UNIVERSIDADE ESTADUAL PAULISTA – UNESP
FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

redes neurais

Introdução e Principais Conceitos

Carlos Roberto Minussi

Anna Diva Plasencia Lotufo (Colaboradora)

Ilha Solteira – SP, maio-2008.

1. Estrutura de Redes Neurais

Apresenta-se, a seguir, o modelo de neurônio mais conhecido na literatura especializada, ou seja, o neurônio de McCulloch & Pitts ([5]). Ressalta-se que centenas de tipos de neurônios têm sido identificados. Cada qual distingue-se dos demais pela forma do corpo celular. Estas diferenças morfológicas exibem especializações funcionais importantes. A identificação das funções dos vários tipos de neurônios representa um dos tópicos mais importantes dos estudos referentes à compreensão do cérebro humano. Os resultados destes estudos poderão orientar o desenvolvimento de redes neurais artificiais ainda mais eficientes, principalmente com relação à capacidade e velocidade do aprendizado.

1.1. Modelo Biológico

Os modelos são formados, basicamente, pelas seguintes partes:

(a) **Corpo celular (Soma)** : Parte central do neurônio responsável pela recepção e geração dos impulsos nervosos.

(b) **Sinapse** : Ponto de contato entre a terminação axônica de um neurônio e o dendrito do outro. Funcionam como válvulas, sendo capazes de controlar a transmissão de impulsos (fluxo de informação) entre os neurônios. Esta capacidade é definida como sendo eficiência sináptica.

(c) **Dendrito** : Os dendritos têm a função de receber as informações, ou impulsos nervosos de outros neurônios e conduzi-las ao corpo celular.

(d) **Axônio** : Um axônio pode medir cerca de 0,1 milímetro podendo chegar a 1 metro. Próximo de seu final, o axônio divide-se em vários ramos (dendritos), que se interconectam com os demais neurônios através das sinapses.

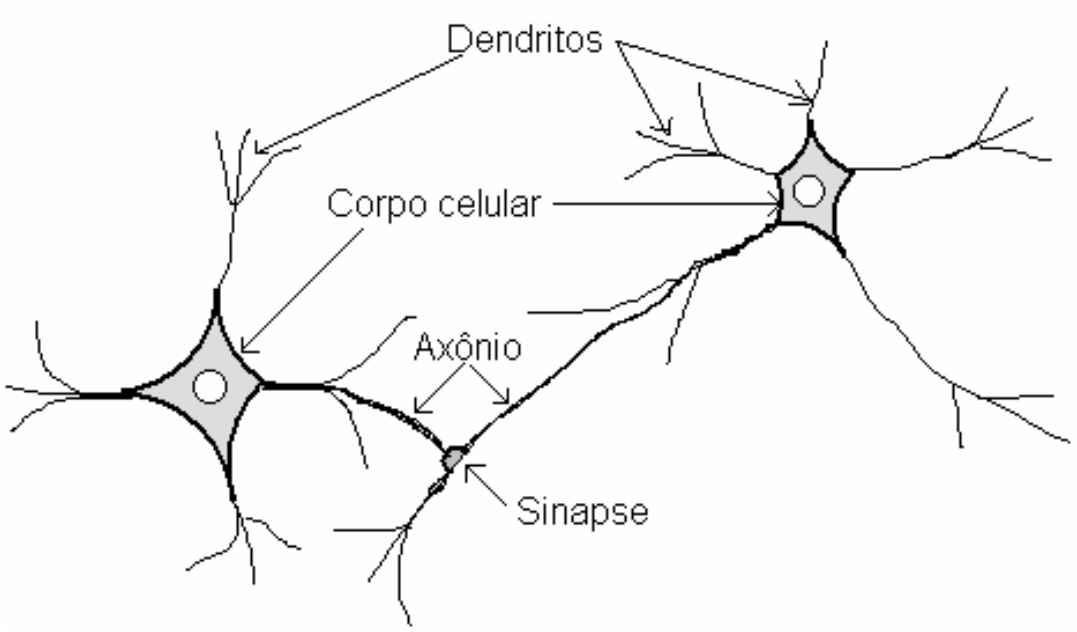


Figura 1. Componentes de um neurônio.

1.2. Neurônio Artificial

Os modelos de neurônios artificiais foram desenvolvidos baseados no funcionamento dos neurônios biológicos. Vários modelos foram propostos na literatura. A seguir apresenta-se o modelo de McCulloch-Pitts, que é o mais empregado, principalmente em problemas de reconhecimento de padrão.

1.2.1. Neurônio de McCulloch-Pitts

O modelo de McCulloch-Pitts ([5]) foi desenvolvido em 1943, o qual descreve um neurônio (Figura 2) cuja atividade *s* é a soma de *n* entradas ponderadas por seus respectivos pesos. Esta atividade alimenta uma função não-linear f(.) que produz um sinal que será enviado aos demais neurônios. As não-linearidades mais empregadas são: relé, lógica *threshold* e sigmóide, conforme mostra-se na Subseção (1.2.2). O neurônio de McCulloch-Pitts pode conter também um peso bias w_0 alimentado por uma constante x0 = +1 que desempenha o controle do nível de saída do neurônio.

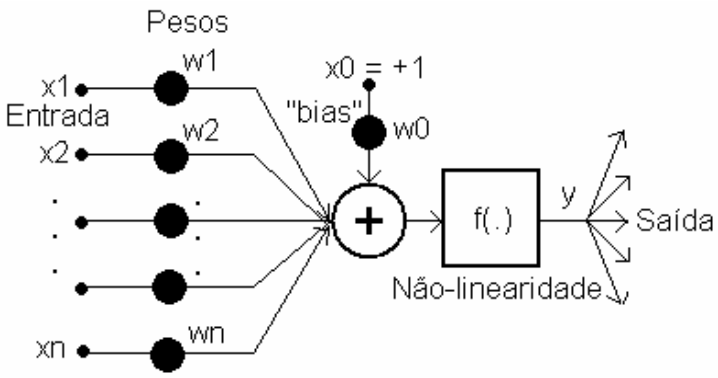


Figura 2. Modelo do neurônio de McCulloch-Pitts.

1.2.2. Não-Linearidades

O sinal *s* do neurônio é usualmente processado por uma função de ativação f(.) que produz o sinal de saída do neurônio. As formas mais utilizadas desta função de ativação são as seguintes:

1. **Relé**

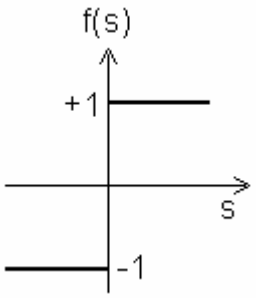


Figura 3. Função relé.

$$f(s) = \begin{cases} +1, \text{ se } s \geq 0 \\ -1, \text{ se } s \leq 0 \end{cases}$$

2. **Lógica *threshold***

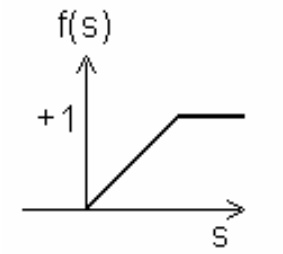


Figura 4. Função lógica *thereshold*.

3. Função Sigmóide (1)

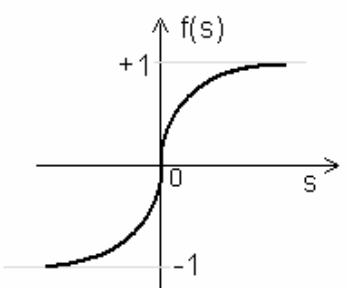


Figura 5. Função sigmóide (1).

$$f(s) = (1 - e^{-\lambda s}) / (1 + e^{-\lambda s})$$

sendo:
λ = inclinação da curva.

4. Função Sigmóide (2)

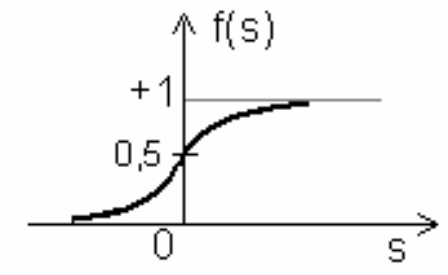


Figura 6. Função sigmóide (2).

$$f(s) = 1 / (1 + e^{-\lambda s})$$

2. Estrutura de Redes Neurais Artificiais

Uma RNA consiste de elementos de processamento (neurônios) e suas conexões (sinapses) (Figura 7). Cada neurônio pode ter várias entradas, porém somente uma saída. Cada saída pode ser utilizada como entrada a vários neurônios (através de ramificações). Assim, como cada neurônio pode receber várias entradas procedentes de outros neurônios. Cada conexão entre neurônios possui um peso que determina sua contribuição na decisão de disparo, controlando, desta forma, os estímulos.

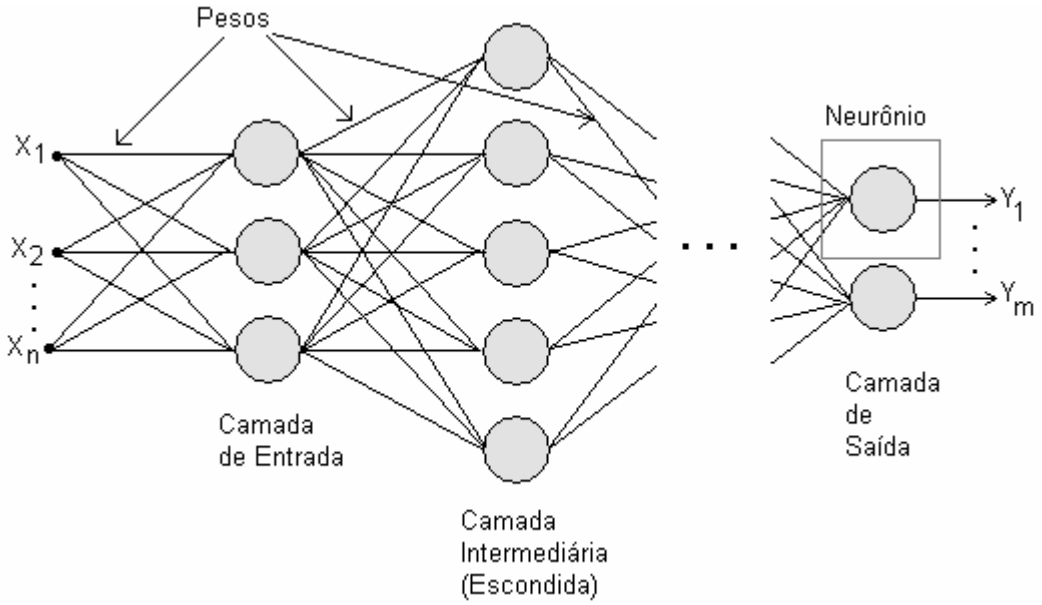


Figura 7. Rede neural artificial.

3. Classificação das Redes Neurais

As redes neurais podem ser classificadas em dois tipos quanto a sua estrutura: redes recorrentes (*feedforward*) e redes não-recorrentes.

Definição 1. *Redes Neurais Recorrentes.* Uma rede é definida como *recorrente* se ela contém laços de realimentação, ou seja, contém conexões das saídas de uma determinada camada para a entrada da mesma ou de camadas anteriores.

As entradas de um neurônio são as saídas dos demais neurônios da camada anterior. As redes que possuem esta estrutura desenvolvem uma memória a longo prazo nos neurônios internos. Nesta classe de redes neurais, *e.g.*, encontra-se a rede de Hopfield ([7]).

Definição 2. *Redes Neurais Não-recorrentes.* Esta rede caracteriza-se estruturalmente por estar disposta em camadas. Nestas redes cada camada de neurônios recebe sinais somente das camadas anteriores, ou seja, elas não possuem laços de realimentação ([3, 7]).

Redes não-recorrentes ou *feedforward* não possuem memória, sendo que, sua saída é exclusivamente determinada em função da entrada e dos valores dos pesos ([3, 7]). A rede neural mostrada na Figura 7 é não-recorrente.

4. Tipos de Treinamento de Redes Neurais

A propriedade mais importante das redes neurais é a habilidade de aprender e com isso melhorar seu desempenho. Isso é feito através de um processo iterativo de ajustes aplicados a seus pesos que correspondem ao treinamento.

Denomina-se algoritmo de treinamento a um conjunto de regras bem definidas para a solução de um problema de treinamento. Existem muitos tipos de algoritmos de treinamento específicos para determinados modelos de redes neurais. Estes algoritmos diferem entre si, principalmente, pelo modo como os pesos são modificados.

Outro fator importante é a maneira pela qual uma rede neural se relaciona com o ambiente. Nesse contexto existem, basicamente, os seguintes paradigmas de treinamento:

Definição 3. *Treinamento Supervisionado.* Consiste no ajuste de pesos de uma rede neural para fornecer saídas desejadas, considerando-se o conjunto de padrões de entrada ([9]).

O treinamento supervisionado necessita de um de vetor de entrada e vetor alvo representando a saída desejada. Juntos eles são chamados de par treinado. Um dos algoritmos mais difundidos para treinamento deste tipo de rede é o algoritmo retropropagação (*backpropagation* (BP) no idioma inglês). Este algoritmo foi proposto por Werbos ([8]) em 1974.

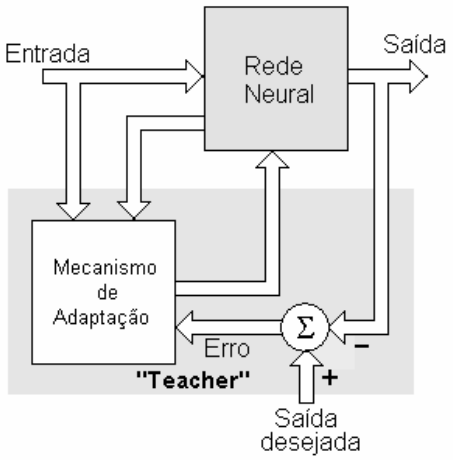


Figura 8. Treinamento supervisionado.

Definição 4. *Treinamento Não-supervisionado.* Consiste no ajuste de pesos de uma rede neural, levando-se em conta somente o conjunto de padrões de entrada. É, portanto, um procedimento de treinamento auto-organizável.

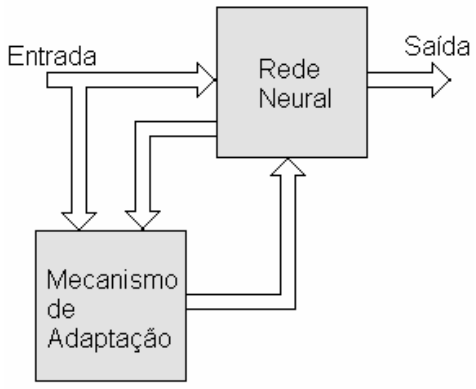


Figura 9. Treinamento não-supervisionado.

5. Modelos de Redes Neurais

Como abordado na Seção 2, uma rede neural é constituída de um arranjo de neurônios funcionando em paralelo e dispostos em camadas. Deste modo, nesta seção, serão abordados os modelos da rede neural e os mecanismos de treinamento.

5.1. O Modelo ADALINE

O modelo de neurônio ADALINE (*ADAptive LInear Element*) ([9]) é mostrado na Figura 10. A saída é uma combinação linear das entradas. Na implementação discreta, estes elementos recebem, no instante k, um vetor padrão de entrada:

$$\mathbf{X}_k = [x_0 = +1 \;\; x_{1k} \;\; x_{2k} \ldots x_{nk}]^T \qquad (5.1.1)$$

e uma resposta desejada d_k. Os componentes do vetor padrão de entrada são ponderados por um conjunto de coeficientes, ou seja, pelo vetor de pesos:

$$\mathbf{W} = [w_0 \;\; w_1 \;\; w_2 \ldots w_n]^T \qquad (5.1.2)$$

A soma das entradas ponderadas é, então, avaliada (calculada), produzindo uma combinação linear correspondente ao produto interno:

$$s_k = < \mathbf{X}_k , \mathbf{W} > \qquad (5.1.3)$$

Os componentes de $\mathbf{X}_k$ podem ser valores reais ou binários. Porém, os pesos são valores essencialmente reais.

Durante o processo de treinamento, os padrões de entrada e de respostas desejadas correspondentes são apresentados à rede neural. Um algoritmo de adaptação ajusta, automaticamente, os pesos de tal forma que as saídas fiquem próximas dos valores desejados.

A rede neural ADALINE consiste no mecanismo de adaptação linear em série com relé (função não-linear), que é empregada para produzir uma saída binária ± 1:

$$Y_k = \mathrm{sgn}(s_k) \qquad (5.1.4)$$

sendo:

Sgn(.) = função sinal.

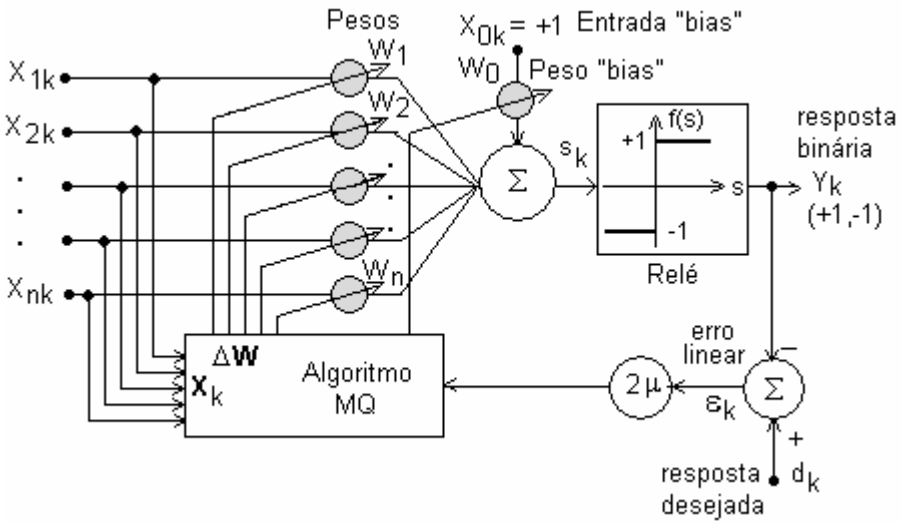


Figura 10. Rede neural ADALINE.

5.2. O Modelo da Rede Neural MADALINE

A rede neural MADALINE (*Multi-ADALINE*) ([9]) é constituída por vários elementos ADALINE. Contudo, o processo de treinamento é bem mais complexo, se comparado ao ADALINE. A rede neural apresentada na Figura 7, associada a algum mecanismo de adaptação de pesos (treinamento), é uma rede MADALINE. Neste sentido, na seqüência, serão abordados os principais conceitos e modelos de treinamento de redes neurais. Em destaque, será apresentado o algoritmo *backpropagation* ([8]), que é, certamente, o mais conhecido e empregado na literatura especializada. Posteriormente, serão enfocados outros tipos de redes neurais que, também, são bastantes empregados: rede de Hopfield ([3, 7]), rede de neural de Kohonen ([7]) e redes da família ART (*Adaptive Resonance Theory*) ([1]), entre outros. Deve-se ressaltar que o treinamento de redes multineurônios / multicamadas é bastante complexo. O treinamento (processo adaptativo) é um procedimento que emprega algum método de otimização para o ajuste de pesos. Sabe-se que os métodos de otimização determinísticos, via de regra, empregam derivadas de funções. Deste modo, o relés, que é uma função não-diferenciável, é inadequado para uso em redes MADALINE. Deve-se, portanto, buscar outras alternativas de funções para uso neste tipo de redes neurais, *e.g.*, as funções sigmoidais que são a base do desenvolvimento de redes neurais com treinamento via algoritmo BP. Este assunto será abordado da seqüência.

6. REALIZAÇÃO DE REDES NEURAIS

A realização de uma RNA (Rede Neural Artificial) consiste na especificação de sua estrutura neural (número de camada, número de neurônios por camada, tipo de não-linearidade, etc.) e na adaptação dos pesos (treinamento). Este procedimento tem como propósito a extração do conhecimento do processo-alvo, ou seja, as aplicações (análise / diagnóstico de problemas complexos). Estes problemas complexos referem-se aos casos, principalmente, aqueles que não se dispõem de modelos matemáticos (sistemas agrícolas, biológicos, análise de sinais, previsões, etc.), ou quando há necessidade de soluções em tempo-real, por exemplo, análise de sistemas de energia elétrica de grande porte. O funcionamento das RNAs é constituído de duas fases: (1) treinamento; (2) teste e análise. A obtenção de modelos das RNA é efetivada por meio de estímulos de entrada / saída (treinamento supervisionado) ou por estímulos somente de entrada

(treinamento não-supervisonado, ou auto-organizável). Este processo se dá via adaptação de pesos executado por alguma técnica dedicada. O treinamento, via de regra, é uma tarefa realizada de modo *off-line*. Consome a totalidade de tempo de realização, enquanto que, uma vez finalizado o treinamento, a fase de análise é realizada sem custo computacional, daí a importância das RNA nas aplicações em tempo-real.

Assim sendo, na seqüência, serão abordados os primeiros estudos sobre a capacidade das RNA. Começa-se considerando-se um único neurônio, buscando estabelecer o que se pode fazer numa unidade neural mais simples. Posteriormente, progressivamente, busca-se verificar a capacidade de RNAs mais complexas.

6.1. Capacidade de um Único Neurônio

Considera-se o modelo neural de McCulloch-Pitts com não-linearidade relé e duas entradas [x1, x2]. Este modelo é mostrado na Figura 11.

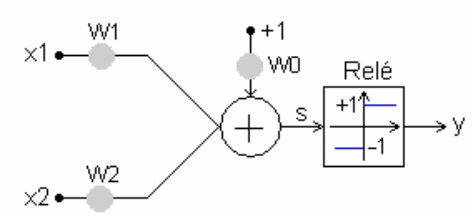


Figura 11. Modelo do neurônio McCulloch-Pitts para duas entradas.

A saída intermediária (s) é dada por:

$$s = W1\,x1 + W2\,x2 + W0 \qquad (6.1.1)$$

A saída (y) constitui-se de valores +1 (para valores $s > 0$) ou –1 (para valores $s < 0$). Deste modo, $s = 0$ representa o lugar geométrico de comutação. Assim, a partir da equação (6.1.1), define-se a seguinte equação (hiperplano de comutação, que no caso de 2 entradas, torna-se uma reta de comutação):

$$x2 = -\frac{W1}{W2}\,x1 - \frac{W0}{W2} \qquad (6.1.2)$$

para W1e W2 ≠ 0.

sendo:

$-\frac{W1}{W2}$ = coeficiente angular da reta;

$-\frac{W0}{W2}$ = deslocamento da reta no eixo de x2.

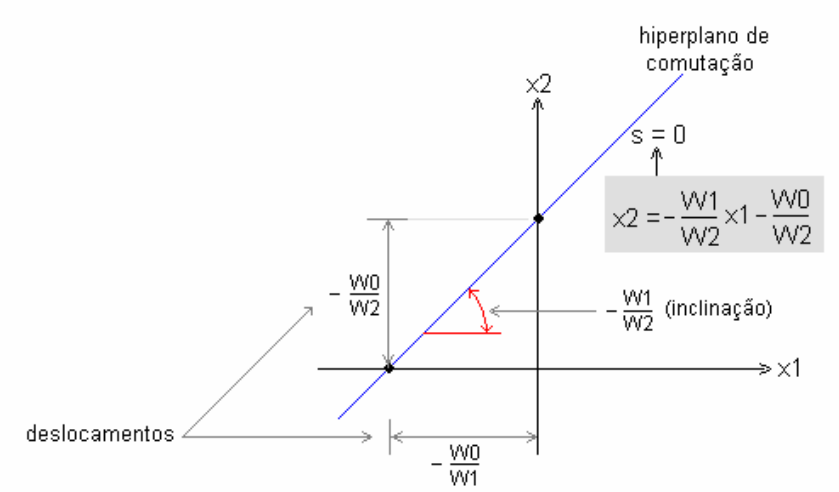


Figura 12. Representação geométrica da reta de comutação.

Supondo-se que se deseja realizar uma função lógica AND, que é uma das mais simples, conforme mostrada na Tabela 1.

Tabela 1. Especificação da função lógica AND.

