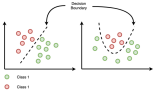
**Ciência de Dados Regressão logística**

**Renato Moraes Silva renato.silva@facens.br**

**Características **

**Entrada**: **conjunto rotulado** de observações representado por variáveis contínuas ou categóricas 

**Meta**: encontrar **fronteira de decisão** que separe classes

**Saída**: **probabilidade** de uma certa amostra

desconhecida pertencer a uma classe

**Pipeline**

Tiago A. Almeida

Pipeline conjunto de dados

*x*11 *x*12 ... *x*21 *x*22 ...

*x*1n *x*2n

*y*1 *y*2

*f*(**x**)

...... .. ...

Regressão

Classificador

*x*m1 *x*m2 ...

*x*mn

*y*m

Logística

**Treinamento**

***Objetivo****:* predizer corretamente o rótulo de novos dados

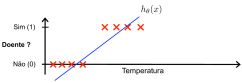
xin x ... **x** xi1 i2 i novo dado

*f*(**x**)

Classificador

*f*(**x**i)

Predição **Teste**

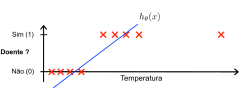
**Regressão Linear **

Seja: *hθ*(*x*) = *θ*0 + *θ*1*x*1 + *θ*2*x*2 + *· · ·* + *θnxn*

**Classificador**

****Se *hθ*(*x*) *≥* 0*.*5, classe (*y*) = 1

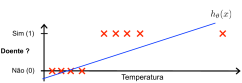
Se *hθ*(*x*) *<* 0*.*5, classe (*y*) = 0

**Regressão Linear **

**Classificador**

****Se *hθ*(*x*) *≥* 0*.*5, classe (*y*) = 1

Se *hθ*(*x*) *<* 0*.*5, classe (*y*) = 0

**Regressão Linear **

**Classificador**

****Se *hθ*(*x*) *≥* 0*.*5, classe (*y*) = 1

Se *hθ*(*x*) *<* 0*.*5, classe (*y*) = 0

**Regressão Logística **

Na classificação *y ∈* 0*,* 1

Na regressão, *hθ*(*x*) pode ser *>* 1 ou *<* 0: 0 *≤ hθ*(*x*) *≤* 1

Na regressão logística?

**Regressão Logística **

Na classificação *y ∈* 0*,* 1

Na regressão, *hθ*(*x*) pode ser *>* 1 ou *<* 0: 0 *≤ hθ*(*x*) *≤* 1

Na regressão logística?

Como: **alterar a representação da hipótese?**

**Hipótese **

Na regressão linear : *hθ*(*x*) = *θTx*

Entretanto, deseja-se que 0 *≤ hθ*(*x*) *≤* 1

Para isso, é necessário usar função logística *g*(*x*) (ou sigmoidal): *hθ*(*x*) = *gθTx*

**Hipótese **

**Função logística**

****Hipótese: *hθ*(*x*) = *gθTx*

Mapeamento: *g*(*z*) = 1 

1 + *e−z*

**Hipótese **

**Função logística**

****Hipótese: *hθ*(*x*) = *gθTx*

Mapeamento: *g*(*z*) = 1 

1 + *e−z*

**Para *z →* +*∞*, *g*(*z*) *→* 1 e para 

*z → −∞*, *g*(*z*) *→* 0. Precisamente,

*g*(0) = 0*.*5

**Hipótese **

**Função logística**

****Hipótese: *hθ*(*x*) = *gθTx*

Mapeamento: *g*(*z*) = 1 

1 + *e−z*

**Para *z →* +*∞*, *g*(*z*) *→* 1 e para 

*z → −∞*, *g*(*z*) *→* 0. Precisamente,

*g*(0) = 0*.*5

Portanto: *hθ*(*x*) = 1 

1 + *e−θT x*

**Hipótese **

Hipótese: *hθ*(*x*) = 1

1 + *e−θT x*

*hθ*(*x*) = probabilidade da amostra pertencer à classe *y* = 1, dado que ela possui os atributos *x*

