Regressão logística

Classificação –

Online transaction → Fraudulento / Não

E-mail → spam / não

Tumor → maligno / benigno

Saída: $y \in \{0, 1\}$ — classif. binária

$y = 0$: negative class, benign

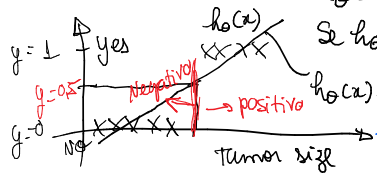
$y = 1$: positive class, malignant

$y \in \{0, 1, 2, 3\}$

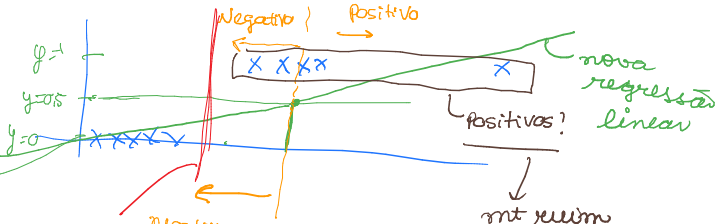
↳ multi-class classif. problem

Se eu usasse linear regression, poderia estabelecer: Se $h_\theta(x) \geq 0,5$, $y = 1$

Se $h_\theta(x) < 0,5$, predict $y = 0$



✗ → com linear reg. com esse exemplo, a linha fica pior



O correta seria separar aqui, mas adicionar um training example à direita alteraria significativamente a reta.

Logistic Regression

$0 \leq h_\theta(x) \leq 1$

classification algorithm, apesar do nome.

└ output é DISCRETE VALUE, ao contrário de uma regressão.

Binary Classification

$x^{(i)} \rightarrow y^{(i)}$ será o label p/ aquele training example.

Qual hipótese usaremos?

- Queremos nosso classificador de hipóteses entre zero e 1.

$0 \leq h_\theta(x) \leq 1$

$0 \leq g(z) \leq 1$

linear reg:

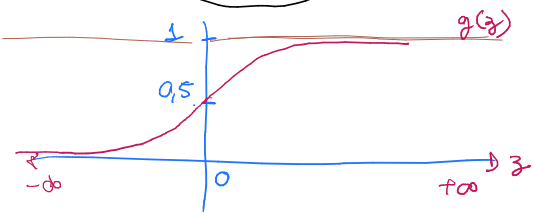
$h_\theta(x) = \theta^T x$

logística:

$h_\theta(x) = g(\theta^T x)$

$g(z) = \frac{1}{1 + e^{-z}}$ — Sigmoid function ou logistic function

$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$ — preciso ajustar θ aos nossos dados. Essa hipótese permitirá as predições



Interpretação do output da hipótese:

$h_\theta(x)$ → probabilidade estimada de que $y=1$ para a entrada x.

Ex:

$$x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tamanho do tumor} \end{bmatrix}$$

$h_\theta(x) = 0,7$

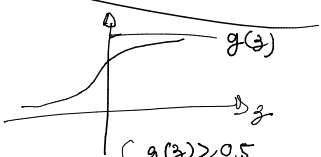
a prob. de ser maligno é 70%

$h_\theta(x) = P(y=1 \mid x; \theta)$

Prob. de $y=1$, dado x, parametrizado por θ.

Decision Boundary

$h_\theta(x) = g(\theta^T x)$



$g(z) \geq 0,5$

Quero tentar entender melhor quando essa hipótese prediz que $y=1$ ou $y=0$.

$h_\theta(x) = g(\theta^T x)$

Quando $z \geq 0$

$g(\theta^T x) \geq 0,5$ qd $\theta^T x \geq 0$ → prediz 1

Prediga $y=0$ se $h_\theta(x) < 0,5$

↳ se $\theta^T x < 0$.

