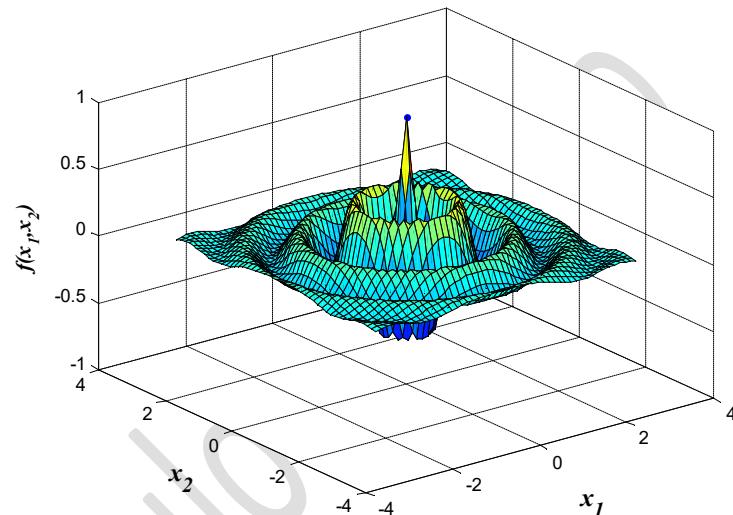


Figura 71: Função que se pretende determinar o ponto de valor máximo.

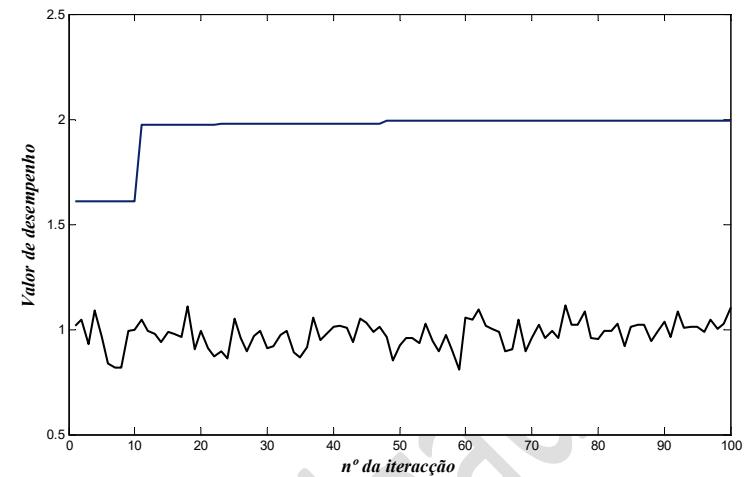


Figura 72: Valores de desempenho da partícula $gBest$ (linha a azul) e desempenho médio do enxame (linha a preto) durante o processo iterativo do método.

11. Optimização por colónias de formigas

Optimização por Colônias de Formigas é uma técnica de pesquisa para problemas combinatórios difíceis, baseado numa população de formigas e inspirado no rasto de feromona que elas depositam no solo quando se deslocam da colmeia para zonas de alimento. Outras formigas perdem a presença de feromona e tendem a seguir os caminhos para os quais a feromona é mais intensa. Com este mecanismo as formigas são capazes de transportar comida para a sua colônia de modo muito expedido e eficaz.

O processo de construção da solução é deste modo estocástico e orientado por um modelo de feromona. Este é um conjunto de parâmetros associados com um gráfico de componentes (nós ou arestas) cujos valores são modificados no decorrer do tempo pelas formigas.

A feromona deixada no solo pelas formigas é uma forma de comunicação indireta e assíncrona que, pela manipulação do ambiente, transportam informação para outras formigas, as quais reagem a mudanças do seu valor. Uma pequena quantidade de feromona num caminho pode conduzir outras formigas a seguirem o mesmo caminho. Todavia, as formigas não têm que estar no mesmo local ou à mesma hora com outras

formigas para comunicarem entre si. Este mecanismo de comunicação, designado por “stigmergic”*, apresenta as seguintes características:

- As formigas atuam como indivíduos;
- A feromona, como transportador de informação, é usada para criar um campo de dissipação;
- O ambiente é um depósito ou montra da informação.

