Introdução às redes Neurais Artificiais (RNAs) v1.1

por João Paulo Schwarz Schüler
jpss@schulers.com - http://www.schulers.com/jpss
(*enviar correções, sugestões ao jpss*)

1 Inspiração Biológica

As redes neurais artificiais inspiram-se em modelos biológicos. Todos os modelos vistos no presente texto são apenas biologicamente inspirados não apresentado comportamento idêntico a uma rede neural biológica. Mesmo assim, diversas características importantes encontradas nas redes biológicas podem ser encontradas nas redes artificiais como o reconhecimento de padrões, processamento distribuído e paralelo (PDP) e tolerância a falhas.

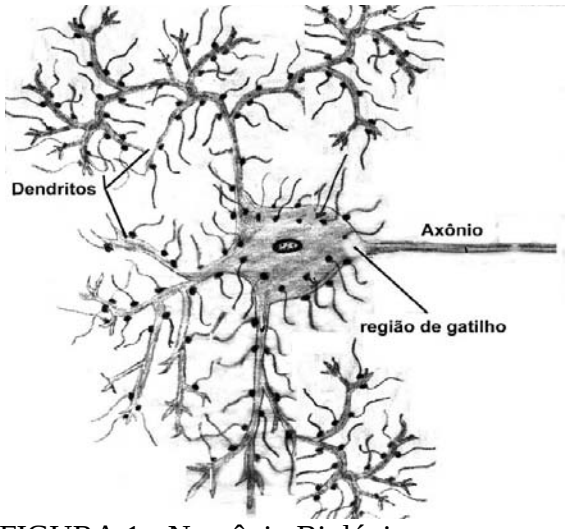


FIGURA 1 - Neurônio Biológico

Na figura 1, encontra-se um neurônio natural. Os neurônios recebem sinais elétricos ou químicos pelas sinapses. As sinapses são o ponto de ligação entre dois neurônios. Em média, cada neurônio apresenta 10.000 sinapses. Alguns neurônios chegam a apresentar 100.000 à 200.000 sinapses. Os neurônios biológicos são um meio de processamento de informação e ao mesmo tempo um meio de armazenamento de informação.

O neurônio biológico recebe sinais pelas sinapses e pode disparar ou não enviando um único sinal por seu axônio que usualmente está ligado a milhares de outros neurônios. A sinapse fica entre o nerônio que recebe um sinal e o axônio do neurônio que envia o sinal.

As sinapses podem ser excitatórias ou inibitórias. As sinapses excitatórias favorecem o disparo do neurônio enquanto que as sinapses inibitórias inibem o disparo dos neurônios. Sendo assim, o disparo do neurônio ocorre de acordo com o padrão de sinal de entrada.

2 ADALINE

2.1 Introdução

Existem diversos modelos de neurônios artificiais. Aborda-se o ADALINE (Adaptative Linear Element) aqui por ser simples e por servir de introdução a outros tipos de neurônios artificiais. Observe na figura 2 os valores w_n representando o tipo de sinapse. Valores positivos para w_n significam que a sinapse n é excitatória. Valores negativos para w_n significam que a sinapse é inibitória. As entradas x_n podem valer -1 ou +1 significando respectivamente o não recebimento ou recebimento de disparo de outro neurônio pela sinapse n.

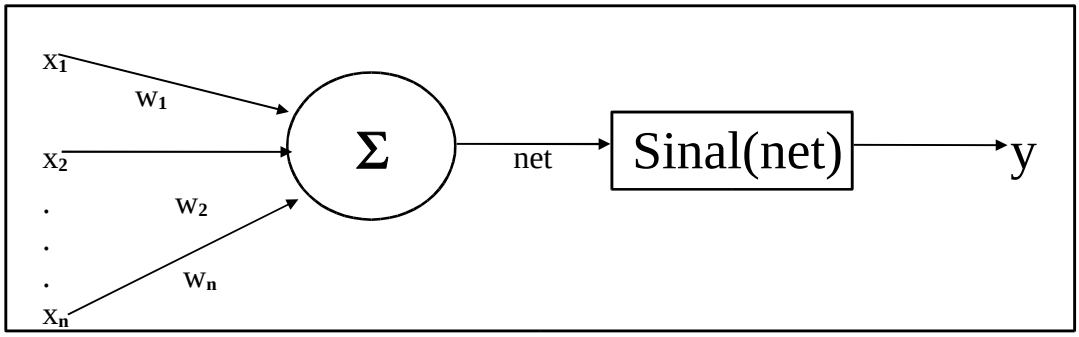


FIGURA 2 -- Modelo do ADALINE.

O ADALINE apresenta:

1. Um vetor de entradas **X** = (x_0, x_1, x_2, x_3, ... x_n);
2. Um vetor de pesos **W** = (w_0,w_2, w_3, ... w_n);
3. Um somador **Σ** que realiza o produto escalar ***WX*** ou soma ponderada do tipo $w_0x_0 + w_1x_1 + w_2x_2 + ... + w_nx_n$ que resulta no valor **net**;
4. A função de ativação **S**INAL sobre **net** que resulta na saída **Y;**
5. O valor de saída **Y**;

Os pesos w_0, w_1, ..., w_n são valores pertencentes ao conjunto dos números reais podendo ser inibitórios (negativos) ou excitatórios (positivos).

