Regressão Logística

Slides extraídos do livro "*Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*" de Aurélien Géron

Sumário

Sumário

Regressão Logística

- Alguns algoritmos de regressão podem ser usados para classificação (e vice-versa).
- A **Regressão Logística** (também chamada de Regressão Logit) é comumente usada para estimar a probabilidade de uma instância pertencer a uma classe particular.
- Se a probabilidade estimada for maior que um limiar (geralmente 50%), então o modelo prevê que a instância pertence àquela classe (classe positiva, rotulada como "1"). Caso contrário, ele prevê que não pertence (classe negativa, rotulada como "0").
- Isso o torna um classificador binário.

Estimando Probabilidades

- Assim como um modelo de Regressão Linear, um modelo de Regressão Logística calcula uma soma ponderada das features de entrada (mais um termo de viés).
- Mas em vez de produzir o resultado diretamente, ele produz o **sigmoide** desse resultado.

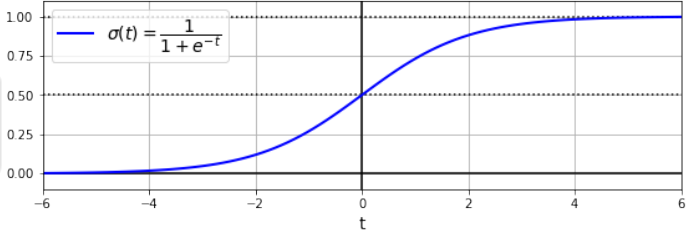
Equação 4-13. Probabilidade estimada do modelo de Regressão Logística (vetorizada)

$$\hat{p} = h_\theta(x) = \sigma(\theta^T x)$$

O logístico—notado $\sigma(\cdot)$—é uma **função sigmoide** (em forma de S) que produz um número entre 0 e 1.

Equação 4-14. Função logística

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$



Uma vez que o modelo de Regressão Logística estimou a probabilidade $\hat{p}$, ele pode fazer sua previsão $\hat{y}$ facilmente.

Equação 4-15. Previsão do modelo de Regressão Logística (limiar de 50%)

$$\hat{y} = \begin{cases} 0 & \text{se } \hat{p} < 0.5 \\ 1 & \text{se } \hat{p} \geq 0.5 \end{cases}$$

- Note que $\sigma(t) \geq 0.5$ quando $t \geq 0$, e $\sigma(t) < 0.5$ quando $t < 0$.
- Portanto, um modelo de Regressão Logística prevê 1 se $\theta^T x$ for positivo e 0 se for negativo.

Treinamento e Função de Custo

O objetivo do treinamento é ajustar o vetor de parâmetros θ para que o modelo estime altas probabilidades para instâncias positivas ($y = 1$) e baixas probabilidades para instâncias negativas ($y = 0$).

Equação 4-16. Função de custo de uma única instância

$$c(\theta) = \begin{cases} -\log(\hat{p}) & \text{se } y = 1 \\ -\log(1 - \hat{p}) & \text{se } y = 0 \end{cases}$$

A função de custo sobre todo o conjunto de treinamento é a média dos custos sobre todas as instâncias de treinamento. Ela pode ser escrita em uma única expressão chamada de **log loss**.

Equação 4-17. Função de custo da Regressão Logística (log loss)

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log(\hat{p}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)})]$$

Boa Notícia

Não há solução de forma fechada, mas esta função de custo é convexa, então o Gradiente Descendente (ou qualquer outro algoritmo de otimização) tem a garantia de encontrar o mínimo global.

Equação 4-18. Derivadas parciais da função de custo logística

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (\sigma(\theta^T x^{(i)}) - y^{(i)}) x_j^{(i)}$$

- Esta equação parece muito com a equação da derivada parcial do MSE.
- Para cada instância, ela calcula o erro de previsão e o multiplica pelo valor da j-ésima feature, e então calcula a média sobre todas as instâncias de treinamento.
- Uma vez que você tem o vetor gradiente contendo todas as derivadas parciais, você pode usá-lo no algoritmo de Gradiente Descendente.

Limites de Decisão: Exemplo com o Dataset Iris

Vamos usar o dataset Iris para ilustrar a Regressão Logística.

- O dataset contém o comprimento e a largura da sépala e da pétala de 150 flores de íris de três espécies diferentes: Iris setosa, Iris versicolor e Iris virginica.
- Construir um classificador para detectar o tipo *Iris virginica* com base apenas na feature de largura da pétala.

