Backpropagation - Pseudocódigo

Rodrigo Gonçalves Flexa - 202106840003,
Milton Costa Marques Neto - 202106840001

Introdução

Este documento visa apresentar e explicar um pseudocódigo para o algoritmo Backpropagation. O pseudocódigo descreve as etapas necessárias para configurar a rede neural, desde a inicialização dos pesos e biases até a avaliação do modelo. As seções subsequentes explicam cada parte do pseudocódigo 1, resumindo ao final com um diagrama de blocos. Futuramente, usaremos esse pseudocódigo como base para uma implementação em python mais robusta.

Algorithm 1 Pseudocódigo - Backpropagation

Entradas: conjunto de dados $(\mathbf{X}, \mathbf{Y})$ **Parâmetros:** η (taxa de aprendizado), N (número máximo de épocas), L (número de camadas), M_l (neurônios na camada l), função de ativação σ

Divisão dos Dados: 60% treino, 20% validação, 20% teste

Normalize e Divida $(\mathbf{X}, \mathbf{Y})$ (conjuntos de treino, validação e teste)
Inicialize w_{lj} e b_l aleatoriamente para todos l, j
$counter \leftarrow 0$ ▷ contador de épocas sem melhora
$min\_val\_error \leftarrow \infty$ ▷ menor erro de validação
for $e = 1$ **to** N **do** ▷ Loop para cada época
    for each $(\mathbf{x}, \mathbf{y})$ **in** conjunto de treino **do** ▷ Loop para cada exemplo
        for $l = 1$ **to** L **do** ▷ Propagação para frente
            $s^l \leftarrow w^l \mathbf{x}^{l-1} + b^l$
            $z^l \leftarrow \sigma(s^l)$ ▷ Aplica função de ativação
        end for
        $\hat{y} \leftarrow z^L$ ▷ A saída final é o valor previsto $\hat{y}$
        $\delta^L \leftarrow (\hat{y} - y) \odot \sigma'(s^L)$ ▷ Calcula o erro na última camada
        $\frac{\partial E}{\partial w^L} \leftarrow \delta^L (\mathbf{z}^{L-1})^T$
        $\frac{\partial E}{\partial b^L} \leftarrow \delta^L$
        $w^L \leftarrow w^L - \eta \frac{\partial E}{\partial w^L}$ ▷ Atualiza pesos da última camada
        $b^L \leftarrow b^L - \eta \frac{\partial E}{\partial b^L}$ ▷ Atualiza biases da última camada
        for $l = L - 1$ **down to** 1 **do** ▷ Loop de backpropagation
            $\delta^l \leftarrow (w^{l+1})^T \delta^{l+1} \odot \sigma'(s^l)$ ▷ Propaga o erro para trás
            $\frac{\partial E}{\partial w^l} \leftarrow \delta^l (\mathbf{z}^{l-1})^T$
            $\frac{\partial E}{\partial b^l} \leftarrow \delta^l$
            $w^l \leftarrow w^l - \eta \frac{\partial E}{\partial w^l}$ ▷ Atualiza pesos
            $b^l \leftarrow b^l - \eta \frac{\partial E}{\partial b^l}$ ▷ Atualiza biases
        end for
    end for
    Calcule o erro médio quadrático no conjunto de validação
    if erro de validação $< min\_val\_error$ **then**
        $min\_val\_error \leftarrow$ erro de validação
        $counter \leftarrow 0$
    else
        $counter \leftarrow counter + 1$
    end if
    if $counter = 5$ **then**
        break ▷ Parada antecipada
    end if
end for

Explicação do pseudocódigo

Inicialização (Linhas 1-4)

Antes do início do treinamento, várias etapas de inicialização são realizadas para preparar a rede neural para o processo de aprendizado:

- **Divisão dos Dados:** O conjunto de dados (X, Y) é dividido em três partes: 60% para treinamento, 20% para validação, e 20% para teste.
- **Normalização dos Dados:** Os conjuntos de treino, validação e teste são normalizados. A normalização visa assegurar que o modelo não seja enviesado por variações na escala dos dados.
- **Inicialização de Pesos e Biases:** Os pesos w_{lj} e os biases b_l para cada neurônio nas várias camadas l são inicializados de forma aleatória.
- **Contador de Épocas sem Melhora:** Um contador *counter* é inicializado com o valor 0. Este contador será utilizado para implementar a lógica de parada antecipada, baseando-se no desempenho do modelo no conjunto de validação ao longo das épocas.
- **Menor Erro de Validação:** A variável min_val_error é inicializada com o valor infinito (∞). Este valor rastreará o menor erro encontrado no conjunto de validação, auxiliando na decisão de quando parar o treinamento se não houver melhorias.

Loop de Épocas (Linha 5)

O treinamento ocorre ao longo de várias épocas (N), onde cada época consiste em múltiplas passagens pelo conjunto de dados, ajustando progressivamente os pesos e biases para minimizar o erro de previsão.

Loop para cada exemplo (Linhas 6-7)

Itera-se sobre cada exemplo de treinamento (x, y) individualmente. A entrada da rede é definida como o vetor de características x.

Propagação para Frente (Forward Pass, Linhas 8-10)

A propagação para frente começa com a entrada $\mathbf{x}$, que é transmitida através das camadas da rede. Em cada camada l, calcula-se a pré-ativação $\mathbf{s}^l$ como a soma ponderada das ativações da camada anterior $\mathbf{z}^{l-1}$, usando os pesos $\mathbf{w}^l$ e os biases $\mathbf{b}^l$. Matematicamente, essa operação é expressa como:

$$\mathbf{s}^l = \mathbf{w}^l \mathbf{z}^{l-1} + \mathbf{b}^l$$

onde $\mathbf{z}^0 = \mathbf{x}$ representa a entrada inicial da rede.

A função de ativação σ, tipicamente uma função sigmoide, ReLU ou tanh (configurável), é então aplicada a $\mathbf{s}^l$ para produzir a ativação $\mathbf{z}^l$ da camada l:

$$\mathbf{z}^l = \sigma(\mathbf{s}^l)$$

Este processo transforma linearmente os dados de entrada e adiciona não-linearidades essenciais para modelar relações complexas. As não-linearidades introduzidas pela função de ativação permitem que a rede aprenda e represente padrões sofisticados, ajustando-se a uma ampla variedade de tarefas.

Cálculo do Erro e Atualização dos Pesos na Última Camada (Linhas 11-17)

Após a propagação para frente, a rede neural gera uma saída final $\hat{y}$, que é a previsão da rede para a entrada dada. Esta previsão é obtida diretamente das ativações da última camada L, conforme indicado por:

$$\hat{y} \leftarrow \mathbf{z}^L$$

,

onde $\mathbf{z}^L$ é o vetor de ativações da última camada.

O próximo passo é calcular o gradiente do erro na última camada, representado por δ^L. Este gradiente δ^L quantifica o quanto a saída da rede $\hat{y}$ contribui para o erro total da rede. O cálculo de δ^L envolve a derivada da função de erro em relação à saída da rede, multiplicada elemento a elemento pela derivada da função de ativação $\sigma'(s^L)$:

$$\delta^L \leftarrow \frac{\partial E}{\partial \hat{y}} \odot \sigma'(s^L)$$

Aqui:

- $\frac{\partial E}{\partial \hat{y}} = \hat{y} - y$ é a derivada da função de erro, como o erro médio quadrático (MSE), em relação à saída da rede $\hat{y}$. Este termo representa o erro residual, ou seja, a diferença entre a previsão da rede e o valor verdadeiro.

- $\odot$ representa a multiplicação de Hadamard, ou seja, a multiplicação elemento a elemento entre os vetores.