2.2 Criando Neurônios por Inspeção

Em uma aplicação normal, não se implementa um neurônio do tipo ADALINE com pesos previamente definidos ou arbitrados. Normalmente, é usada uma regra de aprendizado para que o algoritmo de aprendizagem encontre por si os pesos adequados (Ws) para o neurônio em questão; porém, para efeito didático, no presente tópico, estuda-se como implementar redes de ADALINEs definindo seus pesos manualmente.

Apenas para efeito de exemplo, usando-se o ADALINE, a tabela 1 apresenta um neurônio artificial para operar a operação lógica ***AND***. Nessa tabela, usa-se os valores $w_0 = -0.5$; $x_0 = w_1 = w_2 = 1$; **net** = $(1)*x_1 + (1)*x_2 + (-0.5)*(1) = x_1+x_2-0.5$.

TABELA 1 - Neurônio Artificial AND

X_1	X_2	*Net*	$Y = F(net) = x_1$ ***AND*** x_2
-1	-1	(-1) + (-1) - 0.5 = -2.5	-1
-1	1	(-1) + (+1) - 0.5 = -0.5	-1
1	-1	(+1) + (-1) - 0.5 = -0.5	-1
1	1	(+1) + (+1) - 0.5 = +1.5	1

Invertendo-se os pesos de W, encontra-se o neurônio artificial NAND. Nessa tabela, usa-se os valores $w_0 = 0.5$; $x_0=1$; $w_1 = w_2 = -1$; **net** = $(-1)*x_1 + (-1) *x_2 + (0.5)*(1) = - x_1 - x_2 + 0.5$.

TABELA 2 - Neurônio Artificial NAND

X_1	X_2	*Net*	$Y = F(net) = x_1$ *NAND* x_2
-1	-1	- (-1) - (-1) + 0.5 = 2.5	1
-1	1	- (-1) - (+1) + 0.5 = 0.5	1
1	-1	- (+1) - (-1) + 0.5 = 0.5	1
1	1	- (+1) - (+1) + 0.5 = -1.5	-1

Os neurônios artificiais podem ser conectados uns aos outros formando redes em que a saída ***Y*** de um neurônio pode ser conectada a entrada x_c de vários outros neurônios. Qualquer função lógica pode ser implementada através da composição de operações ***NAND***. Sendo assim, qualquer operação lógica pode ser implementada por redes de neurônios artificiais do tipo ADALINE.

Na tabela 1, usou-se o produto escalar dos seus pesos pelas entradas que é $x_1+x_2 -0,5$. Sendo assim, o neurônio retorna 1 quando $x_1+x_2-0,5>0$ e -1 quando $x_1+x_2-0,5<0$. O espaço de entrada bidimensional formado por (x_1,x_2) foi dividido em duas partes pela reta $x_1+x_2-0,5 = 0$. A propriedade do ADALINE e de alguns outros modelos de neurônios de separar o universo de entrada em duas partes por uma reta, plano ou hiperplano é chamada de ***separabilidade linear.*** Um único ADALINE só consegue resolver problemas linearmente separáveis.

Para traçar o gráfico da função AND da tabela 1, segue-se os passos:

1. $x_1=0$ => $0+x_2-0.5=0$ => $x_2=0.5$ => o ponto (0,0.5) é ponto da linha que separa o universo de entrada;
2. $x_2=0$ => $x_1+0-0.5=0$ => $x_1=0.5$ => (0.5,0) é ponto da linha;
3. Grafar a linha que liga os pontos (0.5,0) e (0,0.5).

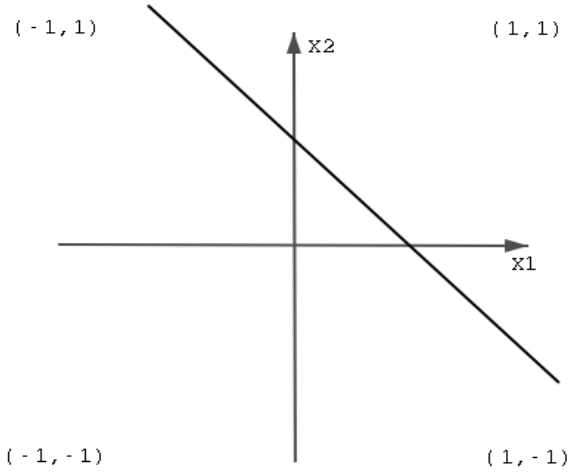


FIGURA 3 - Separabilidade Linear do AND

Observe na figura 3, que a função lógica AND é linearmente separável. É possível traçar uma linha entre o ponto (1,1) que deve retornar 1 e os outros pontos que devem retornar -1. Existem muitas

funções lógicas que não são linearmente separáveis. Na figura 4, observe que para separar as entradas que resultam em 1 das entradas que resultam em -1 foram necessárias duas linhas. Sendo assim, o XOR não é linearmente separável e não pode ser resolvido com um único neurônio. Serão necessários 3 ADALINES para resolver a função XOR por inspeção.

