Escola Politécnica da Universidade de São Paulo

PCS 2059 – Inteligência Artificial

Redes Neurais de Uma Camada “Feed-Forward”

Alunos:

Alexandre Frandulic Shimono

André de Oliveira Lima Ikeda

Luiz Felipe Lu

Índice

1. Introdução 3
2. Unidade da rede neural 5
3. Estruturas das redes neurais 7
4. Redes neurais de uma camada "feed-forward" 8
5. Exemplo 12
6. Conclusão 17
7. Bibliografia 17

1. Introdução

Um **neurônio** é uma célula cerebral cuja principal função é a captação, processamento e disseminação de sinais elétricos, e pode ser decomposto em componentes, conforme ilustra a Figura 1.

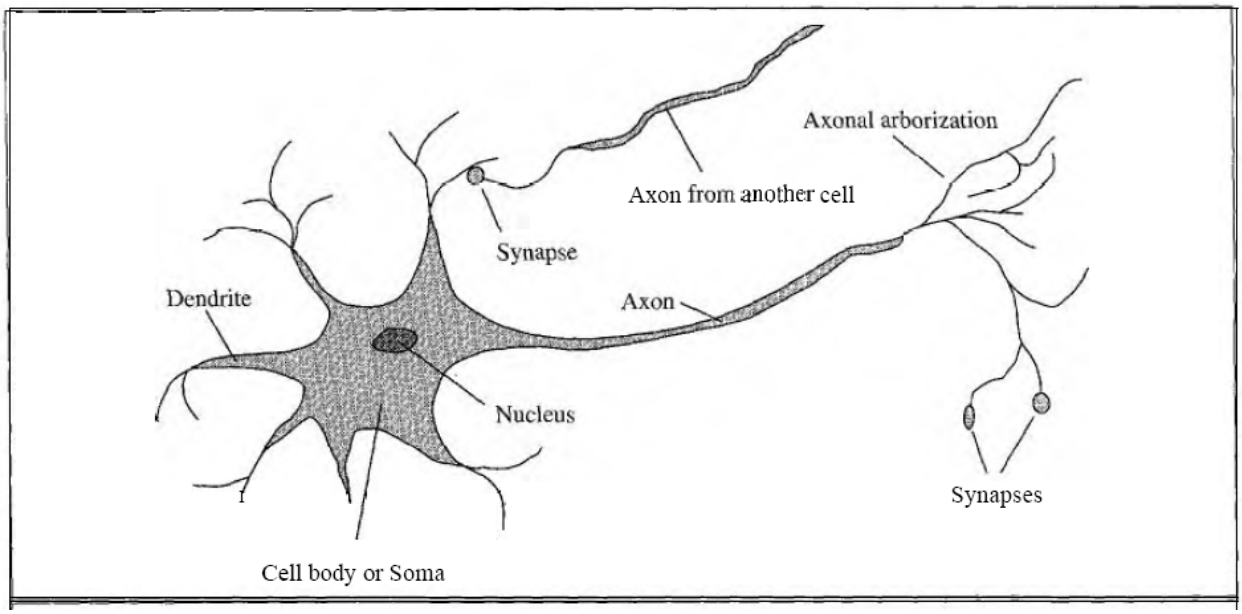

Figura 1– as partes de uma célula nervosa, ou neurônio (fonte: NORVIG, R. 2003 – "Artificial Inteligence – A Modern Approach").

Acredita-se que a capacidade de processamento de informação do cérebro emerge primariamente de redes destes neurônios, e por isso, um dos primeiros trabalhos da área de inteligência artificial foi tentar criar redes neurais artificiais (outros nomes para o capo são conectivismo, processamento distribuído paralelo e computação neural).

Redes neurais artificiais são modelos matemáticos ou computacionais baseados em redes neurais biológicas. Elas consistem de um grupo de neurônios artificiais interconectados, e processam informação utilizando uma abordagem conexionista para a computação.

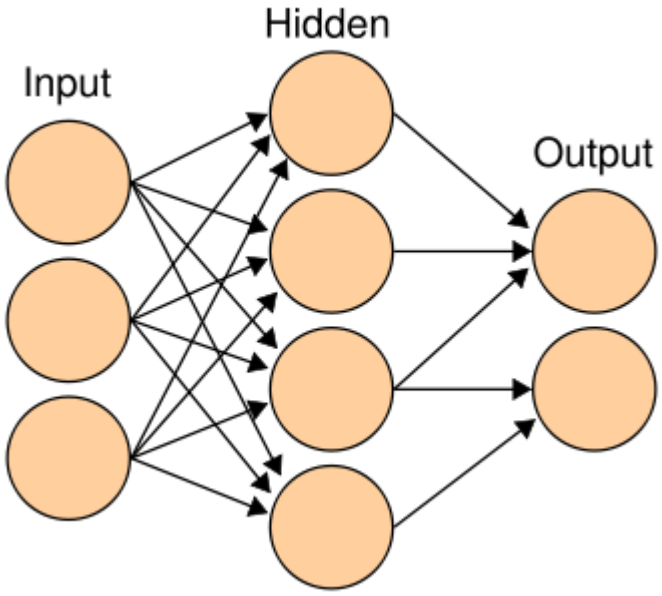

Figura 2 – uma rede neural artificial genérica de 2 camadas (fonte: Wikipedia).

Particularmente, as redes neurais atraíram pesquisadores por suas propriedades abstratas de executar computação distribuída, tolerar entradas com ruído e aprender. Por causa disso as redes neurais ainda são um dos mais populares e eficientes sistemas de aprendizado.