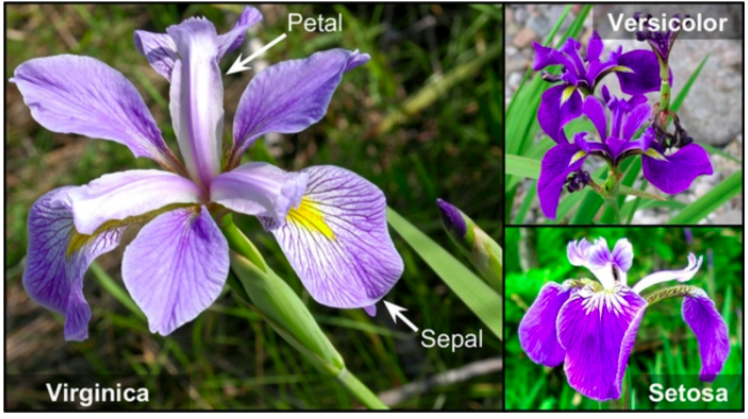


Figure: Flores das três espécies de íris.

Implementação com o Dataset Iris

```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

iris = load_iris(as_frame=True)
X = iris.data[["petal width (cm)"]].values
y = (iris.target_names[iris.target] == 'virginica')
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

log_reg = LogisticRegression(random_state=42)
log_reg.fit(X_train, y_train)
```

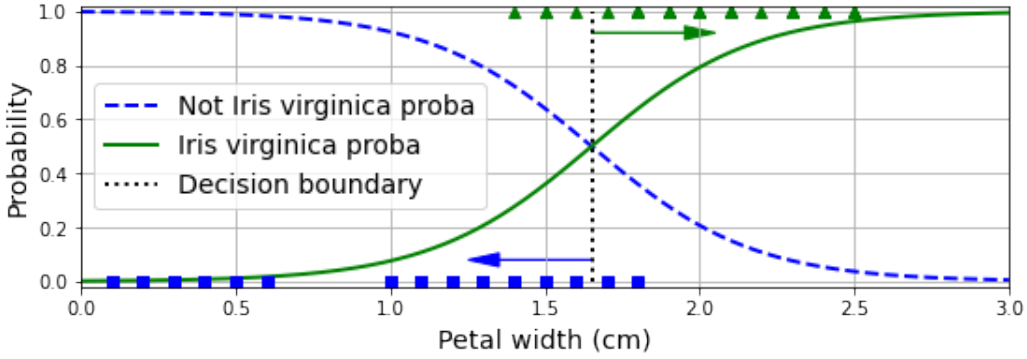


Figure: Probabilidades estimadas e limite de decisão para uma feature.

Limite de decisão em torno de 1.65 cm, onde ambas as probabilidades são iguais a 50%.

Limites de Decisão com Duas Features

- Figura mostra o mesmo dataset, mas desta vez utilizando duas features: largura e comprimento da pétala.
- Linha tracejada representa os pontos onde o modelo estima probabilidade de 50% (**limite de decisão**).
- Note que é um limite linear. Cada linha paralela são os pontos onde o modelo produz uma prob. específica.

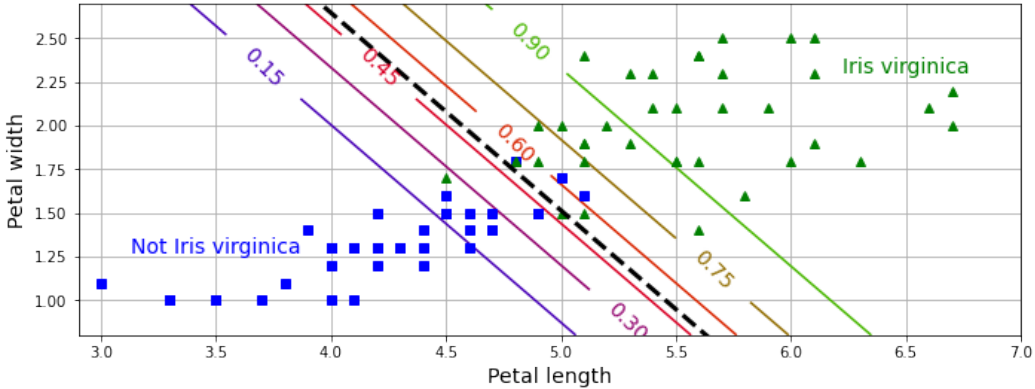


Figure: Limite de decisão linear com duas features.