Deste modo “stigmergy” é uma forma de comunicação indireta e não simbólica mediada pelo ambiente (os insetos trocam informação modificando o seu ambiente), onde a informação é local, legível apenas para os insetos que visitam o local ou vizinhança. O modelo de feromonas serve também para construir o espaço probabilístico de busca. O modelo de feromonas pode ser utilizado como um modelo para os problemas de Otimização Combinacional, definidos como se segue:

Um modelo $P = (S, \Omega, f)$ do problema de otimização combinatória (OC) consistem em:

- Procurar uma solução no espaço S definido por um conjunto finito de variáveis de decisões discretas e um conjunto Ω de restrições sobre essas variáveis.
- Uma função objetiva $f : S \rightarrow \mathbb{R}^+$ a ser minimizada.

O espaço de pesquisa S é definido como se segue: Dado um conjunto de n variáveis discretas X_i , com valores $X_i \in D_i = \{v_i^1, \dots, v_i^m\}$, $i = 1, \dots, n$, com m o valor de cardinalidade do conjunto D_i , i.e. $m = |D_i|$. A instanciação da variável, que é, a atribuição do valor v_i^j à variável X_i , é expresso por $X_i = v_i^j$. Uma solução $s \in S$ é uma lista completa de atribuições, resultado da tarefa em que cada variável de decisão tem um valor de domínio atribuído, que satisfaz as restrições. Se o conjunto Ω de restrições estiver vazio, então cada uma das variáveis de decisão pode assumir qualquer valor do seu domínio, independentemente dos valores das outras variáveis de decisão. Neste caso, chamamos o problema P é sem restrições, e contraponto aos problemas com restrições.

Uma solução $s^* \in S$ é chamada de solução ótima global se $f(s^*) \leq f(s)$, $\forall s \in S$. O conjunto de soluções ótimas é designado por $S^* \subseteq S$. A resolução do problema OC reside em encontrar uma solução $s^* \in S^*$.

Algoritmo

Um conjunto de agentes (computacionais) formigas, concorrentes e assíncronas entre si, movem-se através do espaço do problema, o que se traduzem em soluções parciais do problema a resolver. Elas movem-se usando uma política decisória estocástica e local, baseados em dois parâmetros: trilho (trails) e atratividade, construindo em cada incremento a solução do problema. Quando uma formiga completa a (sua) solução a formiga avalia a solução e modifica a intensidade do trilho, que antes tinha sido usado nas soluções. Esta informação, deixada pela intensidade da feromona, será utilizada na pesquisa pelas futuras formigas.

Uma formiga é uma agente computacional simples que interactivamente constrói uma solução para a instância a resolver. Soluções parciais do problema são vistos como estados. Em cada iteração i a formiga k move-se do estado i para outro estado j , correspondendo a uma mais completa solução parcial. Antes a formiga determina o conjunto de todas as expensões possíveis $A_k^i(t)$ a partir do seu estado atual, i , e move-se para aquele, j , tomando como fator decisório todas as probabilidade de se mover do estado i . A probabilidade da formiga de se mover do estado i para o j depende da combinação de dois valores:

- A atratividade η_{ij} do movimento, calculado por algum indicador heurístico sobre a deseabilidade do movimento;
- A intensidade do trilho (feromona) τ_{ij} , de i para j , que indica o quanto foi profícuo no passado esse movimento.

A intensidade do trilho é normalmente atualizado quando todas as formigas completam as suas soluções, incrementando o decrementando a sua intensidade se fez parte de um “bom” ou “mau” solução, respectivamente.

* <http://en.wikipedia.org/wiki/Stigmergy>

Várias versões deste método tem sido proposto nos últimos anos [80]. Descreveremos em seguida este método ao problema de otimização de rotas num gráfico. Para o efeito o objetivo será encontrar a melhor trajetória, entre o nó inicial e o nó final. O resultado será expresso por uma sequência de nós ligados entre o nó inicial e final, que produz uma menor distância de percurso.