2. Unidade da rede neural

Redes neurais são compostas de unidades conectadas por **ligações diretas**, e cada ligação de um neurônio ao outro serve para propagar sua **ativação**. Na entrada de cada neurônio, é atribuído um **peso** a cada uma destas ligações provenientes de neurônios anteriores (ou da entrada da rede neural), que determina a força do sinal da conexão, ou seja, a relevância deste sinal para o disparo do neurônio. Então, o neurônio calcula a soma das ativações de seus antecessores,depois que os devidos pesos de cada uma destas ligações foi utilizado, e sobre este valor aplica a sua função de ativação, que irá determinar sua saída.

Na Figura 3 pode-se observar um modelo matemático de uma rede neural. Basicamente, o neurônio "dispara" quando a combinação linear das entradas passa de certo limiar, que é determinado pelo **"bias weight",** ou **peso de viés**.

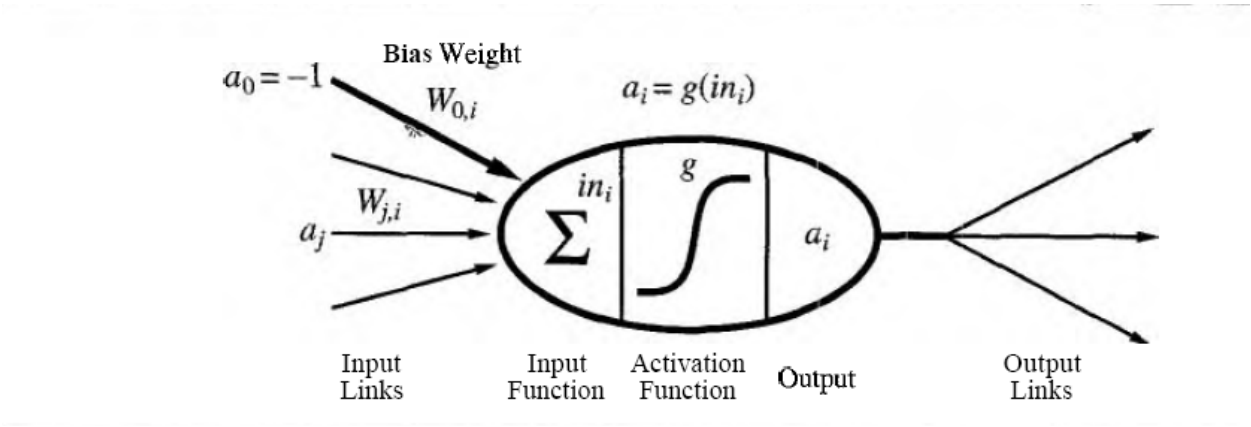

Figura 3 – um modelo matemático de rede neural, elaborado por McCulloch e Pitts (1943) (fonte: NORVIG, R. 2003 – "Artificial Inteligence – A Modern Approach").

A função de ativação deve basicamente atender a dois requerimentos: deve manter a unidade (neurônio) ativa, ou próxima de +1, quando as entradas certas forem fornecidas, e inativa, ou próxima de 0, para as entradas erradas; e a ativação deve ser não linear, caso contrário a rede neural colapsaria numa simples função linear.

Basicamente, duas funções são mais utilizadas como função de ativação: a **degrau**, e a função **sigmóide**, conforme mostra a Figura 4. A função sigmóide tem certa vantagem sobre a função degrau, pois é diferenciável (o que se mostrará muito útil na hora de treinar as redes neurais de uma camada).

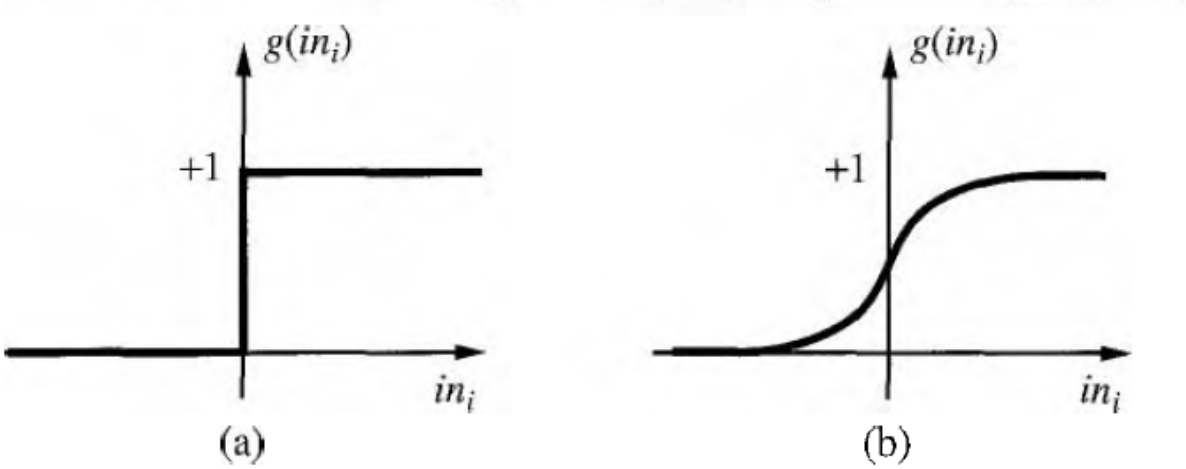

Figura 4 – (a) Função degrau, (b) Função Sigmóide (fonte: NORVIG, R. 2003 – "Artificial Inteligence – A Modern Approach").

