Backpropagation

A Regra Delta Generalizada de Rummelhart, Hinton e Williams.

Grupo:

Adalberto Diniz de Souza

Lucas Bueno dos Reis

Introdução

- A demonstração de limitações de redes neurais de simples camadas foi um fator significante no declínio do interesse em redes neurais nos anos 70.
- A descoberta e difusão de um eficiente método genérico de aprendizagem para redes neurais de multicamadas reviveu as redes neurais como ferramenta para resolver uma grande variedade de problemas.

Introdução

- O objetivo do backpropagation é aplicar o método do gradiente para minimizar o erro quadrático sobre as "saídas" computadas em uma rede.
- O backpropagation é uma modificação do ADALINE de Widrow-Hoff.

Treinamento

- A rede usa treinamento supervisionado. O objetivo está em treinar a rede para alcançar um balanceamento entre habilidade de responder corretamente os parâmetros de entrada que foram usados para o treinamento, e habilidade de dar respostas razoáveis para entradas similares, mas não idênticas, que são usadas no treinamento.

Treinamento

- O treinamento da rede por backpropagation envolve três passos:
  - O feedforward do parâmetro de treinamento de entrada.
  - O cálculo e a retropropagação do erro associado.
  - Ajuste dos pesos.

Treinamento

- Inúmeras variações do backpropagation tem sido desenvolvido para melhorar a velocidade do processo de treinamento.
- Mais que uma camada escondida pode ser benéfico para alguma aplicação, mas uma camada escondida é suficiente.
- O treinamento é lento, a rede treinada pode produzir suas saídas rapidamente.

Treinamento

- Quanto ao tempo de treinamento, vários fatores podem influenciar a sua duração, porém sempre será necessário utilizar algum critério de parada.
- O critério de parada do algoritmo backpropagation não é bem definido, e geralmente é utilizado um número máximo de ciclos. Mas, devem ser considerados a taxa de erro médio por ciclo, e a capacidade de generalização da rede.

Treinamento

- Pode ocorrer que em um determinado instante do treinamento a generalização comece a degenerar, causando o problema de over-training, ou seja a rede se especializa no conjunto de dados do treinamento e perde a capacidade de generalização.

Arquitetura

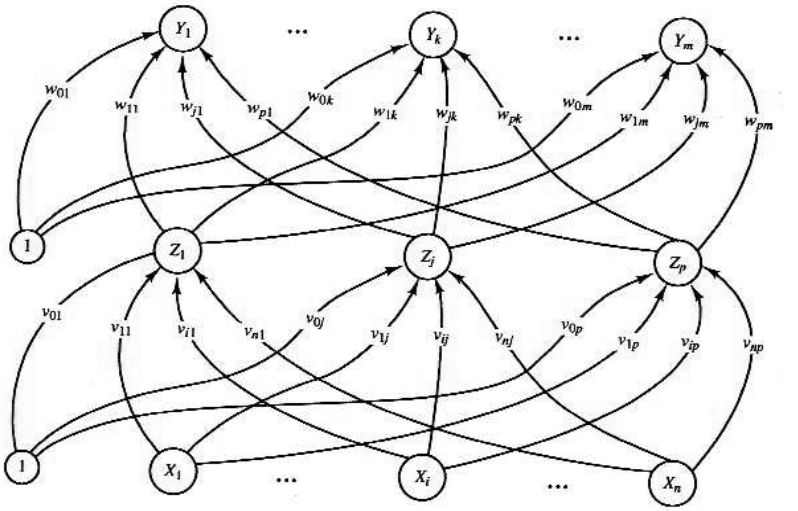
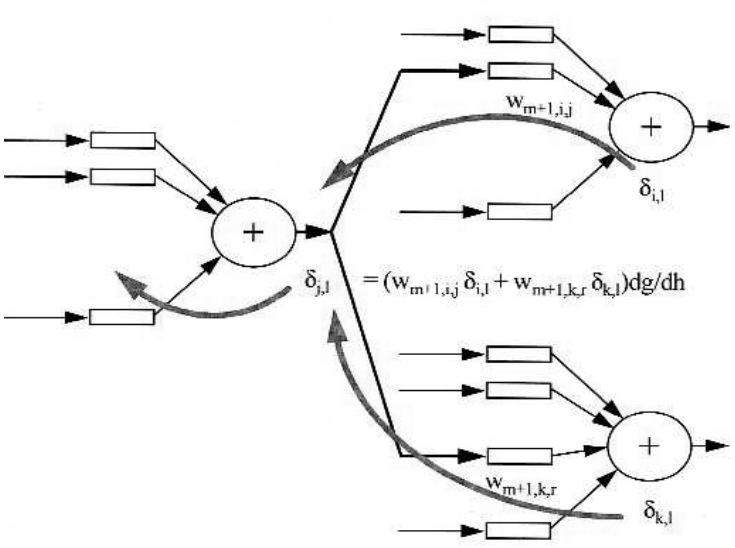


