Regressão Logística

Nina Magalhães de Oliveira

June 13, 2025

1 Introdução

A regressão logística é uma técnica estátistica utilizada para a classificação de dados, construída através da aplicação de uma função sigmoid, também chamada de logit, sobre a regressão linear.

Em um modelo de regressão logística, temos a variável categórica que, em um problema binário, assume os valores 0 (não/negativo/fracasso) ou 1 (sim/positivo/sucesso), e as variáveis quantitativas, as quais podem assumir diversos valores.

Diferentemente da regressão linear, na qual a saída é um valor quantitativo, a regressão logística retorna a probabilidade de um evento ocorrer. Isso é feito por meio da inserção dos dados em uma função sigmoid que irá distríbui-los em um intervalo que varia de 0 a 1. Portanto, a regressão logística calcula a probabilidade de um evento ser 0 ou 1 e, com isso, consegue prever em qual classe determinada entrada se encaixa.

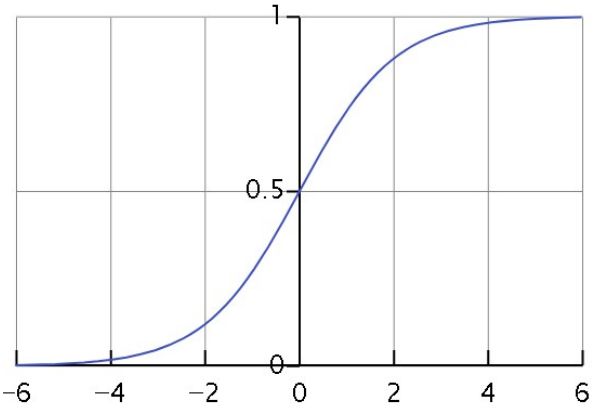

Figure 1: A figura acima é a representação gráfica da função sigmoid. Valores abaixo de 0.5 são classificados como 0, já valores acima de 0.5 pertencem à classe 1

A função da regressão logística, sigmoid, é dada por:

$$Y(s) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(\beta_0+\beta_1 x1+...+\beta_n xn)}}$$

Onde s é a fução da regressão linear e Y(s) é a probabilidade de xi pertencer a classe 1. Sendo assim, é possível rearranjar os termos da função:

$$\beta_0 + \beta_1 x1 + ... + \beta_n xn = \ln(\frac{1}{1-p})$$

Observação: β_0 é chamado de bias (b) ou, coeficiente angular, e β_1,..,β_n são chamdos de pesos (w) ou coeficientes lineares.

2 Modelo da regressão logística

A regressão logística é um algoritmo de aprendizado de máquina supervisionado e exige que uma série de etapas sejam completas de modo a efetuar a classificação.

2.1 Entropia Cruzada Binária

Para treinar um modelo de regressão logística é necessário,primeiro, separar um conjunto n de dados do dataset (T =(x1,y1),(x2,y2),..,(xn,yn) sobre o qual será aplicado uma função de perda/custo/erro, chamada de entropia cruzada binária.

A entropia cruzada binária é encarregada de encontrar a função sigmoid que melhor se ajuste aos dados de treino. Ela mede a diferença entre as probabilidades previstas pelo modelo de teste e os valores verdadeiros de Y(s), penalizando as discrepâncias.

Matematicamente, é descrita como:

$$L(w,b) = \frac{1}{n}\sum_{i}^{n}[yi * log(p'i) + (1 - yi) * log(1 - p'i)]$$

Sendo n o número de entradas do treino e $p'i$ a probabilidade estimada pelo modelo de xi pertencer à classe 1.

2.2 Gradiente Descendente

A aplicação da entropia cruzada binária no modelo, na realidade, inicia-se a partir de sua minimização, já que é dessa maneira que serão encontrados os pesos e os bias da função logística.

Como não há uma fórmula fechada para o vetor dos coeficientes que minimiza a entropia cruzada binária, pode ser utilizado um método de otimização, que é o caso do gradiente descendente.

$$J'(\theta) = \begin{bmatrix} \frac{dJ}{dw} \\ \frac{dJ}{db} \end{bmatrix} = [\dots] = \begin{bmatrix} \frac{1}{N}\sum 2x_i(\hat{y} - y_i) \\ \frac{1}{N}\sum 2(\hat{y} - y_i) \end{bmatrix}$$

Figure 2:
Cálculo do gradiente da função de perda.

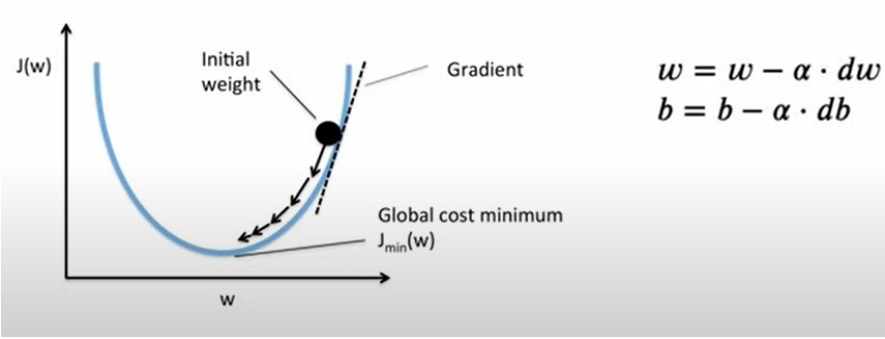

Figure 3:
Gráfico do gradiente descendente da função de perda

Portanto, como mosta a figura 3, o gradiente descendente calcula a direção para o mínimo global, ou seja, quais são os valores dos pesos e do bias que minimizam a função de perda. O cálculo coeficientes pode ser descrito como:

$$w = w' - \sigma * dw$$

$$b = b' - \sigma * db$$

Onde w' e b são os valores iniciais dos pesos e do bias, respectivamente. dw é a derivada da função de perda em relação aos pesos e db é a derivada da função de perda em relação ao bias. E, finalmente, σ é o chamado learning rate, o qual é um parâmetro de ajuste que controla o tamanho dos ajustes feitos aos coeficientes em cada iteração durante o processo de treinamento.