Resumindo, o neurônio (ou unidade da rede neural) executa os seguintes passos para gerar sua saída (utilizando a Figura 3 para ilustrar o exemplo):

2.1. Recebe o sinal de cada uma de suas entradas, ilustradas como "aj";

2.2. Cada uma dessas entradas é multiplicada por seu respectivo peso, ilustrado como "Wj,i";

2.3. É feita a somatória das duplas "ai x Wj,i" , e da entrada "a0" com o peso de viés;

2.4. É aplicada a função de ativação sobre o resultado, ilustrada como "g". Caso o resultado seja positivo, a saída será próxima de 1, caso contrário será próxima de 0.

Um bom exemplo ilustrativo de como funciona a operação de um neurônio é a implementação de portas lógicas em unidades com função degrau, conforme ilustra a Figura 5:

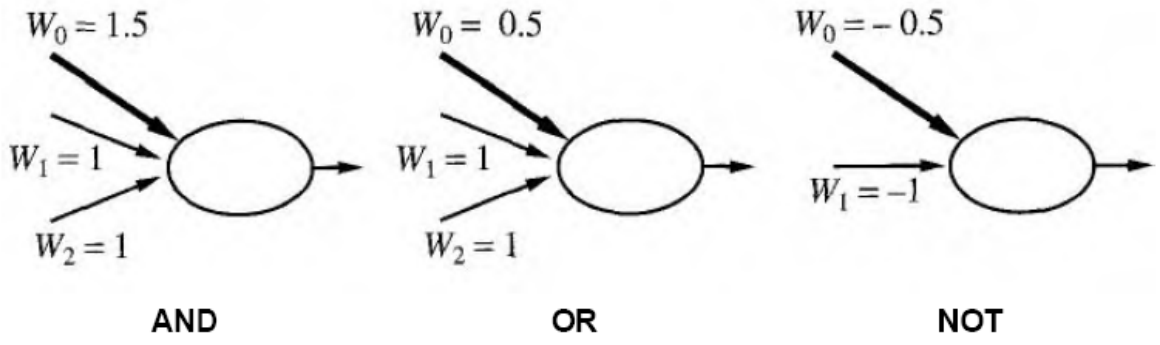

Figura 5 – portas lógicas "AND", "OR" e "NOT" implementadas em neurônios (fonte: NORVIG, R. 2003 – "Artificial Inteligence – A Modern Approach").

Analisando o exemplo da porta lógica "AND", obtêm-se a seguinte tabela:

a0	a1	a2	Somatória ai x Wi	g(Somatória ai x Wi)	AND teórico
-1	1	1	0,5	1	1
-1	1	0	-0,5	0	0
-1	0	1	-0,5	0	0
-1	0	0	-1,5	0	0

Tabela 1 – análise da implementação de uma porta lógica "AND" em um neurônio com função de ativação do tipo degrau.

3. Estruturas das redes neurais

Existem duas categorias principais de redes neurais: **acíclicas** (ou redes "**feed-forward**") e **cíclicas** (ou redes **recorrentes**). As redes "feed-forward" representam uma função de suas entradas, não havendo estado interno além dos pesos das entradas de cada neurônio. Uma rede recorrente por sua vez possui uma realimentação de sua saída, o que significa que os níveis de ativação formam um sistema dinâmico, que podem resultar num estado estável ou exibir oscilações e até mesmo comportamento caótico. Dessa maneira, o próximo estado da rede depende de seu estado atual, que pode depender de entradas anteriores, ou seja, o sistema possui certa memória de curto prazo, o que o torna um modelo mais interessante do cérebro, mas também muito mais difícil de ser entendido.

As redes "feed-forward" podem conter várias unidades entre a entrada e a unidade de saída. Essas unidades são chamadas de **"unidades ocultas" ("hidden units")**. A rede pode ainda ser subdividida em camadas, de modo que cada unidade recebe entradas de unidades da camada imediatamente anterior.

É possível demonstrar que uma rede "feed-forward" é uma função de suas entradas. Na Figura 6 podemos observar os neurônios 3 e 4, cujas saídas irão alimentar o neurônio 5.

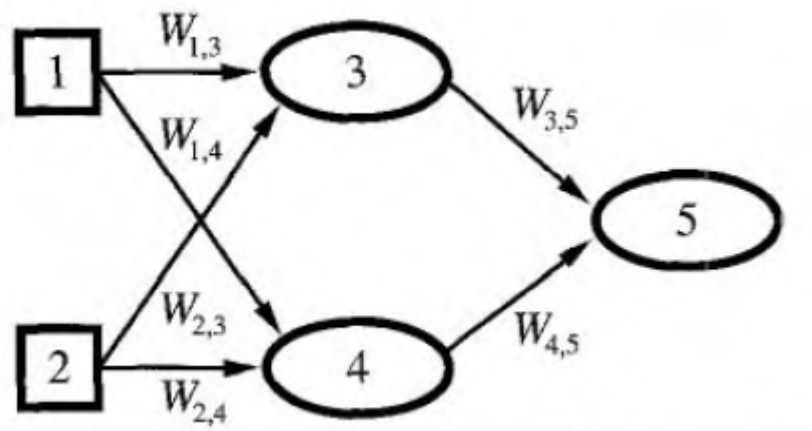

Figura 6 – uma rede neural simples, com uma camada escondida de duas unidades, e uma camada de saída de uma unidade(fonte: NORVIG, R. 2003 – "Artificial Inteligence – A Modern Approach")..

