Redes Neurais Artificiais

Adelardo Adelino Dantas de Medeiros
http://www.dca.ufrn.br/~adelardo/

Junho de 2003

1 Neurônios artificiais

Os neurônios reais têm três partes, como ilustra a figura 1: o *corpo celular*, os *dendritos* e o *axônio*. Os dendritos têm por função receber as informações, ou *impulsos nervosos*, oriundos de outros neurônios e conduzi-las até o corpo celular. A informação é então processada e novos impulsos são gerados. Estes impulsos são transmitidos a outros neurônios, passando através do axônio até os dendritos dos neurônios seguintes.

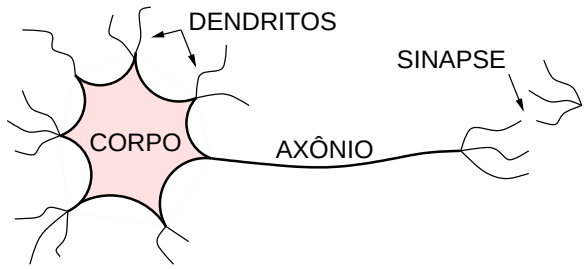


Figura 1: Componentes do neurônio biológico

O ponto de contato entre a extremidade do axônio de um neurônio e o dendrito de outro é chamado de *sinapse*. É pelas sinapses que as células se unem funcionalmente, formando redes neurais. As sinapses funcionam como válvulas, e são capazes de controlar a transmissão de impulsos — isto é, o fluxo de informação — entre os nodos na rede neural. O efeito das sinapses é variável, e é esta variação que dá ao neurônio capacidade de adaptação. Este sistema simples é responsável pela maioria das funções realizadas pelo nosso cérebro. A capacidade de realizar funções complexas surge com a operação em paralelo de todos os 10^{11} nodos do cérebro.

O neurônio artificial mais utilizado é uma simplificação do modelo de comportamento de um neurônio biológico, e baseia-se no esquema da figura 2. Para emular o comportamento das sinapses, os N sinais de entrada z_j são multiplicados pelos seus pesos correspondentes ω_j e somados. O resultado x desta soma é aplicado à função de ativação $f(x)$, que tenta modelar o comportamento do corpo celular na transmissão da informação. A saída y do neurônio é portanto dada pela equação 1:

$$y(z_1, z_2, \cdots, z_N) = f(x) = f\left(\sum_{s=1}^{N} \omega_s z_s\right) \quad (1)$$

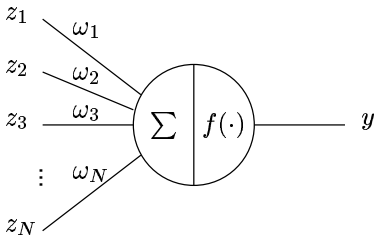


Figura 2: Neurônio artificial básico

A função de ativação deve ser monoticamente crescente e com derivada definida em todos os pontos. Pelo menos em alguns neurônios da rede, deve ser também não-linear e com saturação. As funções de ativação mais usuais são:

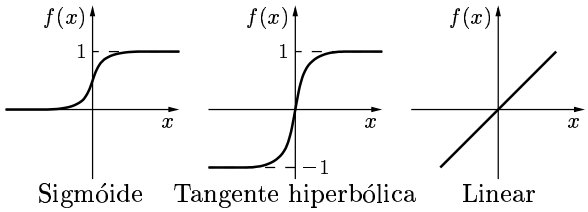
- *sigmóide*, dada pela equação 2.
- *tangente hiperbólica*, dada pela equação 3.
- *linear*, dada pela equação 4.

