INTELIGÊNCIA ARTIFICIAL APLICADA

AULA 5

Prof. Luciano Frontino de Medeiros

UNINTER
CENTRO UNIVERSITÁRIO INTERNACIONAL

CONVERSA INICIAL

Nesta aula você estudará sobre as Redes Neurais Artificiais, uma área de pesquisa dentro da linha conexionista da Inteligência Artificial. Estudaremos sobre a estrutura das RNA dividida em camadas de neurônios, a função de ativação e um dos tipos clássicos de RNA, o Perceptron Simples e o Perceptron Multicamada.

CONTEXTUALIZANDO

Do lado da linha de pesquisa conexionista da IA, temos um dos representantes-chave, que são as Redes Neurais Artificiais. De forma análoga ao comportamento das células inteligentes dos seres vivos, os neurônios, as RNA são capazes de diferenciar padrões quando um conjunto de sinais de diversos padrões são apresentados à sua entrada. Diferente das técnicas da linha simbólica, as RNA aprendem com o ambiente, a partir de conjuntos de amostras representantes dos padrões, de forma bem similar ao que acontece nos seres vivos. Em várias atividades em que é necessário proceder uma análise de padrões complexos, até difíceis para seres humanos, técnicas de RNA são bastante importantes para evidenciar novos conhecimentos. Veremos agora os conceitos importantes relacionados às RNA.

TEMA 1: INTRODUÇÃO A REDES NEURAIS ARTIFICIAIS

O trabalho em Redes Neurais Artificiais (RNA) tem sido motivado desde o começo pelo reconhecimento de que o cérebro humano processa informações de uma forma inteiramente diferente de um computador convencional. O cérebro é um computador altamente complexo, não linear e paralelo possui a capacidade de organizar seus constituintes estruturais, os neurônios, de forma a realizar certos processamentos tais como reconhecimento de padrões, percepção e controle motor, de forma mais rápida que o mais rápido computador existente (HAYKIN, 2004, p. 27).

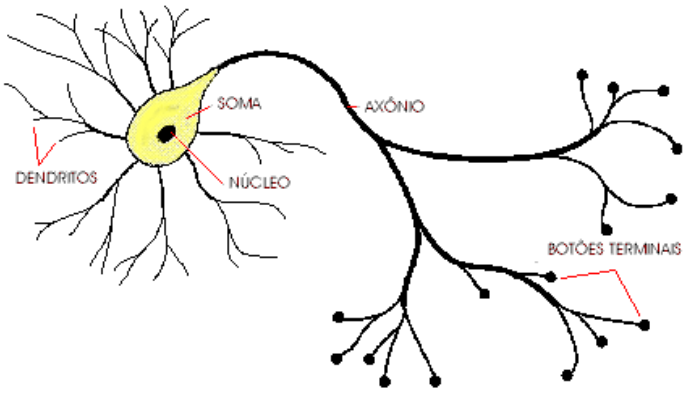
Considere um exemplo da natureza: o sonar de um morcego é um sistema ativo de localização por eco. Além de fornecer informações sobre a distância até o alvo, o sonar transmite também informação sobre a velocidade relativa do alvo, tamanho de várias características do alvo e a elevação do alvo. Toda a complexa computação ocorrendo num cérebro do tamanho de uma ameixa! (HAYKIN, 2004, p. 27).

Os neurônios são as unidades fundamentais dos tecidos do sistema nervoso, incluindo o cérebro. Cada neurônio consiste de um corpo celular, também designado como soma, o qual contém um núcleo. Partindo do corpo da célula existem um número de filamentos denominados dendritos, e um filamento mais longo que é denominado de axônio (Figura 1). Os dendritos ligam-se ao redor da célula a outras células e o axônio faz uma conexão mais longa. A estas conexões dá-se o nome de sinapses.

O sinal de uma célula a outra se faz mediante uma complicada reação eletroquímica. Substâncias químicas transmissoras são lançadas das sinapses e entram pelos dendritos, aumentando ou baixando o potencial elétrico do corpo da célula. Quando o potencial chega a um limiar, um pulso elétrico ou potencial de ação é mandado pelo axônio. O pulso espalha-se ao longo das conexões existentes pelo axônio, eventualmente chegando a outras sinapses e lançando transmissores ao corpo de outras células.