Assim, a saída do neurônio 5, que chamaremos de a5, pode ser descrita como uma função de a3 e a4, as respectivas saídas dos neurônios 3 e 4.

a5 = g(W3,5 x a3 + W4,5 x a4) (1)

Porém, a3 e a4 são funções de a1 e a2, que são as entradas do sistema. Assim:

a3 = g(W1,3 x a1 + W2,3 x a2) (2)

a4 = g(W1,4 x a1 + W2,4 x a2) (3)

Substituindo (2) e (3) em (1), temos a saída a5 como função das entradas a1 e a2:

a5 = g(W3,5 x g(W1,3 x a1 + W2,3 x a2) + W4,5 x g(W1,4 x a1 + W2,4 x a2))

Normalmente, as redes neurais são usadas para problemas de **classificação booleana** (ou seja, fornece um resultado booleano que indica se o elemento pertence a um conjunto ou a outro), sendo que valores de saída acima de 0.5 são interpretados como um conjunto, e abaixo de 0.5 de outro. É possível dividir os intervalos de saída do neurônio de maneira que cada intervalo represente um conjunto, mas é mais comum utilizar um neurônio para cada conjunto (de maneira que uma saída positiva indica que o elemento pertence ao conjunto).

4. Redes neurais de uma camada "feed-forward"

Redes neurais cujas entradas são ligadas diretamente aos neurônios de saída são chamadas de redes neurais de uma camada, ou de redes de **perceptrons**. Como cada unidade de saída é independente das demais, cada peso afeta apenas uma das saídas, conforme mostrado na Figura 7.

Examinando o espaço de hipótese que um perceptron pode representar, pode-se enxergar o perceptron como uma representação de uma função booleana. Além das funções "AND", "OR" e "NOT" vistas anteriormente, o perceptron ainda é capaz de representar funções mais complexas, como por exemplo, a função da maioria (onde o resultado é igual ao valor que possui número de entradas maior que a metade do total de entradas).

Porém, existem funções booleanas que não podem ser representadas pelo perceptron (conforme demonstrado posteriormente). Como definido anteriormente, o perceptron retorna 1 se e somente se a soma com os pesos de todas as entradas (inclusive o peso de viés) for positivo:

$\sum W_j x_j > 0$ ou $\vec{W} \bullet \vec{x} > 0$ (ou seja, o produto escalar entre os pesos e as entradas)

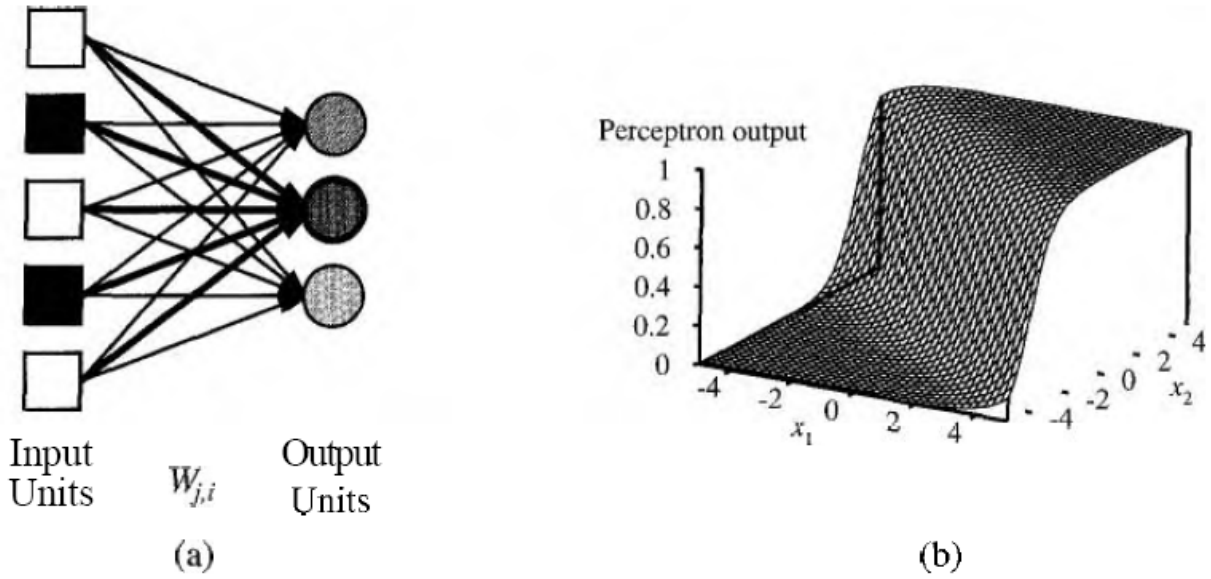

Figura 7 – (a) uma rede neural de uma camada, onde cada entrada é ligada diretamente ao neurônio de saída; (b) Um perceptron de duas entradas com uma função de ativação sigmóide(fonte: NORVIG, R. 2003 – "Artificial Inteligence – A Modern Approach").

A equação define um **hiperplano** (um objeto de n-1 dimensões num espaço de n dimensões) no espaço dos pesos, e por esta razão o perceptron é chamado de **separador linear.** Um exemplo de como essa separação é feita é mostrada na Figura 8.