*P*(*y* = 1*|x*; *θ*) + *P*(*y* = 0*|x*; *θ*) = 1

*P*(*y* = 0*|x*; *θ*) = 1 *− P*(*y* = 1*|x*; *θ*)

**Hipótese **

Hipótese: *hθ*(*x*) = 1

1 + *e−θT x*

*hθ*(*x*) = probabilidade da amostra pertencer à classe *y* = 1, dado que ela possui os atributos *x*

Se *hθ*(*x*) = 0*.*9 para classificação de paciente, então conclui-se que existe 90% de chance do paciente estar doente

**Limite de decisão **Regressão logística

*hθ*(*x*) = *gθTx* *g*(*z*) = 1

1 + *e−z*

Suponha que *y* = 1, se *hθ*(*x*) *≥* 0*.*5 

*θTx ≥* 0

Suponha que *y* = 0, se *hθ*(*x*) *<* 0*.*5

*θTx <* 0

**Função custo Treinamento**:

n



*x*(1)*, y*(1) *,**x*(2)*, y*(2) *, · · · ,**x*(*m*)*, y*(*m*) o

*x* =

 

*x*0 *x*1 ...

*xn*



*x*0 = 1*, y ∈ {*0*,* 1*}*

*hθ*(*x*) = 1

1 + *e−θT x*

Como escolher os **parâmetros** *θ*?

**Função custo**

**Regressão Linear**: *J*(*θ*) = 1*m*X*m*

1

2

*i*=1



*hθ*(*x*(*i*)) *− y*(*i*) 2

**Função custo**

**Regressão Linear**: *J*(*θ*) = 1*m*X*m*

1

2

*i*=1



*hθ*(*x*(*i*)) *− y*(*i*) 2

**Regressão Logística**: *J*(*θ*) = 1*m*X*m i*=1

Cost

*hθ*(*x*(*i*))*, y*(*i*)

**Função custo**

**Regressão Linear**: *J*(*θ*) = 1*m*X*m*

1

2

*i*=1



*hθ*(*x*(*i*)) *− y*(*i*) 2

**Regressão Logística**: *J*(*θ*) = 1*m*X*m i*=1

Cost

*hθ*(*x*(*i*))*, y*(*i*)

Cost

*hθ*(*x*(*i*))*, y*(*i*) =12*hθ*(*x*(*i*)) *− y*(*i*) 2 *hθ*(*x*) = 1

1 + *e−θT x*

**Função custo não é convexa!**

**Funções convexas** 

*Funções Convexas*

Facens - Especialização em Inteligência Artificial Aplicada 19

**Funções convexas** 

*Funções Convexas*

Uma função é chamada convexa se o segmento de reta entre quaisquer dois pontos no gráfico se encontra acima ou sobre o gráfico.

Facens - Especialização em Inteligência Artificial Aplicada 20

**Funções convexas** 

*Funções Convexas*

Uma função é chamada convexa se o segmento de reta entre quaisquer dois pontos no gráfico se encontra acima ou sobre o gráfico.

Facens - Especialização em Inteligência Artificial Aplicada 21

**Funções convexas** 

*Funções Convexas*

Uma função é chamada convexa se o segmento de reta entre quaisquer dois pontos no gráfico se encontra acima ou sobre o gráfico.

Facens - Especialização em Inteligência Artificial Aplicada 22

**Funções convexas** 

*Funções Convexas*

Uma função é chamada convexa se o segmento de reta entre quaisquer dois pontos no gráfico se encontra acima ou sobre o gráfico.

Facens - Especialização em Inteligência Artificial Aplicada 23

**Função Custo **

O algoritmo usado durante o aprendizado da regressão logística é o gradiente descendente.