Se temos a fç:

$$h_\theta(x) = g(\underbrace{\theta_0 + \theta_1 x_1 + \theta_2 x_2}_{z})$$

$$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

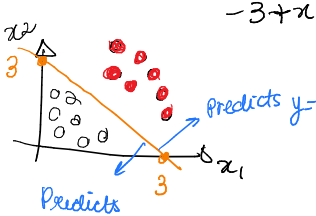
Ele vai predizer $y=1$ se

$$-3 + x_1 + x_2 \geq 0$$

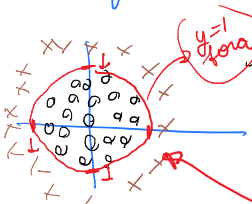
E predizer $y=0$ se

$$-3 + x_1 + x_2 < 0$$

$$x_1 + x_2 \geq 3$$



non-linear decision boundaries



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

$y=1$ se $-1 + x_1^2 + x_2^2 \geq 0$

ou seja

$$x_1^2 + x_2^2 \geq 1$$

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_6 x_1^3 x_2 + \dots)$$

Pergunta:

$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$

How do we fit parameters θ ?

$$x \in \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad x_0 = 1 \,, \quad y \in \{0, 1\}$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

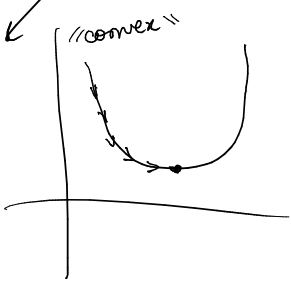
$$\text{Cost}(h_\theta(x), y) = \frac{1}{2}(h_\theta(x) - y)^2$$

Essa cost-function é não-convexa.

non-convex function - many local optima

non-convex

(many local optima)



Com GD, não é garantida a convergência p/ mínimo global.

O problema dessa não-linearidade que aparece aqui é uma de não convexa.

Preciso de uma cost-function convexa e que possamos aplicar gradient descent.

Métodos preditivos
/ prevê rótulos
- Classif.
- Regressão
(prevê números)

Métodos descritivos
- associação
- agrupamento
- detecção de desvios
- padrões sequenciais
- sumarização

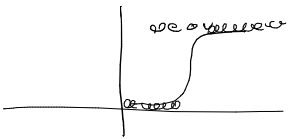
Relação entre a idade e se vai ou ñ pagar um crédito
↳ pagar (atributo-classe)

↳ Idade (atrib. previsor)

Base de treinam.
- naive bayes - analisa a probabilidade
  - risco alto, baixo, moderado
- decision trees
  - classificação
- aprend. por regras
  - ñ gera um modelo!
- KNN - baseado em instância
  - faz cálculo da distância
  - i vizinho q tem a menor distância

Regressão logística

↳ encontrar a melhor função que vai desenhar o "S" do gráfico



Avaliar o algoritmo:

- base de teste ≠ base de treinamento
  ↳ com isso, tenho ideia de acerto do meu modelo.

$$h_\theta(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1 + \dots)}} = p$$

Para construir o algoritmo → transformação LOGIT

$$\log\left(\frac{p}{1-p}\right) = b_0 + b_1 x$$

→ O objetivo é encontrar a melhor linha

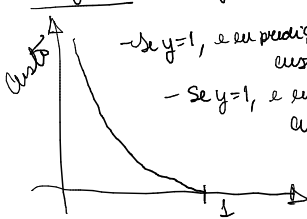
os coeficientes correspondem a pesos p/ cada atributo!

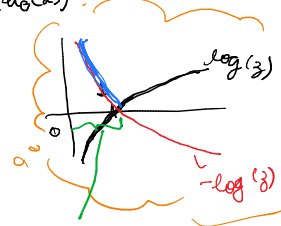
Preciso de uma outra cost-function pq a da regressão linear fica altamente $\cap$-convexa!

Se a regressão dá um valor $h_\theta(x)$, nosso custo será

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

Se $y=1$: Cost-function será $-\log(h_\theta(x))$





estou interessada nesse range!

Se $h_\theta(x)=0$ → $P(y=1|x;\theta)=0$ mas $y=1$, penalizo com custo no alto!

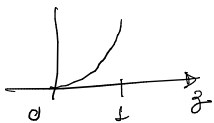
hipótese tá dizendo que a chance de y ser $1=0$.

("a chance do seu tumor ser maligno é zero" mas na verdade $y=1$)

Se $y=0$, o custo será $-\log(1-h_\theta(x))$

Se eu prediço $y=1$, vai pra o custo.

Se predigo $y=0$, custo será zero.



$-\log(1-z)$

Com uma escolha particular de Cost-function, isso nos dará um problema de otimização convexa.

