Notas de Aula - Semana 1

Revisão de Machine Learning e Descida de Gradiente

Redes Neurais

Eduardo Adame

13 de agosto de 2025

Sumário

1 Introdução

Esta primeira aula serve como uma revisão dos conceitos fundamentais de machine learning que são essenciais para compreender redes neurais. O foco principal está na descida de gradiente, algoritmo de otimização que forma a base do treinamento de redes neurais.

Por que começar com uma revisão de ML? Redes neurais são, em essência, uma extensão natural dos métodos de aprendizado de máquina tradicionais. Muitos conceitos fundamentais são compartilhados: função de custo, otimização, regularização, e avaliação de modelos. Ao entender profundamente esses conceitos em um contexto mais simples (regressão linear), será mais fácil aplicá-los em contextos mais complexos (redes neurais profundas).

1.1 Objetivos de Aprendizagem

Ao final desta aula, o estudante será capaz de:

- Implementar o algoritmo de descida de gradiente do zero
- Compreender o efeito da taxa de aprendizado na convergência
- Distinguir entre descida de gradiente em lote (batch) e estocástica (SGD)
- Analisar trajetórias de convergência no espaço de parâmetros
- Identificar problemas comuns na otimização via gradiente

2 Fundamentos Teóricos

2.1 Problema de Regressão Linear

Começamos com o problema clássico de regressão linear. Dado um conjunto de dados $\{(x_i, y_i)\}_{i=1}^n$, queremos encontrar uma função linear que melhor descreva a relação entre as variáveis de entrada x_i e a variável de saída y_i.

Definição 2.1 (Modelo de Regressão Linear) *O modelo de regressão linear múltipla é definido como:*

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i \tag{1}$$

onde β_j são os coeficientes a serem estimados e ε_i é o termo de erro.

Em notação matricial, podemos escrever:

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{2}$$

onde:

- $\mathbf{y} \in \mathbb{R}^n$ é o vetor de variáveis dependentes
- $X \in \mathbb{R}^{n \times (p+1)}$ é a matriz de design (incluindo coluna de 1's para o intercepto)
- $\boldsymbol{\beta} \in \mathbb{R}^{p+1}$ é o vetor de coeficientes
- $\boldsymbol{\varepsilon} \in \mathbb{R}^n$ é o vetor de erros

2.2 Função de Custo

Para encontrar os melhores coeficientes, precisamos definir uma métrica que quantifique quão bem nosso modelo se ajusta aos dados.

Definição 2.2 (Função de Custo MSE) *A função de custo Mean Squared Error (MSE) é definida como:*

$$J(\boldsymbol{\beta}) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{2n}||\mathbf{y} - X\boldsymbol{\beta}||^2 \tag{3}$$

O fator $\frac{1}{2}$ é incluído por conveniência matemática, pois simplifica o cálculo da derivada. O fator $\frac{1}{n}$ normaliza o custo pelo número de amostras.

2.3 Métodos de Solução

Existem duas abordagens principais para minimizar a função de custo:

2.3.1 Solução Analítica

A solução de forma fechada para o problema de mínimos quadrados é:

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \mathbf{y} \tag{4}$$

Vantagens:

- Solução exata (dentro da precisão numérica)
- Uma única operação
- Não requer ajuste de hiperparâmetros

Desvantagens:

- Complexidade computacional $O(p^3)$ para inversão da matriz
- Problemas numéricos quando $X^T X$ é mal-condicionada
- Não escalável para grandes conjuntos de dados
- Não aplicável a funções de custo não-quadráticas

2.3.2 Solução Numérica via Gradiente

A descida de gradiente é um algoritmo iterativo que minimiza a função de custo seguindo a direção de maior decréscimo.

3 Descida de Gradiente

3.1 Derivação do Algoritmo

Para aplicar descida de gradiente, precisamos calcular o gradiente da função de custo em relação aos parâmetros.