3 Aplicação em python

Explicada a lógica matemática por trás da regressão logística, cria-se um algoritmo na linguagem Python que utiliza esses conceitos a fim de classifcar dados do dataset sobre câncer de mama.

3.1 Dataset

O dataset UCI ML Breast Cancer Wisconsin (Diagnostic) foi desenvolvido pela University Medical Centre, Institute of Oncology e pode ser encontrado dentro da biblioteca Sklearn.datasets do python.O dataset,elaborado a partir do estudo de células retiradas de massa mamária, possui 569 linhas e 31 colunas, dividas da seguinte forma:

- 30 colunas são os features, ou seja, trazem as características da célula como,por exemplo, a área e pêrimetro;
- 1 coluna é o target, responsável por classificar determinada célula cancerígena de acordo com o valor atrbuído a ela, ou seja, benigna se for 1 e maligna se for 0.

3.2 Função Sigmoid

Versão computacional da função sigmoid:

```
def sigmoid(s):
    return 1/(1+np.exp(-s))
```

Figure 4:

3.3 Classe LogisticRegression

A classe LogisticRegression possui três funções: função de inicialização, função de treinamento e função de classificação.

3.3.1 Inicialização

Na função de inicialização é feita a parametrização do número de iterações, o valor do learning rate e dos coeficientes (pesos e bias). O valor do learning rate pode ser alterado ao chamar a classe, porém, vale ressaltar que um learning rate muito alto ou muito baixo pode afetar a performance do algoritmo.

```
def __init__(self, lr = 0.001, n_iters= 1000):
    self.lr = lr
    self.n_iters = n_iters
    self.weights = None #iniciamos os pesos como 0
    self.bias = None #iniciamos os bias como 0
```

Figure 5:

3.3.2 Treinamento

A função de treinamento aplicado, sobre os dados de treinamento, engloba 4 etapas fundamentais:

- definir o formato da amostra que será utilizada no treinamento
- aplicação da função sigmoid sobre a regressão linear

- cálculo do gradiente da função de perda, ou seja, da entropia cruzada binária
- definição dos coeficientes pesos e bias que minizam a função de perda

```python
def fit(self, X, y ):

    #definimos o formato da amostra de entrada do modelo
    n_samples , n_features = X.shape
    self.weights = np.zeros(n_features)
    self.bias = 0

    #minimizando a função de perda
    for _ in range(self.n_iters):
        linear = np.dot(X, self.weights) + self.bias #y = β0 + β1x1 + ... +βnxn
        predictions = sigmoid(linear) #y' =  1/1+e**-(β0 + β1x1 + ... +βnxn)

        #calcular o gradiente da entropia cruzada
        dw = (1/n_samples)* 2*(np.dot(X.T,(predictions-y))) #derivada em relação ao peso
        db = (1/n_samples)* 2*(np.sum(predictions-y)) #derivada em relação ao bias

        #encontrando os parâmetros que minimizam a função de perda
        self.weights = self.weights - self.lr*dw
        self.bias = self.bias - self.lr*db
```

Figure 6:

3.3.3 Classificação

Por fim, a função de classificação irá calcular novamente a função sigmoid, agora com o valor atualizado dos coeficientes e, assim, determinar as probabilidades dos atributos do teste. A fim de classificar essas probabilidades entre 0 e 1, define-se que para valores menores ou iguais a 0.5 a classe pertencente é 0 e, caso contrário, a classe pertencente é 1.

```python
def predict(self,X):

    linear = np.dot(X,self.weights) + self.bias
    y_pred = sigmoid(linear)
    class_pred = [0 if y <=0.5 else 1 for y in y_pred]
    return class_pred
```

Figure 7:

4 Resultados

A fim de verificar o resultado do modelo de treinamento, cria-se uma função de acurárica a qual irá quantificar o número de vezes que a saída do treinamento é igual à saída do teste e dividir essa quantidade pelo número de testes. Desse modo, retorna a porcentagem de acertos dentro de um n número de respostas corretas.

Depois de rodar o código algumas vezes, obteve-se acerto em noventa e dois por cento das vezes, portanto, uma acurácia significativa.Conclui-se que o modelo de regressão linear desenvolvido é efetivo na classificação dos atributos do dataset de câncer de mama.

5 Referências

- Logistic Regression in Python - Normalized Nerd (youtube)
- Step by Step Tutorial on Logistic Regression in Python, sklearn, Jupyter Notebook - RegenerativeToday (youtube)

- A Regressão Logística - Algoritmos de Aprendizado de Máquinas - Hashtag Proramação (youtube)
- How to implement Logistic Regression from scratch with Python - AssemblyAI (youtube)
- Aprendizado de Máquina - Aula 06 - Regressão Logística - Unicamp