Em cada interação os valores da feromona, ligada a todos os troços, são atualizados por todas as m formigas que construirão uma solução. A feromona τ_{ij} associada ao troço que liga o nó i ao nó j , é atualizado como se segue:

$$\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij} + \sum_{k=1}^m \Delta \tau_{ij}^k$$

onde ρ é a taxa de evaporação, m é o número de formigas e $\Delta \tau_{ij}^k$ é a quantidade de feromona deixada cair pelo formiga k no troço (i, j) . Três casos distintos podem suceder, que correspondem a diferentes valores de reforço da feromona:

- i) Se a formiga k passou no troço (i, j) sem no entanto ter alcançado o nó destino: $\Delta \tau_{ij}^k = \frac{Q_L}{d_k}$
- ii) Se a formiga k passou no troço (i, j) sem no entanto ter alcançado o nó destino: $\Delta \tau_{ij}^k = \frac{Q_H}{d_k}$
- iii) Outros: $\Delta \tau_{ij}^k = 0$

Em que Q_L e Q_H são contantes, com $Q_L \ll Q_H$ e d_k o comprimento do trajeto realizado pela formiga k .

Na construção da solução, as formigas selecionam o nó seguinte a visitar através de um mecanismo estocástico. Quando a formiga k está na cidade i constrói uma solução parcial de probabilidades s^P de todos os nós destinos possíveis desse ponto. Para o efeito determina a probabilidade de ir para a cidade j de acordo com a seguinte equação:

$$P_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha \cdot \eta_{ij}^\beta}{\sum_{c_{ij} \in N(s^P)} \tau_{ij}^\alpha \cdot \eta_{ij}^\beta} & \text{se } c_{ij} \in N(s^P) \\ 0 & \text{outros} \end{cases}$$

Em que $N(s^P)$ é o conjunto das componentes possíveis para nó destino de i . Os parâmetros α e β controlam a importância relativas da feromona, τ_{ij} , versus a informação heurística η_{ij} , cujo valor é calculado por:

$$\eta_{ij} = \frac{1}{d_{ij}}$$

em que d_{ij} é a distância entre os nós i e j , e o seu valor será nulo caso não exista ligação entre os nós.

Exemplo: dado na aula,

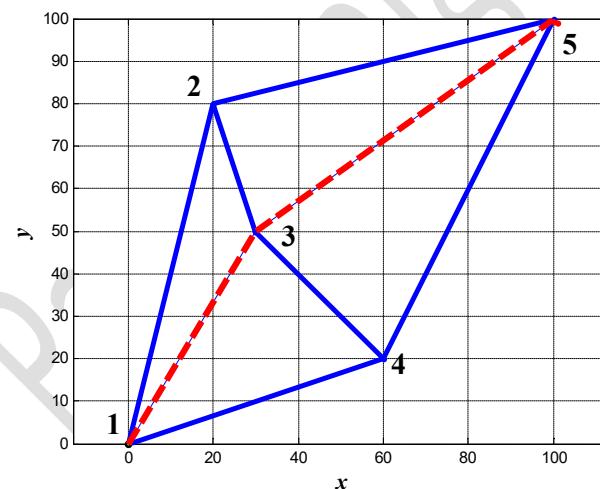


Figura 73: Mapa do problema, com 5 nós, onde o nó 1 é o de partida (colónia) e o nó 5 o de chegada (alimento). A linha a vermelho-tracejado mostra o trajeto ótimo, solução do problema.

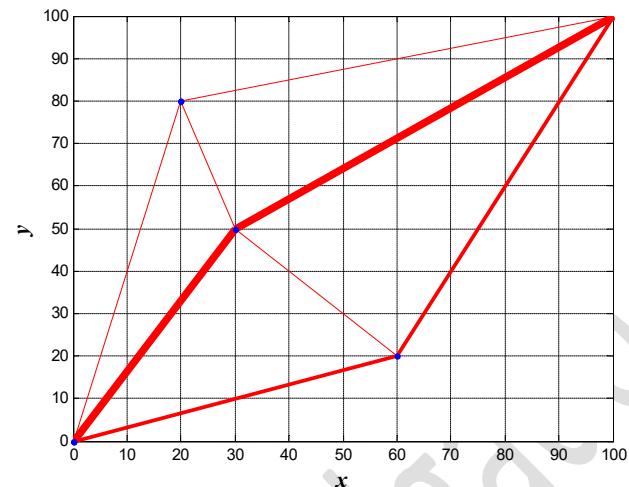


Figura 74 - Mapa de feromona. A intensidade desta hormona é proporcional à espessura das linhas.