Figure 6.1 Backpropagation neural network with one hidden layer.

Arquitetura



Funções de Ativação

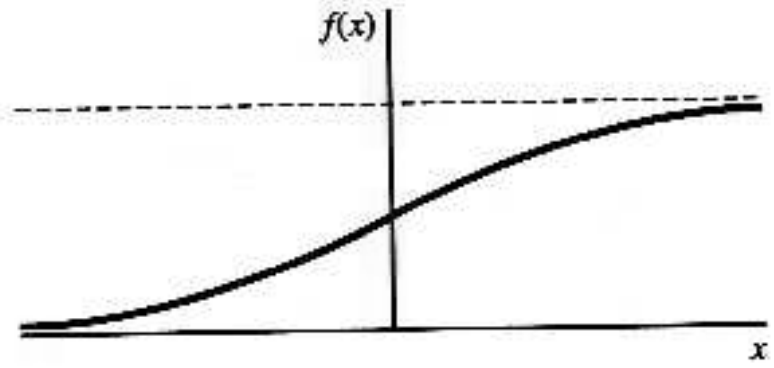


Figure 6.2 Binary sigmoid, range (0, 1).

Funções de Ativação

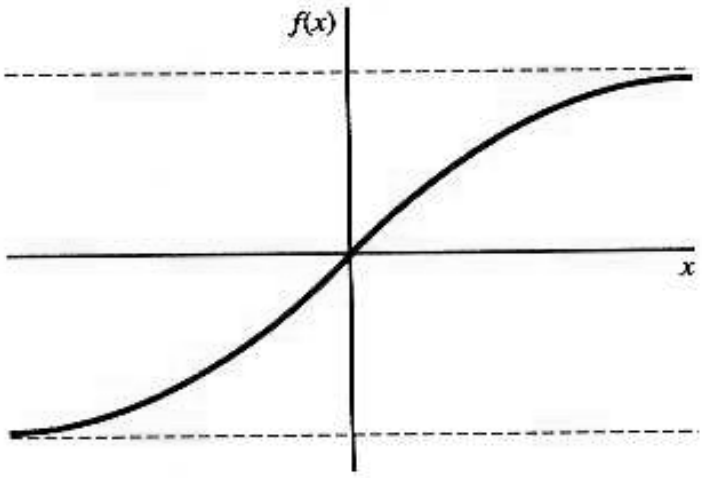


Figure 6.3 Bipolar sigmoid, range (−1, 1).

Aplicação

- Um exemplo que ilustra o treinamento é a resolução do Problema XOR (duas unidades de entrada, quatro unidades escondidas em uma camada escondida e uma unidade de saída):
  - Usando representação de dados binária, com sigmóide binária para função de ativação para todas escondidas e unidades de saída.
  - Usando representação de dados bipolar como função de ativação.

  Em cada um destes exemplos, o mesmo conjunto de pesos iniciais serão usados; valores randômicos serão escolhidos entre –0,5 e +0,5.

Aplicação

- Para resolver a função XOR por Representação Binária:

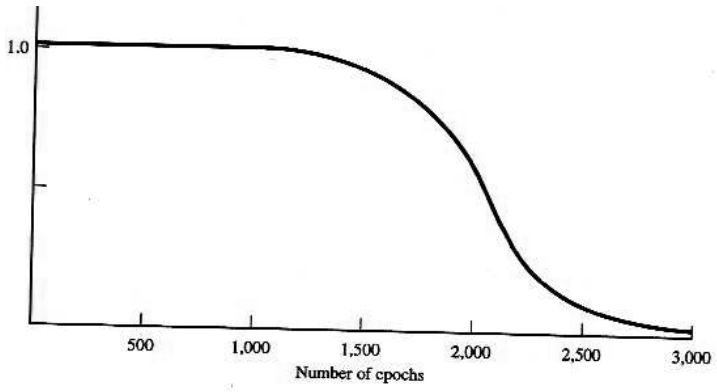


Figure 6.4 Total squared error for binary representation of XOR problem.