Sinapses que incrementam o potencial de outras células são denominadas excitatórias, enquanto que as que decrementam são denominadas inibitórias. Os neurônios podem formar novas conexões com outros neurônios, e tais mecanismos são por meio dos quais se forma a base para o aprendizado do cérebro.

Figura 1: Ilustração de um neurônio biológico



Fonte: adaptado de Medeiros (2007).

Uma rede neural biológica pode, dessa forma, ser abstraída para simular o seu comportamento. Dessa forma, podemos conceituar uma rede neural artificial como um processador maciçamente e paralelamente distribuído constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso (HAYKIN, 2004).

Uma rede neural artificial tem uma série de propriedades (HAYKIN, 2004, p.30):

Não linearidade: neurônios podem ser lineares ou não lineares, dessa forma, permitem aproximações robustas de funções de mapeamento que tenham característica não linear.

Mapeamento Entrada-Saída: A rede aprende a partir de exemplos, estabelecendo mapeamento entre os padrões apresentados na entrada com as saídas dadas pelos exemplos.

Adaptabilidade: redes neurais podem ser treinadas e armazenar o conhecimento nos pesos sinápticos, podendo se adaptar caso o conjunto de amostras utilizado para o treinamento se modifique ao longo do tempo.

Resposta a Evidências: uma rede neural pode perfazer uma tarefa de seleção de um padrão, mas também informar sobre o grau de confiança ou crença referente ao padrão escolhido.

Informação Contextual: o conhecimento é armazenado na própria estrutura e pela ativação da rede neural.

Tolerância a falhas: redes que sejam implementadas em hardware são tolerantes a falhas, em caso de neurônios ou conexões que possam ser danificados, ou mesmo em software utilizando técnicas de poda de redes, que reduzem a quantidade de neurônios ou sinapses, mantendo a mesma condição de *performance*.

Uma rede neural artificial tem a sua construção dependente de alguns elementos:

Número de camadas: as RNA possuem pelo menos uma camada de entrada, de onde recebe os sinais ou características das amostras, e uma camada de saída que apresenta os padrões ou classes, mapeados para os conjuntos de treinamento. Também podem possuir uma ou mais camadas ocultas, como no caso do perceptron multicamada e das redes de base radial.

Quantidade de neurônios por camada: a quantidade de neurônios irá depender da natureza do problema sendo abordado. A camada de entrada terá tantos neurônios conforme as características das amostras do conjunto de treinamento. A camada de saída terá os neurônios referentes às classes a que pertencem as amostras do conjunto de treinamento. A camada oculta ou escondida pode ter a quantidade de neurônios variável, conforme a característica do mapeamento que se deseja.

Tipo de função de transferência: a função de transferência define a ativação do neurônio. Podem ser utilizadas funções discretas (tais como a função degrau, utilizada no perceptron a ser explicado adiante) ou funções contínuas (como a função sigmoide para o perceptron multicamada).

Método de treinamento: ao longo dos anos foram desenvolvidos diversos métodos ou algoritmos de treinamento. Dentre os métodos mais

utilizados está o algoritmo de retropropagação (*backpropagation*), que utiliza a informação do erro na atualização dos pesos.

As RNA podem aprender de diversas formas:

Aprendizagem por correção de erros: a informação do erro é utilizada para modificar os pesos sinápticos.

Aprendizagem baseada em memória: um grande número de exemplos de entrada e saída são armazenados, e uma amostra é comparada com a sua vizinhança para se identificar a classe à qual pertence.

Aprendizagem hebbiana: com base nos estudos de Hebb, utiliza uma regra associativa, que aumenta a força dos pesos positivamente correlacionados ou diminui daqueles negativamente correlacionados.

Aprendizagem competitiva: neste tipo de aprendizagem, os neurônios competem entre si, tendo-se um vencedor que estará ativo em um certo instante (aprendizagem denominada também de *winner-takes-all*).

Aprendizagem de Boltzmann: aprendizagem com características estocáticas derivada das ideias da mecânica estatística. Nesse caso, os neurônios constituem uma estrutura recorrente e operando de maneira binária, controlados por uma função de energia. A atualização dos pesos se dá por correlação, operando em duas condições: uma condição presa (os neurônios visíveis estão presos a estados específicos) e outra condição livre (todos os neurônios podem operar livremente).