12. Sumário

Neste capítulo foram introduzidos os algoritmos genéticos, fortemente inspirados nos mecanismos da seleção natural e da genética. Foi dada ênfase à sua utilização na otimização de parâmetros de funções e de sistemas difusos. Neste contexto foram apresentadas as principais operações genéticas para cromossomas de codificação binária e real. Os exemplos aqui estudados evidenciam a sua robustez e a sua grande potencialidade aplicativa nas mais diversas aplicações de engenharia.

O método PSO introduzido mostra apresentar muitas semelhanças com as técnicas de computação evolutiva tal como os Algoritmos genéticos. O sistema é inicializado com uma população aleatória, procurando por soluções através de um processo geracional. Porém, não possui operadores evolutivos tal como de cruzamento e mutação.

Capítulo V *Apêndice***MÉTODO GRADIENTE DESCENDENTE**

Objetivo: minimizar a função objetivo J sujeita ao conjunto de dado

$D_n = \{(\bar{x}(k), y(k)), k=1, \dots, n\}$, constituído por pares de valores $(\bar{x}(k), y(k))$:

$$J(D_n, \theta) = \sum_{k=1}^n L(y(k), \hat{y}(k))$$

com $\hat{y}(k) = f(\bar{x}(k), \theta)$, onde a função f é parametrizável pelos vector θ :

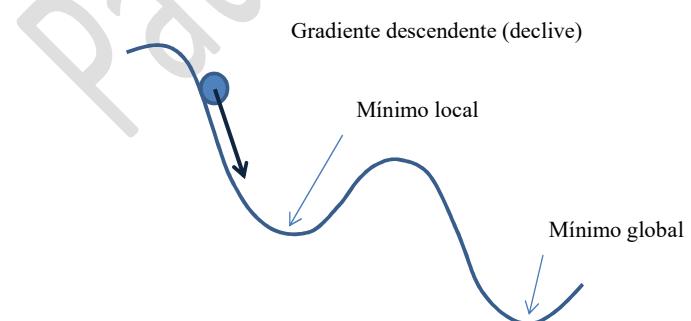
A solução deste problema resume-se a encontrar os melhores parâmetros θ que:

$$\theta^* = \arg \min_{\theta} J(D_n, \theta)$$

O método do Gradiente descendente é um procedimento iterativo onde, em cada iteração s se modifica os valores dos parâmetros θ numa direção contrária ao gradiente de J em relação a θ :

$$\theta^{s+1} = \theta^s - \eta \nabla_{\theta^s} J(D_n, \theta^s)$$

onde n é a taxa de aprendizagem e $\nabla_{\theta^s} J(D_n, \theta^s) = \partial J(D_n, \theta^s) / \partial \theta$. A figura ? mostra como a posição dos parâmetros de move, ajustando a sua posição para níveis de valores de J sucessivamente menores.



REFERÊNCIAS BIBLIOGRÁFICAS

[1] Tou, J. T., Gonzalez, R. C. (1974) *Pattern Recognition*. Reading, MA: Addison-Wesley.

[2] Park, J., Sandberg, J. W. (1991), "Universal approximation using radial basis functions network," *Neural Computation*, vol. 3, pp. 246-257.

[3] Poggio, T., Girosi, F., (1990) "Networks for approximation and learning," *Proc. IEEE*, vol. 78, no. 9, pp. 1481-1497.

[4] Paul V. Yee, Simon Haykin, Regularized Radial Basis Function Networks: Theory and Applications, ISBN: 978-0-471-35349-2, Wiley, April 2001

[5] Jarkko Tikka, Simultaneous input variable and basis function selection for RBF networks Original Research Article *Neurocomputing*, Volume 72, Issues 10–12, June 2009, Pages 2649-2658

[6] Zadeh, L. A., "Fuzzy Sets", *Information and Control* 8, p. 338-353. (1965)

[7] Zadeh, L. A., "The concept of linguistic variable and its application to approximate reasoning," Parte I, *Inf. Sci.*, vol. 8, pp. 199-249, 1975

