∇ data

Regressão Logística

Gustavo Sutter e João G.M Araújo • 29/05/2020

Regressão logística

- Regressão logística é um algoritmo de classificação de Machine Learning
- Ela é extremamente ligada a regressão linear, então vamos revisá-la agora

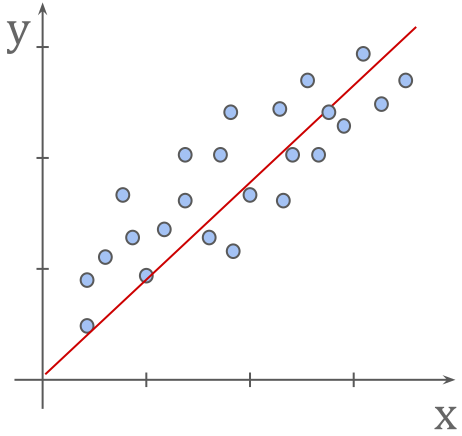
Revisão da regressão linear

- Algoritmo tem como objetivo **encontrar os parâmetros** para que a função hipótese se **aproxime dos dados**
- Para isso definimos uma ***loss function***, que diz o quanto nós estamos errando em função dos nossos parâmetros escolhidos
- Derivando a função de custo podemos a otimizar utilizando o método ***gradient descent***

(Vamos acompanhar agora um exemplo de regressão linear bem simples, com apenas um atributo e com a função de hipótese mais simples possível)

Revisão da regressão linear: função de hipótese

- Nossa hipótese será uma reta com um parâmetro apenas, que controla sua inclinação



$$h_\theta(x) = \theta x$$

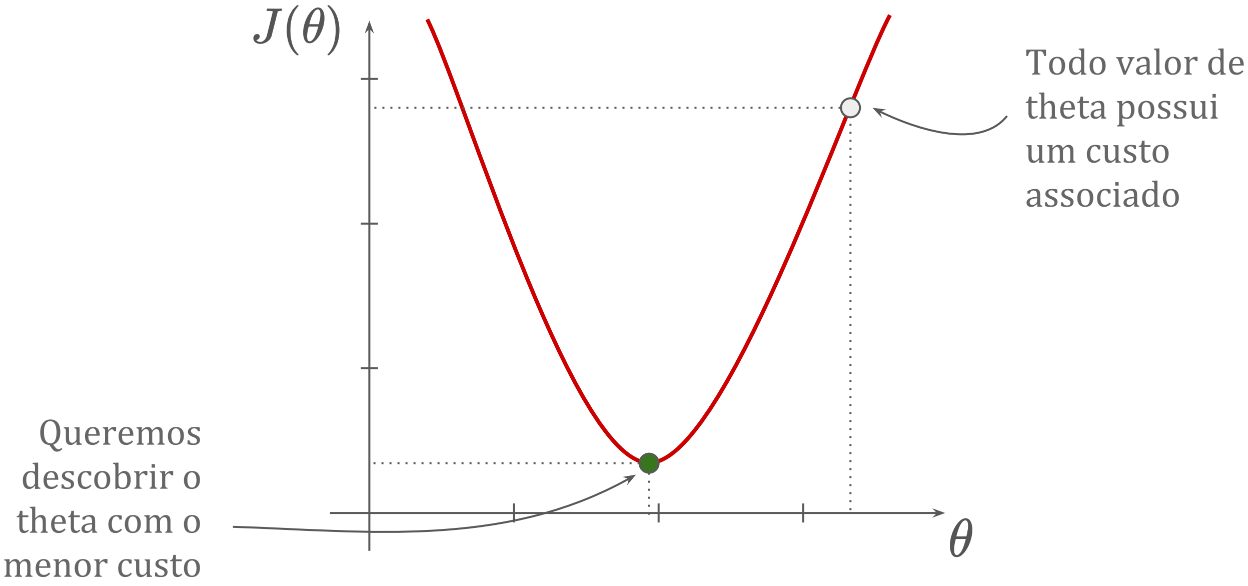
Revisão da regressão linear: função de custo