Pode ser regularizada com penalidades ℓ_1 ou ℓ_2. O Scikit usa ℓ_2 por padrão.

Sumário

Regressão Softmax

- O modelo de Regressão Logística pode ser generalizado para suportar múltiplas classes diretamente. Isso é chamado de **Regressão Softmax** ou Regressão Logística Multinomial.
- Dada uma instância x, calcula uma pontuação $s_k(x)$ para cada classe k (função linear). Em seguida, estima a probabilidade de cada classe aplicando a **função softmax** nas pontuações.

Equação 4-19. Pontuação Softmax para a classe k

$$s_k(x) = (\theta^{(k)})^T x$$

Cada classe tem seu próprio vetor de parâmetros dedicado $\theta^{(k)}$.

Equação 4-20. Função Softmax

$$\hat{p}_k = \sigma(s(x))_k = \frac{\exp(s_k(x))}{\sum_{j=1}^{K} \exp(s_j(x))}$$

A **função softmax** é uma exponencial normalizada (probs. das classes somam 1).

Previsão e Função de Custo da Regressão Softmax

- O classificador de Regressão Softmax prevê a classe com a maior probabilidade estimada (classe com a maior pontuação).

Equação 4-21. Previsão do classificador de Regressão Softmax

$$\hat{y} = \underset{k}{\operatorname{argmax}}\ \sigma(s_k(x)) = \underset{k}{\operatorname{argmax}}\ s_k(x)$$

- O objetivo do treinamento é ter um modelo que estime uma alta probabilidade para a classe alvo.
- Basta minimizar a **entropia cruzada** (generalização da **log loss** para k classes).

Equação 4-22. Função de custo de entropia cruzada

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(\hat{p}_k^{(i)})$$

Implementação e Visualização da Regressão Softmax

A classe `LogisticRegression` do Scikit-Learn usa a Regressão Softmax automaticamente quando você a treina em mais de duas classes.

```
X = iris.data[["petal length (cm)", "petal width (cm)"]].values
y = iris["target"]
# [...]
softmax_reg = LogisticRegression()
softmax_reg.fit(X_train, y_train)

>>> softmax_reg.predict([[5, 2]])
array([2])
>>> softmax_reg.predict_proba([[5, 2]]).round(2)
array([[0. , 0.04, 0.96]])
```

Implementação e Visualização da Regressão Softmax

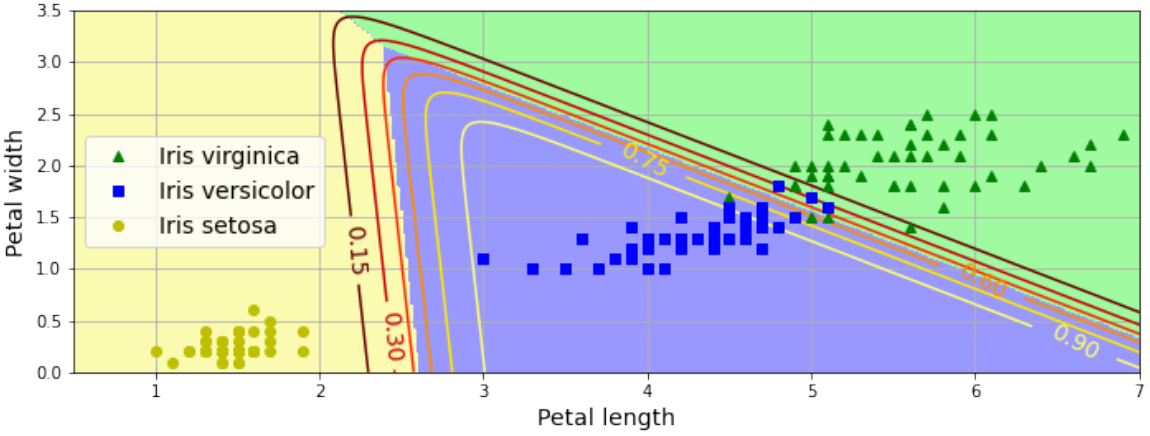


Figure: Limites de decisão da Regressão Softmax. Curvas são probs. da virginica.

Fim