Relatório EP3

Isabela Blucher e Veronica Stocco
9298170 6828626

15 de Abril de 2019

Introdução

Implementamos o algoritmo de regressão logística, com a técnica de gradiente descendente para otimização, para separar dois aglomerados de pontos em qualquer dimensão d.

Implementação

Ambas funções seguem os protótipos dados no EP.

1. Função de *Fitting*:
   - Uma coluna de 1's é adicionada a X e as dimensões de y são ajustadas.
   - Caso um vetor inicial de pesos w não tenha sido fornecido, ele é é gerado aleatóriamente com valores entre (-1, 1), com base em uma distribuição normal multivariada.
   - A matriz de dados X e o vetor de pesos w são multiplicados, resultando em z. Chama-se de h o resultado da aplicação da função sigmoide em z.
   - Com h em mãos, o gradiente é calculado. O vetor de pesos w é atualizado de acordo com a *learning_rate* e o *gradient*. Repete-se esse passo *num_iterations* vezes.
2. Função de *Prediction*:
   - Uma coluna de 1's é adicionada a X.
   - Multiplica-se as matrizes de dados X e pesos w, resultando em z.
   - A predição final é o resultado da função sigmoide aplicada a z.
3. Funções adicionais:
   - `sigmoid(z)`: Retorna o valor da função sigmoide aplicada a z.
   - `cost_function(h, y)`: Retorna a função de custo, uma medida do erro do modelo h para representar a relação entre os dados X e labels y.

Testes e Resultados

Para todos os testes, foram utilizados datasets gerados por `np.random.multivariate_normal`. Para testar se o algoritmo estava funcionando conforme as especificações, executamos o programa com parâmetros que geraram amostras similares àquelas do enunciado. Entretanto, notamos que a separação dos conjuntos de pontos foi muito melhor utilizando os parâmetros *learning_rate* = **0.1** e num_iterations = **10000**. Por esse motivo, realizamos os testes a seguir com esses valores, ao invés de $1e^{-}2$ e 1000, respectivamente.

Testes em 2D

- **Média:** (2, 2) e (10, 2)
- **Covariâncias:**

$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \text{ e } \begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix}.$$

Nas Figuras 1 e 2, podemos comparar os datasets originais (coluna 1) com as separações propostas pelo algoritmo (coluna 2). Os pontos brancos representam a média do dataset.

Em uma análise visual, ambas as separações dos datasets realizadas na Figura 1 aparentam estar corretas. O valor final da função de custo das amostras circulares foi 0.01416, e o das ovais foi 0.03275.

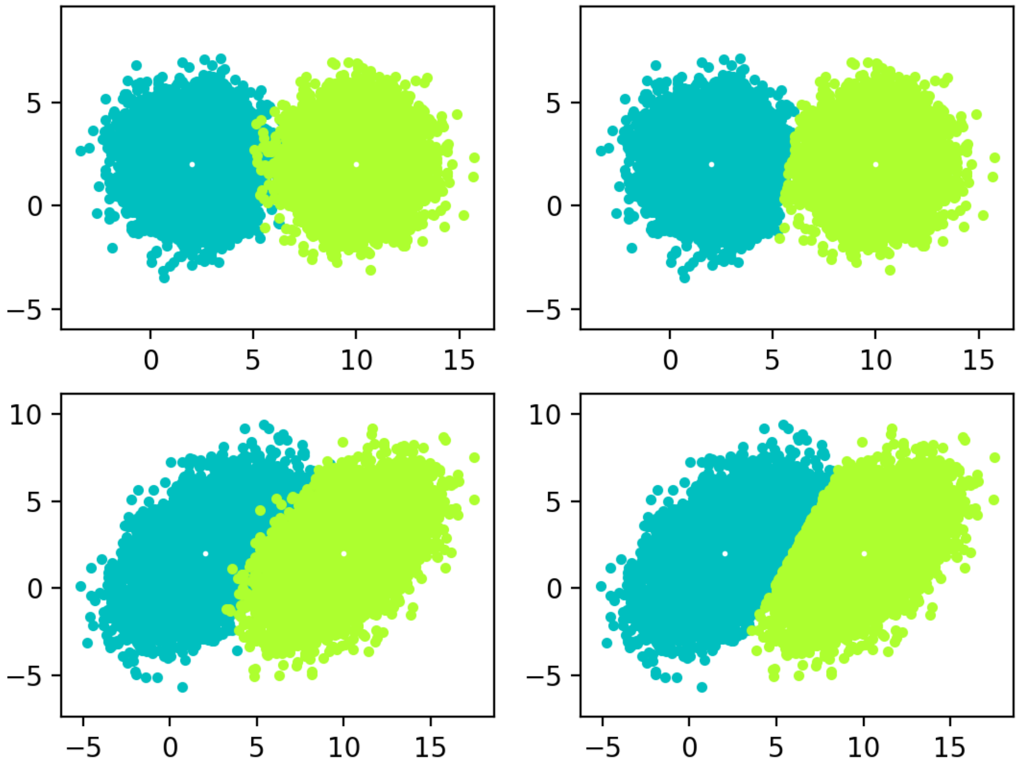

Figura 1: Todos os datasets foram gerados com N = 10 000

Em um segundo teste, o dataset azul possuia dez vezes o número de pontos do dataset verde. Mais uma vez, o algoritmo foi capaz de separar as amostras. Desta vez, com o valor final da função de custo sendo 0.01066 para as amostras circulares e 0.01911 para as ovais.