A aprendizagem ainda pode ser caracterizada como supervisionada (em que há o *feedback* que retorna à rede para orientar o treinamento) e não supervisionada (a rede aprende de forma auto-organizada). Quanto às tarefas que podem ser executadas por uma RNA estão:

- Associação de padrões;
- Reconhecimento de padrões;
- Aproximação de funções;
- Controle;

- Filtragem.

TEMA 2: O PERCEPTRON

O perceptron foi um dispositivo eletrônico inventado em 1957 por Frank Rosenblatt (1928-1971), psicólogo americano, considerado uma espécie de "homem da renascença" devido à sua excelência em várias áreas, incluindo computação, matemática, neurofisiologia, astronomia e música. O perceptron foi construído de acordo com princípios biológicos e que mostrava capacidade de aprendizado. Era organizado em três camadas de unidades (ou neurônios): unidades sensoriais (S), associativas (A) e geradoras de respostas (R), alimentada à frente (Ver figura 2).

Um dos protótipos construídos por Rosenblatt foi o Mark I. O Mark I era um perceptron implementado em hardware com 400 unidades fotosensoras, camada de associação de 512 unidades e camada de saída de 8 unidades.

Figura 2: Esboço do perceptron Mark I

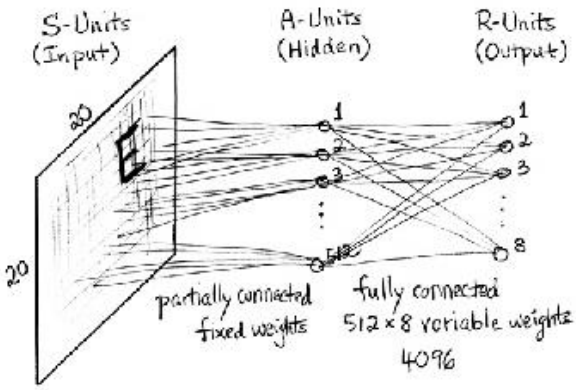
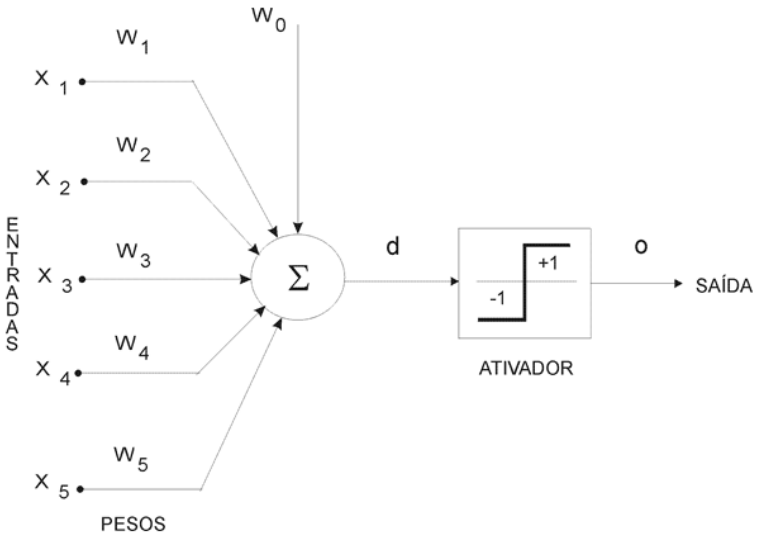


Figura 3: Arquitetura de um perceptron. As variáveis significam: x – valores de entrada; w – valores dos pesos; d – saída intermediária; o – saída ativada



Fonte: adaptado de Medeiros (2007).

Para saber mais...	
Frank Rosenblatt contribuiu para o campo das redes neurais de forma significativa com o invento do perceptron. Para conhecer um pouco mais sobre seu trabalho, acesse o *link*: <http://csis.pace.edu/~ctappert/srd2011/rosenblatt-contributions.htm>, ou por meio do QR Code ao lado.	

O perceptron obtém os sinais do ambiente por meio das entradas, nas quais são apresentados os valores correspondentes aos padrões que queremos classificar, por exemplo, valores de pixels de imagem ou características de um produto. Na figura 3 o perceptron possui 5 (cinco) entradas.