Ele requer que a função seja convexa

A função logística não é sempre convexa

Podemos usar o **logaritmo** para resolver o problema

**Função Custo Entropia cruzada**: combinação de duas funções (uma para cada

rótulo)

*J*(*θ*) = 1*m*X*m*

*i*=1

Cost

*hθ*(*x*(*i*))*, y*(*i*)

(*−* log (*hθ*(*x*)) se *y* = 1

Cost(*hθ*(*x*)*, y*) =

*−* log (1 *− hθ*(*x*)) se *y* = 0

**Função Custo Entropia cruzada**: combinação de duas funções (uma para cada

rótulo)

*J*(*θ*) = 1*m*X*m*

*i*=1

Cost

*hθ*(*x*(*i*))*, y*(*i*)

(*−* log (*hθ*(*x*)) se *y* = 1

Cost(*hθ*(*x*)*, y*) =

*−* log (1 *− hθ*(*x*)) se *y* = 0

*Cost* = 0 se *y* = 1*, hθ*(*x*) = 1 *hθ*(*x*) = 0*, Cost →* inf

**Função Custo Entropia cruzada**: combinação de duas funções (uma para cada

rótulo)

*J*(*θ*) = 1*m*X*m*

*i*=1

Cost

*hθ*(*x*(*i*))*, y*(*i*)

(*−* log (*hθ*(*x*)) se *y* = 1

Cost(*hθ*(*x*)*, y*) =

*Cost* = 0 se *y* = 0*, hθ*(*x*) = 0 *hθ*(*x*) = 1*, Cost →* inf

*−* log (1 *− hθ*(*x*)) se *y* = 0

**Função Custo **

**Entropia cruzada**: combinação de duas funções (uma para cada rótulo)

*J*(*θ*) = *1m*X*m i*=*1*

Cost

*hθ*(*x*(*i*))*, y*(*i*)

Cost(*hθ*(*x*)*, y*) =

(*−* log (*hθ*(*x*))*,* se *y* = 1 *−* log (1 *− hθ*(*x*))*,* se *y* = 0

Como y = 1 ou y = 0, é possível simplificar a função Cost para: Cost(*hθ*(*x*)*, y*) =*−y* log (*hθ*(*x*)) *−* (*1 − y*) log (*1 − hθ*(*x*))

**Função Custo **

**Entropia cruzada**: combinação de duas funções (uma para cada rótulo)

*J*(*θ*) = *1m*X*m i*=*1*

Cost

*hθ*(*x*(*i*))*, y*(*i*)

Cost(*hθ*(*x*)*, y*) =

(*−* log (*hθ*(*x*))*,* se *y* = 1 *−* log (1 *− hθ*(*x*))*,* se *y* = 0

Como y = 1 ou y = 0, é possível simplificar a função Cost para: Se *y* = 0

Cost(*hθ*(*x*)*, y*) =*−y* log (*hθ*(*x*)) *−* (*1 − y*) log (*1 − hθ*(*x*))

**Função Custo **

**Entropia cruzada**: combinação de duas funções (uma para cada rótulo)

*J*(*θ*) = *1m*X*m i*=*1*

Cost

*hθ*(*x*(*i*))*, y*(*i*)

Cost(*hθ*(*x*)*, y*) =

(*−* log (*hθ*(*x*))*,* se *y* = 1 *−* log (1 *− hθ*(*x*))*,* se *y* = 0

Como y = 1 ou y = 0, é possível simplificar a função Cost para: Se *y* = 1

Cost(*hθ*(*x*)*, y*) =*−y* log (*hθ*(*x*)) *−* (*1 − y*) log (*1 − hθ*(*x*))

**Gradiente descendente Regressão logística**

*J*(*θ*) = *1m*X*m i*=*1*

Cost

*hθ*(*x*(*i*))*, y*(*i*)

= *−1m*X*m i*=*1*

h

*y*(*i*)log *hθ*(*x*(*i*))+*1 − y*(*i*) log *1 − hθ*(*x*(*i*)) i

**Objetivo**: ajustar *θ*

min*θJ*(*θ*)

**Classificador** (dado novo *x*): *hθ*(*x*) = 1 1 + *e−θT x*

**Gradiente descendente **

h

*J*(*θ*) = *−1m*X*m i*=*1*

*y*(*i*)log *hθ*(*x*(*i*))+*1 − y*(*i*) log *1 − hθ*(*x*(*i*)) i

Essa forma de calcular o custo é

conhecida como Entropia Cruzada

Deseja-se min*θ J*(*θ*)