Entrada			Saída
Ponto	x1	x2	y (AND)
1	+1	+1	+1
2	+1	-1	-1
3	-1	-1	-1
4	-1	+1	-1

Esta representação [+1, –1] será usada nesta disciplina por razões que a representação binária [0, +1], via de regra, é menos eficiente por, certamente, demandar um

número maior de adaptações para a conclusão do treinamento, conforme será mostrado adiante. Assim, a função lógica AND será ilustrada no plano de fase (Figura 13).

A função lógica AND é definida como:

AND = mín { x1, x2} (6.1.3)

sendo:

mín $\underline{\Delta}$ operador mínimo.

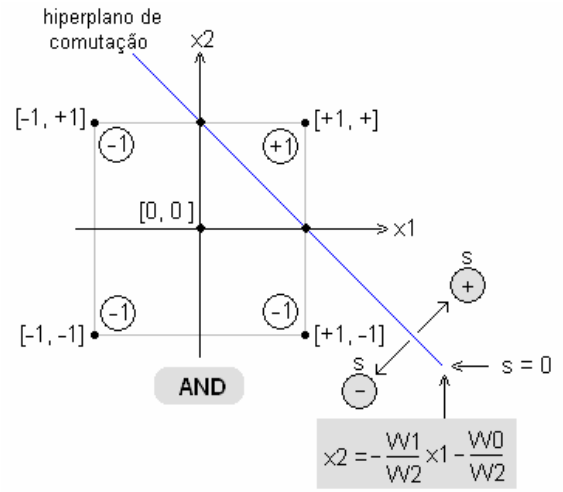


Figura 13. Representação geométrica da função lógica AND.

Deve-se observar que, se W0 for nulo, a reta de comutação passa pela origem [0, 0]. Por conseguinte, neste caso, não há possibilidade de realizar uma das funções lógicas mais simples (função lógica AND). Ou seja, com inclinação (definida pelos parâmetros W1 e W2) igual a 135^o, a reta passa em cima dos pontos 2 e 4 (produzem sinais indefinidos na saída). Se a inclinação for maior do que 135^o, os pontos P1 e P4 produzirão valores positivos na saída, enquanto que para os pontos P2 e P3, a saída será negativa. Por sua vez, se a inclinação for inferior a 135^o, as saídas de P1 e P2 serão positivas, e para P3 e P4 são negativas. Assim, conclui-se que com um único neurônio, para realizar uma das tarefas mais simples, faz-se necessário que o peso *bias* (W0) seja diferente de zero. Esta imposição pode ser relaxada quando se emprega arquiteturas neurais mais elaboradas (multicamadas / multineurônios).

Conclusão 1. O peso *bias* (W0) deve ser diferente de zero, quando se emprega uma RNA mais simples (composta por um único neurônio), ou nos casos com um número bastante reduzido de neurônios.

O que se pode fazer com um único neurônio?

Pode-se realizar tarefas simples, por exemplo, o reconhecimento de uma impressão digital, ou uma imagem de uma pessoa. A resposta será sim (o reconhecimento) ou não. Porém, se resposta for sim, ela não saberá discriminar de quem é a impressão digital, ou a imagem. Por conseguinte, se a tarefa for mais complexas, redes neurais mais elaboradas devem ser empregadas, conforme será abordado mais adiante.

6.2. Realização da Função Lógica AND

A função lógica AND é definida pela equação (6.1.3) e ilustrada conforme é mostrado na Tabela 1.

Realização Mínima: 1 Camada [C1(1)]

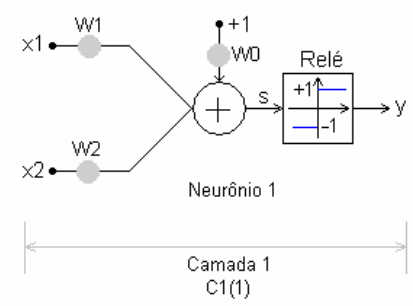


Figura 14. Estrutura neural mínima para a realização da função lógica AND.

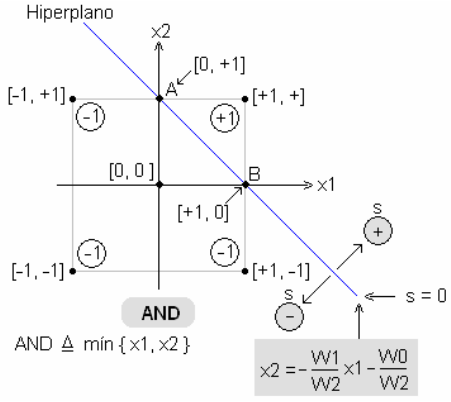


Figura 15. Uso do conceito “linearmente separáveis” para a função lógica AND.

Equação do Hiperplano

a) Ponto [x1, x2] = [0, 0]

s = W1 x1 + W2 x2 + W0

s = 0 + 0 + W0 ← sinal se "s" deve ser igual ao sinal de W0.

Como em [0, 0], s < 0, então, deve-se arbitrar W0 < 0. Assim, arbitra-se

W0 = -1

b) No ponto A = [0, +1]

$$+1 = -\frac{W1}{W2}(0) - \frac{W0}{W2}$$

$$W2 = -W0$$

W2 = 1.

c) No ponto B = [+1, 0]

$$0 = -\frac{W1}{W2}(1) - \frac{W0}{W2}$$

$$\frac{W1}{W2} = -\frac{W0}{W2}$$

$$W1 = -W0$$

W1 = 1.

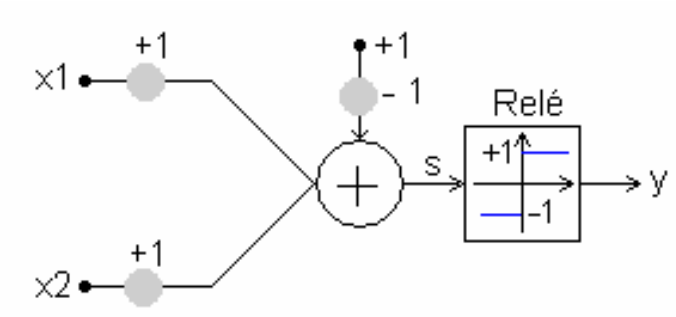


Figura 16. Uma realização neural da função lógica AND.

Tabela 2. Conferência sobre a realização da função lógica AND.

Entrada			Saída Intermediária	Saída y	
padrão	x1	x2	s	Calculada	Desejada
1	+1	+	+1	+1	+1
2	+1	-1	-1	-1	-1
3	-1	+	-1	-1	-1
4	-1	-1	-3	-1	-1
				OK!	

6.3. Realização da Função Lógica OR

A função lógica OR é definida por:

OR = máx {x1, x2 } (6.3.1)
sendo:
máx $\underline{\Delta}$ operador máximo.

Tabela 3. Especificação da função lógica OR.

Entrada			Saída
Ponto	x1	x2	y
1	+1	+1	+1
2	+1	-1	+1
3	-1	-1	-1
4	-1	+1	+1

Realização Mínima: 1 Camada [C1(1)]

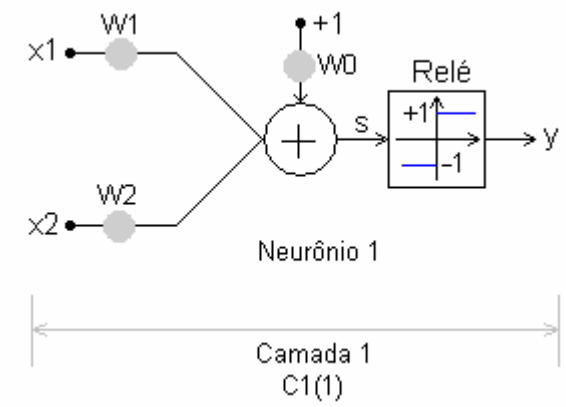


Figura 17. Estrutura neural mínima para a realização da função lógica OR.

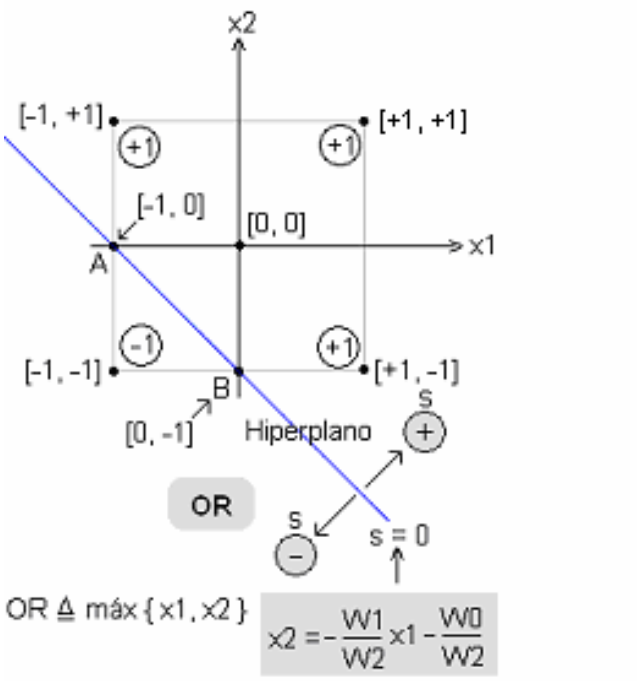


Figura 18. Uso do conceito “linearmente separáveis” para a função lógica OR.

Equação do Hiperplano

a) Ponto [x1, x2] = [0, 0]

s = W1 x1 + W2 x2 + W0

s = 0 + 0 + W0 ← sinal se “s” deve ser igual ao sinal de W0.

Como em [0, 0], s > 0, então, deve-se arbitrar W0 > 0. Assim, arbitra-se

W0 = +1

b) No ponto A = [-1, 0]

$$+0 = -\frac{W1}{W2}(-1) - \frac{W0}{W2}$$

W1 = W0

W1 = 1.

c) No ponto B = [0, -1]

$$-1 = -\frac{W1}{W2}(0) - \frac{W0}{W2}$$

W2 = W0

W2 = 1.

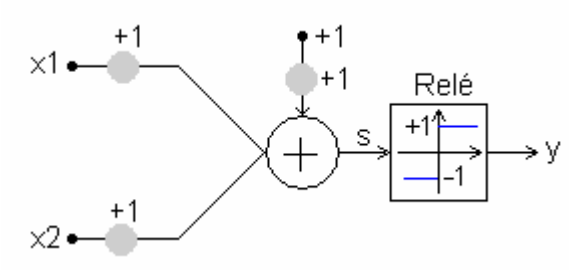


Figura 19. Uma realização neural da função lógica OR.

Tabela 4. Conferência sobre a realização da função lógica OR.

Entrada			Saída Intermediária	Saída y	
padrão	x1	x2	s	Calculada	Desejada
1	+1	+1	+3	+1	+1
2	+1	-1	+1	+1	+1
3	-1	-1	+1	+1	+1
4	-1	+1	-1	-1	-1
				OK!	

6.4. Realização da Função NOR Exclusivo

A função lógica NOR Exclusivo é definida como ilustrado na Tabela 5. Neste caso, observa-se que, usando o modelo de McCulloch-Pitts com não-linearidade relé (modelo adaptativo linear / separabilidade linear), não será possível realizar a referida função lógica com um único neurônio. Portanto, uma arquitetura mínima será:

Realização Mínima: Duas camadas [C1 (2) e C2(1)].

Ou seja, a RNA será composta por duas camadas, sendo que na primeira terá 2 neurônios e, na segunda, terá 1 neuônio.

Conclusão 2. Para a realização de funções mais complexas, há necessidade de arquiteturas neurais multicamadas. ´

NB. Os critérios para especificação de arquiteturas de RNAs serão abordados adiante, por ocasião do estudo de técnicas sistemáticas de treinamento (*e.g.*, o algoritmo retropropagação (*backpropagation*)).

Tabela 5. Especificação da função NOR Exclusiva.

Entrada			Saída
Ponto	x1	x2	y
1	+1	+1	+1
2	+1	-1	-1
3	-1	-1	+1
4	-1	+1	-1

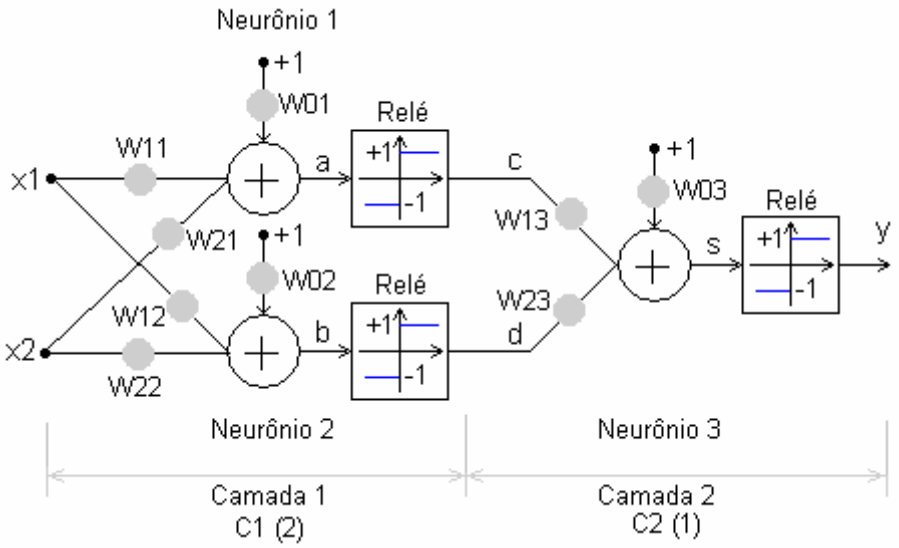


Figura 20. Arquitetura neural para tornar possível a realização mínima da função lógica NOR Exclusivo.

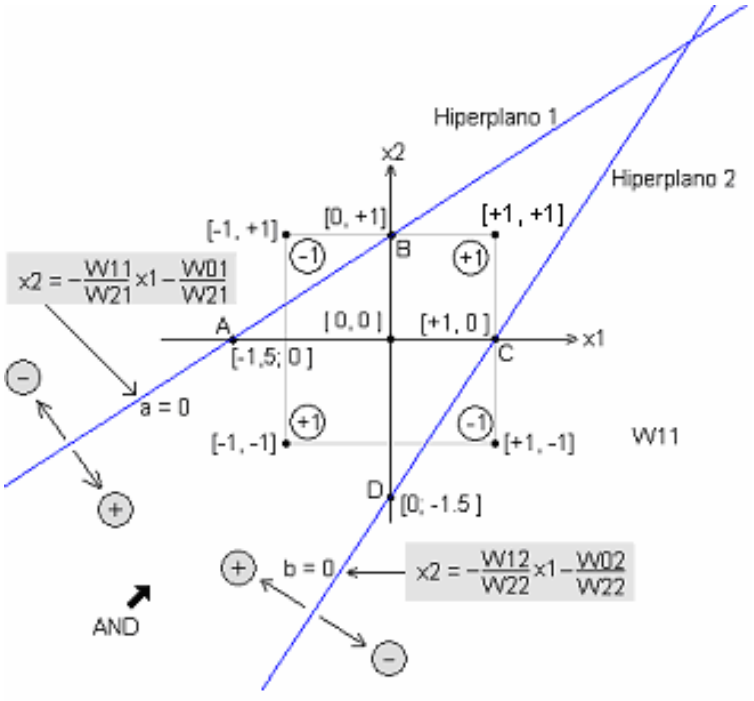


Figura 21. Uso do conceito “linearmente separáveis” para a função lógica NOR Exclusivo.

Hiperplano 1

a) Ponto [x1, x2] = [0, 0]

a = W11 x1 + W21 x2 + W01

a = 0 + 0 + W01 ← sinal de “a “ deve ser igual ao sinal de W01.

Como em [0, 0], a > 0, então, deve-se arbitrar W01 > 0. Assim, arbitra-se

W01 = +1,5

b) No ponto A = [-1,5; 0]

$$+0 = -\frac{W11}{W21}(-1{,}5) - \frac{W01}{W21}$$

1,5 W11 = W01

W11 = +1

c) No ponto B = [0, +1]

$$+1 = -\frac{W11}{W21}(0) - \frac{W01}{W21}$$

$$W21 = -W01$$

$$\boxed{W21 = -1,5.}$$

Hiperplano 2

a) Ponto [x1, x2] = [0, 0]

$$b = W12\,x1 + W22\,x2 + W02$$

$$b = 0 + 0 + W02 \leftarrow$$ sinal se "b" deve ser igual ao sinal de W02.

Como em [0, 0], b > 0, então, deve-se arbitrar W02 > 0. Assim, arbitra-se

$$\boxed{W02 = +1,5}$$

b) No ponto C = [+1; 0]

$$+0 = -\frac{W12}{W22}(1) - \frac{W02}{W22}$$

$$W12 = -W02$$

$$\boxed{W12 = -1,5}$$

c) No ponto D = [0, -1,5]

$$-1,5 = -\frac{W12}{W22}(0) - \frac{W02}{W22}$$

$$1,5\,W22 = W02$$

$$\boxed{W22 = 1}$$

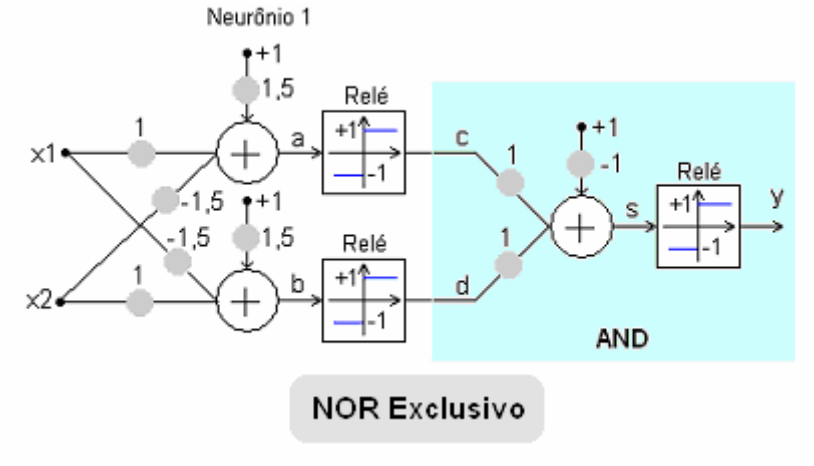


Figura 22. Resultado da realização da função lógica NOR Exclusivo.

Tabela 6. Conferência sobre a realização da função lógica NOR Exclusivo.

Entrada			Saídas Intermediárias					Saída y	
Ponto	x1	X2	a	b	c	d	s	Calculada	Desejada
1	+1	+1	+1	+1	+1	+1	+1	+1	+1
2	+1	-1	+4	-1	+1	-1	-1	-1	-1
3	-1	-1	+2	+2	+1	+1	+1	+1	+1
4	-1	+1	-1	+4	-1	+1	-1	-1	-1
								OK!	

Estes exemplos foram abordados com o propósito de buscar os primeiros entendimentos sobre a adaptação de pesos (treinamento). Certamente, não é esta a forma habitual de se adaptar os pesos. Seria impraticável usá-la em problemas reais. Assim, na seqüência serão apresentadas as técnicas sistemáticas de treinamento. Cada técnica possui particularidades que buscam atender às aplicações específicas. Por exemplo, o algoritmo retropropagação (baseado no gradiente descendente) destina-se à execução do treinamento supervisionado de RNAs *feedforward*, enquanto que nas redes neurais de Kohonen e ART (*Adaptive Resonance Theory*), que são RNA não-supervisionadas, são usados conceitos de verossimilhança.

Ressalta-se que este procedimento do uso do conceito de "separabilidade linear" é robusto, *i.e.*, o diagnóstico pode ser realizado, com sucesso, considerando-se padrões contendo ruído. Por exemplo, supondo-se um ponto P = [0,9; 1,15], para o caso da função lógica AND, a

resposta y será +1. Assim, outros pontos ruidosos também podem ser analisados (vide Figura 23.).

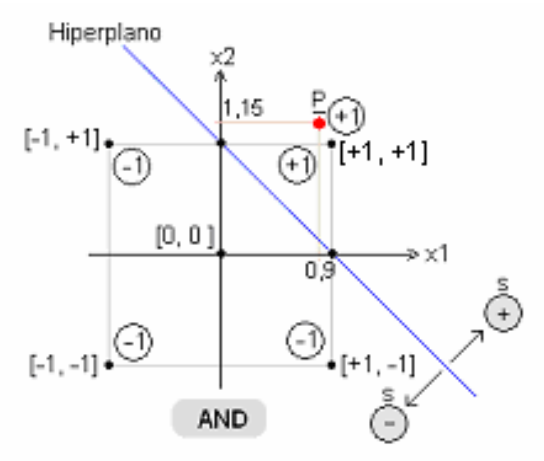


Figura 23. Ilustração da robustez da RNA.

Outra observação refere-se à solução final da adaptação dos pesos. Existem um grande número de conjunto de pesos que podem realizar as funções requeridas com êxito. Por exemplo, na Figura 24, mostram-se algumas soluções possíveis para a função lógica AND, cujos hiperplanos são identificados pelas cores azul, vermelho, laranja, verde, entre outras. Todas estas separações lineares proporcionam o mesmo resultado (produção das saídas desejadas correspondentes).

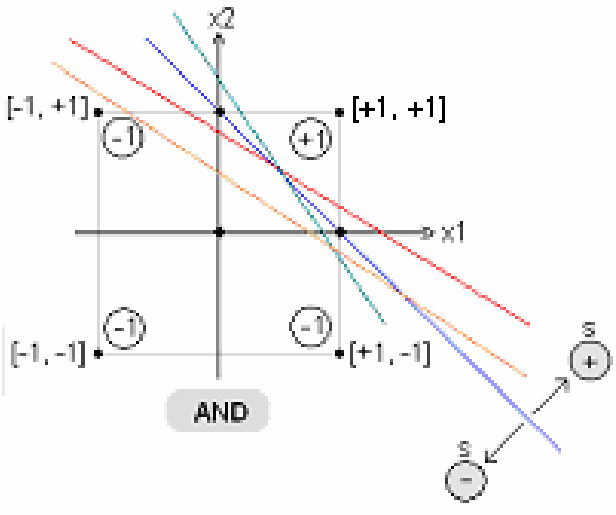


Figura 24. Possíveis soluções para a realização, *e.g.*, da função lógica AND.

6.4. Treinamento Sistemático

O treinamento sistemático consiste no emprego de alguma técnica dedicada, em forma de algoritmo, de adaptação automática de pesos da RNA. Uma das forma mais simples refere-se ao algorimo de Hidrow-Hoff, como mostrada na Seção 6.4.1.

6.4.1. Algoritmo Widrow-Hoff

Considera-se um neurônio de McCulloch-Pitts com não-linaridade tipo relé, conforme mostrada na Figura 25.

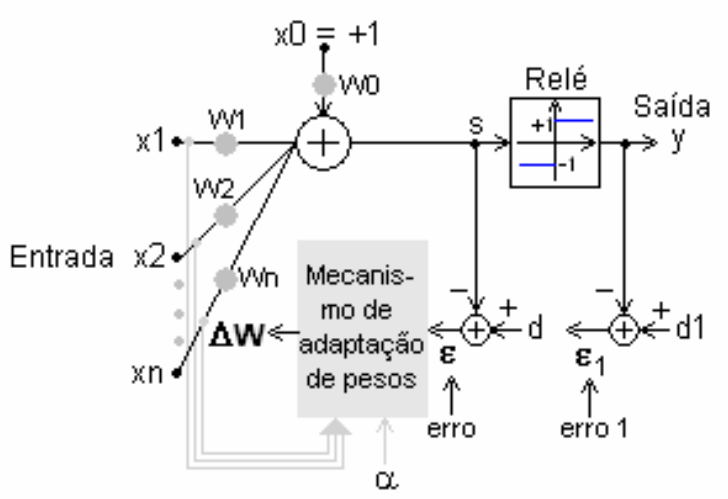


Figura 25. Modelo do neurônio McCulloch-Pitts para n-entradas e mecanismo de adaptação de pesos.