- Para cada valor de theta possível teremos um resultado da hipótese e, portanto, o **quanto estamos errado em relação ao que deveríamos estar prevendo**. Isso é expresso pela função de custo
- Nosso objetivo é encontrar o valor de theta que possui o menor custo, isto é, **minimizar a função de custo**

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Revisão da regressão linear: função de custo

- Podemos mostrar a função de custo graficamente



Revisão da regressão linear: *gradient descent*

- O gradiente da função em um ponto indica sua direção de crescimento
- Se dermos um passo no sentido contrário ao gradiente estamos indo para o ponto que função é minimizada

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)}$$

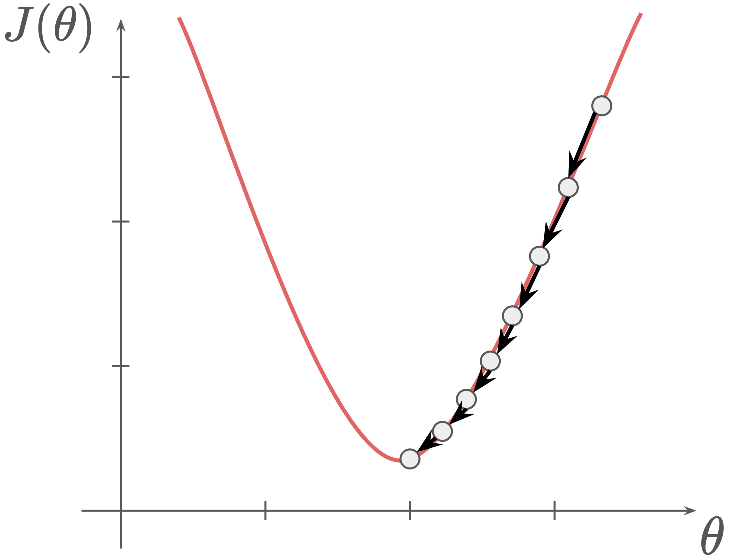
∇

Revisão da regressão linear: *gradient descent*

- A cada passo do algoritmo damos um passo no sentido contrário ao do gradiente

$$\theta := \theta - \alpha \frac{\partial J(\theta)}{\partial \theta}$$

É o parâmetro ***learning rate***, que controla o tamanho de cada passo



Regressão Logística

- Algoritmo de **classificação** construído a partir de uma extensão do algoritmo de regressão linear
- Agora nossa função de hipótese tem outro significado: **a probabilidade do exemplo dado ser da classe positiva**