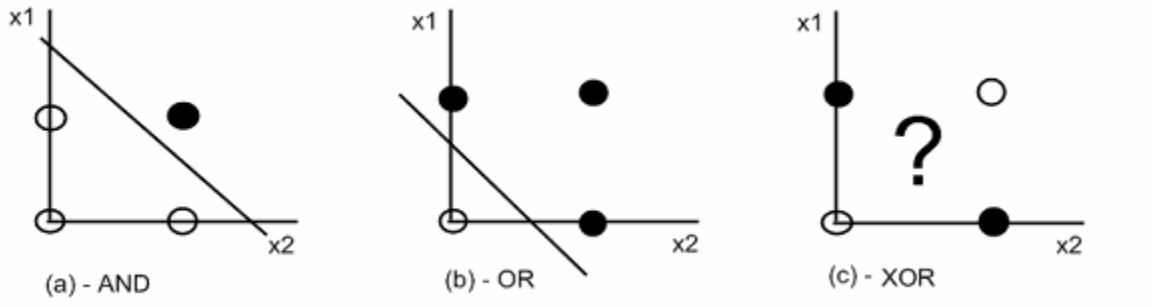

Figura 8 – separabilidade linear em perceptrons com função degrau de ativação. Os pontos negros indicam um ponto de entrada cujo valor de saída é 1. Pontos brancos indicam entradas cuja saída é 0. A linha é equivalente ao limiar do perceptron, ou seja, o local onde o valor de saída muda em função das entradas. Como se pode observar, no caso de um XOR não existe nenhum lugar onde se possa colocar uma linha para se obter o resultado desejado.

Apesar de sua capacidade de expressão de funções limitada, os perceptrons com função degrau têm algumas vantagens, como por exemplo, um **algoritmo simples de aprendizado**, que funciona para qualquer conjunto de treinamento linearmente separável. Infelizmente, o número de funções linearmente separáveis diminui em proporção ao número total de funções possíveis com o aumento do número de variáveis de entrada, conforme mostra a Tabela 2.

Número de entradas	Total de Funções	Funções LS	%
2	16	14	87,5
3	256	104	40,625
4	65536	1882	2,871704102

Tabela 2 – relação entre funções linearmente separáveis e número de variáveis de entrada.

Um **algoritmo de aprendizado** é um método adaptativo pelo qual uma rede de unidades computacionais se auto-organiza para implementar o comportamento desejado. Isto é feito em alguns algoritmos apresentando alguns exemplos do mapeamento de entrada e saída para a rede. Um passo de correção é executado iterativamente até que a rede aprenda a produzir a resposta desejada, conforme ilustra a Figura 9.

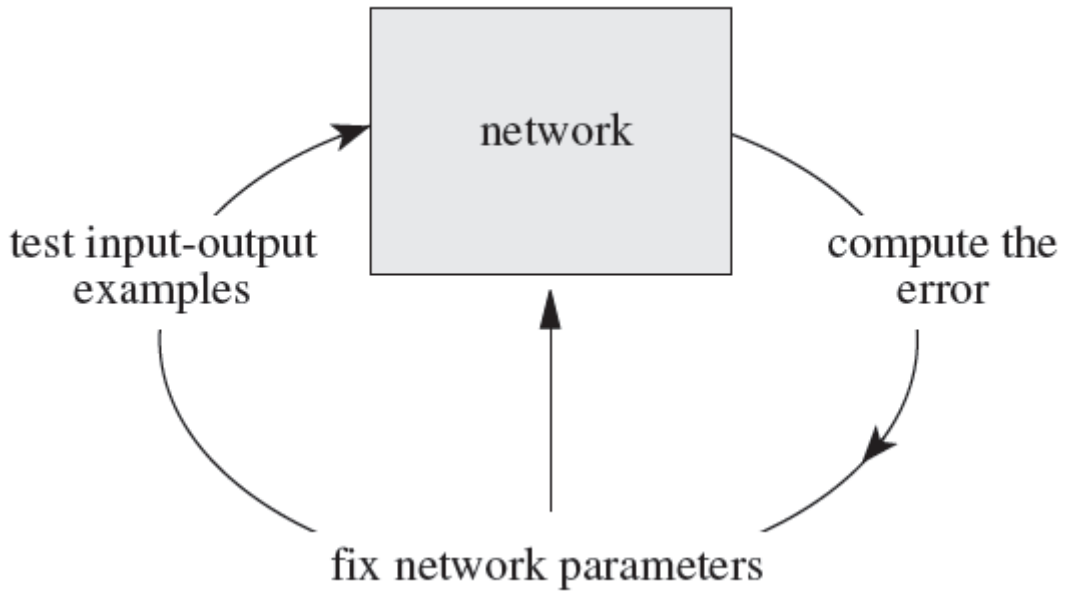

Figura 9 – sistema de aprendizado num sistema paramétrico [Rojas, 1996] .

Basicamente, os tipos de algoritmos que os perceptrons usam são supervisionados com reforço, embora algumas variantes usem aprendizado supervisionado com correção de erros.

4.1. Regra de treinamento de perceptron

Os dois algoritmos aqui apresentados convergem para diferentes hipóteses aceitáveis, sob condições diferentes.

Um meio de aprender um vetor de pesos aceitável é começar por pesos aleatórios, e então iterativamente aplicar ao perceptron cada exemplo de treinamento, modificando os pesos quando o perceptron realiza uma classificação errada. O processo é repetido até que todos os exemplos sejam classificados corretamente. A cada iteração os pesos são modificados de acordo com as regras do treinamento do perceptron, que revisa o peso Wi associado com a entrada xi de acordo com a regra da Equação 1.

$$w_i \leftarrow w_i + \Delta w_i \quad \text{onde} \quad \Delta w_i = \eta(t - o)x_i$$

Equação 1 – iteração para obtenção do próximo valor de Wi a partir do peso anterior."t" é a saída esperada do conjunto de treinamento, o é a saída gerada pelo perceptron e η é uma constante positiva chamada de taxa de aprendizado.