Na figura 4, a equação da reta R1 é $x_1+x_2-0,5=0$ e a equação da reta R2 é $x_1+x_2+0,5=0$. A função XOR retorna +1 quando a entrada está abaixo de R1 ($x_1+x_2-0,5<0$ ou $-x_1-x_2+0,5>0$) e acima de R2 ($x_1+x_2+0,5>0$). Sendo assim, para operar o XOR, são necessários um neurônio para R1, outro para R2 e mais um para operar ($x_1+x_2-0,5<0$) AND ($x_1+x_2+0,5>0$) ou R1 AND R2.

TABELA 3 - XOR

x_1	x_2	x_1 xor x_2
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

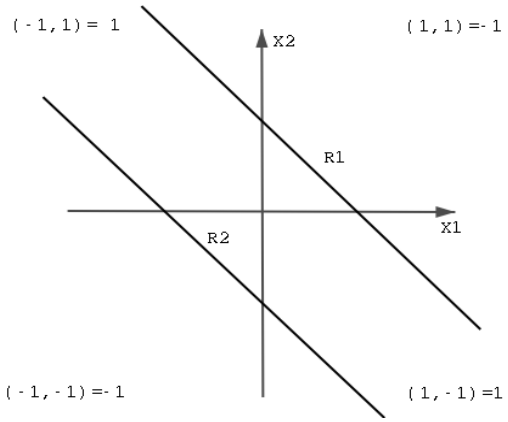


Figura 4 - XOR

Para implementar a rede neural XOR usaremos os neurônios R1, R2 e AND.

TABELA 4 - Rede Neural XOR

x_1	x_2	***R1:SINAL(***$-x_1-x_2+0,5$) $w_1=-1; w_2=-1; w_0=0,5$	***R2:SINAL(*** $x_1+x_2+0,5$) $w_1=1; w_2=1; w_0=0,5$	***R1 AND R2 =*** x_1 ***xor*** x_2 $w_1=1; w_2=1; w_0=-0,5$
-1	-1	1	-1	-1
-1	1	1	1	1
1	-1	1	1	1
1	1	-1	1	-1

Na figura 5, as setas são as sinapses; os números que aparecem aos seus lados representam os seus correspondentes pesos sinápticos; o neurônio R1 opera a operação AND; o neurônio R2 opera a operação OR.

Os pesos usados nas tabelas 1, 2 e 4 foram encontrados por inspeção.

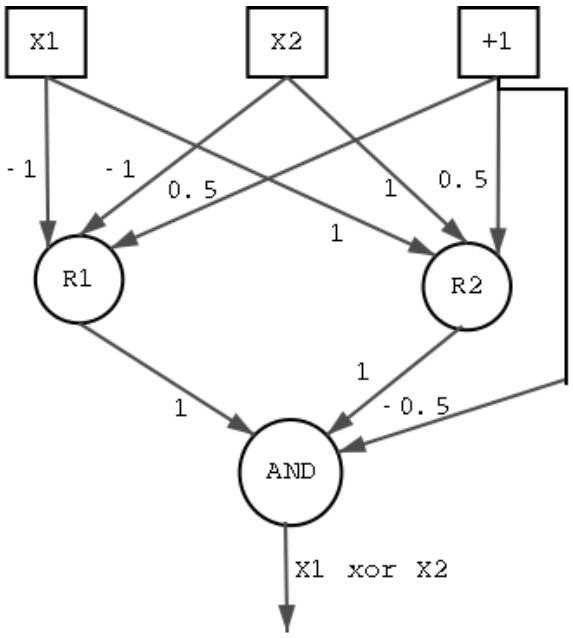


FIGURA 5 - Rede Neural XOR

2.3 Regra de Aprendizagem μ-LMS

No tópico anterior, os pesos foram arbitrados por inspeção. No presente tópico, os pesos resultam de um algoritmo de aprendizagem. É importante observar que um único ADALINE só poderá resolver problemas linearmente separáveis. Lembrando que o nome ADALINE vem de "Adaptative Linear Element", o termo "Adaptative" vem do fato de que existe um algoritmo para adaptar os pesos de forma que ele produza a saída desejada para cada entrada.

No entendimento do autor do presente texto, o algoritmo mais simples para correção de erro é o algoritmo μ-LMS. A fórmula do algoritmo μ-LMS é a que segue:

$W_{(t+1)} = W_t + 2u E_t X_t$ onde:

- W: vetor de pesos;
- μ: constante para correção de erro;
- E: erro instantâneo encontrado. É a saída desejada D menos o valor de net: E = D - net;
- X: vetor de entradas.

Exemplo 1: Seja um ADALINE com 3 entradas ligadas em +1, x_1 e x_2 e pesos iniciados todos em zero. Usando constante de correção de erro μ = 0.1, usa-se o algoritmo μ-LMS para gerar um neurônio artificial que opere a operação lógica or da forma que segue:

1. Define-se as entradas e saídas desejadas. Nesse caso, usamos a tabela da operação lógica OR

que segue:

TABELA 5 - OR

x_0	x_1	x_2	x_1 or x_2 = ***D***
1	-1	-1	-1
1	-1	1	1
1	1	-1	1
1	1	1	1

2. Calcula-se o valor de net para a primeira entrada (1,-1,-1) da tabela OR:
   net = WX = 0*1+0*(-1)+0*(-1)
3. Calcula-se o erro: E = D-net = -1 -0 = -1
4. Corrige-se usando $W_{(t+1)} = W_t + 2u E_t X_t$:
   W_{t+1}=(0,0,0)+2*0.1*(-1)*(1,-1,-1) = (0,0,0)+(-0.2)*(1,-1,-1) = (-0.2,0.2,0.2)
5. Repete-se os passos 2,3,4 para a segunda entrada (1,-1,1) da tabela OR
   net = WX = (-0.2)*1+0.2*(-1)+0.2*1 = -0.2-0.2+0.2 = -0.2
   E = D - net = 1 -(- 0.2) = 1.2
   W_{t+1}=(-0.2,0.2,0.2)+2*0.1*(1.2)*(1,-1,1) = (-0.2,0.2,0.2)+(0.24)*(1,-1,1) =
   = (-0.2,0.2,0.2) + (0.24,-0.24,0.24) = (0.04,-0.04,0.44)
6. Repete-se os passos 2,3,4 para a terceira entrada (1,1,-1) da tabela OR
   net = WX = 0.04*1+(-0.04)*(1)+0.44*(-1) = -0.44
   E = D - net = 1 -(- 0.44) = 1.44
   W_{t+1}= (0.04,-0.04,0.44)+2*0.1*(1.44)*(1,1,-1) =
   = (0.04,-0.04,0.44) + (0.29,0.29,-0.29) = (0.33,0.25,0.15)
7. Na execução normal do algoritmo μ-LMS, os passos 2,3 e 4 deveriam ser repetidos para todas as entradas da tabela OR até que o erro médio quadrado registrado em E seja suficientemente pequeno.

O erro médio quadrado é calculado pela fórmula que segue:

$erro\,médio\,quadrado = \frac{\sum e^2}{L}$ onde L é o número de entradas (padrões) da tabela. Por exemplo, na tabela 5, L é 4; na tabela 7, L é 2. O erro médio quadrado é o somatório dos erros (D-net) elevados ao quadrado para cada uma das entradas dividido pelo número de entradas.

Com o vetor W = (0.33,0.25,0.15), o ADALINE já opera a operação OR conforme mostra a tabela abaixo:

TABELA 6 - Testando os pesos encontrados (0.33,0.25,0.15) para a operação x_1 OR x_2.

x_0	x_1	x_2	***net = WX***	***Sinal(net)***
1	-1	-1	0.33*1+0.25*(-1)+0.15*(-1) = -0.07	-1
1	-1	1	0.33*1+0.25*(-1)+0.15*(1) = 0.23	1
1	1	-1	0.33*1+0.25*(1)+0.15*(-1) = 0.43	1
1	1	1	0.33*1+0.25*(1)+0.15*(1) = 0.73	1

Exemplo 2: Seja um ADALINE com 2 entradas ligadas em x_1 e x_2 e pesos iniciados todos em zero. Usando constante de correção de erro $\mu = 0.1$, usa-se o algoritmo μ-LMS para gerar um neurônio artificial que opere a operação lógica da tabela 7 na forma que segue:

TABELA 7

x_1	x_2	***Valor Desejado na saída***
-1	1	-1
1	-1	1

1. Calcula-se o valor de net para a primeira entrada (-1,1) da tabela.

   net = WX = 0*(-1)+0*(1)=0

2. Calcula-se o erro: E = D-net = -1 -0 = -1

3. Corrige-se usando $W_{(t+1)}=W_t+2uE_tX_t$:

   W_{t+1}=(0,0)+2*0.1*(-1)*(-1,1) = (0,0)+(-0.2)*(-1,1) = (0.2,-0.2)

Com o vetor W = (0.2,-0.2), o ADALINE já opera a operação da TABELA 7 conforme mostra a tabela que segue:

TABELA 8 - Testando os pesos encontrados (0.2,-0.2).

x_1	x_2	***net = WX***	***Sinal(net)***
-1	1	0.2*(-1)+(-0.2)*(1) = -0.4	-1
1	-1	0.2*(1)+(-0.2)*(-1) = 0.4	1

Exemplo 3: Seja um ADALINE com 2 entradas ligadas em x_1 e x_2 e pesos iniciados todos em zero. Usando constante de correção de erro $\mu = 0.1$, usa-se o algoritmo μ-LMS para gerar um neurônio artificial que opere a operação lógica da tabela 7 na forma que segue:

TABELA 9

x_1	x_2	***Valor Desejado na saída***
1	-1	-1
1	1	1

1. Calcula-se o valor de net para a primeira entrada (1,-1) da tabela.

   net = WX = 0*(1)+0*(-1)=0

2. Calcula-se o erro: E = D-net = -1 -0 = -1

3. Corrige-se usando $W_{(t+1)}=W_t+2uE_tX_t$:

   W_{t+1}=(0,0)+2*0.1*(-1)*(1,-1) = (0,0)+(-0.2)*(-1,1) = (-0.2,0.2)

4. Repete-se os passos 1,2,3 para a segunda entrada (1,1) da tabela 9.

   net = WX = (-0.2)*(1)+0.2*(1)=0

   Calcula-se o erro: E = D-net = 1 - 0 = 1

   W_{t+1}=(-0.2,0.2)+2*0.1*(1)*(1,1) = (-0.2,0.2)+(0.2,0.2) = (0,0.4)

Com o vetor W = (0,0.4), o ADALINE já opera a operação da TABELA 9 conforme mostra a tabela que segue:

TABELA 10 - Testando os pesos encontrados (0,0.4).

x_1	x_2	***net = WX***	***Sinal(net)***
1	-1	0*(-1)+(0.4)*(-1) = -0.4	-1
1	1	0*(1)+(0.4)*(1) = 0.4	1

3 Arquiteturas de Rede

A maneira pela qual os neurônios estão ligados por meio de sinapses determina a arquitetura da rede. Diferentes algoritmos de aprendizagem são usados em diferentes arquiteturas de rede. Em várias arquiteturas de rede, aparece o conceito da **camada**. A camada é um conjunto de neurônios normalmente formando uma linha, coluna ou plano com propriedades comuns.