O ajuste de pesos deste neurônio pode ser realizado, através do algoritmo α-MQ (α-Mínimos Quadrados) ou algoritmo Widrow-Hoff:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \alpha\, \varepsilon_k \frac{\mathbf{X}}{|\mathbf{X}|^2} \qquad (6.4.1.1)$$

sendo:

$\mathbf{W} = [\, W1 \;\; W2 \;\; \ldots \;\; Wn \;\; W0 \,]^T$;

$\mathbf{X}$ = [x1 x2 . . . xn x0] ^{T} (vetor padrão de entrada);

k = índice de iteração (adaptação dos pesos);

α = taxa de treinamento;

ε $\underline{\Delta}$ d – s

= erro;

d = saída intermediária desejada;

s $\underline{\Delta}$ saída intermediária

= $\mathbf{W}^T \mathbf{X}$.

No processo de adaptação o erro é considerado a diferença entre a saída intermediária desejada e a saída intermediária, tendo em vista que o erro medido na saída (y) cria uma inconsistência (a função relé não é diferenciável).

O erro ε_k pode ser expresso por:

$$\varepsilon_k \ \underline{\Delta} \ d_k - \mathbf{W}_k^T \mathbf{X} \qquad (6.4.1.2)$$

A mudança dos pesos produzirá uma mudança no erro da seguinte forma:

$$\Delta\varepsilon_k = \Delta(d_k - \mathbf{W}_k^T \mathbf{X})$$

$$= \Delta d_k - \Delta(\mathbf{W}_k^T \mathbf{X}) \quad (d = \text{valor constante}) \quad (\Delta d_k \rightarrow 0)$$

$$= -\mathbf{X}^T \Delta\mathbf{W}_k. \qquad (6.4.1.3)$$

De acordo com o algoritmo α-MQ, os pesos variam do seguinte modo:

$$\Delta\mathbf{W}_k \ \underline{\Delta} \ \mathbf{W}_{k+1} - \mathbf{W}_k = \alpha \, \varepsilon_k \, \frac{\mathbf{X}}{|\mathbf{X}|^2} \qquad (6.4.1.4)$$

Combinando-se a equação (6.4.1.4) com a equação (6.4.1.3), obtém-se:

$$\Delta\varepsilon_k = -\alpha \, \varepsilon_k \, \mathbf{X} \, \frac{\mathbf{X}}{|\mathbf{X}|^2}$$

$$= -\alpha \, \varepsilon_k \qquad (6.4.1.5)$$

A equação (6.4.1.5) pode ser reescrita como:

$\varepsilon_{k+1} - \varepsilon_k = -\alpha\, \varepsilon_k$, ou:

$$\varepsilon_{k+1} = (1 - \alpha)\ \varepsilon_k \qquad (6.4.1.6)$$

Esta equação representa um sistema dinâmico discreto que possui um comportamento assintoticamente estável ($\varepsilon_{k+1} \to 0$, quando $(k+1) \to \infty$), se, e somente se:

$$|\, 1 - \alpha \,| < 1 \qquad (6.4.1.7)$$

Neste caso, o parâmetro α deve ser escolhido dentro do seguinte limite:

$$0 < \alpha < 2 \qquad (6.4.1.8)$$

de modo a garantir a estabilidade do processo iterativo. Nota-se que o parâmetro α independe do conjunto de padrões do treinamento.

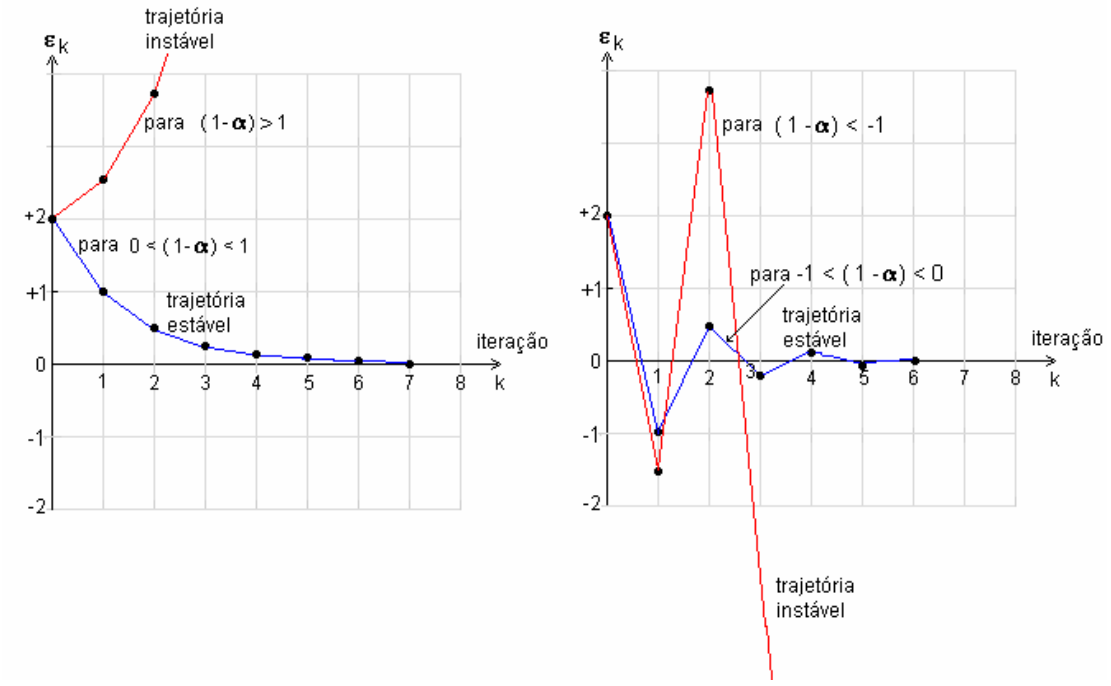


Figura 26. Ilustração do processo de convergência do algoritmo α-MQ.

6.4.1.1. Algoritmo

O algoritmo McCulloch-Pitts pode ser descrito da seguinte forma. A partir da arbitragem (aleatória, ou pseudo-aleatória) dos pesos iniciais da rede neural, deve-se definir uma seqüência aleatória, ou pseudo-aleatória, indicando a ordem em que os padrões de treinamento devem ser apresentados à RNA, para fins de treinamento. A ordenação seqüência natural pode ser também empregada, contudo, a ordenação aleatória é mais plausível do ponto de vista mental, tendo em vista que é esta forma em que os estímulos ocorrem no mundo real.

Posteriormente, para cada par de padrões (**X**, y), calcula-se o erro produzido na saída da RNA (erro $\underline{\Delta}$ d – y). Se o erro for superior a um determinado patamar preespecificado, procede-se a adaptação dos pesos usando a equação (6.4.1.1). Este procedimento dever ser repetido até que, para todo par de padrão (**X**, y), o erro ocasionado na saída estiver abaixo do referido patamar preespecificado. Neste caso, o treinamento estará concluído, ou seja, a RNA estará apta a realizar a generalização.

Exemplo. Supondo-se que se deseja realizar a função AND com duas entradas (Vide Figura 27), via o algoritmo α-MQ.

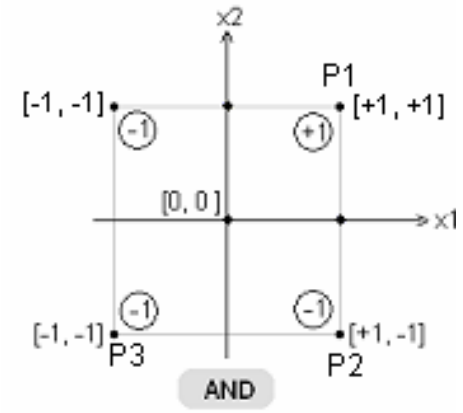


Figura 27. Ilustração da função lógica AND.

Resolução.

Passo 1. (Geração dos pesos iniciais)

Supondo-se que os pesos inciais são assim definidos:

$\mathbf{W} = [1\ 1\ 1\ ]^T$ (por conveniência, deixa-se de lado a geração randômica/pseudo-rendômica dos pesos inciais).

Passo 2. (estabelecer a ordem aleatória de apresentação dos vetores padrões).

S = [0,1329 0,4562 0,3095 0,7875] ^{T} (geração de uma seqüência de 4 números pseudo-randômicos, a partir de uma semente dada).

I = [1 2 3 4]^{T} (vetor de índices de ordenação).

S = [0,7875 0,4562 0,3095 0,1329] ^{T}

I = [4 2 3 1]^{T} (vetor de índices de ordenação seguindo o critério de ordenação decrescente de **S**).

Assim, por este critério a ordem de apresentação dos vetores padrões será {4, 2, 3, 1}.

Adaptação para o vetor padrão 4

P4 = [–1, +1]

Tomando-se o padrão de entrada 4 (P4), o hiperplano definido pelos pesos W1 =1, W2 = 1 e W0 =1 é mostrado na Figura 28.

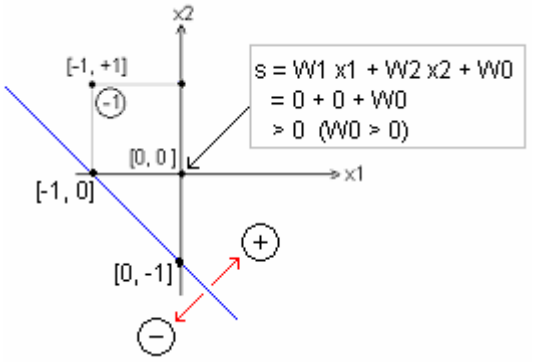


Figura 28.

Aplicando-se a equação de adaptação (6.4.1.1), tem-se:

$$\begin{bmatrix} W1 \\ W2 \\ W0 \end{bmatrix}^{(k+1)} = \begin{bmatrix} W1 \\ W2 \\ W0 \end{bmatrix}^{(k)} + \alpha\,\varepsilon(k)\,\frac{1}{|\mathbf{X}|^2}\begin{bmatrix} x1 \\ x2 \\ x0 \end{bmatrix}$$

Adotando-se α = 1, 5 e d(4) = –0,5 (saída intermediária desejada) que produz s = 1 (-1) + 1 (1) + 1 = 1que produzirá uma saída y(4) = + 1. Assim, tem-se:

$$\begin{bmatrix} W1 \\ W2 \\ W0 \end{bmatrix}^{(1)} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}^{(0)} + (1.5)(-1{,}5)\,\frac{1}{3}\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} W1 \\ W2 \\ W0 \end{bmatrix}^{(1)} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}^{(0)} - 0{,}75\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$$

$\varepsilon(0) = d(4) - y(4)$

$= -0{,}5 - 1$

$= -1{,}5$

$$\begin{bmatrix} W1 \\ W2 \\ W0 \end{bmatrix}^{(1)} = \begin{bmatrix} 1{,}75 \\ 0{,}25 \\ 0{,}25 \end{bmatrix}$$

Inclinação $= -\frac{1{,}75}{0{,}25}$

$= -7$

Deslocamento $= -\frac{0{,}25}{1{,}75}$ (no eixo x1)

$= -0{,}143$

Deslocamento $= -\frac{0{,}25}{0{,}25}$ (no eixo x2)

$= -1.$

Considerando-se estes pesos, a saída y(4) vale:

$s(4) = 1{,}75\,(-1) + 0{,}25\,(1) + 0{,}25$

$= -1{,}25$

$y(4) = -1$ (OK!).

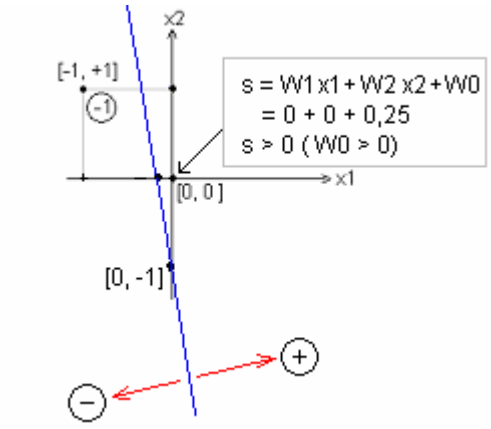


Figura 29. Resultado da adaptação de pesos referente ao padrão de entrada P4.

Adaptação para o vetor padrão 2

P2 = [+1, –1]

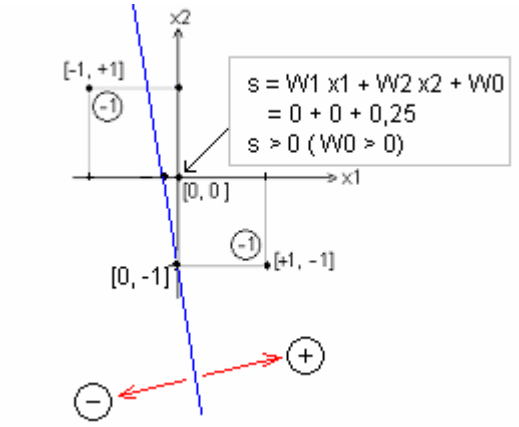


Figura 30. Situação da adaptação de pesos considerando-se os pontos P4 e P2.

Arbitrando-se a saída intermediária desejada d(2) = –0,5 (y(2) = –1), os pesos são assim adaptados:

$$\begin{bmatrix} W1 \\ W2 \\ W0 \end{bmatrix}^{(1)} = \begin{bmatrix} 1{,}75 \\ 0{,}25 \\ 0{,}25 \end{bmatrix}^{(0)} + (1{,}5)(-1{,}5)\,\frac{1}{3}\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} W1 \\ W2 \\ W0 \end{bmatrix}^{(1)} = \begin{bmatrix} 1{,}75 \\ 0{,}25 \\ 0{,}25 \end{bmatrix}^{(0)} + -0{,}75\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} W1 \\ W2 \\ W0 \end{bmatrix}^{(1)} = \begin{bmatrix} 1{,}75 \\ 0{,}25 \\ 0{,}25 \end{bmatrix}^{(0)} + \begin{bmatrix} -0{,}75 \\ 0{,}75 \\ -0{,}75 \end{bmatrix}$$

$$\begin{bmatrix} W1 \\ W2 \\ W0 \end{bmatrix}^{(1)} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

Inclinação = $-\frac{1}{1}$

= -1

Deslocamento = $-\frac{(-1)}{(1)}$ (no eixo x1)

= +1

Deslocamento = $-\frac{(-1)}{(1)}$ (no eixo x2)

= +1.

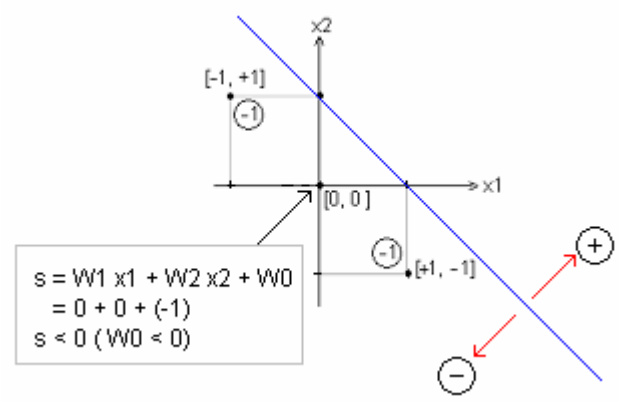


Figura 31. resultado da adaptação de pesos considerando-se o ponto P2.

Para este perfil de pesos, a saída y(2) vale:

y(2) = sgn[1 (1) + 1 (-1) – 1]

= –1 (OK!)

<u>Adaptação para o vetor padrão 3</u>

P3 = [–1, –1]

s(3) = 1 (-1) + 1 (-1) – 1

= –3

y(3) = –1 (OK!) (não necessidade de realizar a adaptação de pesos para o ponto P3).

Adaptação para o vetor padrão 1

P1 = [+1, +1]

s(1) = 1 (1) + 1 (1) – 1

= +1

y(1) = +1 (OK!) (não necessidade de realizar a adaptação de pesos para o ponto P1)

Deste modo, o treinamento pode ser considerado finalizado com pesos finais:

$$\begin{bmatrix} W1 \\ W2 \\ W0 \end{bmatrix} = \begin{bmatrix} +1 \\ +1 \\ -1 \end{bmatrix} .$$

Referências Bibliográficas

[1] CARPENTER, G. A. and GROSSBERG, S. “Pattern Recognition by Self-Organizing Neural Networks”, *The MIT Press*, Cambridge, Massachusetts, USA, 1991.

[2] FOGEL, D. B. “Evolutionary Computation: Toward a New Philosophy of Machine Intelligence”, *IEEE Press*, New York, 1995.

[3] HAYKIN, S. “Neural Networks: A Comprehensive Foundation”, *Prentice-Hall*, Upper Saddle River, New Jersey, USA, 1994.

[4] KARTALOPOULOS, S. V. “Understanding Neural Networks and Fuzzy Logic”, *IEEE Press*, New York, 1996.

[5] W. S. McCulloch, and Pitts, W. “A Logical Calculus of the Ideas Immanent in Nervous Activity", *Bulletin of Mathematical Biophysics*, No. 9, (1943). pp. 127-147.

[6] KUNG, S. Y. “Digital Neural Networks”, *PTR Prentice-Hall*, Englewood Cliffs, N. Jersey, USA, 1993.

[7] SIMPSON, P. K. “Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations”, *Pergamon Press*, New York, 1989.

[8] WERBOS, P. J. “Beyond Regression: New Tools For Prediction And Analysis in The Behavioral Sciences”, *Master Thesis*, Harvard University, 1974.

[9] WIDROW, B.; LEHR, M. A. “30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation”, *Proceedings of the IEEE*, Vol. 78, No. 9, September – 1990, pp. 1415-1442.

UNIVERSIDADE ESTADUAL PAULISTA – UNESP
FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Rede Neural Retropropagação

Carlos Roberto Minussi

Ilha Solteira – SP, maio-2008.

7. Rede Neural *Feedforward* Retropropagação

As redes neurais *feedforward* com não-linearidade tipo relé apresentam grandes dificuldades de realização, tendo em vista que as técnicas de treinamento disponíveis são pouco eficientes. Deste modo, uma arquitetura neural alternativa foi proposta por Werbos ([1]), que consiste na substituição da não-linearidade relé por uma função sigmoidal e no emprego de um mecanismo de adaptação de pesos (treinamento) chamado "Retropropagação" (*Back-propagation*). Por conseguinte, por abuso de linguagem, na seqüência, chamar-se-á a rede neural *feedforward*, usando a técnica de treinamento retropropagação, como sendo Rede Neural Retropropagação. Trata-se de um procedimento de ajuste pesos baseado no erro quadrático dos neurônios da saída da rede neural em que tal erro é propagado no sentido inverso (da saída para a entrada). As variações dos pesos são determinadas usando-se o algoritmo gradiente descendente ([2]). O diagrama esquemático desta rede neural é mostrado na Figura 32 (2 camadas com 2 neurônios e 1 neurônio, respectivamente).

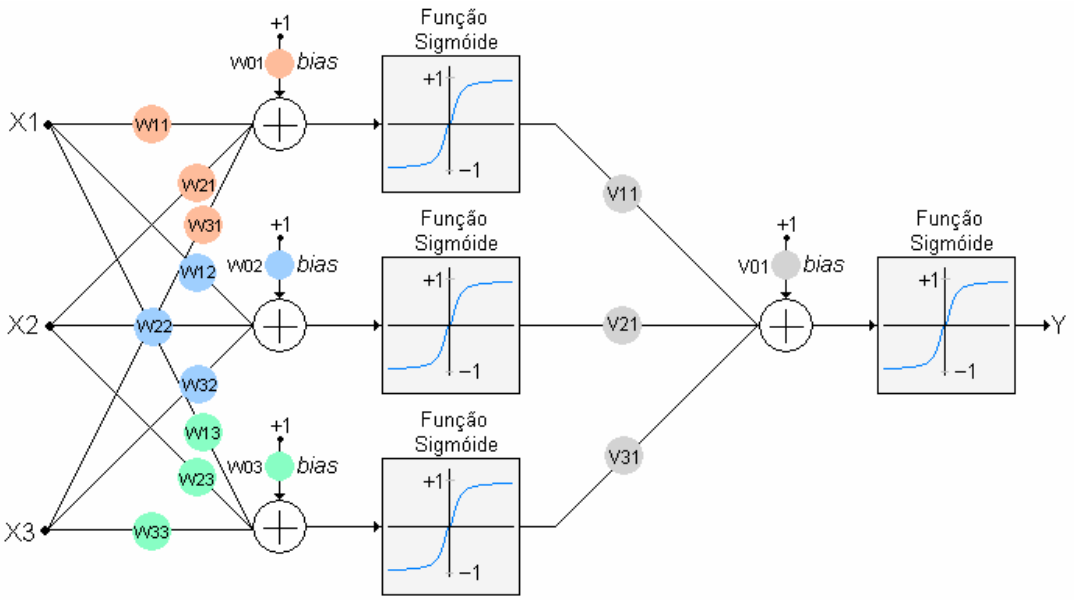
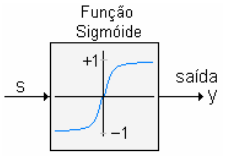


Figura 32. Rede neural *feedforward* com não-linearidade sigmoidal.

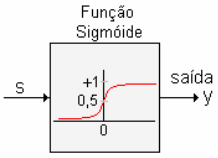
A função sigmóide é expressa por, tomando como referência a entrada (s) e saída (y) desta função (Figura 33):

$$y = \frac{\{1 - e^{(-\lambda s)}\}}{\{1 + e^{(-\lambda s)}\}} \quad (\text{para } -1 \leq y \leq 1) \qquad (7.1)$$



ou:

$$y = \frac{1}{\{1 + \exp(-\lambda s)\}} \quad (\text{para } 0 \leq y \leq 1) \qquad (7.2)$$



sendo:

s = saída intermediária do neurônio;

λ = inclinação da função sigmóide;

e Δ número de Neper (função exponencial)

= 2,7183.

As funções sigmoidais (7.1) e (7.2) devem ser empregadas quando se trabalha com grandezas de saída (y), com variações entre –1 e +1 e entre 0 e +1, respectivamente.

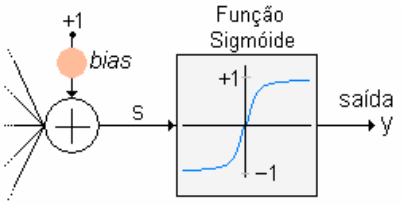


Figura 33. Não linearidade definida pela função sigmóide.

O efeito da variação do parâmetro λ na função sigmóide pode ser melhor compreendido através da ilustração mostrada na Figura 34.

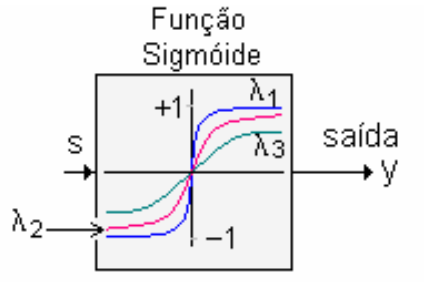


Figura 34. Comportamento da função sigmóide com a variação de λ (λ1 > λ2 > λ3 > 0).

Observa-se que à medida que λ aumenta, a função sigmóide se aproxima da função relé (função relé ≡ função sigmóide com λ = ∞). Portanto, o parâmetro λ controla a inclinação da função sigmóide, conforme será abordado mais adiante.

Exercício 1.

Calcular os valores da saída e os valores intermediários da rede neural mostrada na Figura 35, considerando-se os vetores de entrada $\mathbf{X} = [x1\ x2]^T$ ($\mathbf{X}^1 = [\ 1\ 1\ ]^T$, $\mathbf{X}^2 = [\ 1\ -1\ ]^T$, $\mathbf{X}^3 = [-1\ -1\ ]^T$ e $\mathbf{X}^4 = [-1\ 1\ ]^T$), λ = 0,5 e pesos indicados na referida Figura.

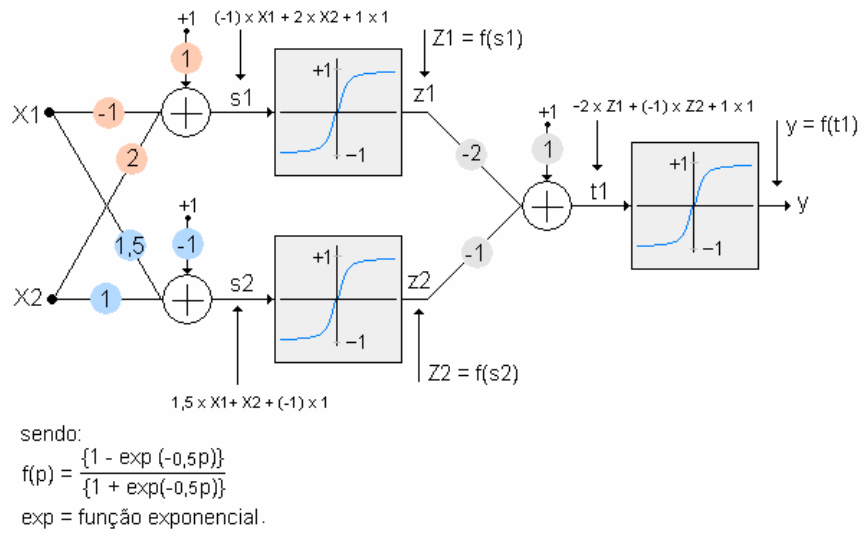


Figura 35. Rede neural.