$$h_\theta(x) = \sigma(\theta^T x) = P(y = 1|x; \theta)$$

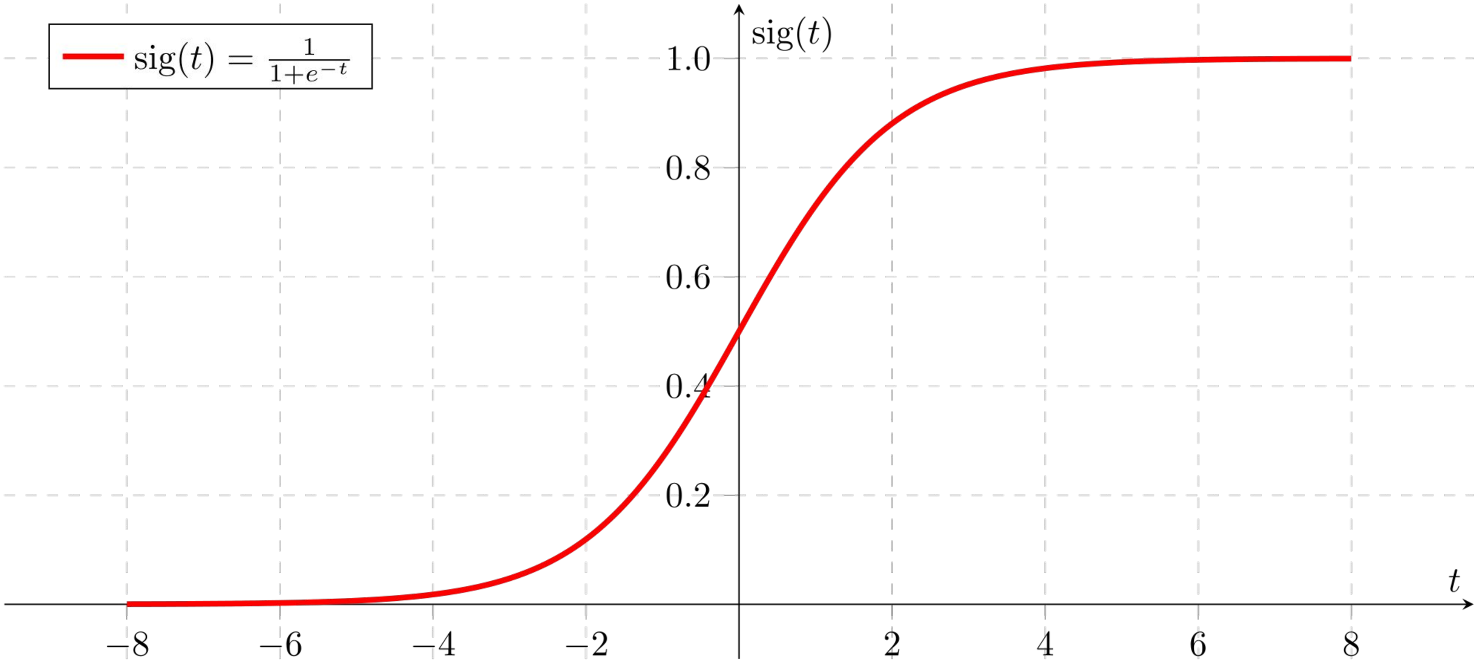
Função logística/sigmóide

Função sigmóide

- A função sigmóide é uma função monotonicamente crescente que coloca o nosso valor no intervalo [0, 1], podendo expressar um probabilidade

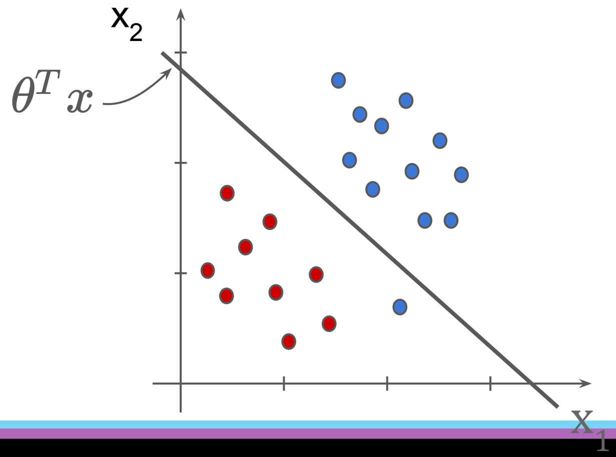
$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Função sigmóide



Decision boundary (fronteira de decisão)

- O hiperplano definido por θ^T é a *decision boundary* do nosso classificador, isto é, para um lado os exemplos são de uma classe e para o outro lado são de outra.
- Note o nome: hiper**plano**, a divisão do espaço é linear nos parâmetros



$$\begin{cases} y = 1, \text{ se } \theta^T x \geq 0 \\ y = 0, \text{ se } \theta^T x < 0 \end{cases}$$