3.1 Rede Alimentada Adiante com Camada Única

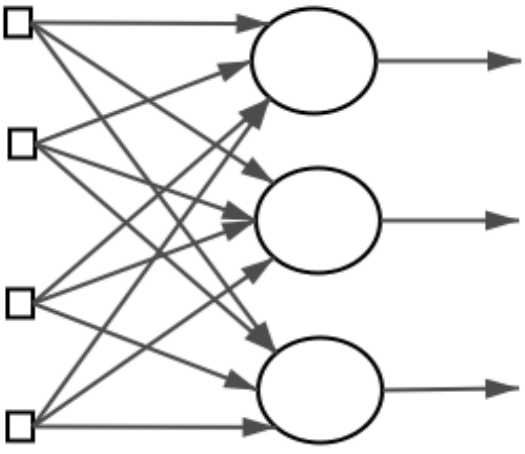


FIGURA 6 - Rede Alimentada Adiante com camada única.

Quando afirma-se que uma rede é alimentada adiante, entende-se que todos os neurônios de uma camada recebem sinais de camadas anteriores e enviam os seus sinais a camadas posteriores. A rede alimentada adiante com camada única é uma rede que recebe os seus sinais de entrada da camada de entrada e os processa em uma única camada de neurônios.

3.2 Rede Alimentada Adiante com Múltiplas Camadas

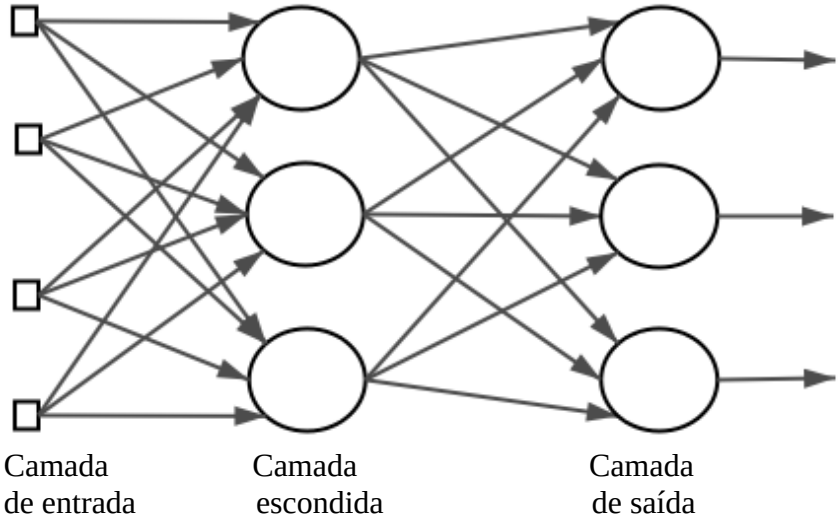


FIGURA 7 - Rede Alimentada Adiante com múltiplas camadas.

Também conhecida como rede alimentada diretamente com Múltiplas camadas, a rede alimentada adiante com múltiplas camadas apresenta neurônios que recebem sinais das camadas anteriores e enviam sinais às camadas posteriores. Essa rede é semelhante a rede alimentada adiante com camada única; porém, pode apresentar várias camadas de neurônios. Em qualquer rede alimentada adiante, não podem ocorrer ciclos ou realimentação.

