Fundamentos das Redes Neurais Artificiais: Backpropagation

Arthur Dantas Mangussi e Ana Carolina Lorena

Instituto Tecnológico de Aeronáutica (ITA)

20 de Outubro de 2025

Redes Neurais Alimentadas Adiante

- Uma **rede neural feedforward** propaga a informação apenas da entrada para a saída.
- Não há conexões recorrentes nem atrasadas.
- A **não linearidade** é essencial para resolver problemas como o *XOR*.
- Utiliza-se uma **função de ativação** (ou transferência) para introduzir essa não linearidade.
- Exemplos de funções de ativação:
  - Sigmoide, tangente hiperbólica, ReLU, entre outras.

Função de Ativação

O que é Função de Ativação?

- Regra para mapeamento das entradas somadas, v, do neurônio até sua saída e, por uma escolha adequada, isto significa a introdução de uma capacidade de processar não linearidade na rede;
- Na prática, estas funções são escolhidas de tal forma a serem monotônicas e saturar nos extremos $[0, 1]$ ou $[-1, 1]$;
- Existem vários tipos de funções de ativação, sendo que as primeiras a serem utilizadas são as clássicas: linear, sigmoide, linear por partes, função de limiar, arco tangente, seno, cosseno, tangente hiperbólica, etc.

Perceptron de Múltiplas Camadas (MLP)

- Uma rede com múltiplas camadas é chamada de **Perceptron de Múltiplas Camadas (MLP)**.
- Estrutura geral: **MLP**(m, n, o) onde:
  - $m =$ número de entradas;
  - $n =$ número de neurônios nas camadas ocultas;
  - $o =$ número de saídas.
- Camadas:
  - Camada de entrada;
  - Uma ou mais camadas ocultas;
  - Camada de saída.
- Cada neurônio aplica uma **soma ponderada** e uma **função de ativação**.
- O treinamento é feito com o **algoritmo de backpropagation**, proposto por Rumelhart, McClelland e Williams (1986).

Derivação Matemática do Algoritmo de Backpropagation

Definição

- O algoritmo de backpropagation é utilizado para aprender quais são os pesos de uma rede neural de múltiplas camadas;
- Para isso, é utilizado o método do gradiente descendente para minimizar a soma dos erros quadrados entre os valores de saída da rede e os valores desejados;
- Conceitualmente, ocorre uma propagação de ativações ao longo das sinapses para produzir uma saída e os erros são retropropagados para realizar as mudanças dos pesos.

Derivação Matemática do Algoritmo de Backpropagation

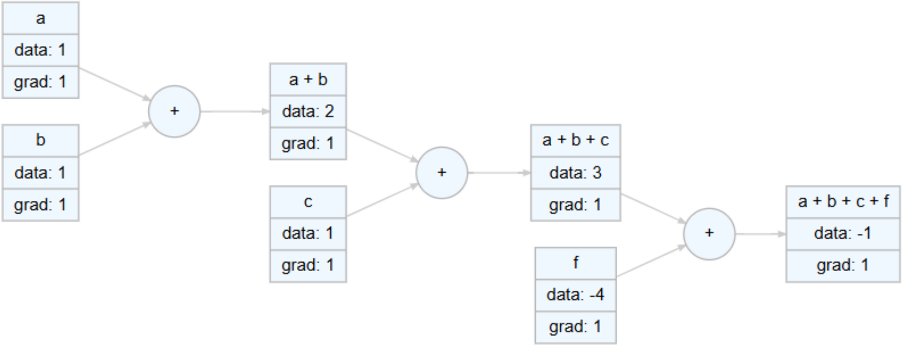

Figure: Exemplo básico da lógica do algoritmo de Backpropagation.

Derivação Matemática do Algoritmo de Backpropagation - Notação

- k denota que o neurônio está na camada de saída, ou a direita do neurônio j;
- j denota que o neurônio está na camada oculta, ou a direita do neurônio i;
- i denota que o neurônio está na camada de entrada;
- $w_{kj}(n)$ denota o peso da sinapse que liga a camada oculta à camada de saída na iteração n
- $w_{ji}(n)$ denota o peso da sinapse que liga a camada de entrada à camada oculta na iteração n
- $e_j(n)$ denota o sinal de erro na saída do neurônio j na iteração n
- $d_j(n)$ denota a saída esperada na saída no neurônio j na iteração n
- $y_j(n)$ denota o valor de saída do neurônio j na iteração n
- $v_j(n)$ denota o campo local induzido no neurônio j na iteração n
- $\varphi_j(.)$ denota a função ativação associada ao neurônio j
- b_j denota o bias aplicado ao neurônio j
- η denota a taxa de aprendizagem