Fazendo parte da estrutura interna do perceptron, temos os pesos ou sinapses. Os pesos assumirão valores tais que, quando aplicarmos um padrão na entrada, obteremos uma saída intermediária d. O aprendizado da rede ficará armazenado nos pesos e seus valores são obtidos mediante um processo de treinamento. O valor w0. é chamado de bias, sendo fixo e entendido como uma espécie de ajuste fino, o qual não multiplica com entrada nenhuma.

A saída intermediária d é calculada mediante o somatório da multiplicação entre cada entrada e seu peso. Ou seja,

d = w1x1 + w2x2 + w3x3 + w4x4 + w5x5 + w0

ou, de forma genérica (com n=5),

$$d = w_0 + \sum_{i=1}^{n} x_i w_i \qquad (1)$$

Em linguagem de programação (Pascal), poderíamos construir o algoritmo a seguir representando este cálculo:

```
function Soma: double;
var w, x: array[0..5] of double;
   d: double;
   i: integer;
begin
...
// Aqui seriam associados os valores para os vetores w e x
...
     d := w[0];
     for i := 1 to 5 do
          d := d + w[i]*x[i];
     Result := d;
end;
```

Após o cálculo da saída intermediária d, precisamos então calcular agora a saída ativada o (do inglês *output*). Como visto na figura 3, a saída ativada o depende do resultado da saída intermediária d. Caso o resultado obtido em d for maior ou igual a 0 (zero), o será igual a +1(um); se d for menor que 0 (zero), o será igual a –1 (menos um). Ou, em forma de código,

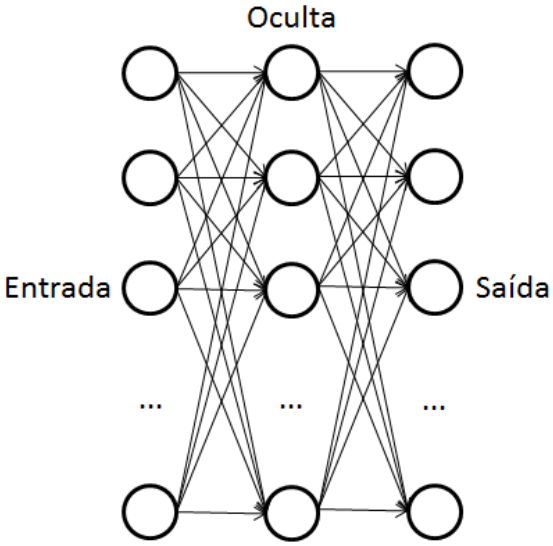
```
function SaidaAtivada(d: double): integer;
var i: integer;
begin
     if d > = 0 then
          i := 1
     else
          i := -1;
     Result := i;
end;
```

TEMA 3: PERCEPTRON MULTICAMADA

O perceptron multicamada se diferencia do perceptron simples por incluir camadas escondidas ou ocultas na RNA. Com a inclusão de camadas ocultas, o número de pesos ou sinapses aumenta consideravelmente (Figura 4). A consequência disso é a possibilidade de melhorar o mapeamento das entradas com as saídas. Um perceptron simples trabalha de forma linear, enquanto que o perceptron multicamada tem condições de lidar com conjuntos de treinamento cuja separabilidade seja não linear.

A arquitetura com camadas ocultas requer algoritmos de aprendizagem que contemplem a atualização dos pesos relacionados às camadas internas. O processo de ativação acontece primeiramente nas camadas ocultas para depois chegar até a camada de saída. A retroalimentação do erro também é feita nos pesos que conectam a(s) camada(s) oculta(s).

Figura 4: Representação de uma rede perceptron multicamada (MLP), com uma camada oculta, com todos os nós conectados.



A dinâmica do perceptron multicamada envolve o processamento de dois tipos de sinais (figura 5) se propagando pela rede (HAYKIN, 2004, p. 186-187):

Sinal funcional: é o sinal apresentado à camada de entrada referente aos atributos do vetor de amostras, que propaga-se para a frente na rede, nó por nó, ativando os neurônios até a camada de saída.

Sinal de erro: tem origem em um neurônio da camada de saída, porém se propagando para trás na rede, ajustando os valores dos pesos ou sinapses.

O algoritmo de retropropagação para o perceptron multicamada envolve a seguinte notação constante na Tabela 1. O algoritmo envolve o processo chamado de descida de gradiente. Esse processo busca calcular o gradiente local do erro (a direção para onde tende a crescer o valor do erro médio calculado), utilizando-o para corrigir os pesos sinápticos na direção contrária a esse gradiente, em busca do erro mínimo local.