3.3 Redes Recorrentes

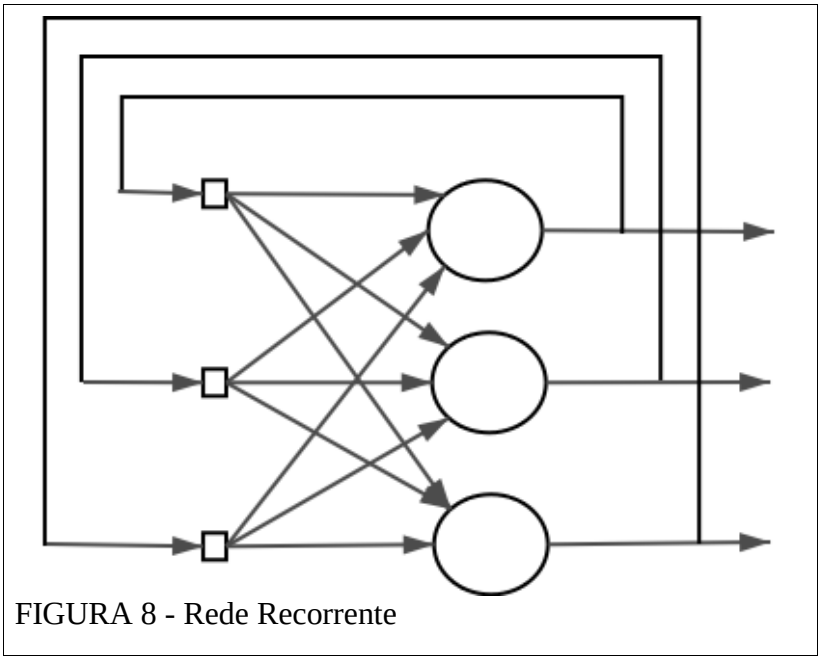


FIGURA 8 - Rede Recorrente

A principal diferença entre a rede alimentada adiante e a rede recorrente está do fato de que a rede recorrente apresenta pelo menos um laço. Na figura 8, cada neurônio se comunica com todos os outros neurônios da rede.

4 Revisão de Produto Escalar

O produto escalar de dois vetores retorna o produto dos seus módulos e o cosseno do ângulo formado pelos mesmos: $\vec{W}.\vec{X} = |\vec{W}| * |\vec{X}| * \cos(a)$ = $w_1x_1 + w_2x_2 + ... + w_nx_n$. O produto escalar de dois

vetores de módulo 1 resulta no cosseno do ângulo formado entre os mesmos.

Vetores de módulo 1 são designados por vetores unitários ou versores. A operação de transformar um vetor não nulo em um vetor unitário chama-se normalização. Para normalizar um vetor X, usa-se a fórmula: $\frac{\vec{X}}{|\vec{x}|}$. O vetor normalizado mantém a direção e sentido do seu vetor original perdendo somente o seu módulo.

O cosseno do ângulo formado por dois vetores tende a 1 quando o ângulo tende a 0. Sendo assim, quanto mais próximos forem os vetores, mais próximo de 1 será o cosseno do ângulo formado entre os mesmos. O produto escalar de dois vetores normalizados é 1 quando os vetores são iguais e -1 quando os vetores são opostos.

Em diversos modelos de neurônios artificiais, o valor de **net** é o produto escalar entre W e X. Supondo vetores W e X normalizados, o valor de **net** é o cosseno do ângulo formado entre os vetores W e X. Quanto mais alto for o valor de **net**, mais próximos são os vetores.

Exemplo: calcular o cosseno do ângulo formado pelos vetores X=(-1,-1) e W=(2,2).

$$|\vec{X}| = \sqrt{((-1)^2 + (-1)^2)} = \sqrt{2} \qquad |\vec{W}| = \sqrt{(2^2 + 2^2)} = \sqrt{8} = 2\sqrt{2}$$

$\vec{W}.\vec{X} = 2*(-1) + 2*(-1) = -4 = \sqrt{2}*2\sqrt{2}*\cos(a)$ => $\cos(a) = \frac{-4}{(\sqrt{2}*2\sqrt{2})} = -1$. Conclui-se que os vetores W e X apontam em direções opostas.

5 Introdução à Rede Neural Competitiva

5.1 Inspiração Biológica

Para tentar entender o funcionamento de uma rede neural biológica, é melhor escolher um ser vivo que possua um sistema nervoso bem simples. O sistema nervoso ótico do *Limulus* foi estudado na década de 30 por H. K. Hartline que ganhou o prêmio Nobel por este trabalho. O *Limulus polyphemus* é um artrópodo bastante comum na costa leste dos EUA. A classe dos artrópodos inclui insetos, aranhas e siris. O *Limulus* conviveu com os dinossauros é considerado como fóssil vivo por muitos.

Os olhos laterais do *Limulus* possuem cerca de 800 *ommatidias* que são unidades receptoras de sinais luminosos. Cada *ommatidia* possui seu próprio axônio. *Ommatidias v*izinhas são lateralmente inibidas. Quando uma *ommatidia* dispara em resposta a um sinal visual, ela inibe o disparo de suas vizinhas. Entre outros efeitos, a inibição lateral é responsável por incrementar o contraste das imagens percebidas.

Analisando um ser vivo bem diferente, o córtex humano é semelhante a uma folha de 1 metro quadrado com uma espessura de cerca de 3 milímetros formada por neurônios que processam informação em paralelo. O córtex apresenta dobras sucessivas possibilitando que ele apresente grande área e pequeno volume.

5.2 Modelo Artificial

Baseando-se em conhecimentos biológicos, a rede neural competitiva apresenta neurônios em uma única camada unidimensional ou bidimensional com inibição lateral. Quando um neurônio dispara, ele inibe os seus vizinhos. Se dois neurônios vizinhos disparam, eles se inibem mutuamente e o neurônio que dispara com maior força ou freqüência vence disparando sozinho.

A designação “rede neural competitiva” vem do fato de que os neurônios competem entre si. O neurônio de maior **net** vence. Normalmente, é mais fácil implementar a rede competitiva com um juiz externo. O juiz externo seleciona o neurônio de maior **net** e o define como vencedor. Somente o neurônio vencedor dispara.

A figura 6 apresenta uma arquitetura de rede neural que pode ser usada para implementar redes competitivas. Considerado a existência de um juiz externo que define o neurônio vencedor, não é necessário implementar as sinapses de inibição lateral. No caso em que todos os neurônios são alimentados pelo mesmo vetor de entrada normalizado e todos os neurônios apresentam vetores de pesos normalizados, o neurônio vencedor é o neurônio que apresenta o seu vetor de pesos mais parecido com o vetor de entrada.