Aplicação

- Para resolver a função XOR por Representação Bipolar:

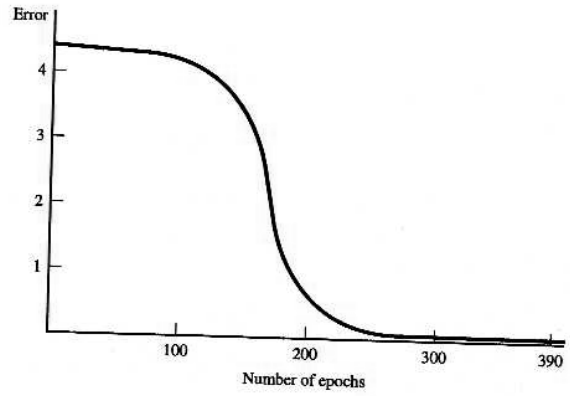


Figure 6.5 Total squared error for bipolar representation of XOR problem.

Algoritmo

- Passo 0 – Inicialize os pesos (conjunto de pequenos valores randômicos).
- Passo 1 – Quando parando pela condição falsa, faça o Passo 2-9.
  - Passo 2 – Para cada par de treinamento, faça o Passo 3-8.
  - **FeedFoward**
    - Passo 3 – Cada unidade de entrada (X_i, i=1, ... , n) receba o valor de entrada x_i e faça broadcasts do sinal para todas unidades da camada acima.
    - Passo 4 – Cada unidade escondida (Z_j, j=1, ... , p) soma a seu peso os valores de entrada,

$$z\_in_j = v_{oj} + \sum_{i=1}^{n} x_i v_{ij}$$

Algoritmo

Aplique esta função de ativação para computar o valor de saída,

$$z_j = f(z\_in_j)$$

e envie este valor para todas unidades da camada acima (unidade de saída).

- Passo 5 – Cada unidade de saída (Yk, k=1, ... , m) soma a seu peso os valores de entrada,

$$y\_in_k = w_{ok} + \sum_{j=1}^{p} z_j w_{jk}$$

e aplique a função de ativação para computar os valores de saída,

$$y_k = f(y\_in_k)$$

Algoritmo

- **Backpropagation do erro**
  - Passo 6 – Cada unidade de saída (Yk, k=1, ... , m) recebe um alvo padrão correspondendo a entrada do padrão de treinamento, computando estes erros dos termos de informação,

$$\delta_k = (t_k - y_k) f'(y\_in_k),$$

calcule os pesos corretos dos termos,

$$\Delta w_{jk} = \alpha \delta_k z_j,$$

calcule os "bias" corretos dos termos,

$$\Delta w_{ok} = \alpha \delta_k,$$

e envie δ_k para unidade de camada abaixo.

Algoritmo

- Passo 7 – Cada unidade escondida (Z_j, j=1, ... , p) soma esta entrada delta (da unidade de camada acima),

$$\delta\_in_k = \sum_{k=1}^{m} \delta_k w_{jk},$$

multiplique pela derivada da função de ativação para calcular o erro de informação do termo,

$$\delta_j = (\delta\_in_j) f'(z\_in_j),$$

calcule os pesos corretos dos termos,

$$\Delta v_{ij} = \alpha \delta_j x_i,$$

e calcule os "bias" corretos dos termos,

$$\Delta v_{0j} = \alpha \delta_j$$

Algoritmo

- **Atualize os pesos e "bias"**
  - Passo 8 – Cada unidade de saída (Y_k, k=1, ... , m) atualize os "bias" e pesos (j=0, ... , p):

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$$

Cada unidade escondida (Z_j, j=1, ... , p) atualize os "bias" e pesos (i=0, ... , n):

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$$

- Passo 9 – Teste a condição de parada

Bibliografia

- Fausett, Laurene - Fundamental Of Neural Networks. Prentice Hall International Editions, 1994.
- Kovács, Zsolt L. - Redes Neurais Artificiais. Collegium Cognitio, 1996.