A figura 3 apresenta o gráfico destas funções.

$$f(x) = \frac{1}{1+e^{-x}} \qquad \dot{f}(x) = f(x)[1-f(x)] \qquad (2)$$

$$f(x) = \frac{1-e^{-x}}{1+e^{-x}} \qquad \dot{f}(x) = 2f(x)[1-f(x)] \qquad (3)$$

$$f(x) = x \qquad \dot{f}(x) = 1 \qquad (4)$$



Sigmóide Tangente hiperbólica Linear

Figura 3: Funções de ativacão mais usuais

2 Redes neurais

A capacidade de armazenamento de informação de uma rede neural reside na conectividade entre os neurônios. O arranjo mais comum para a rede é se dispor os neurônios em camadas, de tal forma que as saídas dos neurônios de uma camada sejam as entradas dos neurônios da camada seguinte. Os sinais de entrada da rede são os sinais de entrada para os neurônios da primeira camada. As saídas dos neurônios da última camada são também as saídas da rede.

Este tipo de configuração possui ao menos uma camada de entrada e uma camada de saída. Entretanto, para que a rede possa aprender problemas com um mínimo de complexidade, deve existir menos uma camada intermediária de neurônios, denominada de camada escondida. A função de ativação dos neurônios desta camada oculta deve ser não-linear.

A figura 4 mostra o esquema mais usual de arranjo de neurônios, onde se tem uma única camada intermediária e a saída de um neurônio de uma camada está ligada a todos os neurônios da camada seguinte.

Para uma rede do tipo da figura 4, estão definidos os seguintes parâmetros:

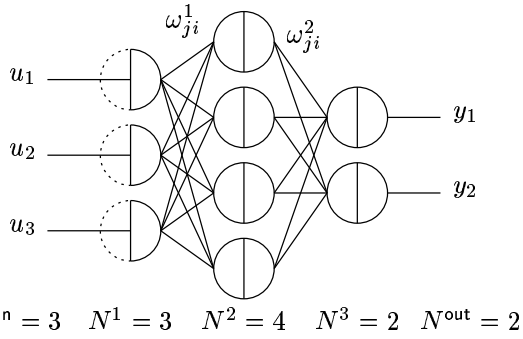


Figura 4: Rede neural com uma camada oculta

N^{in} Número de entradas da rede neural

u_j j-ésimo sinal de entrada da rede neural, $j = 1, \cdots, N^{\text{in}}$.

N^{out} Número de saídas da rede neural

y_j j-ésimo sinal de saída da rede neural, $j = 1, \cdots, N^{\text{out}}$.

M Número de camadas da rede neural ($M \geq 2$, usualmente $M \geq 3$ para que se tenha ao menos uma camada oculta).

N^k Número de neurônios na k-ésima camada, $k = 1, \cdots, M$.

ω^k_{ji} Peso do arco que conecta a saída do i-ésimo neurônio da camada k à entrada do j-ésimo neurônio da camada $k+1$, $k = 1, \cdots, M-1$, $j = 1, \cdots, N^{k+1}$, $i = 1, \cdots, N^k$.

x^k_j Somatório ponderado das entradas do j-ésimo neurônio da k-ésima camada, $k = 1, \cdots, M$, $j = 1, \cdots, N^k$.

z^k_j Saída do j-ésimo neurônio da k-ésima camada, $k = 1, \cdots, M$, $j = 1, \cdots, N^k$.

A partir da definição básica do neurônio (equação 1) e levando-se em conta a particularidade da primeira camada, chega-se a:

$$x^k_j = \begin{cases} \sum_{s=1}^{N^{k-1}} \omega^{k-1}_{js} \cdot z^{k-1}_s & k > 1 \\ u_j & k = 1 \end{cases} \qquad (5)$$

$$z^k_j = f(x^k_j) \qquad (6)$$

No tipo de rede da figura 4, o número de neurônios na primeira e na última camada são

iguais ao número de entradas e saídas da rede ($N^1 = N^{\text{in}}$, $N^M = N^{\text{out}}$), sendo a saída da rede dada pelos neurônios da última camada:

$$y_j = z_j^M \tag{7}$$

3 Aprendizado

O processo de aprendizado em uma rede como a da figura 4 consiste essencialmente em se definir valores corretos para os pesos ω_{ji}^k. Existem duas categorias de técnicas de aprendizado:

Supervisionado: a rede aprende a partir de um conjunto de pares entrada-saída fornecidos por um supervisor.