Por simplicidade de implementação, pode-se determinar o neurônio vencedor simplesmente procurando pelo neurônio que apresenta o seu vetor de pesos mais parecido com o vetor de entrada. A determinação de qual neurônio apresenta vetor de pesos mais parecido com o vetor de entrada é feita procurando pela menor distância euclidiana entre os vetores de pesos e a entrada.

Para efeito de exemplo, abordando-se a figura 9 com uma imagem formada por 5x5 pixels em que cada pixel pode assumir um ton de cinza de 0 a 255 usando 8 bits de profundidade, observa-se que cada imagem é composta por 8*25 bits ou 200 bits. Certamente, não é possível construir por mapeamento uma função que leva cada imagem possível a uma saída específica desejada. Para resolver esse problema, clusteriza-se a entrada em clusters para então tratar os clusters individualmente. Supondo que a imagem da figura 9 seja clusterizada em 256 clusters diferentes, pode-se representar um cluster em apenas 8 bits resultando em uma enorme redução de dimensionalidade.

Através de uma rede neural competitiva e de um algoritmo de aprendizagem adequado, é possível clusterizar imagens em quantas classes for desejado.

X_1	X_2	X_3	X_4	X_5
X_6	X_7	X_8	X_9	X_{10}
X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}
X_{21}	X_{22}	X_{23}	X_{24}	X_{25}

FIGURA 9 – Vetor de Entrada para Clusterização de Padrões em Imagem 5x5.

5.3 Rede SOM

Aproveitando diversas características inspiradas na biologia de uma rede neural competitiva, a rede neural SOM apresenta uma camada normalmente unidimensional de entrada e uma camada normalmente bidimensional competitiva. Conforme pode ser observado na figura 10, cada neurônio da camada bidimensional recebe sinais de todos os neurônios da camada de entrada. Se a entrada é normalizada, o neurônio vencedor é aquele que apresentar maior valor para o produto escalar WX. Para entradas não normalizadas, deve ser calculado o neurônio vencedor pela menor distância euclidiana entre a entrada e os pesos do neurônio.

A rede SOM é indicada para problemas de clusterização. Quando um neurônio vence, o seu vetor de pesos é corrigido na direção do vetor de entrada e os seus vizinhos também recebem uma correção menor na direção do vetor de entrada. O neurônio vencedor é corrigido pela fórmula:

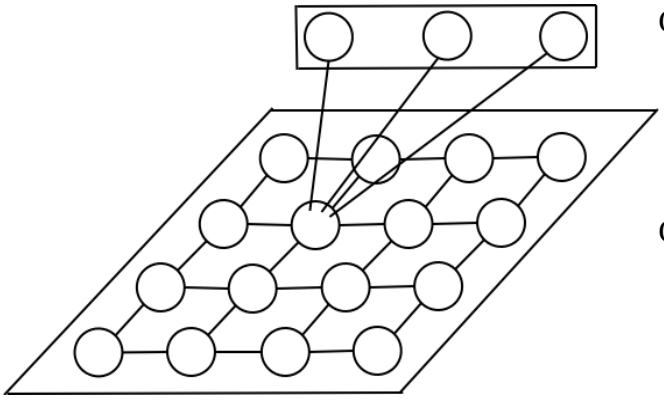


FIGURA 10 - Arquitetura da Rede Neural SOM
Cada neurônio da camada competitiva está conectado a todas as entradas.

$$\vec{W}_{(t+1)} = \vec{W}_t + a(\vec{X}_t - \vec{W}_t)$$

onde:

W: vetor de pesos do neurônio vencedor

a : constante de correção

X: vetor de entrada

Na prática, a mesma fórmula será aplicada à vizinhança do neurônio vencedor; porém, a constante de correção *a* será menor. Definir quantos e quais neurônios serão atingidos na correção da vizinhança é uma questão de projeto. A rede SOM já apresenta bons resultados com apenas 4 ou 8 neurônios vizinhos mais próximos atingidos pela correção.

A rede SOM apresenta diversas características interessantes. A camada competitiva da rede SOM processa informação em paralelo e clusteriza a entrada sem um professor. Sendo assim, a rede SOM apresenta aprendizado não supervisionado.

Exemplo 1: Dada a seguinte rede SOM:

- camada de entrada com apenas um neurônio;
- camada competitiva com 2 neurônios e pesos todos em 0;
- vizinhança formada por 1 neurônio ao lado;
- constante de correção de 0.5 para o neurônio vencedor;
- constante de correção de 0.1 para o neurônio vizinho;
- usar distância euclidiana para calcular o vencedor;

Nesse exemplo, deseja-se (a) desenhar a arquitetura da rede e (b) clusterizar as entradas -2,-1,1,2.

a) Arquitetura da Rede:

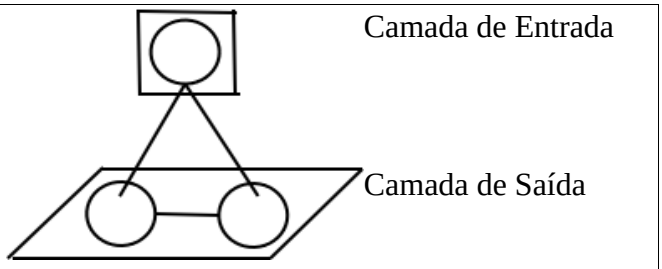


FIGURA 11 - Rede SOM BEM SIMPLES

b) clusterizar as entradas -2,-1,1,2.