Tabela 7. Cálculo da rede neural.

Entradas		**Saídas intermediárias**					**Saídas**
X1	X2	s1	s2	z1	z2	t1	y
1	1	2	1,5	0,4621	0,3584	-0,2826	-0,0705
1	–1	-2	-0,5	-0,4621	-0,1244	2,0486	0,4716
–1	–1	0	-3,5	0	-0,7039	1,7039	0,4020
–1	1	4	-1,5	0,7616	-0,3584	-0,1648	-0,0412

Supondo agora que a saída desejada, por exemplo, para o padrão $\mathbf{X}1 = [\ 1\ \ 1]^T$ seja igual a 0,5. O valor da saída apurado na Tabela 7 corresponde a –0,0705. Portanto, produzindo um erro (d – y) igual a 0,5705, conforme é mostrado na Figura 36.

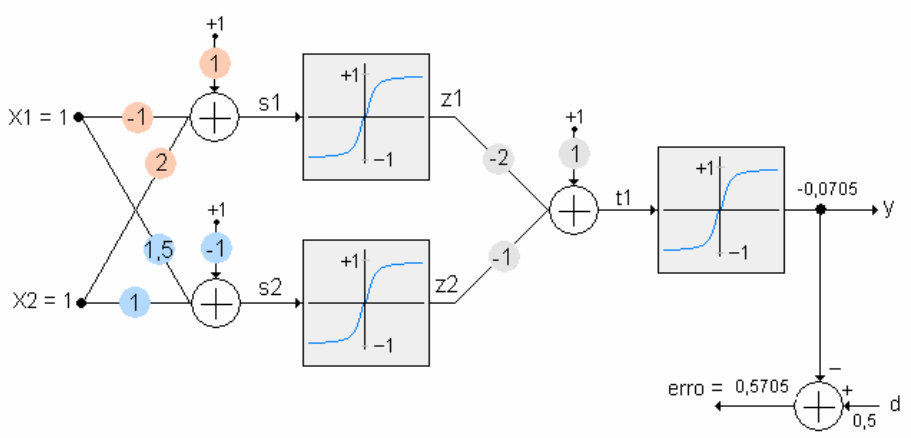


Figura 36.

Deste modo, este erro deve ser reduzido de 0,5705 para um erro que satisfaça a:

$$|\text{erro}| \leq \text{tolerância preespecificada.} \qquad (7.3)$$

ou seja, deve-se determinar variações nos pesos da rede neural que produzam este resultado desejado. A técnica a ser aplicada, neste caso, é o algoritmo retropropagação, o qual é baseado no método do gradiente descendente ([2]):

$$\begin{aligned} \Delta \mathbf{W}^{(k)} &= \mathbf{W}^{(k+1)} - \mathbf{W}^{(k)} \\ &= \alpha \, (-\nabla \varepsilon^{2(k)}) \end{aligned} \tag{7.4}$$

sendo:

$\Delta \mathbf{W}$ = variação do vetor de pesos **W**;

k = índice que indica o número da iteração;

α = taxa de treinamento;

$\nabla \varepsilon$ = vetor gradiente do erro quadrático;

ε^2 = erro quadrático da rede neural.

Este algoritmo encontra-se ilustrado na Figura 37.

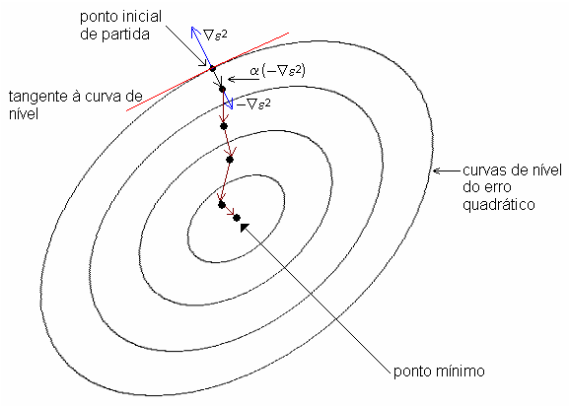


Figura 37. Método gradiente descendente.

O ajuste de peso segue na direção contrária ao gradiente de ε^2. O parâmentro α atua como um fator de escala, diminuindo ou aumentando o tamanho do vetor $(-\nabla\varepsilon^2)$. Deste modo, α

constitui num parâmentro de controle da estabilidade do processo de busca do ponto de equilíbrio mínimo. A arbitragem de α deve ser realizada com critério, conforme será abordado mais adiante.

8. Treinamento usando o algoritmo Retropropagação (*Backpropagation*)

O treinamento, via Retropropagação (RP), é iniciado pela apresentação de um padrão **X** à rede, o qual produzirá uma saída **Y**. Em seguida calcula-se o erro de cada saída (diferença entre o valor desejado e a saída). O próximo passo consiste na determinação do erro propagado no sentido inverso, através da rede associada à derivada parcial do erro quadrático de cada elemento com relação aos pesos e, finalmente, ajustando os pesos de cada elemento. Um novo padrão é apresentado – assim, o processo é repetido, para todos os padrões, até a convergência (expressão (3) tolerância preestabelecida). Os pesos iniciais são normalmente adotados como números randômicos. O algoritmo RP consiste na adaptação de pesos, tal que, é minimizado o erro quadrático da rede. A soma do erro quadrático instantâneo de cada neurônio alocado na última camada (saída da rede) é dada por:

$$\varepsilon^2 = \sum_{i=1}^{ns} \varepsilon_i^2 \tag{8.1}$$

sendo:

ε_i = $d_i - y_i$;

d_i = saída desejada do i-ésimo elemento da última camada da rede;

y_i = saída do i-ésimo elemento da última camada da rede;

ns = número de neurônios da última camada da rede.

Considerando-se o neurônio de índice i da rede, e utilizando-se o método do gradiente descendente, o ajuste de pesos pode ser formulado como:

$$\mathbf{U}_i\,(k+1) = \mathbf{U}_i\,(k) + \Delta\mathbf{U}_i\,(k) \tag{8.2}$$

sendo:

$\Delta \mathbf{U}_i(k) = -\alpha [\nabla_i(k)]$;

$\nabla_i(k)$ = gradiente do erro quadrático com relação aos pesos do neurônio i avaliado em h;

$\mathbf{U}_i$ = vetor contendo os pesos do neurônio i

$= [\, w_{0i} \quad w_{1i} \quad w_{2i} \quad \ldots \quad w_{ni} \,]^T$.

Na Eq. (8.2) a direção adotada para minimizar a função objetivo do erro quadrático corresponde à direção contrária ao gradiente. O parâmetro α determina o comprimento do vetor $[\Delta \mathbf{U}_i(k)]$.

O algoritmo RP é abordado na literatura sob várias formas com o propósito de torná-lo mais rápido computacionalmente. Uma formulação bastante interessante é o algoritmo RP com momento. Então, efetuando-se o cálculo do gradiente como indicado na Eq. (8.2), considerando-se a função sigmóide definida pela Eq. (7.1) ou (7.2) e o termo momento, obtém-se o seguinte esquema de adaptação de pesos:

$$u_{ij}(k+1) = u_{ij}(k) + \Delta u_{ij}(k) \tag{8.3}$$

sendo:

$\Delta u_{ij}(k) = 2\,\alpha\,(1-\eta)\,\delta_j\,x_i + \eta\,\Delta u_{ij}(k\text{-}1)$; (8.4)

u_{ij} = peso correspondente à interligação entre o i-ésimo e j-ésimo neurônio;

η = constante momento ($0 \le \eta < 1$).

Se o elemento j encontrar-se na última camada, então:

$$\delta_j = \theta_j\ \varepsilon_j \tag{8.5}$$

em que:

θ = derivada da função sigmóide dada pela Eq. (7.1) ou (7.2), respectivamente, com relação a s_j

$= 0{,}5\,\lambda\,(1 - y^2)$ (8.6)

$= \lambda\, y\,(1 - y)$. (8.7)

Se o elemento j encontrar-se nas demais camadas, tem-se:

$$\delta_j = \theta_j \sum_{k \in R(j)} w_{jk}\, \delta_k \qquad (8.8)$$

sendo:

R(j) = conjunto dos índices dos elementos que se encontram na fileira seguinte à fileira do elemento j e que estão interligados ao elemento j.

A derivadas da função sigmóide (Equações (7.1) e (7.2)) podem ser calculadas da seguinte forma. A derivada da função sigmoidal (7.1) vale:

$$\frac{\partial y}{\partial s} = \frac{\partial}{\partial s}\left\{\frac{1 - e^{(-\lambda s)}}{1+e^{(-\lambda s)}}\right\}$$

$$= \frac{[1+e^{(-\lambda s)}]\lambda e^{(-\lambda s)} - [1-e^{(-\lambda s)}](-\lambda e^{(-\lambda s)})}{[1+e^{(-\lambda s)}]^2}$$

$$= \frac{2\lambda e^{(-\lambda s)}}{[1+e^{(-\lambda s)}]^2}$$

$$= \frac{\lambda}{2}\frac{4\lambda e^{(-\lambda s)}}{[1+e^{(-\lambda s)}]^2}.$$

Nesta expressão pode-se acrescentar no numerador os termos (1 –1) e $[\{e^{-\lambda s}\}^2 - \{e^{-\lambda s}\}^2]$, ou seja, estão sendo acrescentados 2 valores nulos, o que não afetará o resultado final:

$$\frac{\partial y}{\partial s} = \frac{\lambda}{2}\frac{4\lambda e^{(-\lambda s)} + 1 - 1 + \{e^{(-\lambda s)}\}^2 - \{e^{(-\lambda s)}\}^2}{[1+e^{(-\lambda s)}]^2}$$

$$= \frac{\lambda}{2}\frac{\{1+ e^{(-\lambda s)}\}^2 - \{1- e^{(-\lambda s)}\}^2}{[1+e^{(-\lambda s)}]^2}$$

$$= \frac{\lambda}{2}\{\frac{[1+ e^{(-\lambda s)}]^2}{[1+e^{(-\lambda s)}]^2} - \frac{[1- e^{(-\lambda s)}]^2}{[1+e^{(-\lambda s)}]^2}\}$$

$$= 0{,}5\ \lambda\ (1 - y^2)$$

A derivada parcial da função sigmóide dada pela Equação (7.2), pode ser descrita por:

$$\frac{\partial y}{\partial s} = \frac{\partial}{\partial s}\left\{\frac{1}{1+e^{(-\lambda s)}}\right\}$$

$$= \frac{(-1)\lambda e^{(-\lambda s)}}{[1+e^{(-\lambda s)}]^2}$$

$$= \lambda\{\frac{(1-1)+ e^{(-\lambda s)}}{[1+e^{(-\lambda s)}]^2}\}$$

$$= \lambda\{\frac{[1+e^{(-\lambda s)}]-1}{[1+e^{(-\lambda s)}]^2}\}$$

$$= \lambda\{\frac{[1+e^{(-\lambda s)}]}{[1+e^{(-\lambda s)}]^2} - \frac{1}{[1+e^{(-\lambda s)}]^2}\}$$

$$= \lambda\{\frac{1}{1+e^{(-\lambda s)}} - \frac{1}{[1+e^{(-\lambda s)}]^2}\}$$

$$= \lambda\ (y - y^2)$$

$$= \lambda\ y\ (1 - y).$$

Na Figura 38, mostra-se como pode-se realizar o treinamento da rede neural feedforward de 3 camadas (2, 3 e 2 neurônios nas camadas de entrada, escondida e de saída, respectivamente), via retropropagação. Como forma ilustrativa, omite-se o termo momento.

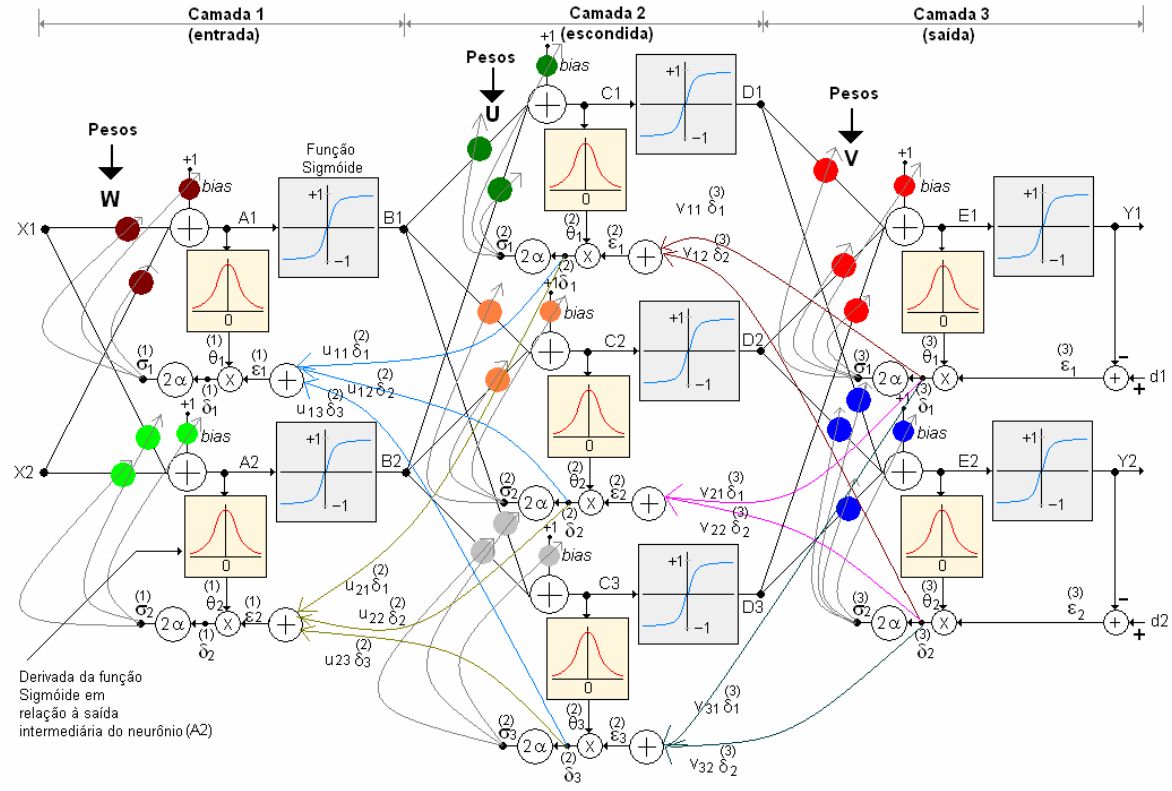


Figura 38. Ilustração do treinamento da rede neural via técnica retropropagação.

Considerando-se a Figura 38, os pesos são assim definidos:

Matrizes de Pesos

W Δ matriz de pesos da camada de entrada

$$= \begin{bmatrix} w11 & w12 \\ w21 & w22 \\ w01 & w02 \end{bmatrix};$$

U Δ matriz de pesos da camada escondida;

$$= \begin{bmatrix} u11 & u12 & u13 \\ u21 & u22 & u23 \\ u01 & u02 & u03 \end{bmatrix};$$

V Δ matriz de pesos da camada de saída;

$$= \begin{bmatrix} v11 & v12 \\ v21 & v22 \\ v31 & v32 \\ v01 & v02 \end{bmatrix}.$$

Atualização dos Pesos

Considerando-se a aplicação na entrada do vetor $\mathbf{X} = [\ X1\ X2\ ]^T$ (estímulo de entrada) cuja saída associada é definida pelo vetor $\mathbf{Y} = [\ Y1\ Y2]^T$, a (k+1)-ésima atualização atualização dos pesos, via algoritmo retropropagação, é dada por:

$$\begin{bmatrix} v11 \\ v21 \\ v31 \\ v01 \end{bmatrix}^{(k+1)} = \begin{bmatrix} v11 \\ v21 \\ v31 \\ v01 \end{bmatrix}^{(k)} + \overset{(3)}{\sigma_1} \begin{bmatrix} D1 \\ D2 \\ D3 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} v12 \\ v22 \\ v32 \\ v02 \end{bmatrix}^{(k+1)} = \begin{bmatrix} v12 \\ v22 \\ v32 \\ v02 \end{bmatrix}^{(k)} + \overset{(3)}{\sigma_2} \begin{bmatrix} D1 \\ D2 \\ D3 \\ 1 \end{bmatrix}$$

sendo:

$$\overset{(3)}{\sigma_1} = 2\alpha\ \overset{(3)}{\varepsilon_1}\ \overset{(3)}{\theta_1}$$

$$\overset{(3)}{\sigma_2} = 2\alpha\ \overset{(3)}{\varepsilon_2}\ \overset{(3)}{\theta_2}$$

$$\overset{(3)}{\varepsilon_1} = d1 - Y1$$

$$\overset{(3)}{\varepsilon_2} = d2 - Y2$$

$$\begin{bmatrix} u11 \\ u21 \\ u01 \end{bmatrix}^{(k+1)} = \begin{bmatrix} u11 \\ u21 \\ u01 \end{bmatrix}^{(k)} + \overset{(2)}{\sigma_1} \begin{bmatrix} B1 \\ B2 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u12 \\ u22 \\ u02 \end{bmatrix}^{(k+1)} = \begin{bmatrix} u12 \\ u22 \\ u02 \end{bmatrix}^{(k)} + \overset{(2)}{\sigma_2} \begin{bmatrix} B1 \\ B2 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u13 \\ u23 \\ u03 \end{bmatrix}^{(k+1)} = \begin{bmatrix} u13 \\ u23 \\ u03 \end{bmatrix}^{(k)} + \overset{(2)}{\sigma_3} \begin{bmatrix} B1 \\ B2 \\ 1 \end{bmatrix}$$

sendo:

$$\overset{(2)}{\sigma_1} = 2\alpha\, \overset{(2)}{\varepsilon_1}\, \overset{(2)}{\theta_1}$$

$$\overset{(2)}{\sigma_2} = 2\alpha\, \overset{(2)}{\varepsilon_2}\, \overset{(2)}{\theta_2}$$

$$\overset{(2)}{\sigma_3} = 2\alpha\, \overset{(2)}{\varepsilon_3}\, \overset{(2)}{\theta_3}$$

$$\overset{(2)}{\varepsilon_1} = v11\, \overset{(3)}{\delta_1} + v12\, \overset{(3)}{\delta_2}$$

$$\overset{(2)}{\varepsilon_2} = v21\, \overset{(3)}{\delta_1} + v22\, \overset{(3)}{\delta_2}$$

$$\overset{(2)}{\varepsilon_3} = v31\, \overset{(3)}{\delta_1} + v32\, \overset{(3)}{\delta_2}$$

$$\begin{bmatrix} w11 \\ w21 \\ w01 \end{bmatrix}^{(k+1)} = \begin{bmatrix} w11 \\ w21 \\ w01 \end{bmatrix}^{(k)} + \overset{(1)}{\sigma_1} \begin{bmatrix} X1 \\ X2 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} w12 \\ w22 \\ w02 \end{bmatrix}^{(k+1)} = \begin{bmatrix} w12 \\ w22 \\ w02 \end{bmatrix}^{(k)} + \overset{(1)}{\sigma_2} \begin{bmatrix} X1 \\ X2 \\ 1 \end{bmatrix}$$

sendo:

$$\overset{(1)}{\sigma_1} = 2\alpha\, \overset{(1)}{\varepsilon_1}\, \overset{(1)}{\theta_1}$$

$$\sigma_2^{(1)} = 2\alpha\, \varepsilon_2^{(1)}\, \theta_2^{(1)}$$

$$\varepsilon_1^{(1)} = u11\, \delta_1^{(2)} + u12\, \delta_2^{(2)} + u13\, \delta_3^{(2)}$$

$$\varepsilon_2^{(1)} = u21\, \delta_1^{(2)} + u22\, \delta_2^{(2)} + u23\, \delta_3^{(2)}$$

Exercício 2.

Considerando-se a rede neural do Exercício 1, o vetor padrão de entrada X1 = [1 1]^T, o valor desejado (d) igual a 0,5 e α = 0,8, determinar 1 iteração do algoritmo retropropagação. Na Figura 39 mostra-se os principais dados (valores dos pesos, saídas intermediárias, etc.). O erro, neste caso, corresponde a ε = 0,7169.

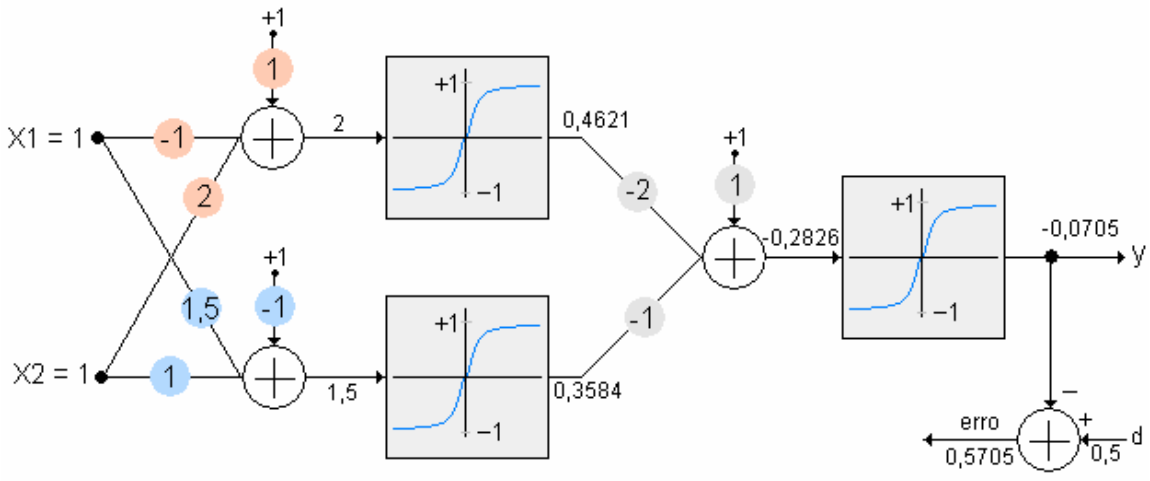


Figura 39. Representação da rede neural referente ao exercício 2.

Na Figura 40 apresentam-se os resultados do ajuste de pesos (1 iteração) pela aplicação do algoritmo retropropagação. Com a realização de 1 iteração, os valores dos pesos atualizados apresentados estão indicados na referida figura.

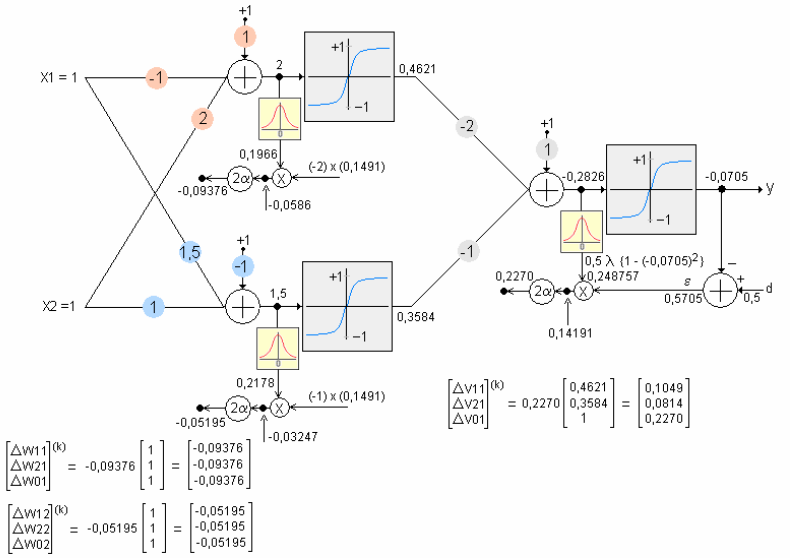


Figura 40. Uma iteração do algoritmo retropropagação.