**Método do gradiente**:

Repita *{*

*θj* := *θj − α∂∂θjJ*(*θ*)

*}*

**Atualização simultânea para todo *θj*

**Gradiente descendente **

h

*J*(*θ*) = *−1m*X*m i*=*1*

*y*(*i*)log *hθ*(*x*(*i*))+*1 − y*(*i*) log *1 − hθ*(*x*(*i*)) i

Essa forma de calcular o custo é conhecida como Entropia Cruzada

Deseja-se min*θ J*(*θ*)

**Método do gradiente**:

Repita *{*

Parece idêntico à Regressão Linear

*θj* := *θj − α∂∂θjJ*(*θ*)

*}*

**Atualização simultânea para todo *θj*

**Gradiente descendente **

h

*J*(*θ*) = *−1m*X*m i*=*1*

*y*(*i*)log *hθ*(*x*(*i*))+*1 − y*(*i*) log *1 − hθ*(*x*(*i*)) i

Essa forma de calcular o custo é conhecida como Entropia Cruzada

Deseja-se min*θ J*(*θ*)

**Método do gradiente**:

Repita *{*

*hθ*(*x*) = *1 1* + *e−θT x*

*θj* := *θj − α∂∂θjJ*(*θ*)

*}*

**Atualização simultânea para todo *θj*

**Gradiente descendente **

h

*y*(*i*)log *hθ*(*x*(*i*))+*1 − y*(*i*) log *1 − hθ*(*x*(*i*)) i

*J*(*θ*) = *−1m*X*m*

*i*=*1*

Essa forma de calcular o custo é

conhecida como Entropia Cruzada

Deseja-se min*θ J*(*θ*)

**Método do gradiente**:

Repita *{*

*θj* := *θj − α*1*m*X*m*

*i*=1

*hθ*(*x*) = *1*

*1* + *e−θT x*

*hθ*(*x*(*i*)) *− y*(*i*) *x*(*i*)

*}*

**Atualização simultânea para todo *θj*

**Técnicas avançadas de otimização **

Existem funções de otimização que desenvolvem o papel do GD de forma muito mais eficiente

Gradiente descendente estocástico

Gradiente conjugado

BFGS

L-BFGS

**Técnicas avançadas de otimização **

Vantagens

Normalmente são **mais rápidas**

****Necessárias em programas de **grande porte**

****Não há necessidade de **ajustar manualmente** o valor do passo *α*

Desvantagens

Maior **complexidade** de interpretação e implementação

**Técnicas avançadas de otimização **

**Implementação**

1 import scipy.optimize

2

3 MaxIter = 100 # qtd. máxima de iterações

4

5 theta = np.zeros(n+1) # Inicializa os thetas

6

7 # minimiza a funcao de custo

8 result = scipy.optimize.minimize(

9 fun=funcaoCusto, # função objetivo a ser minimizada 10 x0=theta, # variável que deverá ser otimizada 11 args=(X, Y), # outros argumentos da função 12 method='BFGS', # método de otimização

13 jac=True, # indica que vetor gradiente será calculado na função objetivo e os thetas serão retornados

14 options={'maxiter': MaxIter, 'disp':True}) 15

16 theta = result.x #coleta os thetas retornados 17

18 custo, grad = funcaoCusto(theta, X, Y)