Não-supervisionado: não é apresentado nenhum conhecimento prévio à rede, que deve fazer deduções a partir de eventuais redundâncias existentes no sinal de entrada.

3.1 *Back-propagation*

A técnica de aprendizado supervisionado mais usual é o treinamento por *back-propagation*. Tomam-se as entradas de um dos pontos de treino e geram-se as saídas da rede. Calcula-se então um sinal de erro pontual, função das diferenças entre os sinais de saída gerados e os desejados para este ponto de treino.

Em seguida, calcula-se o gradiente do erro pontual com relação aos pesos da rede. Este gradiente dá a direção de máxima redução local no erro pontual. Procede-se então a uma modificação incremental dos pesos nesta direção.

Em uma iteração, este procedimento é efetuado para cada ponto de treino. A iteração se encerra com o cálculo do erro global da rede, função das diferenças entre os sinais de saída gerados e os desejados para todos os pontos. Fazem-se tantas iterações quantas necessárias até que este erro global seja aceitável ou que o número máximo de iterações seja atingido.

Para a obtenção das equações matemáticas do aprendizado por *back-propagation*, define-se:

P Número de pontos de treino

u_j^l valor do j-ésimo sinal de entrada da rede no l-ésimo ponto de treino, $j = 1, \cdots, N^{\text{in}}$, $l = 1, \cdots, P$.

d_j^l valor desejado para o j-ésimo sinal de saída da rede no l-ésimo ponto de treino, $j = 1, \cdots, N^{\text{out}}$, $l = 1, \cdots, P$.

E^l erro pontual da rede quando submetida ao l-ésimo ponto de treino, $l = 1, \cdots, P$.

H erro global da rede.

O algoritmo se baseia na minimização, para cada ponto de treino, do seguinte erro pontual:

$$E^l(\cdot) = \frac{1}{2} \sum_{s=1}^{N^{\text{out}}} (d_s^l - y_s|_{\forall j\ u_j = u_j^l})^2 \tag{8}$$

O erro global é dado por:

$$H(\cdot) = \frac{1}{P} \sum_{r=1}^{P} E^r \tag{9}$$

Para cada ponto de treino, a variação dos pesos será definida na direção oposta à do gradiente. Supondo $k > 1$:

$$\begin{aligned} \Delta\omega_{ji}^{k-1} &= -\alpha \cdot \frac{\partial E^l}{\partial \omega_{ji}^{k-1}} \\ &= -\alpha \cdot \frac{\partial E^l}{\partial z_j^k} \cdot \frac{\partial z_j^k}{\partial x_j^k} \cdot \frac{\partial x_j^k}{\partial \omega_{ji}^{k-1}} \end{aligned} \tag{10}$$

A primeira derivada na equação 10 é denominada δ_j^k, enquanto que a segunda e a terceira podem ser facilmente calculadas a partir das equações 6 e 5, respectivamente:

$$\frac{\partial E^l}{\partial z_j^k} = \delta_j^k \qquad \frac{\partial z_j^k}{\partial x_j^k} = \dot{f}(x_j^k) \qquad \frac{\partial x_j^k}{\partial \omega_{ji}^{k-1}} = z_i^{k-1}$$

Com isso:

$$\Delta\omega_{ji}^{k-1} = -\alpha \cdot \delta_j^k \cdot \dot{f}(x_j^k) \cdot z_i^{k-1} \tag{11}$$

$$\omega_{ji}^{k-1}(t+1) \leftarrow \omega_{ji}^{k-1}(t) + \Delta\omega_{ji}^{k-1}(t) \tag{12}$$

O cálculo de δ_j^k depende da camada onde o neurônio j se encontra. Na última camada ($k =$

M), ele pode ser facilmente calculado a partir das equações 7 e 8:

$$\delta_j^M = \frac{\partial E^l}{\partial z_j^M} = \frac{\partial \left[\frac{1}{2} \sum_{s=1}^{N^{\text{out}}} (d_s^l - z_s^M)^2 \right]}{\partial z_j^M} = \\ = z_j^M - d_j^l \qquad (13)$$