Na Figura 41 mostram-se os pesos atualizados e o erro resultante (ε = 0, 4596). Observa-se que o erro diminuiu. Assim, realizando várias iterações o erro final deverá atender o critério de convergência (equação (7.3)).

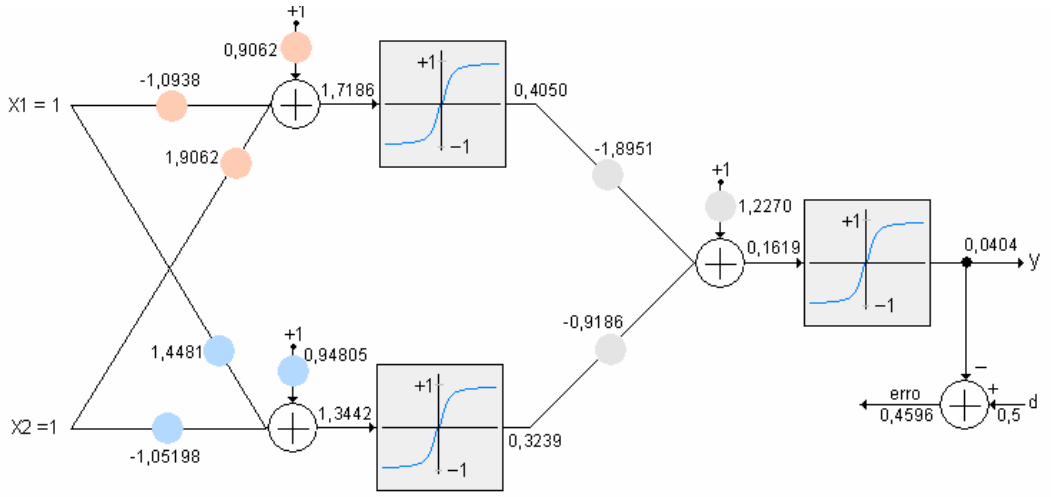


Figura 41. Rede neural com pesos atualizados e o erro resultante.

8.1 Especificação de uma Rede Neural

1. Número de camadas

   Recomendação:

   Empregar 3 camadas (1 de entrada, 1 escondida (intermediária) e 1 de saída).

2. Número de neurônios da camada de entrada

   Recomendação:

   $N_E = \dim(\mathbf{X})$ (8.1.1)

   sendo:

   N_E = número de neurônio da camada de entrada;

   dim Δ dimensão do vetor padrão de entrada

   = número de componente do vetor padrão de entrada **X**.

3. Número de neurônios da camada de saída

   Recomendação:

   $N_S = \dim(\mathbf{Y})$ (8.1.2)

   sendo:

   N_S = número de neurônio da camada de saída;

   dim Δ dimensão do vetor padrão de saída **Y**.

4. Número de neurônios da camada escondida

   Recomendação:

   $N_I \geq \text{máx}\{N_E , N_S\}$ (8.1.3)

   sendo:

   máx = operador máximo.

Este critério evita que haja o estrangulamento das informações entre as camadas de entrada e de saída. Contudo, N_I não deve exceder muito em relação ao número indicado por máx$\{N_E , N_S\}$, com o objetivo de não aumentar o tempo de processamento do treinamento sem necessidade, visto que a rede neural possui naturalmente uma certa redundância, ou seja, a capacidade de armazenamento das informações é, via de regra, bem superior à necessidade para a realização da tarefa escolhida.

8.2. Outras Recomendações

1. Arbitragem do parâmetro que define a inclinação da função sigmóide

   a) Para uso de dados binários: empregar valores grandes para λ, ou seja, à medida em que λ aumenta, a função sigmóide tende a se aproximar da função relé (realização binária);

   b) Para uso de dados analógicos (contínuos), a recomendação é para a adoção de valores λ pequenos, e.g., $0{,}3 \leq \lambda \leq 0{,}5$;

   c) A referência [2] recomenda que a taxa de treinamento seja escolhida de acordo com o seguinte critério:

$$0 < \alpha < \frac{1}{\text{traço}(\mathbf{R})} \tag{8.2.1}$$

   sendo:

   $\mathbf{R}$ = $E(\mathbf{Z}^T \mathbf{Z})$

   E(.) = valor esperado

   $\mathbf{Z}$ = matriz contendo todos os vetores padrões usados na de treinamento da rede neural;

   traço(.) = somatório dos elementos diagonais de uma matriz quadrada.

   Por exemplo, considerando-se os padrões usados no Exercício 1, a matriz Z será assim definida:

$$\mathbf{Z} = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

Tendo em vista que esta escolha demanda um certo tempo de processamento para ser executada, como alternativa recomenda-se a adoção de um valor para α pequeno, *e.g.*, $\alpha = 0.5$;

d) Outra sugestão para arbitragem de α, baseada na experiência com treinamento de redes neurais retropropagação:

$$0 < \alpha < \alpha_{máx} \tag{8.2.2}$$

sendo:

$$\alpha_{máx}\, \lambda = 1 \tag{8.2.3}$$

ou seja, se adotarmos $\lambda = 0{,}5$, o limite de α será $\alpha_{máx} = 2$.

8.3. Deficiência do Treinamento via Técnica Retropropagação

1. Existência de Pontos Mínimos Locais

O treinamento da rede neural será considerado concluído quando houver a convergência para um ponto mínimo global. Porém, a função erro quadrático combinado com o emprego de funções sigmoidais apresentam um grande número de pontos mínimos locais. A técnica retropropagação, que baseada no método do gradiente descendente (uso de derivadas), apresenta a deficiência de convergir para um ponto mínimo local e, portanto, o processo será finalizado sem que houvesse atendido o objetivo final que é a convergência para um ponto mínimo global. Daí decorre a necessidade de se fixar um valor pequeno para a taxa de treinamento. Esta recomendação visa reduzir o risco de uma convergência para um ponto mínimo local.

2. Existência de paralisia no Mecanismo de Adaptção de Pesos

Outro detalhe que contribui para a redução da eficiência, quando não impossibilita a conclusão do treinamento, refere-se à realimentação através da derivada da função sigmóide (parâmetro θ). Isto ocorre quando o valor da saída intermediária do neurônio for muito grande em módulo, ou

seja, quanto (s) for excessivamente negativa ou excessivamente positiva. Nestes casos, o valor de θ será muito pequeno. O processo chega à saturação (fenômeno chamado paralisia) quando a saída do neurônio (y) for ± 1, para a função sigmóide fornecida pela eq. (1), 0 ou +1 quando se tratar da função sigmóide dada pela eq. (2). Para melhor entender tal comportamento, analisa-se a Figura 42, tomando-se como exemplo a função sigmóide definida pela Eq. (7.1).

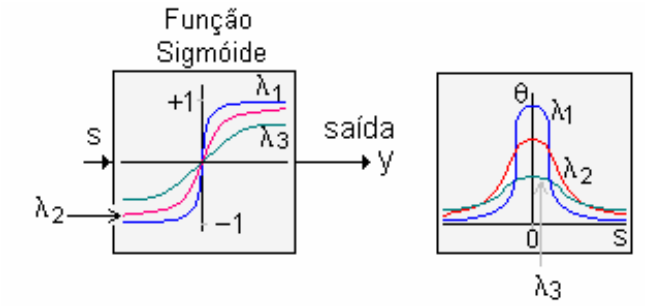


Figura 42. Comportamentos da função sigmóide e sua derivada θ, em termos dos parâmetros λ1, λ2 e λ3 (λ1 > λ2 > λ3).

A amplitude máxima de θ corresponde a:

$\theta_{máx}$ = 0,5 λ (1 – 0^2), para s = 0 (y = 0)

= 0,5 λ, para a função sigmóide da Eq. (7.1)

$\theta_{máx}$ = λ 0,5 (1 – 0,5), correnpondente a s = 0 (y = 0,5)

= 0,25 λ, para a função sigmóide da Eq. (7.2).

A partir deste resultados, conclui-se que ao diminuirmos λ, diminui-se a amplitude de θ e abre-se a base de θ. Portanto, um valor de λ pequeno é desejável, pois amplia-se o espaço de busca da solução (adaptação dos pesos). Contudo, neste caso, a amplitude de θ será menor. Este detalhe pode ser compensado usando-se a expressão (8.2.3), *i.e.*, se diminuimos o valor de λ, pode-se aumentar o valor de α até o valor $\alpha_{máx} = 1/\lambda$.

Com estas medidas, o treinamento da rede neural, via técnica retropropagação, torna-se mais eficiente, ainda que esta técnica tem se mostrado bastante lenta ([1]).

Para tornar ainda mais eficiente a técnica retropropagação, segere-se o uso de um controlador nebuloso (*fuzzy*) para adaptação da taxa de treinamento de forma ótima ([3]), bem como incluir a adaptação, de forma semelhante ao ajuste de pesos, em relação à inclinação da função sigmóide (λ) e um parâmetro (a ser incluido) para produzir o deslocamento da função sigmóide ([4]). Este procedimento atua de forma semelhante à rede neural baseada em funções de base radial (RBF *neural network*).

8.4. Procedimentos de Treinamento

Os pesos da rede são iniciados randomicamente considerando-se o intervalo {0,1}.

O treinamento via RP pode ser efetuado, basicamente, de duas formas:

- Procedimento 1. Consiste em ajustar os pesos da rede (considerando-se todas as camadas), fazendo-se com que haja convergência para cada padrão, até que se complete o conjunto de padrões de entrada. O processo deverá ser repetido até a total convergência, i.e., o erro quadrático seja inferior a uma tolerância preestabelecida para os padrões considerados.

- Procedimento 2. Este procedimento é idêntico ao primeiro, porém, fazendo-se somente uma iteração (ajuste de pesos) por padrão.

8.5. Referências Bibliográficas

[1] Werbos, P. J. "Beyond Regression: New Tools For Prediction And Analysis in The Behavioral Sciences", PhD. Thesis, Harvard University, 1974.

[2] Widrow, B.; Lehr, M.A. “30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation”, Proceedings of the IEEE, Vol. 78, No. 9, Sep. 1990, pp. 1415-1442.

[3] Arabshahi, P.; Choi, J. J.; Marks II, R. J. and Caudell, T. P. “Fuzzy Parameter Adaptation in Optimization: Some Neural Net Training Examples”, IEEE Computational Science & Engineering, Vol. 3, No. 1, pp. 57-65, 1996.

[4] Lopes, M. L. M.; Minussi, C. R. and Lotufo, A. P. “Electrical Load Forecasting Formulation By A Fast Neural Network”, Engineering Intelligent Systems For Electrical Engineering And Communications, Vol. 11, No. 1, pp. 51-57, 2003.

Redes Neurais de Hopfield

A rede neural de Hopfield é uma das concepções mais conhecidas de memória associativa. A seguir, apresenta-se um algoritmo referente a uma das possíveis redes de Hopfield, tomando-se como base a ilustração mostrada na Figura 1. Este algoritmo refere-se a uma versão da rede original de Hopfield ([1]), a qual é empregada como memória endereçada por conteúdo.

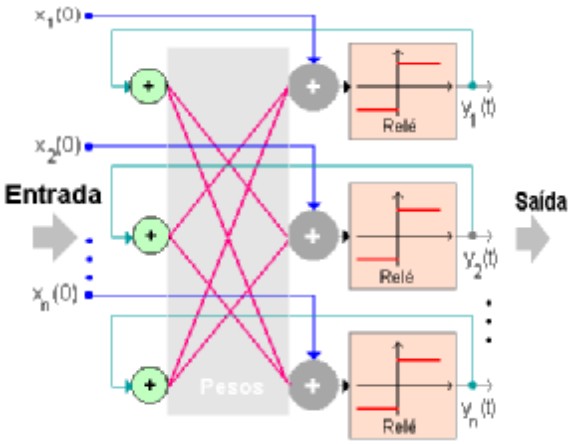


Figura 1. Uma possível configuração de uma rede neural de Hopfield.

Algoritmo

Passo 1. Definição dos pesos das conexões:

$$w_{ij} = \begin{cases} \sum_{r=1}^{M} x_i^{(r)} x_j^{(r)}, & i \neq j, \\ 0, & i = j. \end{cases} \qquad (1)$$

sendo:

w_{ij} = peso referente à conexão do neurônio i para o neurônio j

$x_i^{(r)}$ = (com valores +1 ou –1) é o i-ésimo elemento do padrão r;

M = representa o número de padrões a serem armazenados;

r = índice referente ao número do padrão.

Passo 2. Iniciar a rede com o padrão de entrada desconhecido:

$$s_i(0) = x_i(0),\ i = 1, 2, \ldots, N \qquad (2)$$

sendo:

N = número de componentes do vetor x;

$s_i(0)$ = saída do nó i correspondente ao tempo discreto t;

$x_i(0)$ = i-ésimo elemento do padrão de entrada desconhecido.

Passo 3. Processo iterativo:

$$s_j(t+1) = f_j\left\{\sum_{i=1}^{N} w_{ij}\, s_i(t)\right\},\ j = 1, 2, \ldots, N. \qquad (3)$$

sendo:

fj(.) Δ função não-linear

= sgn(.);

sgn = função sinal.

Este processo deve ser repetido até que:

$$\left| s_j(t+1) - s_j(t) \right| < \text{tolerância predefinida.}$$

A solução “s” no final do processo iterativo, se convergir, será idêntica a um dos padrões $x^{(r)}$, justamente aquele que melhor se aproximar caracteristicamente.

A rede de Hopfield é uma rede simétrica, *i.e.*, os pesos, correspondentes às conexões, definidos pela equação (1), valem:

$$w_{ij} = w_{ji}, \forall\ i, j;\ j \neq i$$
$$w_{ii} = 0. \qquad (4)$$

O sistema dinâmico definido pelas equações (3) e (4) é não-linear com multiplicidade de pontos de equilíbrio. Um estudo da estabilidade deste sistema pode ser realizado usando-se a teoria de estabilidade de Liapunov. Utilizando-se uma função (função energia) de Liapunov, a rede fica perfeitamente descrita por uma superfície de energia. Então, a partir de uma condição inicial, representando um ponto sobre esta superfície, a trajetória do sistema evoluirá, a cada instante, para níveis de energia mais baixos, até atingir o ponto de equilíbrio estável (ponto mínimo da superfície de energia) mais "próximo". Os padrões a serem armazenados, definidos pela estrutura (2), são pontos de equilíbrio estáveis.

Em vista disto, a rede de Hopfield apresenta limitações, quando empregada como memória endereçada por conteúdo:

1) o número de padrões que podem ser armazenados e precisamente reconstituídos é consideravelmente limitado. O aumento do número de padrões a serem armazenados representa uma elevação do número de pontos de equilíbrio. Assim, se muitos padrões forem armazenados, aumentará a probabilidade da rede convergir para um padrão não desejado, ou estranho ao conjunto dos padrões armazenados;
2) a rede pode apresentar problemas de convergência, se existir nos padrões de entrada, muitos "bits" coincidentes. Isto representa a proximidade de pontos de equilíbrio.

Estas duas observações representam, em última análise, a redução e proximidade de domínios (regiões) de estabilidade. Portanto, se o padrão desconhecido conter ruído significativo, o resultado obtido poderá não ser satisfatório (baixa convergência ou instabilidade).

Como forma de evitar que tais problemas ocorram, recomenda-se que ([1]) que o M não seja superior ao um pequeno percentual de N, ou seja:

$$M \leq 15\%\ N \qquad (5)$$

Exemplo 1

Supondo-se que se deseja armazenar os seguintes vetores padrões:

$A = [\ 1\ \ 1\ \ 1\ \ -1\ ]^T$,

$B = [\ 1\ \ 1\ \ 1\ \ 1\ \ ]^T$ e

$C = [-1\ -1\ \ 1\ \ \ 1\ ]^T$.

Tem-se, portanto,'

$N = 4$

$M = 3$.

Aplicando-se o Passo 1 do Algoritmo, obtém-se:

$w11 = 0$

$w22 = 0$

$w33 = 0$

$w44 = 0$

$w12 = A(1)*A(2)+B(1)*B(2)+C(1)*C(2)$

$= 1 \times 1 + 1 \times 1 + -1 \times (-1) = 3$

$w13 = A(1)*A(3)+B(1)*B(3)+C(1)*C(3)$

$= 1 \times 1 + 1 \times 1 + -1 \times 1 = 1$

$w14 = A(1)*A(4)+B(1)*B(4)+C(1)*C(4)$

$= 1 \times (-1) + 1 \times 1 + -1 \times 1 = -1$

$w21 = w12$

$w23 = A(2)*A(3)+B(2)*B(3)+C(2)*C(3)$

$= 1 \times 1 + 1 \times 1 + -1 \times 1 = 1$

$w24 = A(2)*A(4)+B(2)*B(4)+C(2)*C(4)$

$= 1 \times -1 + 1 \times 1 + -1 \times 1 = -1$

$w31 = w13$

$w32 = w23$

$w34 = A(3)*A(4)+B(3)*B(4)+C(3)*C(4)$

$= 1 \times (-1) + 1 \times 1 + 1 \times 1 = 1$

$w41 = w14$

$w42 = w24$

$w43 = w34$

$A = [\ 1\ \ 1\ \ 1\ \ -1\ ]^T$,

$B = [\ 1\ \ 1\ \ 1\ \ 1\ \ ]^T$ e

$C = [-1\ -1\ \ 1\ \ 1\ ]^T$.

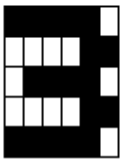
$$W = \begin{bmatrix} 0 & w12 & w13 & w14 \\ w21 & 0 & w23 & w24 \\ w31 & w32 & 0 & w34 \\ w41 & w42 & w43 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 3 & 1 & -1 \\ 3 & 0 & 1 & -1 \\ 1 & 1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{bmatrix}$$

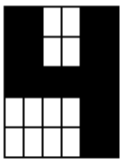
Exemplo 2

Padrões a Serem Armazenados

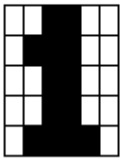
Rede Neural de Hopfield



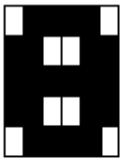
Padrão 1



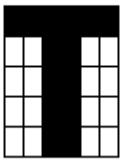
Padrão 2



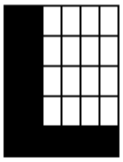
Padrão 3



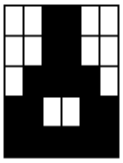
Padrão 4



Padrão 5



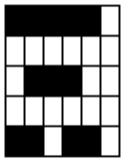
Padrão 6



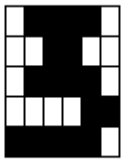
Padrão 7

Identificação de Padrão Desconhecido

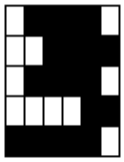
Rede Neural de Hopfield



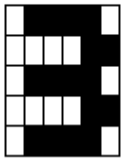
Padrão desconhecido



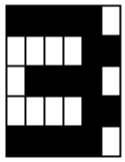
Iteração 1



Iteração 2



Iteração 3



Iteração 4

Exemplo 3

Otimização: Programação Linear

Uma formulação geral do problema pode ser colocada na seguinte forma:

- minimizar uma função escalar:

$$f(\mathbf{x}) \tag{6}$$

sujeito às restrições:

$$\mathbf{h}(\mathbf{x}) \geq 0 \tag{7}$$

sendo:

$\mathbf{x} = [\, x_1\ x_2\ x_3 \ldots x_n \,]^T$.

Este problema pode ser formulado como ([2]):

- minimizar : $\mathbf{c}^T \boldsymbol{v}$ (8)

sujeito às restrições : $\mathbf{L}v = \mathbf{b}$ (9)

$$0 \leq v_i \leq v_{max} \tag{10}$$

sendo: v_{max} = limite superior da variável v_i

$\mathbf{c}$ = vetor custo.

A equação (10) corresponde às restrições de igualdades e, feita as devidas manipulações matemáticas, também às restrições de desigualdades. A equação (8) é a função objetivo a ser minimizada. As equações (9) e (10) definem uma região factível poliedral em um espaço n-dimensional. Este problema pode ser resolvido utilizando-se uma rede neural tipo Hopfield, como proposto na referência [2], utilizando-se n neurônios. A variável v_i está relacionada com a saída deste neurônio, tipo Mc-Culloch-Pitts, com função de ativação sigmoidal, cujos valores de saída estão compreendidos entre 0 e v_{max}. A rede neural proposta por Wang ([2]), para a solução do problema de programação linear, composta por n neurônios, é mostrada na Figura 2.

A equação dinâmica da rede apresentada na Figura 2 é descrita pelo seguinte sistema de equações diferencial e algébrica:

$$\dot{\mathbf{u}}(t) = -\rho\, \mathbf{L}^T \mathbf{L}\, \boldsymbol{v}(t) + \rho\, \mathbf{L}^T \mathbf{b} - \kappa \exp(-\eta t)\, \mathbf{c} \tag{11}$$

$$v_i(t) = v_{max} / \{\, 1 + \exp[\, -\varphi\, u_i(t)]\}. \tag{12}$$

sendo:

ρ, κ e η são parâmetros escalares positivos;

t = tempo.

A equação (12) representa a função sigmóide com amplitude v_{max} e inclinação φ, em que a entrada e saída são, respectivamente, u e *v*.

Os pesos da rede neural são definidos a partir da estrutura do problema de programação linear:

- **Matriz de Pesos :** $\mathbf{W} = -\rho\,\mathbf{L}^T\mathbf{L}$ (13)
- **Pesos *Bias*** **:** $F(t) = \rho\,\mathbf{L}^T\mathbf{b} - \kappa \exp(-\eta\, t)\,\mathbf{c}$ (14)

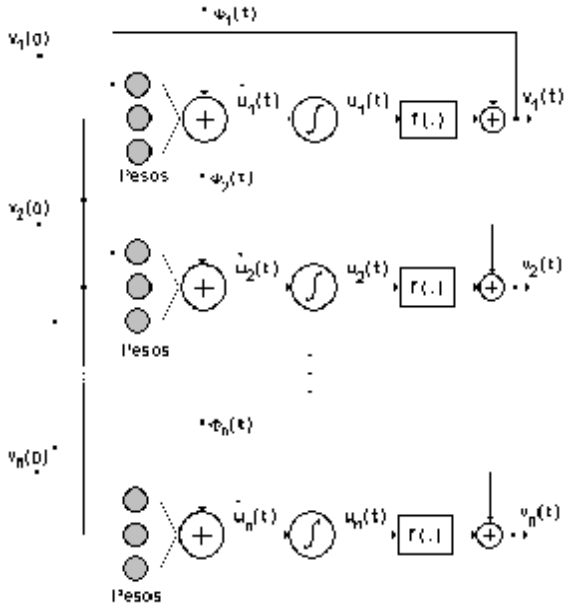


Figura 2. Esquema da rede neural.

Assim, dadas as condições iniciais do problema, o vetor saída – definido em função do estado do sistema – deverá evoluir para um ponto de equilíbrio que corresponde à solução do problema. Este mecanismo consiste em "forçar" o atendimento das restrições e minimizar a função objetivo. Em razão do termo decaimento no vetor peso *bias* (Φ(t)) esta rede neural recorrente é um sistema dinâmico variante no tempo. Como mencionada anteriormente, esta rede neural pode ser implementada através do desenvolvimento de circuitos dedicados (Chua e Lin ([3])).

Referências

[1] LIPPMANN, R. P. “An Introduction to Computing With Neural Nets”, *IEEE ASSP Magazine*, April 1987, pp. 4 – 22.

[2] WANG, J. “Analysis And Design of a Recurrent Neural Network for Linear Programming”, *IEEE Transaction on Circuits and Systems - 1: Fundamental Theory And Applications*, Vol. 40, No. 9, 1993, pp. 613–618.

[3] CHUA, L. O. and LIN, G. N. “Nonlinear Programming Without Computation”, *IEEE Transactions on Circuits and Systems*, Vol.. CAS-31, No. 2, (1984pp.182 - 188.