[8] - E. Cox, "Difusa Fundamentals," *IEEE Spectrum*, pp. 58-61, Oct. 1992

[9] Lin, Chin-Teng and Lee, C.S. George, "Neural Fuzzy Systems, A Neuro-Fuzzy Synergism to Intelligent Systems", Prentice Hall, 1996

[10] Dubois, D. and H. Prade, "Fuzzy Sets and Systems: Theory and Application", New York: Academic Press, 1980

[11] Dubois, D. and H. Prade, "A review of fuzzy set aggregation connectives", *Inf. Sci.* vol. 36, pp. 85-121, 1985

[12] Yager, R. R. "On a general class of fuzzy connectives", *Fuzzy Sets Syst.*, Vol. 4, pp. 235-242, 1980

[13] Klir, George J. and Bo Yuan, "Fuzzy Sets and Fuzzy Logic, Theory and Applications", Prentice Hall PTR, 1995

[14] Yager, R., et al. Ed, "Fuzzy sets and applications: Selected papers by L. A. Zadeh", New York: Jonh Wiley, 1987.

[15] Zadeh, L. A. 1978, "Fuzzy sets as a basis for a theory of possibility." *Fuzzy Sets Syst.* Vol.1 1(1), pp. 3-28

[16] Klir, George J. and Tina A. Folger, "Fuzzy sets, uncertainty and information", Prentice Hall, Englewood Cliffs, N. J., 1988.

[17] Pedrycz, W., "Fuzzy Control and Fuzzy Systems," New York: John Wiley, 1989

[18] Terano, Toshiro, Kiyoji Asai e Michio Sugeno, "Applied Fuzzy Systems", AP Professional, 1994

[19] Fukami , S., M. Mizumoto, and K. Tanak, "Some Considerations of Fuzzy conditional inference," *Fuzzy Sets Syst.*, vol. 4, pp. 243-273, 1980

[20] Mamdani, E. H., "Applications of fuzy algorithms for simple dynamic plant," *Proc. IEE*, vol. 121, pp. 1585-1588, 1974

[21] Larsen, P. M., "Industrial applications of fuzzy logic control," *Int. J. Man. Mach. Studies*, vol. 12, nº1, pp. 3-10, 1980

[22] Zadeh, L. A., "Outline of a new approach to the analysis of complex systems and decision process," *IEEE Trans. Syst. Man and Cybern.*, vol. SMC-3, nº1, pp. 28-44, 1973

[23] Mendel, J. M., "Fuzzy Logic Systems for Engineering: a Tutorial", *Proc. IEEE*, vol. 83 nº 3, pp345-377, 1995

[24] Kosko, B., "Additive Fuzzy Systems: from functions approximation to learning," In *Fuzzy Logic and Neural Network Handbook*, C. H. Chen, pp. 9.1-9.22, McGraw-Hill, Inc., 1996

[25] Kosko, B. " Global Stability of Generalized Additive Fuzzy Systems," *IEEE Trans. on Syst., Man, and Cyber.*, vol. 28, pp. 441-452, August 1998

[26] Yager, R. R., "Essentials of fuzzy Modeling and Control," New York: Wiley, 1994

[27] Mouzouris, G. C. and J. M. Mendel, "Non-singleton fuzzy logic systems," *Proc. 1994 IEEE Conf. On Fuzzy Syst.*, Orlando, FL, June 1994

[28] Mouzouris, G. C. and Mendel, J. M., "Nonsingleton Fuzzy Systemns: Theory and Application," *IEEE Trans. on Fuzzy Systems*, vol. 5, nº 1, Feb. 1997.

[29] Filev, D. P. and R. R. Yager, "A generalized defuzzification method via BAD distributions," *Int. J. Intell. Syst.*, vol. 6, pp. 687-697, 1991

[30] Hellendoorn, H. and C. Thomas, "Defuzzification in fuzzy controllers," *J. Intell. Syst.*, vol. 1, pp. 109-123, 1993

[31] Terano, T., K. Asai, and M. Sugeno, "Fuzzy Systems Theory and Its Applications," New York: Academic, 1992

[32] Wang, Li-Xin, "Adaptive Fuzzy Systems and Control, design and Stability analysis," PTR Prentice-Hall Englewood Cliffs, 1994.