Função de custo

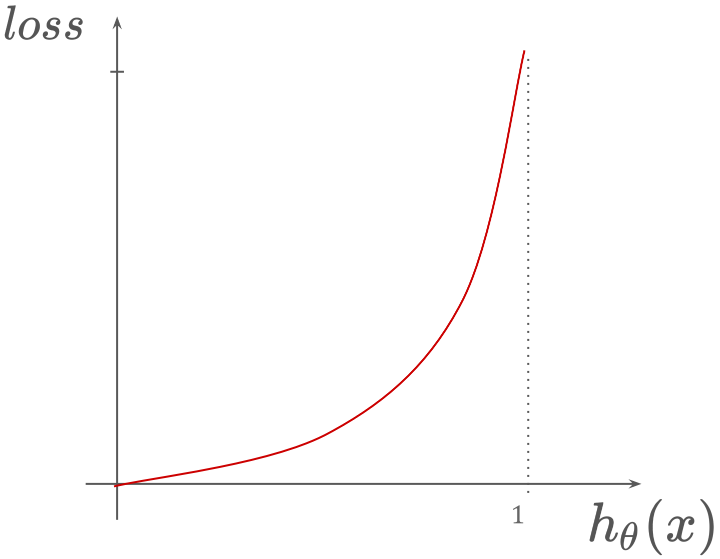
- Para a regressão logística utilizamos outra *loss function*, chamada ***cross entropy loss (entropia cruzada)***
- Essa função é útil para comparar duas distribuições de probabilidade

$$loss(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) \text{ , se } y = 1 \\ \\ -log(1 - h_\theta(x)) \text{ , se } y = 0 \end{cases}$$

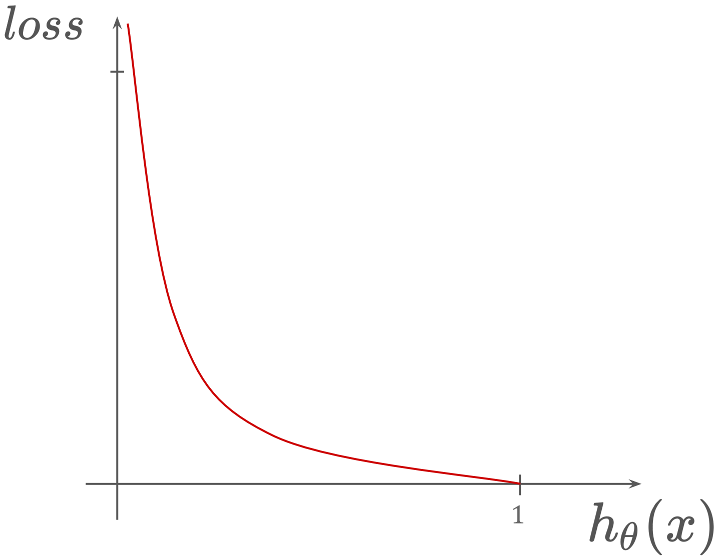
também podemos reescrever da seguinte forma:

$$loss(h_\theta(x), y) = -[ylog(h_\theta(x)) + (1-y)log(1-h_\theta(x))]$$

Função de custo



Se y = 0



Se y = 1

Função de custo

- Note que nosso output está no intervalo (0, 1)
- Se $y = 0$, a loss diminui a medida que $h(\theta)$ tende a 0
- Já se $y = 1$, a loss diminui à medida que $h(\theta)$ tende a 1
- Ou seja, a loss diminui à medida que $h(\theta)$ tende ao valor da classe correta

Função de custo

- Perceba que essa perda é para cada exemplo do conjunto, então precisamos **tirar a média de todos as instâncias** para obter nosso custo.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} loss(h_\theta(x^{(i)}), y^{(i)})$$

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

Gradient Descent

- Assim como na regressão linear vamos utilizar *gradient descent* para minimizar nossa função de custo
- Derivando a função de custo obtemos o seguinte gradiente