Rede Neural de Kohonen

A rede de Kohonen é uma rede neural não-supervisionada de mapeamentos auto-organizável, conforme mostrada na Figura 1. É composta por um reticulado (grade) bi-dimensional e um conjunto de vetores pesos, fixado inicialmente em valores aleatórios entre 0 e 1, $\boldsymbol{w}_j = [\, w_{1j}\ w_{2j}\ \ldots\ w_{nj}]^T$, $1 = 1, \ldots, nn$, e $\boldsymbol{x} = [\, x_1\ x_2\ \ldots\ x_n]^T$ um vetor de entrada, sendo *nn* o número de neurônios sobre a grade da rede neural. Trata-se, portanto, de um mapeamento de $\Re^n$ em $\Re^2$.

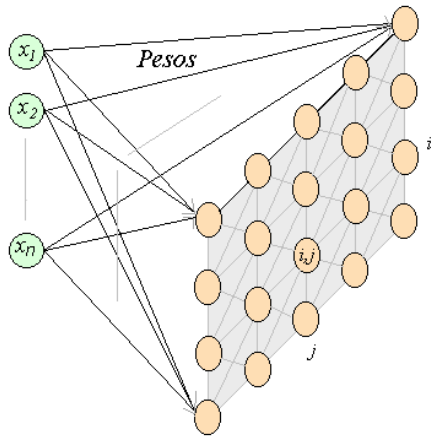


Figura 1. Estrutura da rede neural de Kohonen.

Cada neurônio sobre a grade está conectado a entrada da rede, através do vetor de entrada conforme é mostrado na Figura 2.

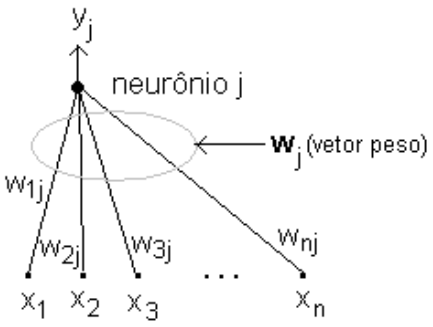


Figura 2. Conexão do vetor padrão de entrada X com o j-ésimo neurônio da rede de Kohonen.

Deste modo a saída y_j (atividade do j-ésimo neurônio) pode ser calculado da seguinte forma:

$$y_j = < \mathbf{W}_j, \mathbf{x} > \quad (1)$$

sendo:

y_j = atividade (saída) do j-ésimo neurônio sobre a grade de Kohonen;

$\mathbf{W}_j$ = $[w_{1j} w_{2j} \ldots w_{nj}]^T$

$\mathbf{x}$ = $[x_1 x_2 \ldots x_n]^T$

NB:

- Kohonen propôs esta rede baseada na regra de aprendizado de Hebb (Donald Hebb). Propôs, por conseguinte, um procedimento sistemático e simplificado do método de Hebb;
- o aprendizado é similar ao que ocorre no cérebro (neo-cortex cerebral).

O treinamento (ou aprendizagem) é um processo através do qual os parâmetros de uma rede neural artificial são ajustados utilizando uma forma contínua de estímulos. No caso do treinamento por competição, dado um estímulo à rede (vetor de entrada), as unidades de saída disputam entre si para serem ativadas. A unidade vencedora terá seus pesos atualizados no treinamento. A forma mais extrema de competição entre neurônios é chamada o vencedor leva tudo (*winner take all*). Como o próprio nome sugere, apenas um neurônio terá sua saída ativada. Já o mapeamento auto-organizável, proposto por Kohonen, que organiza os dados de entrada em agrupamentos (*clusters*), corresponde a um tipo de treinamento competitiva e não-supervisionada. Nesta forma de treinamento, as unidades que atualizam seus pesos, o fazem formando um novo vetor peso que é uma combinação linear dos pesos antigos e o atual vetor de entrada. O método consiste na adaptação de pesos da rede neural como mostrado na Figura 3 em que os vetores **x** e **w** encontram-se normalizados (comprimentos unitários). Considera-se um determinado vetor de pesos $\mathbf{w}_h$ correspondente ao tempo discreto h. O novo valor $\mathbf{w}_{h+1}$ pode ser encontrado da seguinte forma:

$$\mathbf{w}_{h+1} = \mathbf{w}_h + \alpha (\mathbf{x} - \mathbf{w}_h). \quad (2)$$

sendo:

α = taxa de treinamento ($0 \leq \alpha \leq 1$);

H = índice de atualização.

A taxa de treinamento α pode ser um valor constante ou uma função decrescente.

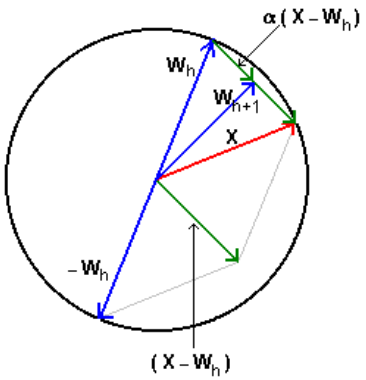


Figura 3. Regra de adaptação de pesos da rede de Kohonen.

O comprimento do vetor *w*, em cada atualização, é menor do que 1. Portanto, após cada adaptação de pesos, há necessidade de uma nova normalização do vetor de pesos.

O neurônio vencedor é escolhido como sendo aquele que apresentar maior atividade (y) sobre a grade de Kohonen, ou seja:

$$NV = \max_{i} \{<\mathbf{w}_i, \mathbf{x}>\} \quad (3)$$

Ou:

$$= \min_{i} \{ \| \mathbf{x} - \mathbf{w}_i \| \} \quad (4)$$

sendo:

NV = neurônio vencedor;

$\| \cdot \|$ = norma Euclidiana.

Ao neurônio vencedor (Equação 3), atribui-se índice k. Deve-se observar que, quando os vetores $\mathbf{w}_i$ e $\mathbf{x}$ encontram-se normalizados (comprimentos unitários) ,as indicações do neurônio vencedor, através do máximo produto interno (Equação 3) e a através da mínima distância (Equação 4), são rigorosamente as mesmas, ou seja, estes dois procedimentos são equivalentes. Usando-se, então, a regra de adaptação, Kohonen propôs que o ajuste da rede neural em que os pesos são ajustados considerando-se vizinhanças em torno do neurônio vencedor, como mostrado na Figura 4, sendo:

$NC_k(S_i)$ = vizinhança do neurônio vencedor k, referente à região S_i, em que $S_1 \subset$ (contido) $S_2 \subset S_3$.

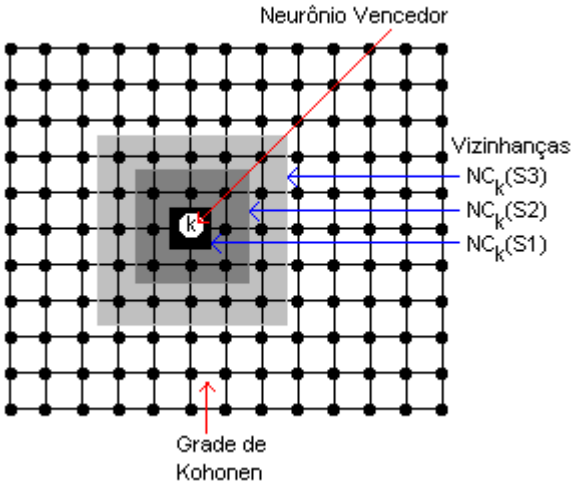


Figura 4. Vizinhanças (quadradas) do neurônio vencedor.

Na Figura 5 são mostrados alguns tipos mais freqüentes de vizinhanças encontrados na literatura especializada.

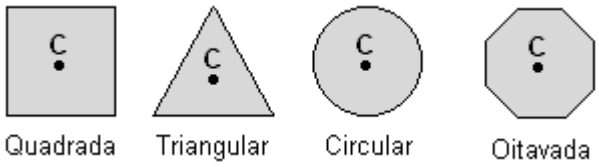


Figura 5. Tipos de vizinhanças mais utilizadas.

Algoritmo Básico (conceitual)

Passo 1. Inicie os pesos da rede de Kohonen entre a entrada e a grade. Estes pesos podem ser gerados randomicamente com valores compreendidos entre 0 e 1.

Passo 2. Normalize os vetores de pesos.

Passo 3. Normalize todos os vetores padrões de entrada.

Passo 4. Apresente um novo vetor padrão de entrada.

Passo 5. Calcule a distância ou o produto interno para todos os neurônios sobre a grade de Kohonen.

Passo 6. Encontre o neurônio vencedor (Equação (3) ou 4). Designar o neurônio vencedor pelo índice k.

Passo 7. Adaptar os vetores pesos do neurônio vencedor e dos demais neurônios contidos na vizinhança escolhida (vide Figura 4).

$$\mathbf{w}_i (h+1) = \mathbf{w}_i(h) + \alpha (\mathbf{x} - \mathbf{w}_i (h)). \quad (5)$$

$i \in NC_k$.

Passo 8. Renormalise todos os vetores de pesos adaptados no passo 7. Este procedimento se faz necessário tendo em vista que o vetor peso, após a adaptação, não possui comprimento unitário, sendo portanto, necessário torná-lo (com comprimento unitário).

Passo 9. Retornar ao Passo 4. Este procedimento deve ser repetido, considerando-se um número fixo de iterações, ou até que as variações (módulos) dos pesos sejam inferiores a uma certa tolerância preestabelecida.

NB:

- A auto-organização se dá à medida que aumente o número de ciclos de adaptação.
- Treinamento lento;
- Aumento da velocidade de treinamento:
  - Uso do conceito de consciência ([1]);
  - Uso de lógica nebulosa ([4]).

Referências Bibliográficas

[1] DeSieno, D. “Adding a Conscience to Competitive Learning”, *IEEE International*

Conference Neural Networks, San Diego, CA, 1987, vol. 1, pp. 117-124.

[2] Kohonen, T. “Self-organization and Associative Memory”, *Springer-Verlag*, 2nd Edition, Berling, Germany, 1989.

[3] Kohonen, T. “Self-organizing Map”, *Proceedings of IEEE*, 1990, pp. 1464 – 1480, vol 78, no. 09.

[4] Terano, T.; Asai, K. and Sugeno, M. Fuzzy Systems Theory and its Application, *Academic Press*, New York, USA, 1987.

UNIVERSIDADE ESTADUAL PAULISTA – UNESP
FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Rede Neural ART

Carlos Roberto Minussi

Ilha Solteira – SP, maio-2008.

10. Rede Neural ART (*Adaptive Resonance Theory*) Nebulosa

Características:

- Treinamento rápido;
- Treinamento não-supervisionado;
- Estabilidade / plasticidade.

NB. A representação dos vetores, nas redes neurais da família ART, é adotada por linha e não por coluna (como habitualmente feita).

Na Figura 1 apresenta-se a estrutura da rede neural ART nebulosa.

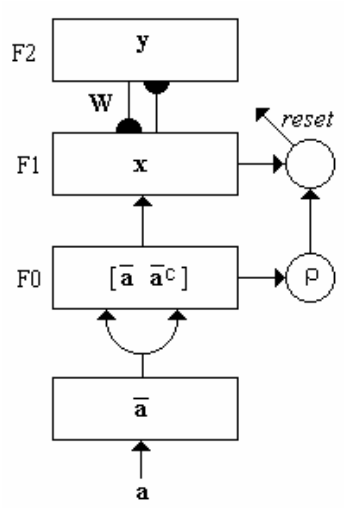


Figura 1. Rede neural ART Nebulosa.

A rede neural ART é composta por três camadas: F_0 (camada de entrada), F_1 (camada de comparação) e F_2 (camada de reconhecimento que armazena as categorias (*clusters*)). O algoritmo desta rede neural consiste, basicamente, nos seguintes passos (Carpenter et al, 1992):

Passo 1: Parâmetros da Rede

Os parâmetros utilizados no processamento da rede ART nebulosa são:

1. Parâmetro de Escolha : $\alpha > 0$;
2. Taxa de Treinamento : $\beta \in [0,1]$;
3. Parâmetro de Vigilância : $\rho \in [0,1]$.

Passo2: Dados de Entrada

Os dados de entrada são denotados pelo vetor $\mathbf{a} = [\ a_1\ a_2\ \ldots\ a_M\ ]$ M-dimensional. Este vetor é normalizado com o intuito de evitar a proliferação de categorias. Assim:

$$\bar{\mathbf{a}} = \frac{\mathbf{a}}{|\mathbf{a}|} \tag{1}$$

sendo:

$\bar{\mathbf{a}}$ = vetor de entrada normalizado;

$$|\mathbf{a}| = \sum_i a_i. \tag{2}$$

Passo 3: Codificação do vetor de entrada

A codificação de complemento é realizada para preservar a amplitude da informação, ou seja:

$$\bar{a}_i^{\,c} = 1 - \bar{a}_i \tag{3}$$

em que:

$\bar{\mathbf{a}}^{\,c}$ = vetor complementar de entrada normalizado.

Assim sendo, o vetor de entrada será um vetor 2M-dimensional, sendo denotado por:

$$\mathbf{I} = [\ \bar{\mathbf{a}}\ \bar{\mathbf{a}}^{\,c}\ ]$$

$$= [\ \bar{a}_1\ \bar{a}_2 \ldots \bar{a}_M\ \bar{a}_1^{\,c}\ \bar{a}_2^{\,c} \ldots \bar{a}_M^{\,c}\ ] \tag{4}$$

$|\mathbf{I}| = \sum_{i=1}^{M} \bar{a}_i + \sum_{i=1}^{M} \bar{a}_i{}^c = M$ (todos os vetores com normalização e codificação complementada terão mesmo comprimento M).

Passo 4: Vetor de Atividade

O vetor de atividade de F_2 é simbolizado por $\mathbf{y} = [\ y_1\ y_2\ \ldots\ y_N\ ]$, sendo N o número de categorias criadas em F_2. Deste modo, tem-se:

$$y_j = \begin{cases} 1, & \text{se o nó j de } F_2 \text{ é ativo} \\ 0 & \text{caso contrário} \end{cases} \tag{5}$$

Passo 5: Inicialização dos Pesos

Inicialmente todos os pesos possuem valor igual a 1, ou seja:

$$w_{j1}(0) = \ldots = w_{j2M}(0) = 1 \tag{6}$$

indicando que não existe nenhuma categoria ativa.

Passo 6: Escolha da categoria

Dado o vetor de entrada **I** em F_1, para cada nó j em F_2, a função de escolha T_j é determinada por:

$$T_j = \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|} \tag{7}$$

sendo:

$\wedge$ = operador AND nebuloso, definido por:

$(\mathbf{I} \wedge \mathbf{w})_i$ = min (Ii, wi). (8)

A categoria é escolhida como sendo o nó J ativo, ou seja:

$$J = \arg \max_{j=1,\ldots,N} T_j \tag{9}$$

Usando-se a equação (9), se existir mais de uma categoria ativa, a categoria

escolhida será aquela que possuir menor índice.

Passo 7: Ressonância ou *Reset*

A ressonância ocorre se o critério de vigilância (10) for satisfeito:

$$\frac{|\mathbf{I} \wedge \mathbf{w}_J|}{|\mathrm{I}|} \geq \rho \qquad (10)$$

Caso o critério definido pela equação (10) não seja satisfeito, ocorre o *reset*. No *reset*, o nó J de F_2 é excluído do processo de busca dado por (9), ou seja, $T_J = 0$. Então, é escolhida uma nova categoria através de (9) para o processo de ressonância. Este procedimento será realizado até que a rede encontre uma categoria que satisfaça (10).

Passo 8: Atualização dos Pesos (Treinamento)

Após o vetor de entrada **I** ter completado o estado de ressonância, segue o processo de treinamento, no qual ocorre a modificação do vetor peso dado por:

$$\mathbf{w}_J^{novo} = \beta\ (\mathbf{I} \wedge \mathbf{w}_J^{velho}) + (1-\beta)\ \mathbf{w}_J^{velho} \qquad (11)$$

sendo:

J = categoria ativa.

$\mathbf{w}_J^{novo}$ = vetor peso atualizado

$\mathbf{w}_J^{velho}$ = vetor peso referente à atualização anterior.

Se β = 1, tem-se o treinamento rápido.

Passo 9: Tomar um novo padrão e retornar ao Passo 6. Se não mais padrão, finalizar o processo de treinamento.

Rede Neural ARTMAP Nebulosa

A rede neural ARTMAP é uma arquitetura em que o treinamento é realizado de modo supervisionado e auto-organizável. Destinada-se à aproximação de funções não-lineares multidimensionais. Esta rede é composta por dois módulos ARTa e ARTb possuem a mesma estrutura da rede neural ART descrita anteriormente, exceto quando uma vigilância básica é usada para controlar o sistema. O módulo Inter-ART é responsável pela verificação se há casamento da entrada (ART_a) e da saída (ART_b). As matrizes pesos associadas aos módulos ART_a ($\mathbf{w}^a$) e ART_b ($\mathbf{w}^b$), assim, semelhantes ao módulo Inter-ART ($\mathbf{w}^{ab}$), são iniciadas com valores iguais a 1, ou seja, todas as atividades encontram-se inativas. Estas atividades são ativadas à medida que ocorre ressonância entre os padrões de entrada e de saída. Toda vez que os pares de entrada (**a**, **b**), associados aos módulos ART_a e ART_b, são confirmados (as entradas **a** e **b** referem-se às categorias J e K ativas, respectivamente), de acordo com o teste do *match tracking*:

$$\left|\mathbf{x}^{ab}\right|_i = \frac{\left|\mathbf{y}_i \wedge \mathbf{w}^{ab_J}\right|}{\left|\mathbf{y}_i\right|} \tag{12}$$

$\left|\mathbf{x}^{ab}\right|_i \geq \rho ab \rightarrow$ OK, o par de treinamento deve ser confirmado nas matrizes de pesos com índices J e K.

$\left|\mathbf{x}^{ab}\right|_i < \rho ab \rightarrow$ deve-se buscar um outro índice J (com relação aos vetores de entrada **a**) vetor, até que o critério seja satisfeito.

Os pesos $\mathbf{w}^a$, $\mathbf{w}^b$ e $\mathbf{w}^{ab}$ devem ser adaptados usando-se a Eq. (11) e:

$$\mathbf{w}^b{}_K{}^{novo} = \beta\,(\mathbf{I} \wedge \mathbf{w}^b{}_K{}^{velho}) + (1-\beta)\,\mathbf{w}^b{}_K{}^{velho} \tag{13}$$

$$w^{ab}{}_{JK}{}^{novo} = 1. \tag{14}$$

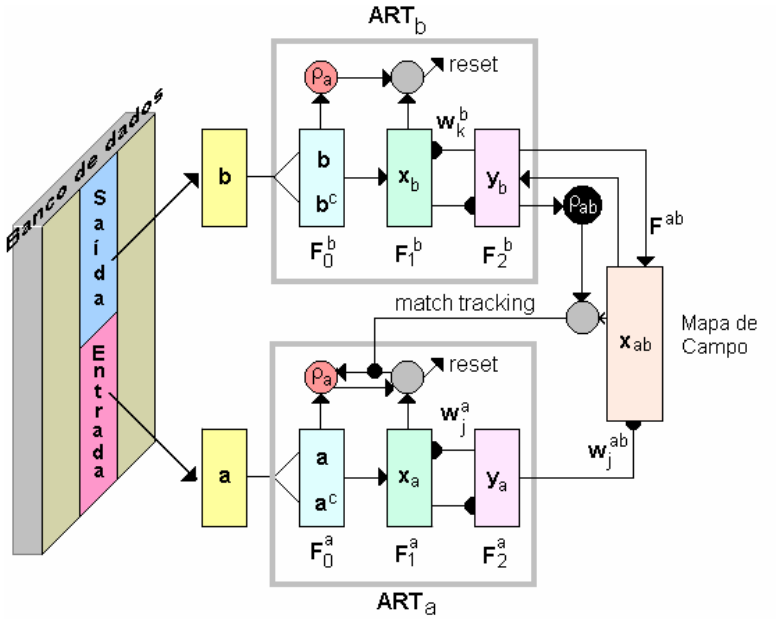


Figura 2. Rede neural ARTMAP nebulosa.

Características:

- Treinamento rápido;
- Treinamento supervisionado;
- Estabilidade / plasticidade.

Exemplo

Rede Neural ARTMAP Nebulosa

(Treinamento)

Considere 3 pares de vetores como mostrados na Figura 3 e Tabela 1.

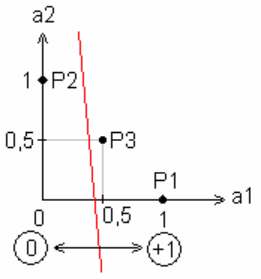


Figura 3.

Tabela 1. Dados de entrada e de saída para o treinamento da rede neural ARTMAP nebulosa.

Padrão	Entrada	Saída
P1	[1 0]	[1]
P2	[0 1]	[0]
P3	[0,5 0,5]	[1]

Deve observar que tais vetores não necessitam ser normalizados, visto que atendem o espectro de variação (0, 1). Assim sendo, na Tabela 2 são apresentados os vetores de entrada com codificação de complemento.

Tabela 2. Dados de entrada e de saída para o treinamento da rede neural ARTMAP nebulosa com codificação de complemento.

Padrão	Entrada com complemento (**I**a)	Saída com complemento (**I**b)
P1	[1 0 0 1]	[1 0]
P2	[0 1 1 0]	[0 1]
P3	[0,5 0,5 0,5 0,5]	[1 0]

Definição de parâmetros:

Ma = número de componentes dos vetores de entrada
= 4

Mb = número de componentes dos vetores de saída
= 2

N = número de neurônios
= 3

Arbitra-se Na = Nb = 3 (igual ao número de pares de padrões).

Dados:

$\rho a = 0,95$

$\rho b = 1$

$\rho ab = 0,95$

α = 0,1

β = 1.

Iniciação de pesos:

$\mathbf{W}a(0) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$, dimensão = (Na x Ma)

$\mathbf{W}b(0) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$, dimensão = (Nb x Mb)

$Wab(0) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$, dimensão = (Na x Nb).

Treinamento

NB. A ordem de apresentação dos pares de vetores para o treinamento pode ser adotada seqüencialmente (1, 2, 3, etc.) ou por ordem aleatória. A ordem aleatória (ou pseudo-aleatória) é preferível, tendo em vista ser mais plausível do ponto de vista mental. Contudo, neste exemplo, ir-se-á adotar a ordem seqüencial como forma ilustrativa.

Para o Par de Treinamento 1

ARTa

a1 = [1 0 0 1]

$Tj = \dfrac{\left| a1 \wedge wj^a \right|}{\alpha + \left| wj^a \right|}$

$T1 = \dfrac{\left| [1\,0\,0\,1] \wedge [1\,1\,1\,1] \right|}{0{,}1 + \left| [1\,1\,1\,1] \right|}$

ARTb

b1 = [1 0]

$Tk = \dfrac{\left| b1 \wedge wk^b \right|}{\alpha + \left| wk^b \right|}$

$T1 = \dfrac{\left| [1\,0] \wedge [1\,1] \right|}{0{,}1 + \left| [1\,1] \right|}$

$= \frac{|[1\ 0\ 0\ 1\ ]|}{0{,}1 + 4}$

$= \frac{2}{4{,}1}$

$= 0{,}4878$

T2 = 0,4878

T3 = 0,4878

Índices de ordenação: (1, 2, 3)

Escolhe-se o de menor índice: J = 1

$= \frac{|[1\ 0\ ]|}{0{,}1 + 2}$

$= \frac{1}{2{,}1}$

$= 0{,}4776$

T2 = 0,4776

T3 = 0,4776

Índices de ordenação: (1, 2, 3)

Escolhe-se o de menor índice: K = 1

Teste de Vigilância:

$|x^a| = \frac{|a1 \wedge w1^a|}{|a1|}$

$|x^a| = \frac{|[1\ 0\ 0\ 1\ ] \wedge [\ 1\ 1\ 1\ 1\ ]|}{|[1\ 0\ 0\ 1\ ]|}$

$= \frac{|[1\ 0\ 0\ 1\ ]|}{|[1\ 0\ 0\ 1\ ]|}$

$|x^a| = 1 \geq \rho a \rightarrow$ OK!,

índice J = 1 (ativado).

Teste de Vigilância:

$|x^b| = \frac{|b1 \wedge w1b|}{|b1|}$

$|x^b| = \frac{|[1\ 0\ ] \wedge [\ 1\ 1\ ]|}{|[1\ 0\ ]|}$

$= \frac{|[1\ 0\ ]|}{|[1\ 0\ ]|}$

$|x^b| = 1 \geq \rho b \rightarrow$ OK!,

índice K = 1 (ativado).