Para explicar o funcionamento do algoritmo, basta imaginar algumas situações hipotéticas. Caso o conjunto de pesos já tenha sido classificado corretamente pelo perceptron, então não haverão mudanças para a próxima iteração. Caso contrário, ele irá incrementar ou diminuir os pesos atuais em função da diferença entre o resultado esperado e o obtido. Pode ser provado que o procedimento converge para um numero finito de aplicações da regra de treinamento do perceptron que classificam corretamente todos os conjuntos de treinamento, desde que os exemplos sejam linearmente separáveis e que seja fornecido uma taxa de aprendizado pequena o suficiente. Se os dados não forem linearmente separáveis, a convergência não é garantida.

4.2. Regra do Delta e Gradiente Descendente

Caso o espaço de amostras não seja linearmente separável, o algoritmo acima não garante a convergência. Para superar essa dificuldade, foi desenvolvido o algoritmo que utiliza a regra do delta, ou seja, que utiliza o gradiente descendente para procurar o espaço de amostras por vetores que melhor se adaptam ao conjunto de treinamento.

O algoritmo se baseia no método dos mínimos quadrados, realizando uma busca de otimização no espaço dos pesos possíveis, ao percorrer o espaço dos possíveis valores do erro em função dos pesos adotados, e percorrer este espaço na direção da derivada negativa de maior módulo, ou seja, na direção em que há uma maior diminuição do erro. A cada iteração, o peso de cada entrada é calculado, levando em conta o peso anterior, o erro utilizando o peso anterior, a derivada da função de ativação para a iteração anterior, x anterior, e uma taxa de aprendizado α, que simboliza a velocidade com que o vetor de resposta caminha, conforme mostra a Equação 2.

$$W_j \leftarrow W_j + a \times Err \times g'(in) \times x_j$$ onde **Err = y(desejado) – g(in)**

Equação 2 – valor do novo peso Wj, em função do peso anterior.

Para a função sigmóide, a derivada de g vale g(1 – g). Já para perceptrons de função degrau, em que a derivada não está definida, a regra de aprendizado desenvolvida por Rosenblatt (1957) é idêntica a Equação 1, exceto que a derivada de g está omitida. Uma vez que a derivada é a mesma para todos os pesos, a sua omissão muda apenas a magnitude, e não o sentido da atualização para cada exemplo.

O cálculo do próximo peso é feito até que uma meta de parada seja atingida, que pode ser quando a variação dos pesos passar a ser menor que um determinado limite pré-estabelecido, por exemplo.

5. Exemplo

Como exemplo, pode-se mostrar como calcular os pesos para a função NAND. A Tabela 3 ilustra as relações entre entradas e saídas.

a0	a1	a3
1	1	0
1	0	1
0	1	1
0	0	1

Tabela 3 – relações de entrada e saída para uma porta "NAND"

Utilizando um perceptron (X_1, X_2 input, X_0*W_0=b, TH=0.5), obtemos a seguinte tabela:

Escola Politécnica da Universidade de São Paulo

PCS 2059 – Inteligência Artificial

Redes neurais de uma camada "feed-forward"

5.1. Função Degrau

			Entradas			Pesos Iniciais			Saida Calculada			Soma	Saida do neurônio	Erro	Correção	Pesos Finais		
Limiar	Taxa de aprendizado	Valores do sensor			Saída desejada													
TH	LR	X0	X1	X2	Z	w0	w1	w2	C0	C1	C2	S	N	E	R	W0	W1	W2
PASSO 1																		
0,5	0,1	1	0	0	1	0	0	0	0	0	0	0	0	1	0,1	0,1	0	0
0,5	0,1	1	0	1	1	0,1	0	0	0,1	0	0	0,1	0	1	0,1	0,2	0	0,1
0,5	0,1	1	1	0	1	0,2	0	0,1	0,2	0	0	0,2	0	1	0,1	0,3	0,1	0,1
0,5	0,1	1	1	1	0	0,3	0,1	0,1	0,3	0,1	0,1	0,5	0	0	0	0,3	0,1	0,1
PASSO 2																		
0,5	0,1	1	0	0	1	0,3	0,1	0,1	0,3	0	0	0,3	0	1	0,1	0,4	0,1	0,1
0,5	0,1	1	0	1	1	0,4	0,1	0,1	0,4	0	0,1	0,5	0	1	0,1	0,5	0,1	0,2
0,5	0,1	1	1	0	1	0,5	0,1	0,2	0,5	0,1	0	0,6	1	0	0	0,5	0,1	0,2
0,5	0,1	1	1	1	0	0,5	0,1	0,2	0,5	0,1	0,2	0,8	1	-1	-0,1	0,4	0	0,1
PASSO 3																		
0,5	0,1	1	0	0	1	0,4	0	0,1	0,4	0	0	0,4	0	1	0,1	0,5	0	0,1
0,5	0,1	1	0	1	1	0,5	0	0,1	0,5	0	0,1	0,6	1	0	0	0,5	0	0,1
0,5	0,1	1	1	0	1	0,5	0	0,1	0,5	0	0	0,5	0	1	0,1	0,6	0,1	0,1
0,5	0,1	1	1	1	0	0,6	0,1	0,1	0,6	0,1	0,1	0,8	1	-1	-0,1	0,5	0	0
PASSO 4																		
0,5	0,1	1	0	0	1	0,5	0	0	0,5	0	0	0,5	0	1	0,1	0,6	0	0
0,5	0,1	1	0	1	1	0,6	0	0	0,6	0	0	0,6	1	0	0	0,6	0	0
0,5	0,1	1	1	0	1	0,6	0	0	0,6	0	0	0,6	1	0	0	0,6	0	0
0,5	0,1	1	1	1	0	0,6	0	0	0,6	0	0	0,6	1	-1	-0,1	0,5	-0,1	-0,1
PASSO 5																		
0,5	0,1	1	0	0	1	0,5	-0,1	-0,1	0,5	0	0	0,5	0	1	0,1	0,6	-0,1	-0,1
0,5	0,1	1	0	1	1	0,6	-0,1	-0,1	0,6	0	-0,1	0,5	0	1	0,1	0,7	-0,1	0
0,5	0,1	1	1	0	1	0,7	-0,1	0	0,7	-0,1	0	0,6	1	0	0	0,7	-0,1	0
0,5	0,1	1	1	1	0	0,7	-0,1	0	0,7	-0,1	0	0,6	1	-1	-0,1	0,6	-0,2	-0,1