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

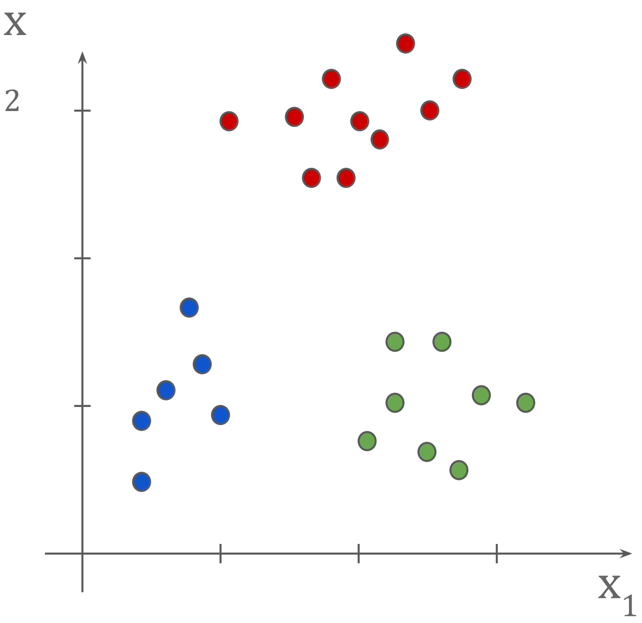
Classificação multiclasses

- Existem dois métodos para aplicar regressão logística a problemas multiclasse
    - **One vs Rest**
    - **Multinomial Logistic Regression**

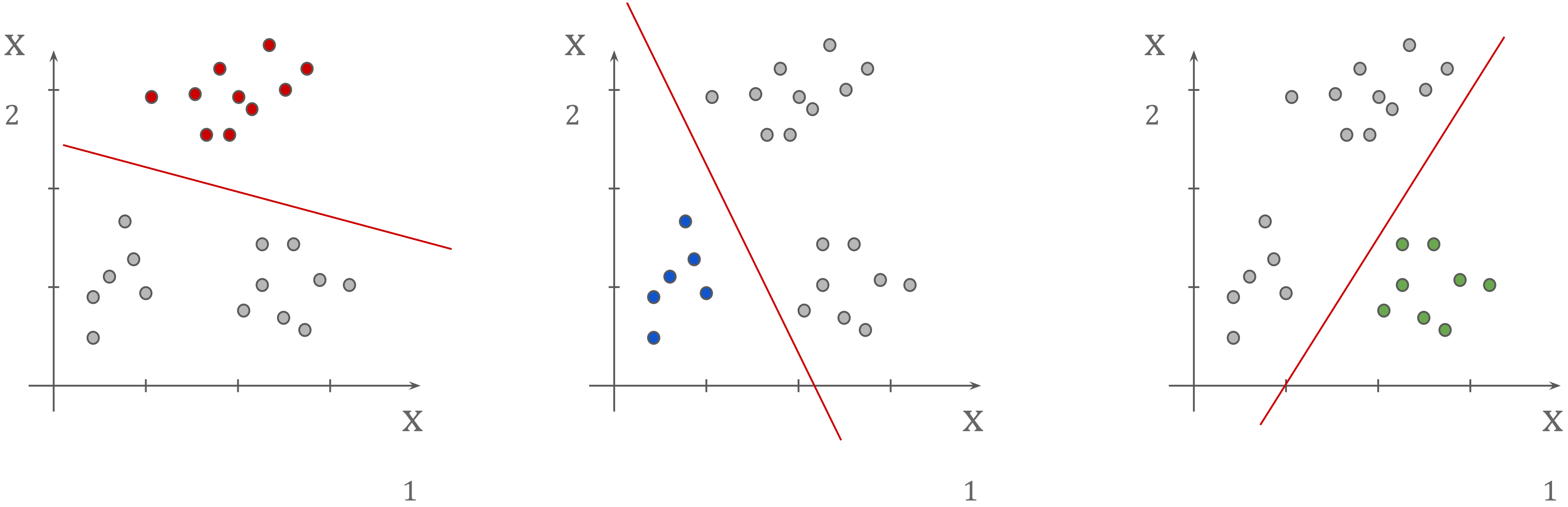
One vs Rest

- Esse método consiste em treinar C classificadores distintos, onde cada um classifica se as instâncias são da classe C_i ou não
- Então escolhemos como resultado a classe cujo classificador deu a maior probabilidade
- É importante notar que os outputs dos classificadores não são probabilidades mutuamente exclusivas e sua soma em geral é diferente de 1