Match Traking:

$|x^{ab}| = \frac{|y^a{}_J \wedge w_J{}^{ab}|}{|y_J|}$

$|x^{ab}| = \frac{|[1\ 0\ 0\ ] \wedge [\ 1\ 1\ 1]|}{|[1\ 0\ 0\ ]|}$

$= \frac{|[1\ 0\ 0\ ]|}{|[1\ 0\ 0\ ]|}$

$|x^{ab}| = 1 \geq \rho ab \rightarrow$ OK!. $\rightarrow$ unidades (J = 1, K = 1) devem ser confirmados (houve ressonância)

Atualização de Pesos:

$W_J^{a\,(novo)} = a1 \wedge W_J^{a\,(velho)}$ (T. rápido)

$W_1^{a\,(novo)} = a1 \wedge W_1^{a\,(velho)}$

$= [1\ 0\ 0\ 1\ ] \wedge [\ 1\ 1\ 1\ 1\ ]$

$= [\ 1\ 0\ 0\ 1]$

$W_{JK}^{ab\,(novo)} = 1,\ W_{Jk}^{ab\,(novo)} = 0$ para $k \neq K$

$W_{JK}^{ab\,(novo)} = [1\ 0\ 0\ ]$

Atualização de Pesos:

$W_K^{b\,(novo)} = b1 \wedge W_K^{b\,(velho)}$

$W_1^{b\,(novo)} = b1 \wedge W_1^{b\,(velho)}$

$= [1\ 0\ ] \wedge [\ 1\ 1\ ]$

$= [1\ 0\ ]$

Matrizes de Pesos Atualizadas:

$$\mathbf{W}a = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix},$$

$$\mathbf{W}b = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \end{bmatrix},$$

$$Wab = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Para o Par de Treinamento 2

ARTa

$a2 = [\ 0\ 1\ 1\ 0\ ]$

$$Tj = \frac{\left| a2 \wedge wj^a \right|}{\alpha + \left| wj^a \right|}$$

$$T1 = \frac{\left|[0\ 1\ 1\ 0\ ] \wedge [1\ 0\ 0\ 1\ ]\right|}{0{,}1 + \left|[\ 1\ 0\ 0\ 1]\right|}$$

$$= \frac{\left|[0\ 0\ 0\ 0\ ]\right|}{0{,}1 + 2}$$

ARTb

$b2 = [\ 0\ 1\ ]$

$$Tk = \frac{\left| b2 \wedge wk^b \right|}{\alpha + \left| wk^b \right|}$$

$$T1 = \frac{\left|[0\ 1\ ] \wedge [1\ 0\ ]\right|}{0{,}1 + \left|[\ 1\ 0\ ]\right|}$$

$$= \frac{\left|[0\ 0\ ]\right|}{0{,}1 + 1}$$

$$= \frac{0}{2{,}1}$$

$$= 0$$

$$T2 = \frac{\left|[0\,1\,1\,0\,] \wedge [1\,1\,1\,1\,]\right|}{0{,}1 + \left|[\,1\,1\,1\,1]\right|}$$

$$= \frac{\left|[0\,1\,1\,0\,]\,\right|}{0{,}1 + \left|[\,1\,1\,1\,1]\right|} =$$

$$= \frac{2}{4{,}1}$$

$$= 0{,}4878$$

$$T3 = 0{,}4878$$

Índices de ordenação: (2, 3, 1)

Escolhe-se o de menor índice: J = 2

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

$$= 0$$

$$T2 = \frac{\left|[0\,1\,\,] \wedge [1\,1\,]\right|}{0{,}1 + \left|[\,1\,1\,]\right|}$$

$$= \frac{\left|[0\,1\,]\,\right|}{0{,}1 + \left|[\,1\,1\,]\right|}$$

$$= \frac{1}{2{,}1}$$

$$= 0{,}4761$$

$$T3 = 0{,}4761$$

Índices de ordenação: (2, 3, 1)

Escolhe-se o de menor índice: K = 2

Teste de Vigilância:

$$\left|x^a\right| = \frac{\left|a2 \wedge w2^a\right|}{\left|\,a2\,\right|}$$

$$\left|x^a\right| = \frac{\left|[0\,1\,1\,0\,] \wedge [\,1\,1\,1\,1\,]\right|}{\left|\,[0\,1\,1\,0\,]\,\right|}$$

$$= \frac{\left|[0\,1\,1\,0\,]\,\right|}{\left|[0\,1\,1\,0\,]\,\right|}$$

$$\left|x^a\right| = 1 \geq \rho a \rightarrow \text{OK!},$$

índice J = 2 (ativado).

Teste de Vigilância:

$$\left|x^b\right| = \frac{\left|b2 \wedge w2b\right|}{\left|\,b2\,\right|}$$

$$\left|x^b\right| = \frac{\left|[0\,1\,\,] \wedge [\,1\,1\,]\right|}{\left|\,[0\,1\,]\,\right|}$$

$$= \frac{\left|[0\,1\,]\,\right|}{\left|[0\,1\,\,]\,\right|}$$

$$\left|x^b\right| = 1 \geq \rho b \rightarrow \text{OK!},$$

índice K = 2 (ativado).

Match Traking:

$$\left|x^{ab}\right| = \frac{\left|y^a{}_J \wedge w_J{}^{ab}\right|}{\left|\,y_J\,\right|}$$

$$\left|x^{ab}\right| = \frac{\left|[0\,1\,0\,] \wedge [\,1\,1\,1]\right|}{\left|\,[0\,1\,0\,]\,\right|}$$

$$= \frac{\left|[0\ 1\ 0\ ]\right|}{\left|[0\ 1\ 0\ ]\right|}$$

$\left|x^{ab}\right| = 1 \geq \rho ab \rightarrow$ OK!. $\rightarrow$ unidades (J = 2, K = 2) devem ser confirmados (houve ressonância)

Atualização de Pesos:

$W_2^{a\ (novo)} = a2 \wedge W_2^{a\ (velho)}$

$= [0\ 1\ 1\ 0\ ] \wedge [\ 1\ 1\ 1\ 1\ ]$

$= [\ 0\ 1\ 1\ 0]$

$W_{JK}^{ab\ (novo)} = 1,\ W_{Jk}^{ab\ (novo)} = 0$ para $k \neq K$

$W_{JK}^{ab\ (novo)} = [0\ 1\ 0\ ]$

Atualização de Pesos:

$W_2^{b\ (novo)} = b2 \wedge W_2^{b\ (velho)}$

$= [0\ 1\ ] \wedge [\ 1\ 1\ ]$

$= [0\ 1\ ]$

Matrizes de Pesos Atualizadas:

$$\mathbf{W}a = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix},$$

$$\mathbf{W}b = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix},$$

$$Wab = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Para o Par de Treinamento 3

ARTa

$a3 = [\ 0{,}5\ \ 0{,}5\ \ 0{,}5\ \ 0{,}5\ ]$

$$Tj = \frac{\left|a3 \wedge wj^a\right|}{\alpha + \left|wj^a\right|}$$

ARTb

$b3 = [\ 1\ 0\ ]$

$$Tk = \frac{\left|b3 \wedge wk^b\right|}{\alpha + \left|wk^b\right|}$$

$$T1 = \frac{\left|[0{,}5\ \ 0{,}5\ \ 0{,}5\ \ 0{,}5\ ] \wedge [1\ 0\ 0\ 1\ ]\right|}{0{,}1 + \left|[\ 1\ 0\ 0\ 1]\right|}$$

$$= \frac{\left|[\,0{,}5\ \ 0\ \ 0\ \ 0{,}5\ ]\right|}{0{,}1 + 2}$$

$$= \frac{1}{2{,}1}$$

$$= 0{,}4761$$

$$T2 = \frac{\left|[0{,}5\ \ 0{,}5\ \ 0{,}5\ \ 0{,}5] \wedge [0\ 1\ 1\ 0\ ]\right|}{0{,}1 + \left|[\ 0\ 1\ 1\ 0]\right|}$$

$$= \frac{\left|[0\ \ 0{,}5\ \ 0{,}5\ \ 0\ ]\ \right|}{0{,}1 + \left|[\ 0\ 1\ 1\ 0]\right|} =$$

$$= \frac{1}{2{,}1}$$

$$= 0{,}4761$$

$$T3 = \frac{\left|[0{,}5\ \ 0{,}5\ \ 0{,}5\ \ 0{,}5] \wedge [1\ 1\ 1\ 1\ ]\right|}{0{,}1 + \left|[\ 1\ 1\ 1\ 1]\right|}$$

$$= \frac{\left|[0{,}5\ \ 0{,}5\ \ 0{,}5\ \ 0{,}5]]\right|}{0{,}1 + \left|[\ 1\ 1\ 1\ 1]\right|}$$

$$= \frac{2}{4{,}1}$$

$$= 0{,}4878$$

Índices de ordenação: (3, 1, 2)

Escolhe-se o de menor índice: J = 3

$$T1 = \frac{\left|[1\ 0\ ] \wedge [1\ 0\ ]\right|}{0{,}1 + \left|[\ 1\ 0\ ]\right|}$$

$$= \frac{\left|[1\ 0\ ]\right|}{0{,}1 + 1}$$

$$= \frac{1}{1{,}1}$$

$$= 0{,}9090$$

$$T2 = \frac{\left|[1\ 0\ \ ] \wedge [0\ 1\ ]\right|}{0{,}1 + \left|[\ 0\ 1\ ]\right|}$$

$$= \frac{\left|[0\ 0\ ]\ \right|}{0{,}1 + \left|[\ 0\ 1\ ]\right|}$$

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

$$= 0$$

$$T3 = \frac{\left|[1\ \ 0] \wedge [1\ 1]\right|}{0{,}1 + \left|[\ 1\ 1]\right|}$$

$$= \frac{\left|[1\ \ 0]\ \right|}{0{,}1 + \left|[\ 1\ 1]\right|}$$

$$= \frac{1}{2{,}1}$$

$$= 0{,}4761$$

Índices de ordenação: (1, 3, 2)

Escolhe-se o de menor índice: K = 1

Teste de Vigilância:

$$\left|x^a\right| = \frac{\left|a3 \wedge w3^a\right|}{\left|\ a3\ \right|}$$

$$\left|x^a\right| = \frac{\left|[0{,}5\ \ 0{,}5\ \ 0{,}5\ \ 0{,}5\ ] \wedge [\,1\ 1\ 1\ 1\ ]\right|}{\left|\,[0{,}5\ \ 0{,}5\ \ 0{,}5\ \ 0{,}5\ ]\,\right|}$$

Teste de Vigilância:

$$\left|x^b\right| = \frac{\left|b3 \wedge w1b\right|}{\left|\ b1\ \right|}$$

$$\left|x^b\right| = \frac{\left|[1\ 0\ \ ] \wedge [\,1\ 0\ ]\right|}{\left|\,[1\ 0]\,\right|}$$

$$= \frac{\left|[0{,}5\ \ 0{,}5\ \ 0{,}5\ \ 0{,}5]\right|}{\left|[0{,}5\ \ 0{,}5\ \ 0{,}5\ \ 0{,}5\ ]\right|} \qquad = \frac{\left|[1\ 0\ ]\right|}{\left|[1\ 0\ \ ]\right|}$$

$\left|x^a\right| = 1 \geq \rho a \rightarrow$ OK!, índice J = 3 (ativado).

$\left|x^b\right| = 1 \geq \rho b \rightarrow$ OK!, índice K = 1

Match Traking:

$$\left|x^{ab}\right| = \frac{\left|y^a{}_J \wedge w_J{}^{ab}\right|}{\left|y_J\right|}$$

$$\left|x^{ab}\right| = \frac{\left|[0\ 0\ 1\ ] \wedge [\ 1\ 1\ 1]\right|}{\left|[0\ 0\ 1\ ]\right|}$$

$$= \frac{\left|[0\ 0\ 1\ ]\right|}{\left|[0\ 0\ 1\ ]\right|}$$

$\left|x^{ab}\right| = 1 \geq \rho ab \rightarrow$ OK!. $\rightarrow$ unidades (J = 3, K = 1) devem ser confirmados (houve ressonância)

Atualização de Pesos:

$W_3{}^{a\ (novo)} = a3 \wedge W_3{}^{a\ (velho)}$

$= [0{,}5\ 0{,}5\ 0{,}5\ 0{,}5] \wedge [\ 1\ 1\ 1\ 1\ ]$

$= [\ 0{,}5\ 0{,}5\ 0{,}5\ 0{,}5]$

$W_{JK}{}^{ab\ (novo)} = 1,\ W_{Jk}{}^{ab\ (novo)} = 0$ para $k \neq K$

$W_{JK}{}^{ab\ (novo)} = [1\ 0\ 0\ ]$

Atualização de Pesos:

$W_1{}^{b\ (novo)} = b3 \wedge W_1{}^{b\ (velho)}$

$= [1\ 0\ ] \wedge [\ 1\ 0\ ]$

$= [1\ 0\ ]$

Matrizes de Pesos Atualizadas:

$$\mathbf{W}a = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 \end{bmatrix},$$

$$\mathbf{W}b = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}, \leftarrow \text{(unidade não-ativada)}$$

$$Wab = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Assim sendo, na Figura 4, apresenta-se o esquema resultante da rede neural ARTMAP para o exemplo apresentado.

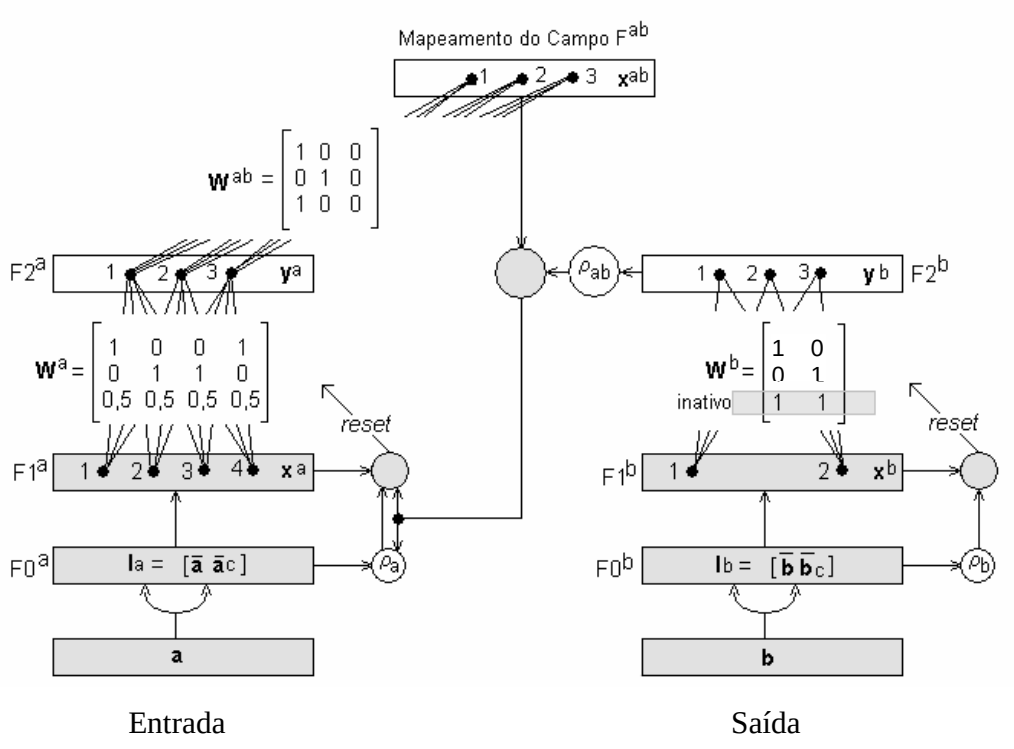


Figura 4. Rede neural ARTMAP nebulosa treinada considerando-se o exemplo mostrado na Figura 3.

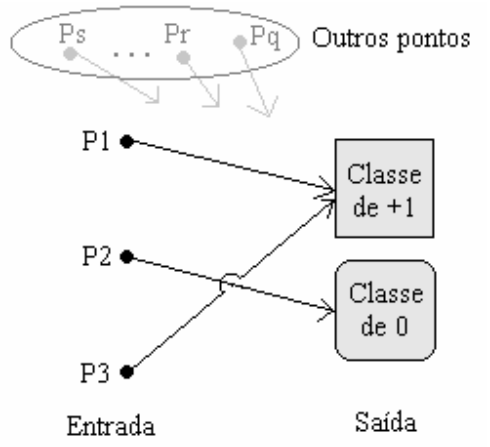


Figura 5. Formação de classes.

Na Figura 6 relacionam-se as principais redes neurais da família ART disponíveis na literatura especializada. A rede neural ART constitui numa arquitetura básica (análoga à Figura 1 (ART nebulosa)). Assim, várias outras redes neurais ART descendentes podem ser geradas a partir do módulo básico. Alguma destas redes estão destacadas na Figura 6. Contudo, outras tanto podem ser perfeitamente concebidas, bastando apenas incluir alguma inovação.

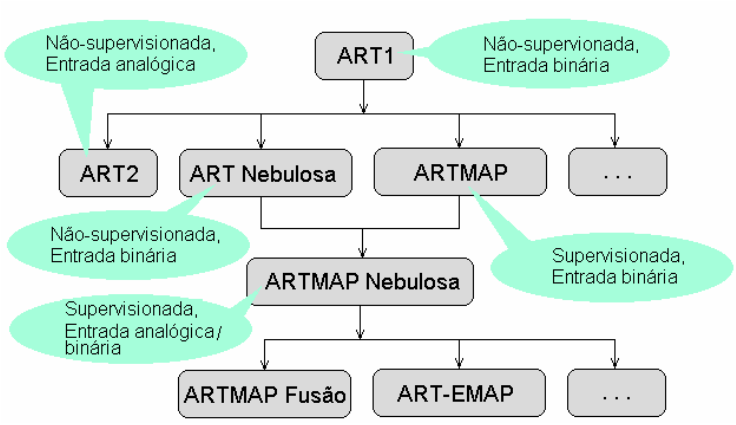


Figura 6. Redes neurais da família ART.

O Efeito do Parâmetro de Vigilância

As redes neurais ART são muito sensíveis a variações de seus parâmetros durante o treinamento. O parâmetro mais crítico é o parâmetro de vigilância (ρ) que controla a resolução do processo de classificação. Se ρ assume um valor baixo, *e.g.*, a rede permite que padrões poucos semelhantes sejam agrupados na mesma categoria de reconhecimento, criando poucas classes. Se para ρ for atribuído um valor alto (próximo de 1), pequenas variações nos padrões de entrada levarão à criação de novas classes. Este efeito pode ser ilustrado na Figura 7.

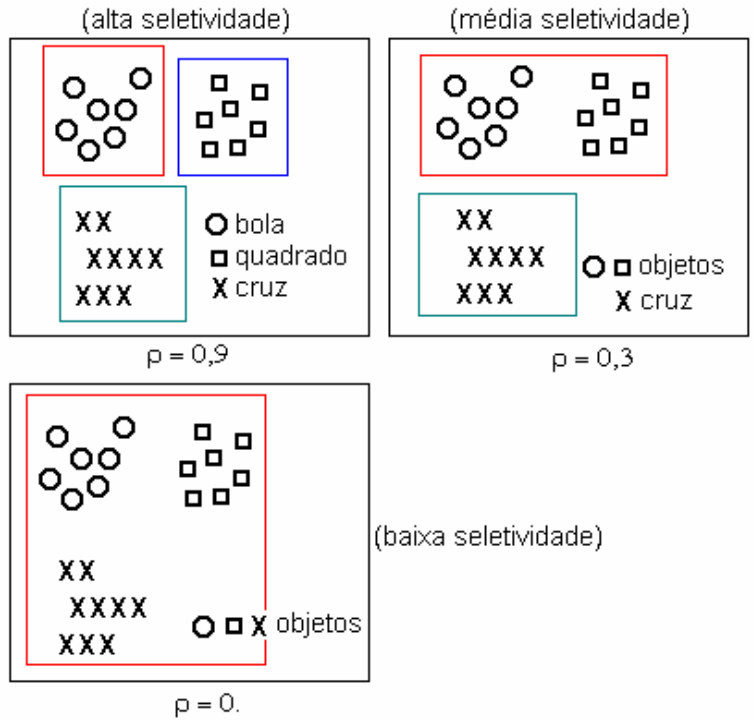


Figura 7. Quadros ilustrativos do efeito do parâmetro de vigilância ρ no treinamento da rede neural ART.

Referências

[1] Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H. and Rosen D. B. (1992). “Fuzzy ARTMAP: A Neural Network Architecture For Incremental Supervised Learning of Analog Multidimensional Maps”, IEEE Transactions on Neural Networks, 3(5): 698-713.

[2] Carpenter, G. A. and Grossberg, S. (1992). “A Self-organizing Neural Network For Supervised Learning, Recognition And Prediction”, IEEE Communications Magazine, 38-49.

Rede Neural ELEANNE (“Efficient LEarning Algorithms for Neural Networks”)

- Técnica de treinamento de redes neurais *feedforward* multicamada proposta Karayiannis & Venetsanopoulos.
- Esta rede neural, via de regra, é composta por duas camadas (de entrada e de saída). Os neurônios da camada de saída possuem, como função de ativação, uma função linear, conforme é ilustrado na Figura 1.

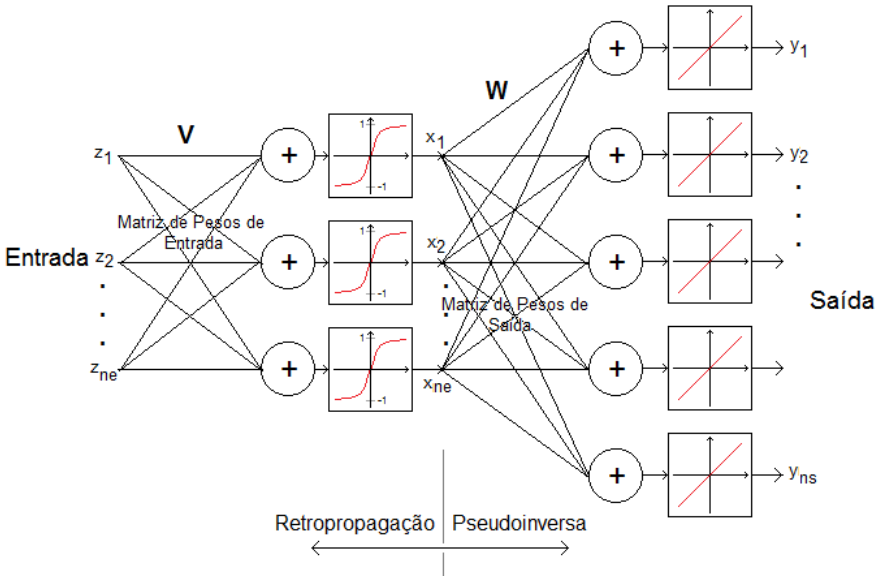


Figura 1. Rede neural *feedforward* multicamadas com treinamento vias algoritmos e retropropagação e pseudoinversa.

- O treinamento dos neurônios da camada de entrada é realizado usando o algoritmo retropropagação (ajuste dos pesos referentes à matriz **V**), enquanto que os de saída são treinados baseado no uso da matriz pseudoinversa (ajuste da matriz **W**).
- Considerando-se a camada de saída rede neural *feedforward* multicamadas mostrada na Figura 2:

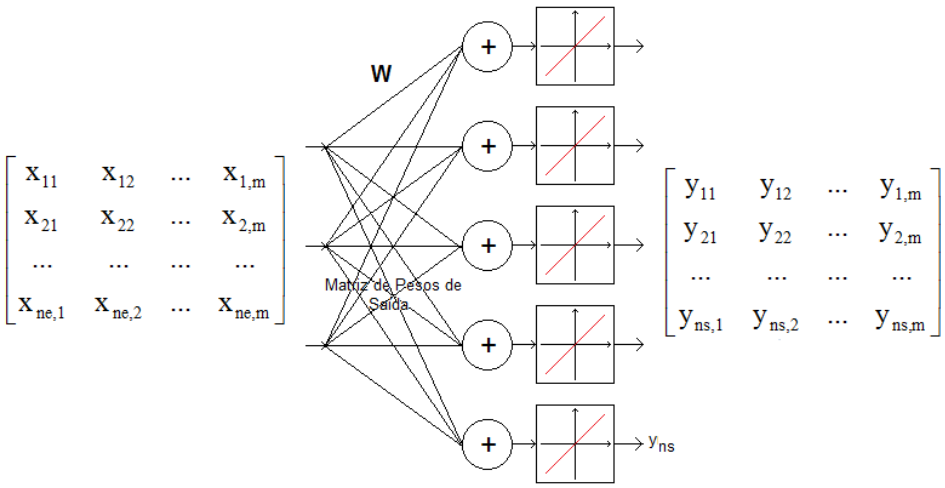


Figura 2. Camada de saída da rede neural *feedforward* multicamadas.