Figura 5: Representação do sinal funcional, que se propaga à frente, e do sinal de erro, que retropropaga na rede.

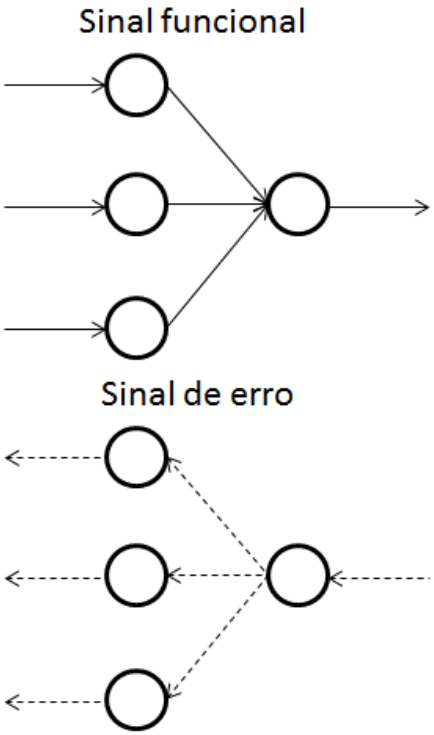


Tabela 1: Descrição das variáveis nas fórmulas correspondentes ao algoritmo de retropropagação do percetpron multicamada, relacionadas ao grafo de propagação do sinal na figura 22

Variável	Descrição
i, j, k	Índices para os neurônios. Usando a notação em que o sinal funcional se propaga da esquerda para a direita, O neurônio i está numa camada mais à esquerda; o índice j.
E	Soma dos erros quadráticos.
ej	Sinal de erro na saída do neurônio j.
dj	Sinal desejado no neurônio de saída j.
yj	Sinal funcional que aparece na saída do neurônio j.

xj	Sinal da amostra apresentada no neurônio j da camada de entrada.
wji	Peso sináptico conectando a entrada do neurônio i à saída do neurônio j.
Δwji	Correção aplicada ao peso sináptico que conecta a entrada do neurônio i à saída do neurônio j.
vj	Soma ponderada de todas as entradas sinápticas no neurônio j acrescida do bias.
δj	Gradiente local para o neurônio j.
f	Função de ativação.
η	Taxa de aprendizado.
α	Taxa de momentum

De forma geral, as etapas referentes ao algoritmo de retropropagação são descritas a seguir:

1. Construa uma rede MLP totalmente conectada, com N neurônios na camada de entrada, M neurônios na camada de saída e P neurônios na camada oculta, conforme as características do problema.

2. Inicialize os pesos com valores aleatórios e os pesos w0 (bias) com valor +1.
   -

3. Escolha uma amostra na forma de um par entrada-saída desejada (xi, di). Assim, tal como na Figura 6, na camada de entrada, os valores do neurônio yi serão iguais aos xi.

4. Propague o sinal dos neurônios da camada de entrada yi para a camada intermediária por meio do somatório ponderado pelos pesos sinátpicos wji para calcular vj (também chamado de campo local induzido) e utilize a função de ativação para calcular os valores de entrada yj para a camada intermediária:

$$v_j = \sum_{i=0}^{N} w_{ji} y_i \quad \text{e} \quad y_j = f(v_j)$$

Uma função bastante utilizada para perceptrons multicamada é a função sigmoide. A função sigmoide tem um comportamento mais “suave” do que a função degrau, utilizada no exemplo do perceptron simples. Enquanto que a função degrau possui apenas duas opções (+1 ou -1), a função sigmoide calcula valores infinitos dependendo do argumento da função. Assim, enquanto que a função degrau é uma função discreta, a função sigmoide é considerada uma função contínua. Veja o gráfico desta função na Figura 7.

A fórmula da função sigmoide[1] é

$$f(v_j) = \frac{1}{1 + e^{-av_j}}, a > 0$$

5. Propague o sinal dos neurônios da camada intermediária yj para a camada de saída por meio do somatório ponderado pelos pesos sinátpicos wkj para calcular o campo local induzido vk, utilizando a função sigmoide para calcular os valores de entrada yk para a camada de saída:

$$v_k = \sum_{j=0}^{P} w_{kj} y_j \quad \text{e} \quad y_k = f(v_k)$$

[1] O símbolo e utilizado é o número de Euler (aproximadamente 2,71828...). Não onfundir com o valor do erro e_j. A variável a regula a inflexão da curva sigmoide, sendo comum adotar $a=1$.