Derivação Matemática do Algoritmo de Backpropagation

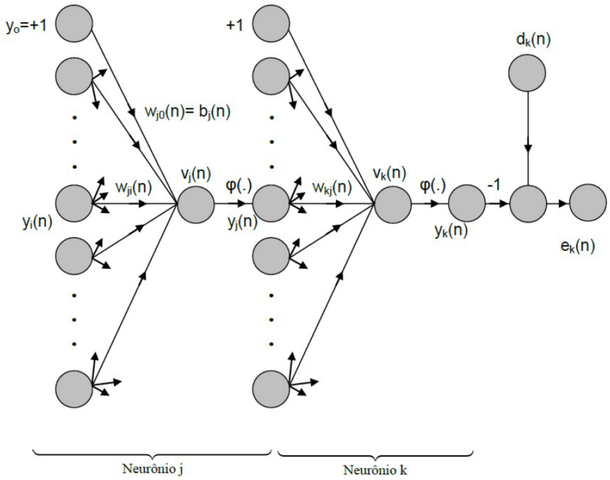

Figure: Arquitetura básica de neurônios *j* e *k*

Derivação Matemática do Algoritmo de Backpropagation

Inicialmente, considerados as seguintes expressões para começar a derivação matemática:

$$e_j(n) = d_j(n) - y_j(n) \tag{1}$$

$$E(n) = \frac{1}{2}\sum_j e_j^2(n) \tag{2}$$

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} \tag{3}$$

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) \tag{4}$$

$$y_j(n) = \varphi_j(n)(v_j(n)) \tag{5}$$

Objetivo: ajustar todos os pesos da rede neural a fim de reduzir o erro total (minimização).

Derivação Matemática do Algoritmo de Backpropagation

A função de erro ($E(n)$) não é diretamente uma função do peso, dessa maneira aplica-se a regra da cadeia:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{6}$$

Derivação Matemática do Algoritmo de Backpropagation

Calculando cada termo da Equação 6:

$$\frac{\partial E(n)}{\partial e_j(n)} = \frac{\partial}{\partial e_j(n)}\left[\frac{1}{2}\sum_j e_j^2(n)\right] = e_j(n) \tag{7}$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = \frac{\partial}{\partial y_j(n)}\left[d_j(n) - y_j(n)\right] = -1 \tag{8}$$

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \frac{\partial}{\partial v_j(n)}\left[\varphi_j(n)(v_j(n))\right] = \varphi_j'(n)(v_j(n)) \tag{9}$$

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = \frac{\partial}{\partial w_{ji}(n)}\left[\sum_{i=0}^{m} w_{ji}(n) y_i(n)\right] = y_i(n) \tag{10}$$

Derivação Matemática do Algoritmo de Backpropagation

Substituindo os resultados de 7, 8, 9 e 10 em 6, obtém-se:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = e_j(n)(-1)\varphi_j'(n)(v_j(n))y_i(n) \tag{11}$$

ou

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi_j'(n)(v_j(n))y_i(n) \tag{12}$$

A seguir é definida uma quantidade denominada de gradiente local do erro, denotada por $\delta_j(n)$, e calculada como:

$$\delta_j(n) = -\frac{\partial E(n)}{\partial v_j(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \tag{13}$$

A partir das derivadas parciais anteriormente calculadas, obtém-se:

$$\delta_j(n) = e_j(n)\varphi_j'(n)\left(v_j(n)\right) \tag{14}$$

Derivação Matemática do Algoritmo de Backpropagation

Porque usar o gradiente local?

De acordo com Haykin (2001) a vantagem em utilizar o gradiente local é que ele aponta para as modificações necessárias nos pesos sinápticos. O gradiente local é uma medida de quanto do erro da saída é devido a um peso particular $w_{ji}(n)$.

Por fim, substituindo o resultado da Equação 12 e 14 em 3, obtém-se:

$$\Delta w_{ji} = -\eta e_j(n)\varphi'_j(n)(v_j(n))y_i(n)$$

$$\Delta w_{ji} = -\eta \delta_i(n) y_i(n) \tag{15}$$

Regra Delta Generalizada é dada pela Equação 15.