PASSO 6																		
	0,5	0,1	1	0	0	1	0,6	-0,2	-0,1	0,6	0	0	0,6	1	0	0	0,6	-0,2	-0,1
	0,5	0,1	1	0	1	1	0,6	-0,2	-0,1	0,6	0	-0,1	0,5	0	1	0,1	0,7	-0,2	0
	0,5	0,1	1	1	0	1	0,7	-0,2	0	0,7	-0,2	0	0,5	0	1	0,1	0,8	-0,1	0
	0,5	0,1	1	1	1	0	0,8	-0,1	0	0,8	-0,1	0	0,7	1	-1	-0,1	0,7	-0,2	-0,1
PASSO 7																			
	0,5	0,1	1	0	0	1	0,7	-0,2	-0,1	0,7	0	0	0,7	1	0	0	0,7	-0,2	-0,1
	0,5	0,1	1	0	1	1	0,7	-0,2	-0,1	0,7	0	-0,1	0,6	1	0	0	0,7	-0,2	-0,1
	0,5	0,1	1	1	0	1	0,7	-0,2	-0,1	0,7	-0,2	0	0,5	0	1	0,1	0,8	-0,1	-0,1
	0,5	0,1	1	1	1	0	0,8	-0,1	-0,1	0,8	-0,1	-0,1	0,6	1	-1	-0,1	0,7	-0,2	-0,2
PASSO 8																			
	0,5	0,1	1	0	0	1	0,7	-0,2	-0,2	0,7	0	0	0,7	1	0	0	0,7	-0,2	-0,2
	0,5	0,1	1	0	1	1	0,7	-0,2	-0,2	0,7	0	-0,2	0,5	0	1	0,1	0,8	-0,2	-0,1
	0,5	**0,1**	**1**	**1**	**0**	**1**	**0,8**	**-0,2**	**-0,1**	**0,8**	**-0,2**	**0**	**0,6**	**1**	**0**	**0**	**0,8**	**-0,2**	**-0,1**
	0,5	**0,1**	**1**	**1**	**1**	**0**	**0,8**	**-0,2**	**-0,1**	**0,8**	**-0,2**	**-0,1**	**0,5**	**0**	**0**	**0**	**0,8**	**-0,2**	**-0,1**
PASSO 9																			
	0,5	**0,1**	**1**	**0**	**0**	**1**	**0,8**	**-0,2**	**-0,1**	**0,8**	**0**	**0**	**0,8**	**1**	**0**	**0**	**0,8**	**-0,2**	**-0,1**
	0,5	**0,1**	**1**	**0**	**1**	**1**	**0,8**	**-0,2**	**-0,1**	**0,8**	**0**	**-0,1**	**0,7**	**1**	**0**	**0**	**0,8**	**-0,2**	**-0,1**

Tabela 4 – exemplo de cálculo dos pesos de um perceptron utilizando a regra do Delta.

Assim, os pesos calculados após 9 passos (ou épocas, como também são chamadas), foram:

W0 = 0,8

W1 = -0,2

W2 = -0,1

Escola Politécnica da Universidade de São Paulo

PCS 2059 – Inteligência Artificial

Redes neurais de uma camada "feed-forward"

Verificação

TH	x0	x1	x2	w0	w1	w2	c0	c1	c2	Soma	Saida
0,5	1	0	0	0,800	-0,200	-0,100	0,800	0,000	0,000	0,800	1
0,5	1	0	1	0,800	-0,200	-0,100	0,800	0,000	-0,100	0,700	1
0,5	1	1	0	0,800	-0,200	-0,100	0,800	-0,200	0,000	0,600	1
0,5	1	1	1	0,800	-0,200	-0,100	0,800	-0,200	-0,100	0,500	0