[33] Zadeh, L. A., "A theory of approximate reasoning," in *Machine Intelligence*, vol. 9, J. Hayes, D. Michie, and L. I. Mikulich, Eds. New York: Halstead, 1979, pp. 149-194

[34] Yager, R. R., "Approximate reasoning as basis for rule based expert systems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-14, 1984

[35] Yager, R. R., "Deductive approximate reasoning systems," *IEEE Trans. Knowl. Data Eng.*, vol 3, pp. 399-414, 1991

[36] Dubois D. and H. Prade, "Fuzzy sets in approximate reasoning – Parte I: Inference with possibility distributions," *Fuzzy Sets Syst.*, vol. 40, pp. 143-202, 1991

[37] Wang, L. A. and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least squares learning," *IEEE Trans. Neural Networks*, vol. 3, pp. 807-814, 1992.

[38] Ying, Hao, "Sufficient Conditions on General Fuzzy Systems as Function Approximators," *Automatica*, Vol. 30, Nº 3, pp. 521-525, 1994

[39] Sugeno, M. and G. T. Kang, "Structure identification of fuzzy model," *Fuzzy Sets and Systems*, vol. 28, pp. 15-33, 1988.

[40] Takagi, T. and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 15, pp. 116-132, 1985

[41] Goodwin, G. C. and K. S. Sin, "Adaptive Filtering Prediction and Control," Englewood Cliffs,NJ: Prentice-Hall, 1984

[42] Widrow, Bernard and Michael A. L., "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation," *Proc. IEEE*, vol 78, no 9, pp. 1415-1442, Sept. 1990

[43] William, R. J. and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, pp. 270-280, 1989

[44] Lin, C. T. and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system", *IEEE Trans. Comput.*, vol. 40(12), pp. 1320-1336, 1991

[45] Nomura, H., I Hayashi, and N. Wakami, "A learning method of fuzzy interface rules by descendent method.", *Proc. IEEE Int. Conf. Fuzzy Syst.*, pp. 203-210, San Diego, 1992

[46] Hertz, J., A. Krogh, and R. G. Palmer, "Introduction to the Theory of Neural Computation", Reading, MA: Addison-Wesley, 1991

[47] Chen, S. C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks*, vol. 2, pp. 302-309, 1991

[48] Wang, Li-Xin and Jerry M. Mendel, "Fuzzy Basis Functions, Universal Approximation, and Orthogonal Least-Squares Learning", *IEEE Trans. on Neural Networks*, Vol. 3, n° 5,pp. 807-814 September 1992

[49] Yen, John and Liang Wang, "Simplifying Fuzzy Rule-Based Models Using Orthogonal Transformation Methods," *IEEE Trans. on Syst., Man, And Cybernetics – Parte B: Cybernetics*, vol. 29, nº1, pp.13-24, 1999

[50] Ljung, L., "System Identification: Theory for the User," Englewood Cliffs, NJ: Prentice-Hall, 1987

[51] Jang, J.-S. R., C.-T Sun and E. Mizutani, "Neuro-Fuzzy and Soft Computing. A computational approach to learning and machine intelligence," Upper Saddle River, NJ: Prentice Hall

[52] Goodwin, G. C. and K. S. Sin, *Adaptive Filtering Prediction and Control*, Englewood Cliffs, NJ: Prentice-Hall, 1984

[53] Jang, J.-S. R., "ANFIS: Adaptive-network-based fuzzy inference systems," *IEEE Trans. Syst. Man, and Cybern.*, vol. 23, pp. 665-685, May 1993

[54] Kosko, B., "Neural networks and fuzzy systems: a dynamical systems approach to machine Intelligence", Englewood Cliffs, NJ: Prentice-Hall.