Teorema 3.1 (Gradiente do MSE) *O gradiente da função de custo MSE em relação aos parâmetros é:*

$$\nabla_{\boldsymbol{\beta}} J(\boldsymbol{\beta}) = \frac{1}{n} X^T (X\boldsymbol{\beta} - \mathbf{y}) \tag{5}$$

Demonstração:

$$J(\boldsymbol{\beta}) = \frac{1}{2n} ||\mathbf{y} - X\boldsymbol{\beta}||^2 \tag{6}$$

$$= \frac{1}{2n} (\mathbf{y} - X\boldsymbol{\beta})^T (\mathbf{y} - X\boldsymbol{\beta}) \tag{7}$$

$$= \frac{1}{2n} (\mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T X\boldsymbol{\beta} + \boldsymbol{\beta}^T X^T X\boldsymbol{\beta}) \tag{8}$$

Derivando em relação a $\boldsymbol{\beta}$:

$$\nabla_{\boldsymbol{\beta}} J(\boldsymbol{\beta}) = \frac{1}{2n} (-2X^T\mathbf{y} + 2X^T X\boldsymbol{\beta}) = \frac{1}{n} X^T (X\boldsymbol{\beta} - \mathbf{y}) \tag{9}$$

3.2 Algoritmo de Descida de Gradiente

1. **Inicialização:** Escolha valores iniciais $\boldsymbol{\beta}^{(0)}$
2. **Para** $t = 0, 1, 2, \ldots$ até convergência:
   (a) Calcule o gradiente: $\mathbf{g}^{(t)} = \nabla_{\boldsymbol{\beta}} J(\boldsymbol{\beta}^{(t)})$
   (b) Atualize os parâmetros: $\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \alpha \mathbf{g}^{(t)}$
3. **Retorne:** $\boldsymbol{\beta}^{(T)}$

onde $\alpha > 0$ é a taxa de aprendizado (learning rate).

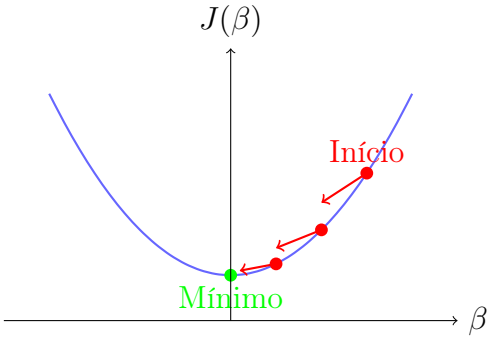

Figura 1: Intuição geométrica da Descida de Gradiente

3.3 Taxa de Aprendizado

A escolha da taxa de aprendizado α é crucial para o sucesso do algoritmo:

- α **muito pequeno:** Convergência lenta, muitas iterações necessárias
- α **adequado:** Convergência rápida e estável
- α **muito grande:** Oscilações, possível divergência

Uma heurística comum é começar com $\alpha = 0.01$ e ajustar conforme necessário.

3.4 Critérios de Parada

O algoritmo deve parar quando:

1. **Convergência do gradiente:** $||\nabla J(\boldsymbol{\beta}^{(t)})|| < \varepsilon$
2. **Convergência dos parâmetros:** $||\boldsymbol{\beta}^{(t+1)} - \boldsymbol{\beta}^{(t)}|| < \varepsilon$
3. **Convergência da função de custo:** $|J(\boldsymbol{\beta}^{(t+1)}) - J(\boldsymbol{\beta}^{(t)})| < \varepsilon$
4. **Número máximo de iterações:** $t > T_{max}$

4 Descida de Gradiente Estocástica (SGD)

4.1 Motivação

Para conjuntos de dados muito grandes, calcular o gradiente usando todas as amostras (batch gradient descent) torna-se computacionalmente custoso. A descida de gradiente estocástica (SGD) oferece uma alternativa eficiente.