6. Calcule o gradiente local para os neurônios da camada de saída δk de acordo com a fórmula (no caso de se utilizar a função sigmoide)

$$\delta_k = ay_k\left(d_k - y_k\right)\left(1 - y_k\right)$$

7. Calcule o gradiente local para os neurônios da camada intermediária δj conforme a fórmula:

$$\delta_j = ay_j\left(1 - y_j\right)\sum_{k=0}^{M}\delta_k w_{kj}$$

8. Atualize o valor dos pesos conforme os valores dos gradientes locais calculados para a camada de saída e a camada intermediária. O índice sobrescrito t+1 indica o próximo valor das atualizações dos pesos, sendo o índice sobrescrito t o valor atual.

$$\Delta w_{kj}^{t+1} = \alpha\Delta w_{kj}^{t} + \eta\delta_k y_k$$

e

$$\Delta w_{ji}^{t+1} = \alpha\Delta w_{ji}^{t} + \eta\delta_j y_j$$

O valor de atualização compõe-se de duas partes: uma referente ao valor atual do peso, multiplicado pela taxa de momento α; e outra referente à modificação calculada pelo gradiente local, multiplicada pela taxa de aprendizagem η. Os pesos sinápticos devem ser atualizados então:

$$w_{kj}^{t+1} = w_{kj}^{t} - \Delta w_{kj}^{t}$$

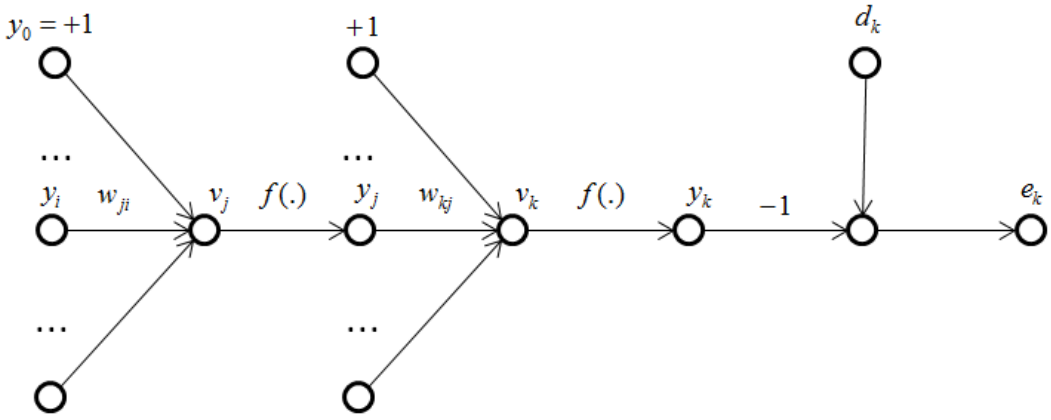
e

$$w_{ji}^{t+1} = w_{ji}^{t} - \Delta w_{ji}^{t}$$

9. Retorne ao passo 3 e escolha outro par entrada-saída. Quando todos os pares forem apresentados à rede, significa que uma época de treinamento foi alcançada. Dessa forma, refaz-se o processo para várias épocas até que o valor global de erro alcance um valor que possibilite a classificação ótima pela rede MLP. O erro global pode ser calculado pela fórmula seguinte, em que C indica o somatório para todas as amostras apresentadas ao perceptron multicamada:

$$E = \frac{1}{2}\sum_{C} e_k^2 = \frac{1}{2}\sum_{C}(d_k - y_k)^2$$

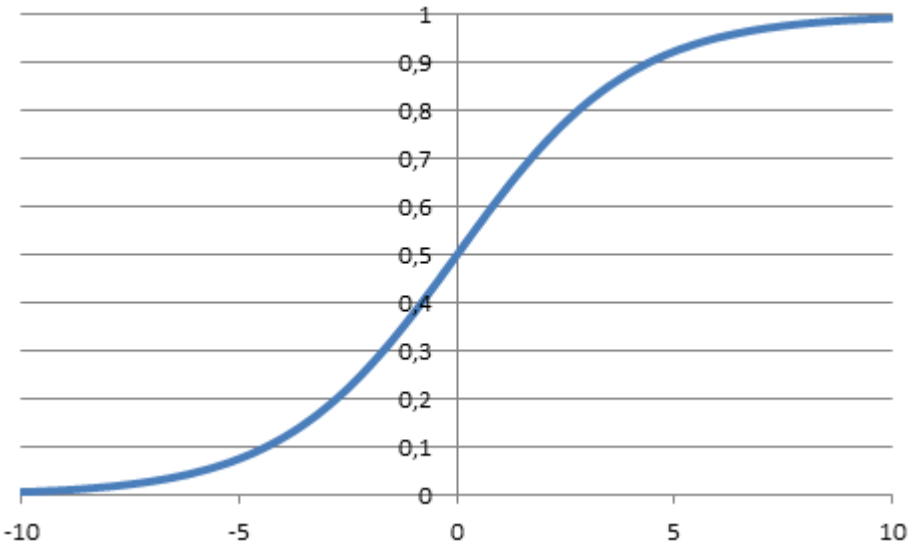
Figura 6: Representação da propagação do sinal funcional à frente na rede MLP, a partir da entrada, passando pela ativação até chegar à camada de saída



Fonte: adaptado de Haykin (2004, p. 192).

Este processo do algoritmo de retropropagação foi descrito utilizando uma camada intermediária, mas pode ser estendido para o caso de a rede ter mais camadas ocultas.

Figura 7: Gráfico da função de ativação sigmoide, que permite obter a ativação dos neurônios da rede MLP de forma suave



Normalização

A utilização do perceptron multicamada para vários tipos problemas requer que os valores dos neurônios das camadas de entrada e saída sejam modificados para trabalhar numa faixa específica, que facilite a codificação do algoritmo e evite possíveis erros relacionados com *overflow* de variáveis.

Utilizando um exemplo de segmentação relativo a investidores de um banco, a Figura 8 mostra algumas variáveis que podem ser utilizadas para segmentar os clientes. Pode-se notar que elas têm condições de assumir valores distintos. Enquanto um valor para o prazo de investimento chega a ter um valor máximo de 35, a faixa para a variável renda pode chegar a 30000. Quando alimentamos a camada de entrada da rede MLP, precisamos fazer com que a rede entenda tais valores como se pertencessem a uma faixa específica. Isso é conseguido a partir da normalização.

Figura 8: Variáveis utilizadas no exemplo de segmentação de clientes investidores, mostrando a variabilidade das faixas para as diferentes variáveis

Variável	Medida	Mínimo	Máximo
Idade	Anos	10	60
Renda	R$	R$ 0	R$ 30000
Prazo de Investimento	Meses	1	36
Escolaridade	-	0	3
Sexo	-	0	1

Valores muito diferentes !!!

Podemos normalizar os valores para uma faixa específica, por exemplo, [0,1]. Ou seja, os neurônios assumirão apenas valores dentro dessa faixa. É necessário então transformar os valores de entradas para que fiquem dentro dessa faixa. A Figura 9 mostra como a variável renda pode ser normalizada. O valor mínimo que a variável pode assumir é zero; então, o neurônio assumirá o valor “0”. O valor máximo que ele poderá assumir será de 30000; então, este valor corresponderá ao valor do neurônio “1”. Para esse tipo de normalização, podemos calcular por regra de três simples. No caso de um valor qualquer, digamos, 5000, o cálculo mostra que na proporção da faixa normalizada [0,1], o valor correspondente será de 0,16. Na Figura 10 temos mais exemplos, que já induzem como expressarmos a fórmula para se utilizar dentro do algoritmo de treinamento.

Figura 9: Ilustração da normalização para a variável renda. Os valores mínimos e máximos são utilizados numa regra de três para obter os valores normalizados na faixa [0,1]