- [55] Kohonen, T. K., "Self-Organization and Associative Memory," 3^a Ed., New York: Springer-Verlag, 1989
- [56] Kohonen, T. K., "The self-Organizing map," Proceedings of the IEEE, vol 78, pp. 1464-1480, 1990
- [57] Kong, S. G., and B. Kosko, "Differential competitive learning for centroid estimation and phoneme recognition," IEEE Trans. Neural Networks, vol. 2, pp. 118-124, 1991
- [58] Dickerson, J. *, and B. Kosko [1993]. "Fuzzy function learning with covariance ellipsoids." Proc. IEEE Int. Conf. Neural Networks, vol. II, pp. 1162-1167, San Francisco, 1993
- [59] Donald F. Specht, "Probabilistic neural networks and general regression neural networks," Chapter 3, edited by C. H. Chen, "Fuzzy Logic and Neural Network Handbook", McGraw-Hill, Inc, 1996.
- [60] Karr, C. L., and E. J. Gentry, "Fuzzy control of PH using genetic algorithms." IEEE Trans. Fuzzy Syst. Vol 1, pp. 46-53, 1993
- [61] Salgado , Paulo , J. Boaventura Cunha, Raul Morais, A. Valente, Carlos Couto, "Controladores difusos da temperatura para uma câmara climática utilizando algoritmos genéticos", pD, 1753-1759, vol. 3, 5^a Jornadas Hispano-Lusas de Ingeniería Eléctrica, Slamanca, 3-5 Julho 1997.
- [62] Goldberg,D. E., "Genetic Algorithms in Search, Optimization, and Machine Learning," reading, MA: Addison-Wesley, 1989.
- [63] Wang , Li Xin and J. M. Mendel [1992]. "Generating fuzzy rules by learning from examples," IEEE Trans. Syste. Man Cybern. Vol 22(6), pp. 1414-1427, 1992
- [64] E. H. Mamdani, "Applications of fuzzy algorithms for simple dynamic plant," Proc. Inst. Elec. Eng., vol. 121, pp. 1585-1588, 1974
- [65] Paulo Salgado, J. Boaventura Cunha, Carlos Couto, "A Computer-Based Environmental Controller for Growth Chambers", Sixth International Conference on Computers in Agriculture, Cancun, Mexico, p. 185-203, June 1998
- [66] Sousa, João M. C., "Fuzzy Model-Based Control", Ph.D. Thesis.
- [67] Hwu, K.I.; Liaw, C.M., "Quantitative speed control for SRM drive using fuzzy adapted inverse model", Aerospace and Electronic Systems, IEEE Transactions on , Volume: 38 Issue: 3 , pp. 995-968, July 2002.
- [68] Abonyi, J.; Babuska, R.; Szeifert, F., "Fuzzy modeling with multivariate membership functions: gray-box identification and control design", Systems, Man and Cybernetics, Part B, IEEE Transactions on , Volume: 31 Issue: 5 , pp. 755 -767, Oct. 2001
- [69] D. Psaltis, A. Sideris, and ^a Yamamura, "A multilayered neural network controller," IEEE Control Syst. Magazine, vol. (, no. 4, pp. 17-21, Apr. 1988.
- [70] J.-S. R. Jang, "Fuzzy controller design without domain experts," in Proc. IEEE Int. Conf. On Fuzzy Syst., March 1992
- [71] Takagi, T., and M. Sugeno, "Derivation of fuzzy control rules from human operator's control actions." Proc. IFAC Symp. Fuzzy Inf. Knowl. Represent. Decis. Anal., pp. 55-60, Marseilles, France, 1983
- [72] Holland, J.H., *Adaptation in Natural and Artificial Systems*. MIT Press, 1975
- [73] D. E. Goldberg, *Genetic Algorithms in search, optimization and machine learning*, Addison-Wesley, 1989.
- [74] Davis, L., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 1991
- [75] Michalewicz, Z., *Genetic Algorithms + Data Structure = Evolution Programs*. Springer-Verlag, 1992
- [76] Holland, J., *Adaptation in natural and artificial systems*. The University of Michigan Press, Ann Arbor, 1975

- [77] Baker, J. E., "Adaptive selection methods for genetic algorithms". Proc. Int. Conf. Genet. Algorithms Their Appl., 101-111, Pittsburgh, 1985.
- [78] Joines, J. and Houck, C., On the use of non stationary penalty functions to solve constrained optimization problems with genetic algorithms. In 1994 IEEE International Symposium Evolutionary Computation, Orlando, Fl, pp. 579-584.
- [79] DeJong, K., *The Analysis and behaviour of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975
- [80] Marco Dorigo, Mauro Birattari, and Thomas Stutzle, Ant Colony Optimization - Artificial Ants as a Computational Intelligence Technique, IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE | NOVEMBER 2006