$J(\theta)$ será convexa e livre de ótimos locais

Gradientes Conjugado, BFGS e L-BFGS

↳ formas + sofisticadas de otimizar θ q podem substituir o gradient-descent.

Primeiro tenho que escrever a fç que avalia as seguintes funções p/ um dado θ.

$J(\theta)$

$\frac{\partial}{\partial \theta_j} J(\theta)$

Multiclass classification problems

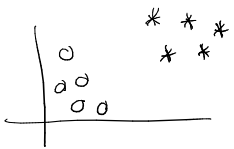
- One vs. all

(e-mail em folders diferentes ou automaticamente tag e-mails

- work $y=1$
- family $y=2$
- friends... $y=3$
- hobby $y=4$

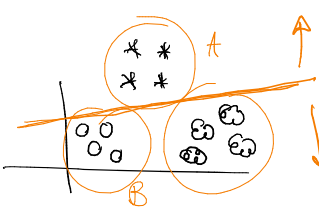
↓ multiclass

Classificação Binária

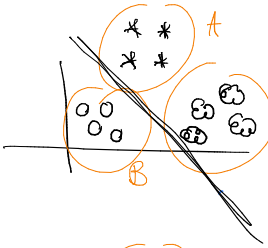


Multiclass classification classification

⇒ I learn it into 3 separate classification problems. A é o positive e o resto é negative



Pergunto: é da classe A?



trato B como positivo

trato C como positivo

onde trato i como a classe positiva

- Summarize, we fit 3 classifiers

$$h_\theta^{(i)}(x) = P(y=1 \mid x, \theta)$$

qual a probabilidade de y ser da classe i, dado x parametrizado por θ

- train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class (i) to predict the probability that $y = i$
- on a new input x, to make a prediction, pick the class i that maximizes

$$\max_i h_\theta^{(i)}(x)$$

Assim, se tenho K classes e uso one vs. all, tenho que treinar K classifiers.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{cost}\left(h_\theta(x^{(i)}), y^{(i)}\right)$$

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

Mas y é sempre 0 ou 1.

Isso equivale a escrever:

$$\text{cost}\left(h_\theta(x), y\right) = -y \log\left(h_\theta(x)\right) - (1-y) \log\left(1-h_\theta(x)\right)$$

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

Esta cost function foi derivada da estatística usando Princípio da MLE

↓

Serve p/ achar eficientemente parâmetros p/ diversos modelos

Além de ser convexa.

To fit parameters θ:

$$\min_\theta J(\theta) \rightarrow \text{get } \theta$$

to make a prediction given new x:

$$\text{output } h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

isso vai medar $P(y=1 \mid x;\theta)$

Vou minimizar cost-function por GDesc.

Quero $\min_{\theta} J(\theta)$

Repeat {

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

}

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

Algoritmo idêntico a reg. linear!

O que muda da linear p/ a logística é que a definição da nossa hipótese mudou.

Antes	Agora:
$h_\theta(x) = \theta^T x$	$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$

Não esquecer do feature scaling!

Vectorized implementation:

$$h = g(X\theta)$$

$$J(\theta) = \frac{1}{m} \cdot (-y^T \log(h) - (1-y)^T \log(1-h))$$

(y: real; h: predito)

$$\theta := \theta - \frac{\alpha}{m} X^T (g(X\theta) - \vec{y})$$

Optimização p/ logistic regression
ficam much more quickly than GD.

→ E os algoritmos escalam melhor p/ very large ML problems.

P/ cada iteração, o GD calcula

$$J(\theta) \text{ e } \frac{\partial}{\partial \theta_j} J(\theta) \qquad \forall j = 0, 1, \dots, n$$

Optimization algorithms

→ GD

→ Conjugate Gradients
→ BFGS
→ L-BFGS

- Não precisa pegar manualmente um α
- often faster than GD
- MORE COMPLEX!

dá as derivadas parciais

```
function [jVal, gradient] = costFunction(theta)
  jVal = (theta(1)-5)^2 + ... + (theta(2)-5)^2;
  gradient(1) = 2*(theta(1)-5)
```

dá J(θ)

isso em implementação!

fminunc - function minimization unconstrained

@ ⇒ ponteiro p/ a costFunction

vce se tem no matlab

fminunc

help fminunc

Se é large ML problem, use esses algoritmos em vez de GD