- Assim, o p-ésimo componente, do vetor de saída (com função de ativação linear), referente ao k-ésimo vetor padrão é dado por:

$$y_{p,k} = \mathbf{W}_k^T \mathbf{x}_k, \forall k, k = 1, 2, \ldots, m \quad (1)$$

sendo:

$\mathbf{y}k$: $[y_{1,k}\ y_{2,k} \ldots y_{ns,k}]^T$, k = 1, 2, . . ., m

$\mathbf{x}_k$ = $[x_{1,k}\ x_{2,k} \ldots x_{ne,k}]^T$, k = 1, 2, . . . m

$\mathbf{W}_k$: k-coluna da matriz **W**;

ns : número de componentes de **x**;

ns : número de componentes de **y**;

m : número de pares de padrões $\{\mathbf{x}_k, \mathbf{y}_k\}$, para k = 1, 2, ... m.

- Combinando todos os m pares de $(\mathbf{x}_k, \mathbf{y}_k)$, obtém-se o seguinte sistema:

$$\mathbf{Y}_m = \mathbf{W}^T \mathbf{X}_m \quad (2)$$

$$= \left[\mathbf{W}^T\mathbf{x}_1 \quad \mathbf{W}^T\mathbf{x}_2 \quad \ldots \quad \mathbf{W}^T\mathbf{x}_m\right]$$

sendo:

$$\mathbf{X}_m = [\mathbf{x}_1\ \mathbf{x}_2 \ldots \mathbf{x}_m]$$

$$= \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1,m} \\ x_{21} & x_{22} & \ldots & x_{2,m} \\ \ldots & \ldots & \ldots & \ldots \\ x_{ne,1} & x_{ne,2} & \ldots & x_{ne,m} \end{bmatrix}$$

$\mathbf{x}_2$

$$\mathbf{Y}_m = [\mathbf{y}_1\ \mathbf{y}_2 \ldots \mathbf{y}_m]$$

$$= \begin{bmatrix} y_{11} & y_{12} & \ldots & y_{1,m} \\ y_{21} & y_{22} & \ldots & y_{2,m} \\ \ldots & \ldots & \ldots & \ldots \\ y_{ns,1} & y_{ns,2} & \ldots & y_{ns,m} \end{bmatrix}$$

$\mathbf{y}_2$

A solução ótima da equação matricial (2), no senso da norma mínima, é dada por:

$$\mathbf{W}^T = \mathbf{Y}_m \mathbf{X}_m^+ \quad (3)$$

sendo:

$\mathbf{X}_m^+$: matriz pseudoinversa de $\mathbf{X}_m$,

ou seja:

pós-multiplicando-se a equação (2) por $\mathbf{X}_m^T$, obtém-se:

$$\mathbf{W}^T \mathbf{X}_m \mathbf{X}_m^T = \mathbf{Y}_m \mathbf{X}_m^T \quad (4)$$

$$\mathbf{W}^T = \mathbf{Y}_m \mathbf{X}_m^+ \quad (5)$$

sendo:

$\mathbf{X}_m^+ = \mathbf{X}_m^T (\mathbf{X}_m \mathbf{X}_m^T)^{-1}$ (matriz pseudoinversa de $\mathbf{X}_m$). (6)

dim{ $\mathbf{X}_m \mathbf{X}_m^T$ } = ne x ne.

NB. A matriz pseudoinversa $\mathbf{X}_m^+$ existirá, se e somente se, m $\geq$ ne. Contudo, isto não garante que ela existe, ou seja, é condição necessária, porém não-suficiente.

- Fazendo:

$$\mathbf{P}m = (\mathbf{X}m\ \mathbf{X}m^T)^{-1} \quad (7)$$

Pode-se obter **P**m, de forma recursiva via lema de inversão de matrizes.

Referência

[1] Karayiannis, N. B. and Venetsanopoulos, A. N. “Efficient Learning Algorithms for Neural Networks (ELEANNE)”,IEEE Transactions on Systems, Man, And Cybernetics. Vol. 23. No. 5, September/October-1993.

MEL – Máquina no Estado Líquido

(LSM - “Liquid State Machine”)

- Autoria de Maass [3]
- Concepção baseada a partir de modelo de processamento de entradas de fluxos contínuos e dinâmicos, por exemplo, os meios líquidos (uma piscina ou uma lagoa);
- **Analogia** (vide Figura 1):

  Uma pedra (entrada) e um meio líquido (estrutura extremamente complexa sendo responsável por grande parte do processamento). O lançamento da pedra na água produz respingos e ondulações no meio líquido caracterizando uma reação do meio líquido. Essas ondulações se propagam na superfície do meio líquido durante certo tempo (durante a fase transitória). Essas ondulações interagirão com outras ondulações. Elas podem ser registradas (por exemplo, por uma câmera acoplada a um microcomputador). A câmera associa as imagens obtidas a um padrão de comportamento das ondas. A MEL, no meio líquido, transforma uma entrada temporal (de baixa dimensão) em estados líquidos (de alta dimensão). A unidade de leitura (*readout*) é uma função que interpreta os estados líquidos em um tempo *t*, gerando uma saída.

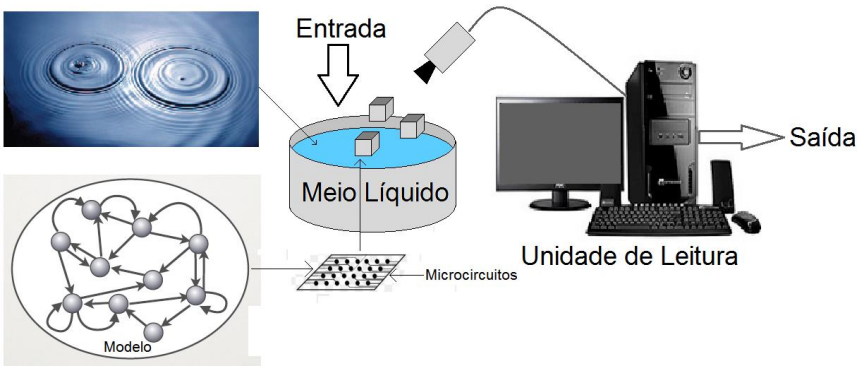


Figura 1. Analogia de uma MEL.

- Uma Rede Neural pode ser considerada como piscinas de neurônios que compõem o meio líquido.
- Elas necessitam de uma codificação capaz de explorar a habilidade das redes neurais naturais (microcircuitos corticais), ao mesmo tempo em que devem ser capazes de aprender suas aptidões biológicas.
- Maass [3] introduziu o conceito de neurônio pulsado, permitindo que a MEL realize um processamento em tempo real.

- Uma entrada de fluxo contínuo u(.) é inserida no meio líquido L^M (vide Figuras 2 e 3). Esse meio líquido atua como um "filtro de líquidos" produzindo o estado líquido $X^M(t)$ (perturbações) para cada instante t. Os estados líquidos são mapeados produzindo a função saída y(t) (padrão de imagem reconhecido ao se visualizar a imagem) por meio da função f^M (Câmera).

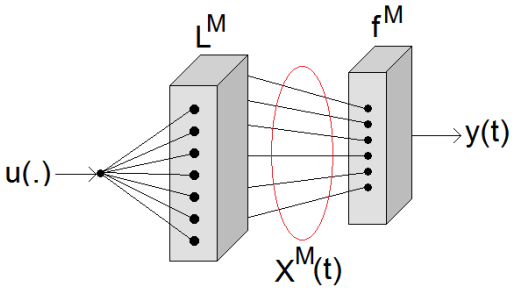


Figura 2. Arquitetura da MEL.

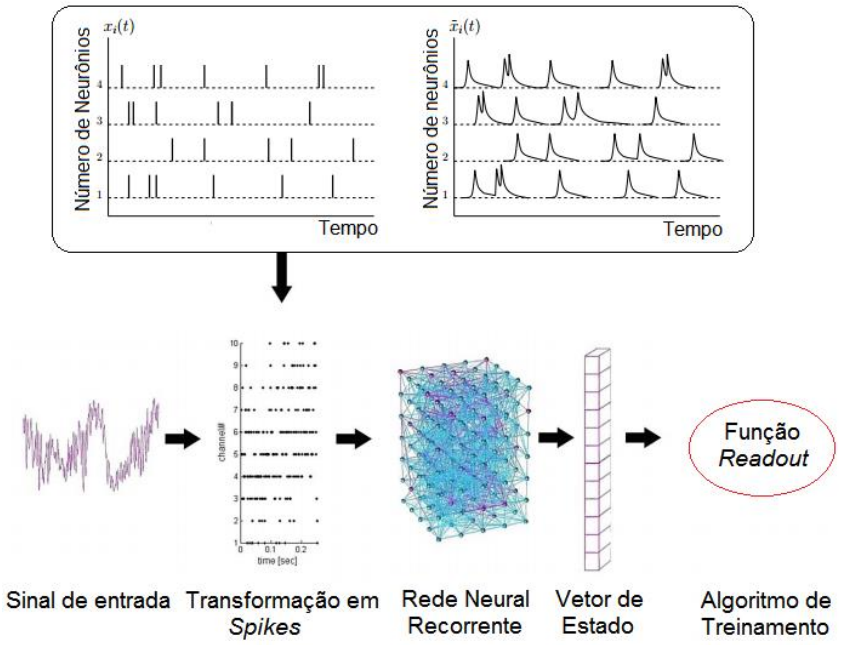


Figura 3.

- o meio líquido pode ser representado em um espaço tridimensional, contendo uma ou mais piscinas de neurônios. As piscinas são compostas por vários neurônios.
- A unidade de leitura possui uma estrutura extremamente simples, se comparada à organização do meio líquido. Entretanto, sua função é de grande relevância, uma vez que é responsável pela análise dos dados de entrada, os quais são pré-processados no meio líquido. Assim, a unidade de leitura pode ser implementada por sistemas simples de classificação como, por exemplo, uma rede neural *perceptron* multicamada com treinamento via técnica retropropagação (*backpropagation*).
- Portanto, toda a complexidade está alocada na interpretação do meio líquido.

- Como dado de entrada, o meio líquido recebe um ou mais fluxos contínuos, gerando como saída vários estados líquidos, os quais são dependentes da topologia das piscinas de neurônios.

- **Algoritmo Sucinto**

1. Definir o microcircuito neural a ser analisado.
2. Gravar os estados $x(t)$ do microcircuito em vários pontos no tempo em resposta para diferentes entradas $[u(.)]$
3. Aplicar um algoritmo de aprendizagem supervisionada a um conjunto de exemplos de treinamento da forma $[x(t), y(t)]$ para formar uma função de leitura (*readout*) f tal que as saídas $f(x(t))$ fiquem tão perto quanto possível das saídas alvo $y(t)$.

Reservoir Computing

- O conceito *Reservoir Computing* (RC) tem sido desenvolvido de forma independente por Maass et al. [3] (*Liquid State Machine*) e por Jaeger [1] (*Echo State Network*) como um procedimento alternativo a análise de séries temporais usando Redes Neurais Recorrentes (RNR).
- A RC difere de uma Rede Neural *Feedforward* Multicamadas (RNFM), ou seja, a RNFM o ajuste de pesos é realizado considerando-se todos os neurônios de todas as camadas, enquanto que na RC (Figura 4) o ajuste de peso é realizado somente nos neurônios da camada de saída.

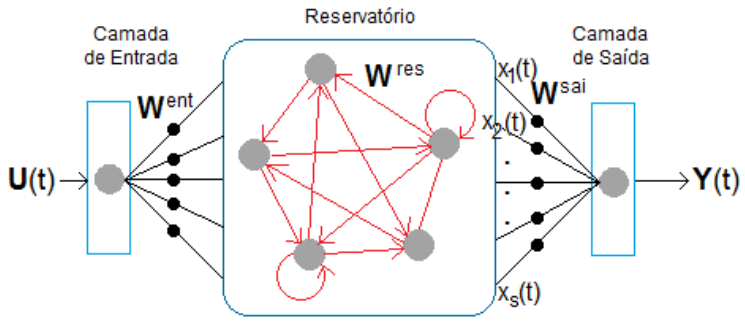


Figura 4. RC.

- No RC somente os pesos $\mathbf{W}^{sai}$ são ajustados. As matrizes de pesos $\mathbf{W}^{ent}$ e $\mathbf{W}^{res}$ são calculados.
- As matrizes de pesos $\mathbf{W}^{ent}$ e $\mathbf{W}^{res}$ são geradas randomicamente.
- A evolução de cada reservatório (estado líquido) é determinada da seguinte forma:

$xj(t+1) = f(\mathbf{W}j^{res}\ \mathbf{X}(t) + \mathbf{W}^{ent}\ \mathbf{U}(t))$

sendo:

f : função tangent hiperbólica;

$\mathbf{U}(t) = [u_1(t)\ u_2(t)\ \ldots\ u_e(t)]^T$

e : número de componentes do vetor **U**(t);

s : número de componentes do vetor estado líquido **X**(t).

- A saída da rede neural (considerando-se a função de ativação linear) é dada por:

$$\mathbf{Y}(t) = \mathbf{w}b + \mathbf{W}^{sai}\,\mathbf{X}(t) \qquad (1)$$

- O sistema é treinado (pesos da última camada ($\mathbf{W}^{sai}$)) visando minimizar o erro da saída de rede neural, ou seja:

$$\text{Min } \{\text{erro} = \| \mathbf{Y}(t) - \underline{\mathbf{Y}}(t) \|\} \qquad (2)$$

sendo:

$\underline{\mathbf{Y}}(t)$: vetor saída desejada.

- A solução do problema (2), considerando-se a equação (1), pode ser obtida via emprego do método da pseudoinversa matricial:

$$\mathbf{W}^{sai} = (\mathbf{X}'^{T}\,\mathbf{X}')^{-1}\,\mathbf{X}'^{T}\mathbf{Y}'$$

sendo:

X' e **Y**' as matrizes **X** e **Y**, respectivamente, incluídos os termos *bias* **w**b.

Referências

[1] Jaeger, H. (2002). "Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and de 'echo state network' approach", Technical Report GMD Report 159, German National Reseach Center For Information Technology, St. Augustin-Gernany.

[2] Lukosevicius, M. and Jaeger, H. (2009)."Reservoir computing approaches to recurrent neural network training", Computer Science Review, Vol.3, No.3, pp. 127-149.

[3] Maass, W.; Natschläger, T. and Markram, H. (2002)."Real-time computing without stable states: a new framework for neural computation based on perturbations, Neural Computing, Vol. 14, No. 11, pp. 2531–2560.

Modelo BELBIC

BELBIC: “Brain Emotional Learning Based Intelligent Controller”

- Proposto por Caro Lucas (iraniano) [2];
- Baseado num modelo de Balkenius & Moren [1];
- Controlador Adaptativo Direto (CAD) (parâmetros do controlador são ajustados visando minimizar o erro).

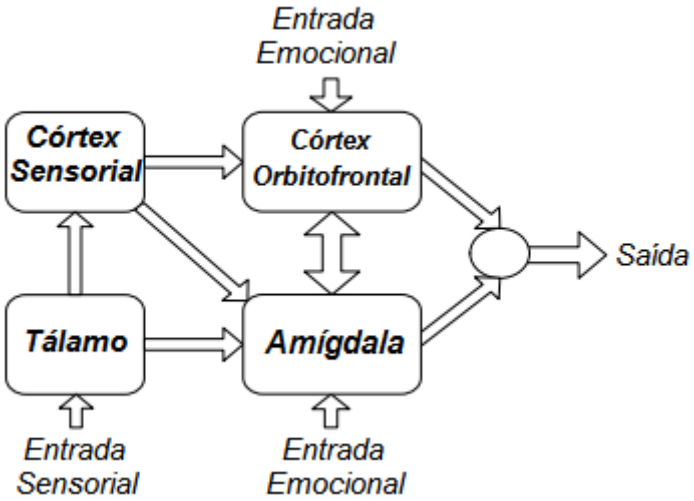


Figura 1. Modelo BELBIC.

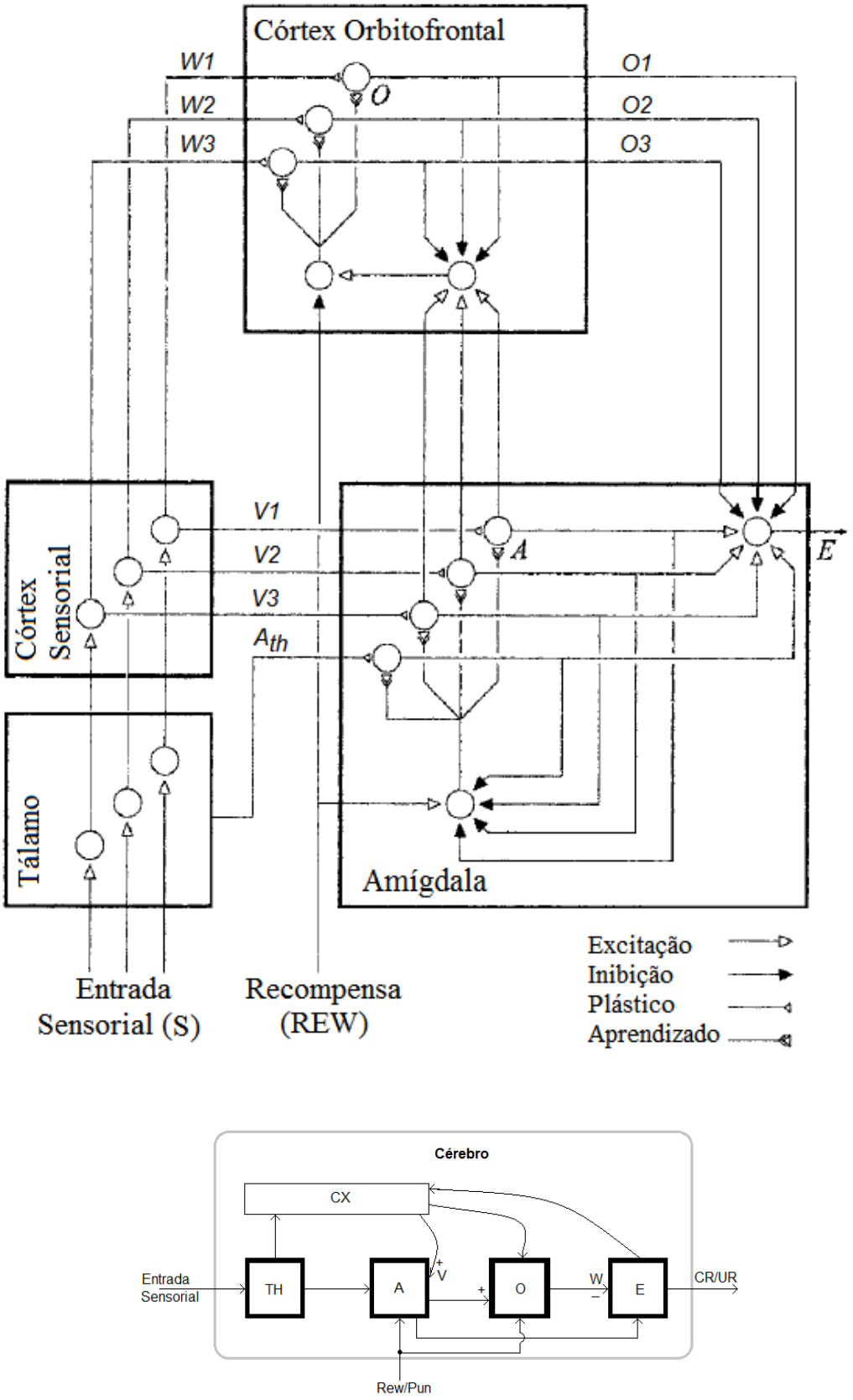


Figura 2. Ilustração das interações no modelo BELBIC.

sendo:

CX : Córtex sensorial;

A : estrutura de entrada na amígdala;

E : Estrutura de saída da amígdala;

O : Córtex Orbitofrontal;

Rew/Pun : Sinais externos identificando a recompensa / punição;

CR/UR : Resposta condicionada/não-condicionada;

V : força associativa da representação cortical para a amígdala que é modificada pelo aprendizado;

W : conexão inibitória do córtex orbitofrontal para a amígdala que é modificada durante o aprendizado.

- A estrutura BELBIC é dividida em duas partes principais (vide Figura 1): (1) amígdala e (2) córtex orbitofrontal;
- A Amígdala recebe entradas do tálamo e das áreas corticais;
- O sistema orbitofrontal recebe contribuições das áreas corticais e da amígdala.
- O BELBIC recebe o reforço de sinal (*REW*);
- A saída do BELBIC é dada por:

$$E = \sum_i Ai - \sum_i Oi \text{ (incluindo o componente } A_{th}) \quad (1)$$

$$E' = \sum_i Ai - \sum_i Oi \text{ (não incluindo o componente } A_{th}) \quad (2)$$

sendo:

A_{th} : conexão do tálamo.

- A conexão do tálamo é calculada da seguinte forma:

$$A_{th} = max(S_i) \quad (3)$$

- Regra de aprendizado da amígdala:

$$\Delta V_i = \alpha_a \{Si\ max\ (0, REW - \sum_i Ai\ )\} \quad (4)$$

sendo:

α_a : taxa de treinamento da amígdala;

REW : sinal de reforço (recompensa / punição);

Vi : pesos das conexões plásticas da amígdala.

- Similarmente, a regra de aprendizado do córtex orbitofrontal:

$$\Delta W_i = \alpha_o \{S_i (E' - REW)\} \quad (5)$$

sendo:

α_o : taxa de treinamento do córtex orbitofrontal;

W_i : pesos das conexões córtex orbitofrontal.

- Os valores dos nós são calculados da seguinte forma:

$$A_i = S_i V_i \quad (6)$$

$$O_i = S_i W_i \quad (7)$$

- O sinal de reforço *REW*:

$$REW = f(e, u) \quad (8)$$

Uma função custo.

sendo:

e : sinal do erro;

y_p : saída da planta.

- De forma similar, as entradas sensoriais podem ser uma função das saídas da planta e saídas do controlador:

$$S_i = g(u, y_p, y_r) \quad (10)$$

Uma função custo.

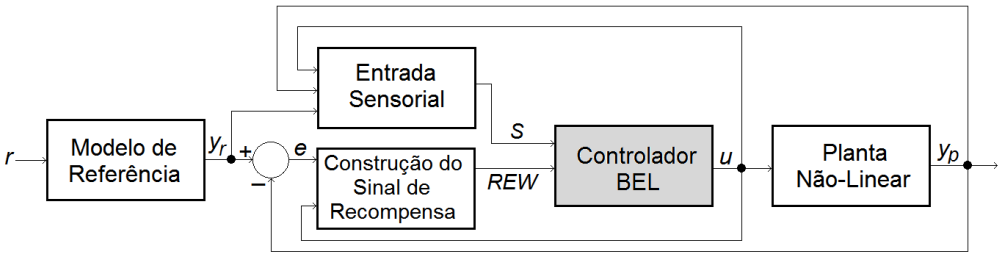


Figura 4. Arquitetura do controlador BEL (“Brain Emotional Learning”).

Referências

[1] BALKENIUS, C. and MOREN, J. (2001). Emotional Learning: A Computational Model of the Amygdala", Cybernetics and Systems: An International Journal, Taylor & Francis, No. 32, pp.611 – 636.

[2] LUCAS, C.; SHAHMIRZADI, D. and SHEIKHOLESIAMI, M. (2004). "Introducing BELBIC: Brain Emotional Learning Base Intelligent Controller", Intelligent Automation and Soft Computing, No. 10, pp. 11-22.