Exemplo de Algoritmo de Aprendizado Supervisionado: Regressão Linear

A regressão linear é um algoritmo simples e fundamental em aprendizado de máquina, usado para prever um valor contínuo com base em variáveis de entrada.

Problema:

Imagine que você tem um conjunto de dados com a área de diferentes casas (em metros quadrados) e os preços dessas casas. Você quer construir um modelo que possa prever o preço de uma casa com base em sua área.

Passos para Implementar a Regressão Linear:

1. **Definição do Modelo**:

o O modelo de regressão linear simples pode ser definido como:

preço = θ0 + θ1 × área (preço é igual a theta zero mais theta um vezes a área)

2. **Função de Custo**:

o A função de custo mede a diferença entre as previsões do modelo e os valores reais. Usaremos o erro quadrático médio (Mean Squared Error - MSE):

3. **Gradiente Descendente**:
o Para minimizar a função de custo, usamos o algoritmo de gradiente descendente para ajustar θ0 e θ1:
o Onde α é a taxa de aprendizado.

4. **Implementação em Python**:

```
import numpy as np
import matplotlib.pyplot as plt

# Dados de exemplo (área em m², preço em milhares de dólares)
areas = np.array([50, 60, 70, 80, 90])
precos = np.array([150, 180, 210, 240, 270])

# Normalização dos dados
areas_mean = np.mean(areas)
areas_std = np.std(areas)
precos_mean = np.mean(precos)
precos_std = np.std(precos)

areas_norm = (areas - areas_mean) / areas_std
precos_norm = (precos - precos_mean) / precos_std

# Função para calcular a previsão
def predict(area, theta):
    return theta[0] + theta[1] * area
```

```python
# Função de custo (MSE)
def compute_cost(areas, precos, theta):
    m = len(precos)
    predictions = predict(areas, theta)
    return (1/(2*m)) * np.sum((predictions - precos)**2)

# Gradiente descendente
def gradient_descent(areas, precos, theta, alpha, num_iters):
    m = len(precos)
    cost_history = []

    for i in range(num_iters):
        predictions = predict(areas, theta)
        errors = predictions - precos

        theta[0] -= alpha * (1/m) * np.sum(errors)
        theta[1] -= alpha * (1/m) * np.sum(errors * areas)

        cost_history.append(compute_cost(areas, precos, theta))
    return theta, cost_history

# Inicializar parâmetros
theta = np.random.rand(2) * 0.01 # Inicializar theta com valores pequenos
alpha = 0.1 # Taxa de aprendizado ajustada
num_iters = 1000 # Número de iterações

# Treinamento do modelo
theta, cost_history = gradient_descent(areas_norm, precos_norm, theta, alpha,
num_iters)

# Exibir os parâmetros finais
print(f'Theta final: {theta}')

# Previsão de preços para novas áreas
nova_area = 100
nova_area_norm = (nova_area - areas_mean) / areas_std
previsao_preco_norm = predict(nova_area_norm, theta)
previsao_preco = previsao_preco_norm * precos_std + precos_mean

print(f'Previsão de preço para uma área de {nova_area} m²: ${previsao_preco:.2f} mil')

# Plotando a linha de regressão
plt.scatter(areas, precos, color='red', marker='x', label='Dados de treinamento')
plt.plot(areas, predict(areas_norm, theta) * precos_std + precos_mean, label='Linha de
Regressão')
plt.xlabel('Área (m²)')
plt.ylabel('Preço (milhares de dólares)')
plt.legend()
plt.show()
```

Análise dos Resultados:

- **Theta final**: O valor de theta[1] = 1 indica que o modelo basicamente aprendeu uma relação direta entre área e preço, o que faz sentido dado que os dados de entrada são lineares e a diferença entre os valores de preço é constante.
- **Previsão de preço**: A previsão de $300.00 mil para uma área de 100 m² parece estar correta dado os dados de entrada, onde cada incremento de 10 m² na área resulta em um aumento de $30.000 no preço.

Conclusão:

Esse resultado é esperado e indica que o modelo aprendeu corretamente a relação linear nos dados fornecidos. Se essa relação é a desejada, o modelo está funcionando como deveria.

Se precisar de mais ajustes ou deseja explorar modelos mais complexos, como adicionar mais dados ou tentar uma regularização, podemos continuar a refinar o código.