- Alimenta-se o neurônio da camada de entrada com a primeira entrada -2.
- Considerando que ambos os neurônios apresentam o mesmo peso 0, escolhe-se arbitrariamente que o vencedor é o neurônio da esquerda.
- Corrige-se o neurônio vencedor na forma como segue:
  $\vec{W}_{(t+1)} = \vec{W}_t + a(\vec{X}_t - \vec{W}_t)$ = 0+0.5(-2 -0) = -1
  O peso do neurônio da esquerda passa a valer -1.
  Corrige-se o vizinho como segue:
  $\vec{W}_{(t+1)} = \vec{W}_t + a(\vec{X}_t - \vec{W}_t)$ = 0+0.1(-2 -0) = -0.2.
- Alimenta-se o neurônio da camada de entrada com a segunda entrada -1.
- Escolhendo o neurônio que apresenta a menor distância euclidiana entre o peso e a entrada como vencedor, vence o neurônio da esquerda. Corrige-se o neurônio vencedor:
  $\vec{W}_{(t+1)} = \vec{W}_t + a(\vec{X}_t - \vec{W}_t)$ = (-1)+0.5(-1 +1) = -1

  Corrige-se o vizinho da direita:

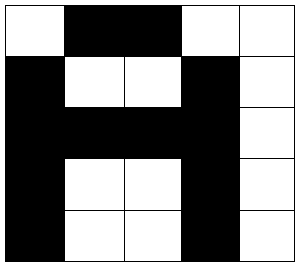
  $\vec{W}_{(t+1)} = \vec{W}_t + a(\vec{X}_t - \vec{W}_t)$ = -0.2+0.1(-1 -(-0.2)) = -0.28
- Alimenta-se o neurônio da camada de entrada com a terceira entrada 1.
  Vence o neurônio da esquerda por ter seu peso mais próximo a entrada.
  Corrige-se o vencedor:
  $\vec{W}_{(t+1)} = \vec{W}_t + a(\vec{X}_t - \vec{W}_t)$ = (-0.28)+0.5(1 -(-0.28)) = 0.36

  Corrige-se o vizinho da esquerda:

  $\vec{W}_{(t+1)} = \vec{W}_t + a(\vec{X}_t - \vec{W}_t)$ = -1+0.1(1 -(-1)) = -0.8

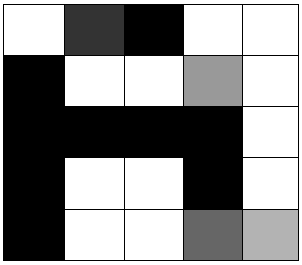
Com o peso do neurônio da esquerda em -0.8 e com o peso do neurônio da direita em 0.36, a rede SOM do exemplo já é classificada em duas classes diferentes as entradas. Quando a entrada é -2 ou -1, dispara ou neurônio da esquerda. Quando a entrada é 1 ou 2, dispara o neurônio da direita. O valor da constante de correção *a* pode sofrer decréscimo em módulo ao longo do tempo para refinar a clusterização.

No exemplo anterior, foram usados vetores unidimensionais. Em aplicações práticas, são usados vetores com múltiplas dimensões. Por exemplo, para alimentar uma rede SOM com os caracteres das figuras 12 e 13, são necessários vetor de entrada e vetor de pesos com 25 dimensões (uma dimensão para cada pixel).



-1	1	1	-1	-1
1	-1	-1	1	-1
1	1	1	1	-1
1	-1	-1	1	-1
1	-1	-1	1	-1

FIGURA 12 - Letra A codificada



-1	0.8	1	-1	-1
1	-1	-1	0	-1
1	1	1	1	-1
1	-1	-1	1	-1
1	-1	-1	0.3	-0.3

FIGURA 13- Letra A com ruído

6 Bibliografia Comentada

[SIM2001] SIMON, Haykin. **Redes Neurais: Princípios e Prática**. Bookman, 2001.

Excelente e vasto livro. Apresenta grande quantidade de modelos de neurônios e de redes neurais artificiais.

[FREE91] FREEMAN, James A; SKAPURA, David M. **Redes Neuronales**. Addinson-Wesley. 1991. p.1-92

É um bom livro para estudar ADALINE e rede SOM entre diversos outros pontos não cobertos no presente texto. Possui modelos implementados em Pascal.

[AND95] ANDERSON, James A. **An Introduction to Neural Networks**. Mit Press. London. England. 1995. p. 1-85;93-103.

Fonte bibliográfica usada na inspiração biológica do *Limulus.* Apresenta diversos modelos de redes neurais competitivas implementados.

[KAN97] KANDEL, Eric R. SCHWARTZ, James H. JESSEL, Thomas M. **Fundamentos da Neurociência e do Comportamento**. Guanabara Koogan. Rio de Janeiro. 1997. P.5-33, 110-160.

Leitura obrigatória para quem deseja ter conhecimentos de neurologia.