Caso o neurônio não esteja na camada de saída ($k \neq M$), pode-se escrever:

$$\delta_j^k = \frac{\partial E^l}{\partial z_j^k} = \frac{1}{N^{k+1}} \sum_{s=1}^{N^{k+1}} \frac{\partial E^l}{\partial x_s^{k+1}} \cdot \frac{\partial x_s^{k+1}}{\partial z_j^k} = \\ = \frac{1}{N^{k+1}} \sum_{s=1}^{N^{k+1}} \frac{\partial E^l}{\partial x_s^{k+1}} \cdot \frac{\partial \left(\sum_{r=1}^{N^k} \omega_{sr}^k \cdot z_r^k \right)}{\partial z_j^k} = \\ = \frac{1}{N^{k+1}} \sum_{s=1}^{N^{k+1}} \frac{\partial E^l}{\partial x_s^{k+1}} \cdot \omega_{sj}^k = \\ = \frac{1}{N^{k+1}} \sum_{s=1}^{N^{k+1}} \frac{\partial E^l}{\partial z_s^{k+1}} \cdot \frac{\partial z_s^{k+1}}{\partial x_s^{k+1}} \cdot \omega_{sj}^k = \\ = \frac{1}{N^{k+1}} \sum_{s=1}^{N^{k+1}} \delta_s^{k+1} \cdot \dot{f}(x_s^{k+1}) \cdot \omega_{sj}^k \qquad (14)$$

4 Algoritmos

4.1 Funcionamento

```
// Calcula a saída de uma rede neural
// Entrada: vetor u de N^in elementos
// Saída: vetor y de N^out elementos
ROTINA CALCULA_SAIDA(VETOR u, VETOR y)
|  INT k,j
|  PARA k de 1 a M
|  |  PARA j de 1 a N^k
|  |  |  CALCULA x_j^k      // Eq. 5
|  |  |  CALCULA z_j^k      // Eq. 6
|  |  FIM PARA
|  FIM PARA
|  PARA j de 1 a N^out
|  |  y_j ← z_j^M
|  FIM PARA
FIM ROTINA
```

4.2 Treinamento

```
// Treina uma rede neural
// Entrada: array u de P vetores (N^in elem)
// Entrada: array d de P vetores (N^out elem)
ROTINA TREINA_REDE(ARRAY u, ARRAY d)
|  INT n,l,k,j,i
|  n ← 0
|  REPITA
|  |  n ← n + 1
|  |  PARA l de 1 a P
|  |  |  // Fase forward
|  |  |  CALCULA_SAIDA(u^L,.)
|  |  |  // Fase backward
|  |  |  PARA k de M a 2
|  |  |  |  PARA j de 1 a N^k
|  |  |  |  |  CALCULA δ_j^k //(13) ou (14)
|  |  |  |  |  PARA i de 1 a N^(k-1)
|  |  |  |  |  |  CALCULA Δω_ji^(k-1) //(11)
|  |  |  |  |  FIM PARA
|  |  |  |  FIM PARA
|  |  |  FIM PARA
|  |  |  // Atualização dos pesos
|  |  |  PARA k de 2 a M
|  |  |  |  PARA j de 1 a N^k
|  |  |  |  |  PARA i de 1 a N^(k-1)
|  |  |  |  |  |  ATUALIZA ω_ji^(k-1) //(12)
|  |  |  |  |  FIM PARA
|  |  |  |  FIM PARA
|  |  |  FIM PARA
|  |  FIM PARA
|  |  CALCULA H //(9)
|  ENQUANTO ((H > ERRO_ACEITAVEL) E
|            (n < NUM_MAX_ITERACOES))
FIM ROTINA
```

Referências

[1] A. d. P. Braga, A. C. P. d. L. F. Carvalho, and T. B. Ludermir. *Redes Neurais Artificiais - Teoria e Aplicações*. Editora LTC, Rio de Janeiro, Brasil, 2000.