Tiago

Técnicas Avançadas de Otimização **Técnicas avançadas de otimização** 

**Implementação**

**Implementação**

**theta =**

**function [J, gradiente] = funcaoCusto(theta)**

**J = [ ];** código para calcular

código para calcular

**gradiente(1)= [ ];**

código para calcular

**gradiente(2)= [ ];**

código para calcular

**gradiente(n+1)= [ ];**

***Overfitting* vs *Underfitting ***

*Overfitting* (associado ao problema de alta variância) especialização nos dados de treinamento

desempenho ruim em novos dados de entrada problema na generalização

*Underfitting* (associado ao problema de alto viés) baixa complexidade

desempenho ruim até mesmo nos dados conhecidos

ago . me

Viés x Variância 

**Viés x Variância**

e

d

a

x

a

T

Ano

**Alto viés** *(Underfit)*

**Alta variância**

*(Overfit)* **OK**

**Overfitting **

**Possíveis soluções**

****Reduzir a quantidade de atributos

**Seleção manual** dos atributos que deverão ser mantidos

Usar algoritmo para **seleção automática** de atributos

Regularização

Manter todos os atributos, mas **reduzir o valor de cada**

**parâmetro** *θ*

**Funciona bem quando há uma **grande quantidade de atributos**, cada um contribuindo um pouco para a predição final *y*

**Underfitting **

**Possíveis soluções**

****Aumentar a quantidade de atributos

Adicionar atributos polinomiais (*x*21*, x*22*, x*1*x*2*, ...*) reduzir o parâmetro de regularização

Tiago

Regularização: regressão linear**Regularização: regressão linear** 

**Regularização: regressão linear** Regularização: regressão linear

x x

Deseja-se reduzir a influência de e

Tiago A. Almeida 

Ti A. Almi

Tiago A. Almeida

Regularização: regressão linear **Regularização: regressão linear** 

x x

Opção: penalizar e

+1000*.✓*23 + 1000*.✓*24

**Regularização: regressão linear **

*J*(*θ*) = 12*m*X*m i*=1

"

*hθ*(*x*(*i*)) *− y*(*i*) 2+ *λ*X*n i*=1

#

*θ*2*j*

**Parâmetro**

**de regularização**

**Regularização: regressão linear **

*J*(*θ*) = 12*m*X*m i*=1

"

*hθ*(*x*(*i*)) *− y*(*i*) 2+ *λ*X*n i*=1

#

*θ*2*j*

Se *λ* for muito grande (1010), então: *θj ≈* 0

Portanto, haverá ***underfitting***

**Regularização: regressão logística **

"

*J*(*θ*) =

*−1m*X*m i*=*1*

*y*(*i*)log *hθ*(*x*(*i*))+*1 − y*(*i*) log *1 − hθ*(*x*(*i*)) #+*λ2m*X*n*

*θ2j*

*i*=*1*

Gradiente descendente Repita{

*θ*0 := *θ*0 *− α*1*m*X*m*

*hθ*(*x*(*i*)) *− y*(*i*) *x*(*i*)0

*θj*:= *θj − α*

"

1

*i*=1

X*m*

#

*hθ*(*x*(*i*)) *− y*(*i*) *x*(*i*)

*j* +*λmθj*

*m*

*i*=1

(*j* = 1*,* 2*,* 3*, ..., n*)

}

**Não usar regularização**

**no índice 0 (bias)**

Tiago A. Almeida

Técnicas Avançadas de Otimização **Técnicas avançadas de otimização** 

**Implementação**

**Implementação**

**function [J, gradiente] = funcaoCusto(theta)**

**J = [ ];** código para calcular

código para calcular

**gradiente(1)= [ ];**

código para calcular

**gradiente(2)= [ ];**

+

**gradiente(3)= [ ];** código para calcular

+

código para calcular

**gradiente(n+1)= [ ];**

**Problemas multiclasse **

1 Para cada classe do problema, construa um classificador binário para recuperar a pontuação de cada registro em relação a esta classe.

2 Atribua a classe cuja ativação seja mais alta.

**Abordagem para o problema: exemplo** *Abordagem para o problema: exemplo* 

0.22

0.36

0.87

Facens - Especialização em Inteligência Artificial Aplicada 33

**Referências **

Aulas do Prof. Tiago Agostinho de Almeida (UFSCar, campus de Sorocaba)

Aulas do Prof. Juvenal J. Duarte

CARVALHO, André Carlos Ponce de Leon Ferreira et al. Inteligência Artificial - Uma Abordagem de Aprendizado de Máquina. Disponível em: Minha Biblioteca, (2nd edição). Grupo GEN, 2021.