Derivação Matemática do Algoritmo de Backpropagation

Nesse ponto é necessário considerar duas situações distintas:

1. Se o neurônio j é um nó de saída, então $\delta_j(n)$ é o produto da derivada da função de ativação, $\varphi'_j(v_j(n))$, e a diferença no j-ésimo componente entre o valor desejado e a saída da rede;
2. Se o neurônio j é um nó oculto da rede, não existe um valor desejado especificado para este neurônio. A solução é retropropagar os valores dos gradientes locais camada por camada através da rede, de tal forma a fazer com que os pesos sejam atualizados.

Derivação Matemática do Algoritmo de Backpropagation

A partir da definição de gradiente local, consideramos o caso em que o neurônio j está em uma camada oculta, então:

$$\delta_j(n) = -\frac{\partial E(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)}$$

$$\delta_j(n) = -\frac{\partial E(n)}{\partial y_j(n)} \varphi_j'(n)(v_j(n)) \tag{16}$$

Considerando que o neurônio k é de saída, a função de erro $E(n)$ é dado por:

$$E(n) = \frac{1}{2} \sum_k e_k^2(n) \tag{17}$$

Derivação Matemática do Algoritmo de Backpropagation

Considere

$$e_k = d_k(n) - y_k(n) = d_k(n) - \varphi_k(v_k(n))$$

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n) y_j(n)$$

Logo,

$$\frac{\partial E(n)}{\partial y_j(n)} = \frac{\partial}{\partial y_j(n)} \left[\frac{1}{2} \sum_k e_k^2(n) \right] = \sum_k e_k(n) \frac{\partial e_k}{\partial y_j(n)} \tag{18}$$

$$\frac{\partial e_k}{\partial y_j(n)} = \frac{\partial e_k}{\partial v_k(n)} \frac{\partial v_k}{\partial y_j(n)} = -\varphi_k'(v_k(n)) \frac{\partial v_k}{\partial y_j(n)}$$

$$\frac{\partial v_k}{\partial y_j(n)} = \frac{\partial}{\partial y_j(n)} \left[\sum_{j=0}^{m} w_{kj}(n) y_j(n) \right] = w_{kj}(n)$$

Derivação Matemática do Algoritmo de Backpropagation

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k(n)[-\varphi'_k(v_k(n))]w_{kj}(n) \tag{19}$$

ou

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_k e_k(n)\varphi'_k(v_k(n))w_{kj}(n) \tag{20}$$

Aplicando a definição do gradiente local

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_k \delta_k(n)w_{kj}(n) \tag{21}$$

Derivação Matemática do Algoritmo de Backpropagation

Então, pode-se considerar dois cenários:

j é um neurônio de saída

$$\delta_j(n) = e_j(n)\varphi_j'(n)\left(v_j(n)\right)$$

j é um neurônio oculto

$$\delta_j(n) = \varphi_j'(n)(v_j(n))\sum_k \delta_k(n)w_{kj}(n)$$

Exemplo - MLP para Operador Lógico XOR

Utilize a arquitetura da rede neural da Figura a seguir para resolver o problema do operador lógico do XOR. Para isso, aplique o algoritmo de backpropagation.

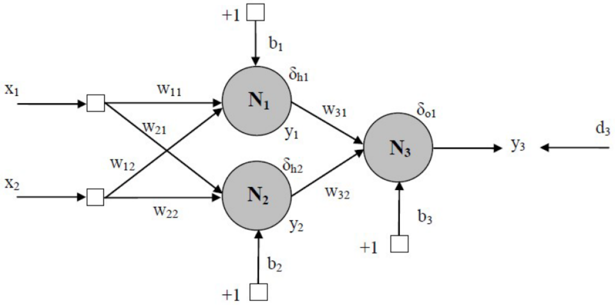

Figure: Redeu Neural MLP(2,2,1)

Implementação do Backpropagation

GitHub para download do código que será apresentado:

1. Parte I - Derivação Matemática

2. Parte II - Implementação

3. References

Referências I

1. Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (9 October 1986). "Learning representations by back-propagating errors". Nature. 323 (6088): 533–536.
2. Haykin, S. Redes Neurais - Principios e Pratica, Bookman, 2 ed., 2000.
3. Oliveira, Mauri Aparecido De. Backpropagation e Redes Neurais - Vol. 1. Rio de Janeiro:Ciência Moderna, 2024.