Definição 4.1 (SGD) *Na descida de gradiente estocástica, o gradiente é estimado usando apenas uma amostra (ou um pequeno lote) por vez:*

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \alpha \nabla J_i(\boldsymbol{\beta}^{(t)}) \tag{10}$$

onde $J_i(\boldsymbol{\beta}) = \frac{1}{2}(y_i - \mathbf{x_i}^T\boldsymbol{\beta})^2$ *é o custo para a amostra* i*.*

4.2 Algoritmo SGD

1. **Inicialização:** $\boldsymbol{\beta}^{(0)}$, defina número de épocas
2. **Para cada época:**
   (a) Embaralhe os dados (shuffle)
   (b) **Para cada amostra** (x_i, y_i):
      i. Calcule predição: $\hat{y}_i = \mathbf{x_i}^T\boldsymbol{\beta}$
      ii. Calcule erro: $e_i = y_i - \hat{y}_i$
      iii. Atualize: $\boldsymbol{\beta} = \boldsymbol{\beta} + \alpha e_i \mathbf{x_i}$

4.3 Vantagens e Desvantagens do SGD

Vantagens:

- Computacionalmente eficiente para grandes conjuntos de dados
- Pode escapar de mínimos locais devido ao ruído
- Permite aprendizado online (dados chegando continuamente)
- Convergência mais rápida em termos de tempo de execução

Desvantagens:

- Convergência com oscilações
- Estimativa ruidosa do gradiente
- Requer mais cuidado na escolha da taxa de aprendizado
- Pode não convergir para o mínimo exato

4.4 Importância do Embaralhamento

O embaralhamento (shuffling) dos dados a cada época é crucial no SGD:

- **Sem embaralhamento:** O algoritmo pode memorizar a ordem dos dados e criar ciclos na trajetória de convergência
- **Com embaralhamento:** Reduz viés sistemático e melhora a convergência

5 Exercícios Propostos

5.1 Exercício 1: Implementação Básica

Implemente o algoritmo de descida de gradiente do zero e compare com:

1. Solução analítica usando $(X^T X)^{-1} X^T y$
2. Solução do scikit-learn usando `LinearRegression`

Os três métodos devem convergir para resultados similares.

5.2 Exercício 2: Análise da Taxa de Aprendizado

Experimente diferentes valores de taxa de aprendizado:

- $\alpha = 0.00001$ (muito pequeno)
- $\alpha = 0.001$ (adequado)
- $\alpha = 0.1$ (muito grande)

Para cada caso:

1. Plote a trajetória no espaço de parâmetros
2. Plote a evolução da função de custo
3. Analise o comportamento da convergência

5.3 Exercício 3: SGD vs. Batch GD

Implemente SGD e compare com batch gradient descent:

1. Execute SGD com embaralhamento
2. Execute SGD sem embaralhamento
3. Compare as trajetórias e tempos de convergência

Dica: Use taxas de aprendizado diferentes para cada método (SGD geralmente precisa de taxa menor).

6 Leitura Complementar

6.1 Livros Recomendados

- **Hands-On Machine Learning** (Aurélien Géron) - Capítulo 4: Training Linear Models
- **Pattern Recognition and Machine Learning** (Christopher Bishop) - Capítulo 3: Linear Models for Regression
- **Deep Learning** (Ian Goodfellow) - Capítulo 4: Numerical Computation

6.2 Recursos Online

- **CS229 Stanford:** https://cs229.stanford.edu/main_notes.pdf
- **Gradient Descent Viz:** https://www.ruder.io/optimizing-gradient-descent/
- **3Blue1Brown - Gradient Descent:** Série de vídeos sobre cálculo e otimização

7 Próxima Aula: Introdução às Redes Neurais

Na próxima semana, expandiremos os conceitos de hoje para redes neurais:

- Do neurônio biológico ao artificial
- Perceptron: o primeiro modelo neural
- Funções de ativação: linear vs. não-linear
- Redes multi-camadas (MLPs)
- Por que precisamos de não-linearidade?

Preparação para próxima aula:

1. Complete todos os exercícios desta semana
2. Revise conceitos de funções matemáticas (sigmoid, tanh, ReLU)
3. Leia sobre o perceptron de Rosenblatt (1958)