One vs Rest



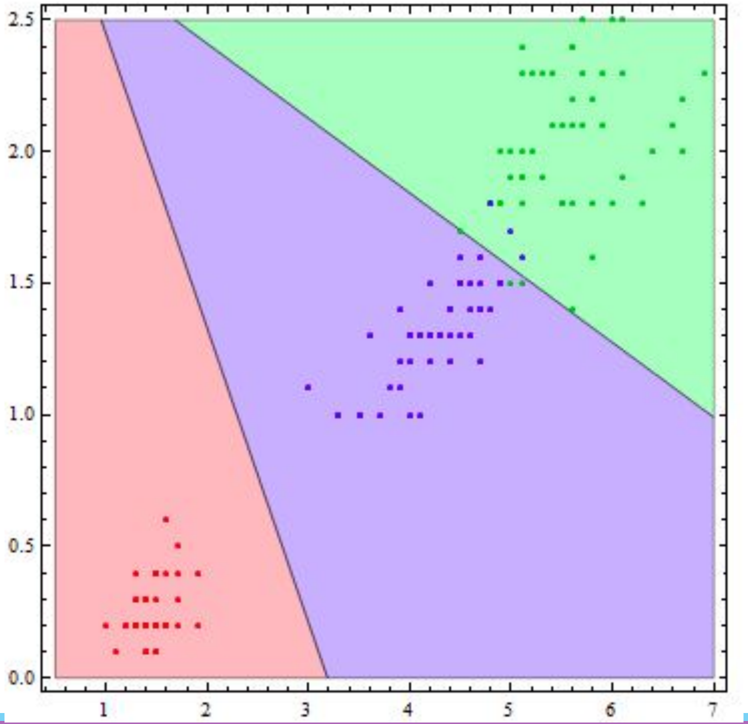
One vs Rest



Multinomial Logistic Regression

- Teremos um conjunto de pesos W_i que geram a probabilidade de i pertencer a classe i
- Teremos vários outputs $W^T_i x = y_i$, então aplicamos a **função softmax**
- Nossa saída é um vetor de probabilidades $P(C_i | x, W_i)$
- $softmax(y) = \frac{e^{y_i}}{\sum e^{y_j}}$ para cada coordenada
- Temos um único classificador e portanto $\Sigma P(C_i \mid x, W_i) = 1$

Multinomial Logistic Regression



Regularização

- Os métodos usados para regressão linear também funcionam para regressão logística: ridge, LASSO e ElasticNet
- Novamente é fundamental que os dados sejam normalizados

Na prática

- Dados numéricos
  - One-Hot Encoding
- Escala das features faz diferença
  - Feature scaling
  - No scikit a regressão normal usa regularização L_2, então é muito importante normalizar as features
- Vários métodos de solução além de gradient descent
  - Métodos de segunda ordem, convergem mais rápido, porém cada um tem suas restrições

Na prática: otimizadores

Reg/Solver	liblinear	lbfgs(default)	newton-cg	sag	saga
Multinom + L2	✘	✅	✅	✅	✅
OVR + L2	✅	✅	✅	✅	✅
Multinom + L1	✘	✘	✘	✘	✅
OVR + L1	✅	✘	✘	✘	✅
ElasticNet	✘	✘	✘	✘	✅
Nenhuma	✘	✅	✅	✅	✅

Leituras avançadas

- Regressão logística a partir do teorema de Bayes (link)
- Cornell: Relação entre naive-bayes e Regressão Logística (link)
- O motivo para usarmos cross entropy ao invés de MSE (link)
- Derivando a função de custo (link)