- $\sigma'(s^L)$ é a derivada da função de ativação aplicada à pré-ativação s^L da última camada. Essa derivada ajusta o gradiente do erro para refletir a sensibilidade da saída da camada L às variações na entrada s^L, conforme modulada pela função de ativação.

Este cálculo de δ^L permite que a rede compute como deve ajustar os pesos e biases na última camada durante o processo de retropropagação para reduzir o erro total da rede, como se segue abaixo:

$$\frac{\partial E}{\partial w^L} \leftarrow \delta^L (\mathbf{z}^{L-1})^T$$

onde: - δ^L é o erro na última camada. - $\mathbf{z}^{L-1}$ são as ativações da camada anterior $(L-1)$. - A multiplicação é feita entre δ^L (que é um vetor coluna) e $\mathbf{z}^{L-1}$ transposta (um vetor linha), resultando em uma matriz que representa os gradientes dos pesos w^L.

O gradiente do erro em relação ao bias b^L é dado diretamente por:

$$\frac{\partial E}{\partial b^L} \leftarrow \delta^L$$

Finalmente, os pesos e biases da última camada são atualizados usando o gradiente descendente. Para cada peso w^L, a atualização é feita subtraindo o produto da taxa de aprendizado η pelo gradiente correspondente:

$$w^L \leftarrow w^L - \eta \frac{\partial E}{\partial w^L}$$

E de maneira semelhante, para o bias b^L:

$$b^L \leftarrow b^L - \eta \frac{\partial E}{\partial b^L}$$

Este processo de cálculo do erro, determinação dos gradientes, e atualização dos pesos e biases na última camada é o que permite à rede neural aprender e ajustar-se a partir dos dados de treinamento.

Backpropagation (Linhas 17-23)

Durante o loop de retropropagação, que começa na penúltima camada e vai até a primeira, o erro é propagado para trás através da rede. Para cada camada l, o gradiente do erro em relação às ativações da camada anterior (δ^l) é calculado multiplicando a transposta dos pesos da camada seguinte (w^{l+1}) pelo gradiente da camada seguinte (δ^{l+1}) e, em seguida, aplicando a multiplicação elemento a elemento pela derivada da função de ativação $\sigma'(s^l)$.

Com δ^l determinado, os gradientes dos pesos e biases ($\frac{\partial E}{\partial w^l}$ e $\frac{\partial E}{\partial b^l}$) são calculados. Esses gradientes são então usados para ajustar os pesos e biases da camada l utilizando o método de gradiente descendente, onde cada peso w^l e bias b^l é atualizado subtraindo o produto da taxa de aprendizado η pelo respectivo gradiente.

Este processo permite que a rede minimize o erro ajustando iterativamente seus parâmetros (pesos e biases) em todas as camadas.

Cálculo do Erro e Parada Antecipada (Linhas 25-35)

Após completar o processo de backpropagation para cada exemplo de treinamento durante uma época, o algoritmo realiza os seguintes passos para monitorar e controlar o treinamento baseado no desempenho do modelo no conjunto de validação:

- **Cálculo do Erro Médio Quadrático:** O erro médio quadrático é calculado no conjunto de validação. Este erro é uma métrica crucial que quantifica a diferença entre os valores previstos pela rede e os valores reais, oferecendo um indicativo de quão bem o modelo está performando.

- **Atualização do Menor Erro e Contador:** Se o erro de validação calculado for menor que o menor erro de validação previamente registrado (min_val_error), este último é atualizado para o novo valor mais baixo, e o contador de épocas sem melhora é zerado. Caso contrário, o contador é incrementado em 1, indicando mais uma época sem melhora no desempenho.

- **Verificação da Condição de Parada Antecipada:** Se o contador atingir 5, significando que não houve melhoria no erro de validação por cinco épocas consecutivas, o treinamento é interrompido antecipadamente. Esta abordagem de parada antecipada ajuda a prevenir o sobreajuste, garantindo que o modelo não apenas aprenda a representar os dados de treinamento, mas também generalize bem para novos dados.