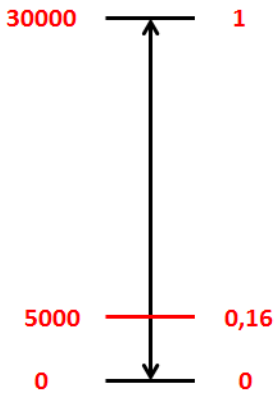
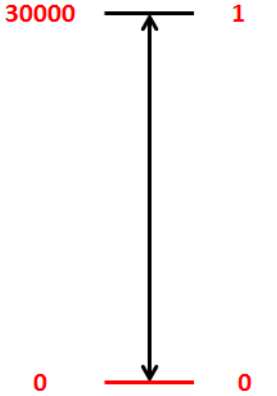
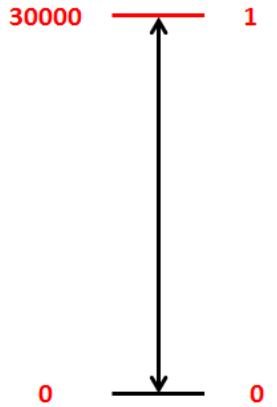
	Diagrama	Cálculo
Exemplo		Pega-se o valor atual da renda (digamos, 5000) Faz-se a diferença 5000 – 0 (0 é o valor mínimo) = 5000 Divide-se pela diferença 30000 – 0 (máximo – mínimo) = 30000 5000 / 30000 = 0,16
Mínimo		Quando o valor é o valor mínimo: 0 – 0 = 0 Divide-se pela diferença (máximo – mínimo) = 30000 – 0 = 30000 Calculando: 0 / 30000 = 0
Máximo		Quando o valor é o valor mínimo: 30000 – 0 = 30000 Divide-se pela diferença (máximo – mínimo) = 30000 – 0 = 30000 Calculando: 30000 / 30000 = 1

Figura 10: Mais exemplos mostrando o valor original da variável renda, e o valor normalizado para ser alimentado à rede neural

Valor	Cálculo	Valor da Entrada do Neurônio
7500	(7500 – 0)/(30000 – 0)	0,25
15000	(15000 – 0)/(30000 – 0)	0,5
23750	(23750 – 0)/(30000 – 0)	0,79
29000	(29000 – 0)/(30000 – 0)	0,97

A fórmula que poderemos utilizar para a normalização será:

$$x_{normalizado} = \frac{x_{entrada} - x_{\min}}{x_{\max} - x_{\min}}$$

SÍNTESE

Neste capítulo foi vista uma introdução ao estudo das Redes Neurais Artificiais, as quais fazem parte da linha de pesquisa conexionista. A base para o estudo é a forma como os neurônios se conectam em uma rede neural, trocando informações entre si para a resolução de algum problema tal como reconhecimento de padrões. Uma rede neural tem as propriedades de não linearidade, mapeamento entrada-saída, adaptabilidade, resposta a evidências e tolerância a falhas. Uma RNA depende da quantidade de camadas de entrada, ocultas ou de saída, da quantidade de neurônios por camada, do tipo de função de transferência e do método de treinamento. RNAs podem aprender de diversas formas: por correção de erros, baseada em memória, aprendizagem hebbiana, aprendizagem competitiva ou aprendizagem

de Boltzmann. A aprendizagem pode ser supervisionada ou não supervisionada. RNAs podem executar diferentes tarefas tais como reconhecimento, associação de padrões, aproximação de funções, controle e filtragem. O perceptron é um modelo clássico de RNA.

O treinamento de um perceptron exige a definição de uma taxa de aprendizagem. O algoritmo básico para treinamento de um perceptron consiste em calcular o erro e propagar o erro para trás na rede de forma a atualizar os valores das sinapses. Se um perceptron precisa aprender a classificar corretamente duas classes, se tais classes forem separáveis linearmente, o perceptron sempre aprenderá a classificar corretamente. O valor ótimo dos pesos é aquele que irá minimizar o erro global. O problema do XOR demonstrou restrições do perceptron simples original estudado por Rosenblatt, que poderia ser resolvido adicionando-se camadas ocultas e tornando-o um perceptron multicamadas. O algoritmo de treinamento de um perceptron multicamadas é denominado algoritmo de retropropagação. Geralmente adota-se como função de transferência para um perceptron multicamadas a função sigmoide. A normalização é um passo importante adotado antes de se fazer o treinamento de uma RNA.

REFERÊNCIAS

RUSSEL, S.; NORVIG, P. **Inteligência Artificial**. Tradução da 2. ed. Rio de Janeiro: Campus, 2004.

MEDEIROS, L. F. **Redes Neurais em Delphi**. 2. ed. Florianópolis: Visualbooks, 2007.

HAYKIN, S. **Redes Neurais**: Princípios e prática. Porto Alegre, Bookman, 2001.