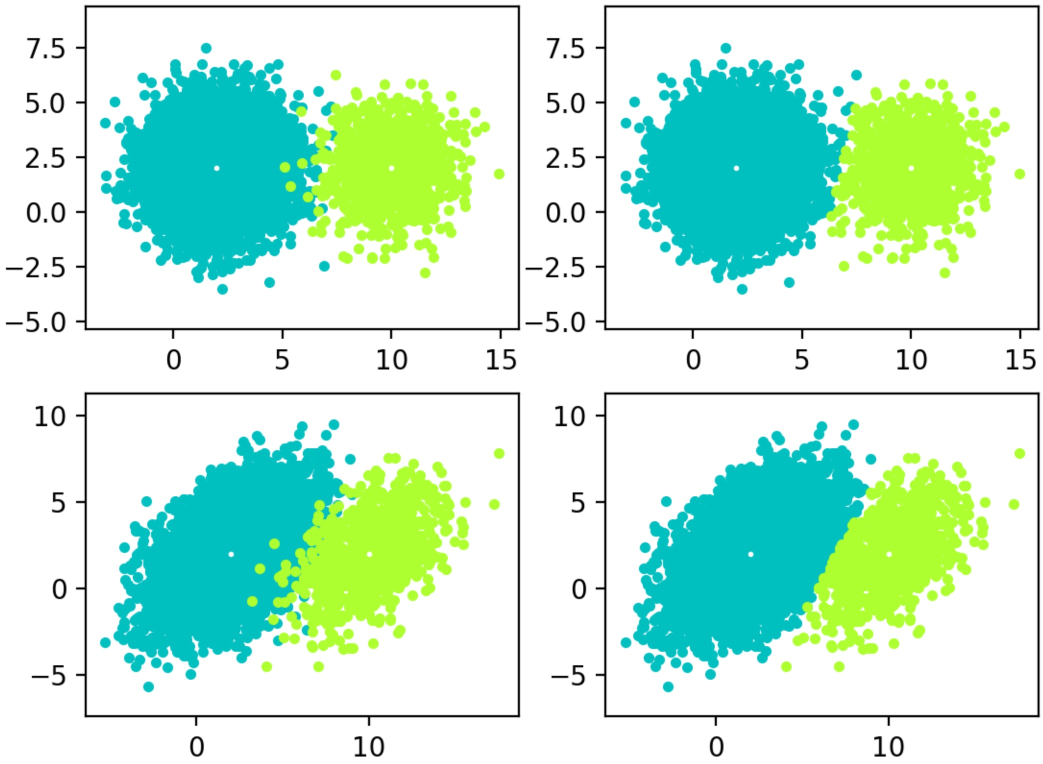

Figura 2: Os datasets em azul (N = 10 000) são 10x maiores que os verdes.

Testes em mais dimensões

Apesar da visualização gráfica não ser possível, também testamos o algoritmo em datasets gerados nas seguintes condições: N = 10 000, learning_rate = 0.1 e num_iterations = 10 000.

1. **3D:**
   - **Média:** (4, 2, 1) e (10, 2, 1)
   - **Covariância:** $\begin{bmatrix} 3 & 2 & 1 \\ 2 & 4 & 0 \\ 1 & 0 & 3 \end{bmatrix}$
2. **4D:**
   - **Média:** (4, 2, 1, 1) e (10, 6, 2, 1)
   - **Covariância:** $\begin{bmatrix} 4 & 1 & 2 & 1 \\ 1 & 3 & 0 & 1 \\ 2 & 0 & 2 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$
3. **Resultados:** Para ambos os casos, executamos cinco testes e calculamos o valor final médio da função de custo. Para os dados 3D, foi de 0.02981, e para os 4D, 0.02236. Esses valores de erro são comparáveis aos obtidos na separação dos datasets ovais da Figura 1 (0.03275), o que nos leva a concluir que a separação foi razoável.

Conclusão

O algoritmo implementado é capaz de separar dois datasets de forma satisfatória. O uso dos parâmetros *learning_rate* = **0.1** e num_iterations = **10000** ao invés daqueles dados no enunciado levou a uma separação muito melhor.

Fontes

Além dos slides de sala e do material do curso *Learning From Data*, consultamos as explicações teóricas disponíveis em: `https://towardsdatascience.com/building-a-logistic-regression-in-python-301d27367c24`.