5.2. Função Sigmóide

Limiar	Taxa de aprendizado	Valores do sensor			Saída desejada														
TH	LR	X0	X1	X2	Z	w0	w1	w2	C0	C1	C2	S	Sigm	N	E	R	W0	W1	W2
0,5	0,1	1	0	0	1	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,500	0	1,000	0,025	0,025	0,000	0,000
0,5	0,1	1	0	1	1	0,025	0,000	0,000	0,025	0,000	0,000	0,025	0,506	1	0,000	0,000	0,025	0,000	0,000
0,5	0,1	1	1	0	1	0,025	0,000	0,000	0,025	0,000	0,000	0,025	0,506	1	0,000	0,000	0,025	0,000	0,000
0,5	0,1	1	1	1	0	0,025	0,000	0,000	0,025	0,000	0,000	0,025	0,506	1	-1,000	-0,025	0,000	-0,025	-0,025
0,5	0,1	1	0	0	1	0,000	-0,025	-0,025	0,000	0,000	0,000	0,000	0,500	1	0,000	0,000	0,000	-0,025	-0,025
0,5	0,1	1	0	1	1	0,000	-0,025	-0,025	0,000	0,000	-0,025	-0,025	0,494	0	1,000	0,025	0,025	-0,025	0,000
0,5	0,1	1	1	0	1	0,025	-0,025	0,000	0,025	-0,025	0,000	0,000	0,500	1	0,000	0,000	0,025	-0,025	0,000
0,5	0,1	1	1	1	0	0,025	-0,025	0,000	0,025	-0,025	0,000	0,000	0,500	1	-1,000	-0,025	0,000	-0,050	-0,025
0,5	0,1	1	0	0	1	0,000	-0,050	-0,025	0,000	0,000	0,000	0,000	0,500	1	0,000	0,000	0,000	-0,050	-0,025
0,5	0,1	1	0	1	1	0,000	-0,050	-0,025	0,000	0,000	-0,025	-0,025	0,494	0	1,000	0,025	0,025	-0,050	0,000
0,5	0,1	1	1	0	1	0,025	-0,050	0,000	0,025	-0,050	0,000	-0,025	0,494	0	1,000	0,025	0,050	-0,025	0,000
0,5	0,1	1	1	1	0	0,050	-0,025	0,000	0,050	-0,025	0,000	0,025	0,506	1	-1,000	-0,025	0,025	-0,050	-0,025
0,5	0,1	1	0	0	1	0,025	-0,050	-0,025	0,025	0,000	0,000	0,025	0,506	1	0,000	0,000	0,025	-0,050	-0,025
0,5	0,1	1	0	1	1	0,025	-0,050	-0,025	0,025	0,000	-0,025	0,000	0,500	0	1,000	0,025	0,050	-0,050	0,000
0,5	0,1	1	1	0	1	0,050	-0,050	0,000	0,050	-0,050	0,000	0,000	0,500	0	1,000	0,025	0,075	-0,025	0,000
0,5	0,1	1	1	1	0	0,075	-0,025	0,000	0,075	-0,025	0,000	0,050	0,512	1	-1,000	-0,025	0,050	-0,050	-0,025
0,5	0,1	1	0	0	1	0,050	-0,050	-0,025	0,050	0,000	0,000	0,050	0,513	1	**0,000**	0,000	0,050	-0,050	-0,025
0,5	0,1	1	0	1	1	0,050	-0,050	-0,025	0,050	0,000	-0,025	0,025	0,506	1	**0,000**	0,000	0,050	-0,050	-0,025
0,5	0,1	1	1	0	1	0,050	-0,050	-0,025	0,050	-0,050	0,000	0,000	0,500	1	**0,000**	0,000	0,050	-0,050	-0,025
0,5	0,1	1	1	1	0	0,050	-0,050	-0,025	0,050	-0,050	-0,025	-0,025	0,494	0	**0,000**	0,000	0,050	-0,050	-0,025

Os pesos obtidos foram:

W0 = 0,05

W1 = -0,05

W2 = -0,025

Verificação

TH	x0	x1	x2	w0	w1	w2	c0	c1	c2	Soma	Sigm	Saida
0,5	1	0	0	0,050	-0,050	-0,025	0,050	0,000	0,000	0,050	0,513	1
0,5	1	0	1	0,050	-0,050	-0,025	0,050	0,000	-0,025	0,025	0,506	1
0,5	1	1	0	0,050	-0,050	-0,025	0,050	-0,050	0,000	0,000	0,501	1
0,5	1	1	1	0,050	-0,050	-0,025	0,050	-0,050	-0,025	-0,025	0,494	0

6. Conclusão e observações

O uso de redes neurais tem se mostrado eficaz na resolução de problemas. Em particular os perceptrons são muito eficientes em problemas de classificação, embora sua utilidade seja limitada pelo número de funções linearmente decrescente com o aumento do número de variáveis de entrada.

Um outro ponto que torna os perceptrons muito úteis é o algoritmo simples e eficaz de atribuição de pesos ás entradas do neurônio. Embora o número de funções implementáveis seja relativamente pequeno para grandes números de variáveis, a facilidade de sua implementação justifica seu estudo e seu uso.

O fato de simular o modelo de processamento do cérebro e ser um modelo fiel de um neurônio possibilitou avanços na medicina, como por exemplo uma melhor compreensão de como funciona a visão humana e a arquitetura dos olhos, e também um avanço no desenvolvimento da visão de máquinas, o que certamente auxiliará na criação de aparelhos de auxílio para pessoas com deficiências visuais (como por exemplo no livro "Vision", de David Marr).

No campo da engenharia, a rede neural pode ser muito útil á medida que pode realizar funções lógicas mais complexas utilizando apenas um neurônio, economizando dessa maneira matéria prima e tornando a produção mais barata.

7. Bibliografia

NORVIG, Peter; RUSSEL, Stuart . **Artificial Intelligence – A modern approach – Second Edition.** Pearson Education Inc. Upper Saddle River, New Jersey. 2003.

ROJAS, Raúl. **Neural Networks – A Systematic Introduction.** Springer-Verlag. Berlin, 1996.

MICHELL, Tom M. **Machine Learning** . McGrawHill. 1997

WIKIPEDIA. **Perceptron**. Disponvel em http://en.wikipedia.org/wiki/Perceptron. Acesso em 02/10/2008.