Mineração de Dados

Classificação

Sumário

Conceitos Básicos

Aprendizado Supervisionado e Não Supervisionado

- Aprendizado Supervisionado
  - Classificação e regressão
  - Supervisão: os dados de treinamento são acompanhados por valores esperados
  - Os valores esperados podem vir de observações, medições, indicações de um especialista, etc
  - Novas instâncias são classificadas com o que se aprendeu sobre os dados de treinamento
- Aprendizado Não Supervisionado
  - Não há um valor esperado nos dados de treinamento
  - Deve-se estabelecer uma relação entre elementos de um mesmo grupo com base em atributos

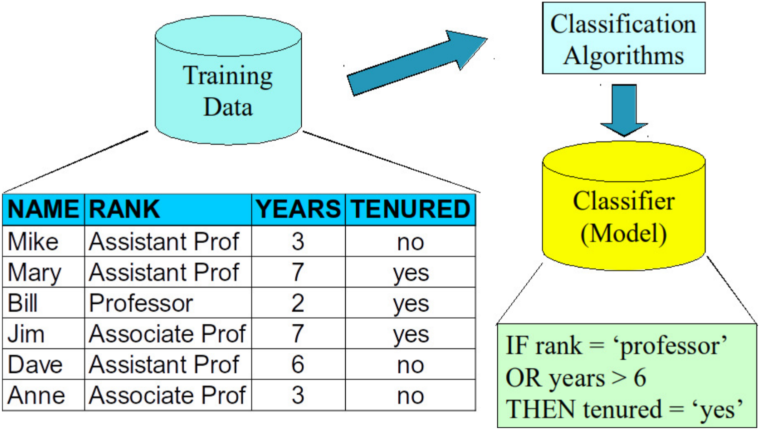
Regressão e Classificação

- Regressão
  - Predição numérica
  - Os modelos são funções de valores contínuos
- Classificação
  - O modelo deve prever uma classe para um conjunto de valores de entrada
- Algumas aplicações de classificação
  - Aprovação de empréstimos
  - Diagnóstico médico
  - Detecção de fraude
  - Categorização de páginas

Processo de Classificação

- Construção e utilização do modelo
- Construção
  - Como relacionar os atributos com o valor esperado
  - Cada tupla é associada a uma classe
  - Um conjunto de dados de treinamento é usado para criar o modelo
  - Um modelo pode ser representado por superfícies separadoras, regras de classificação, árvores de decisão ou expressões aritméticas

Construção do Modelo

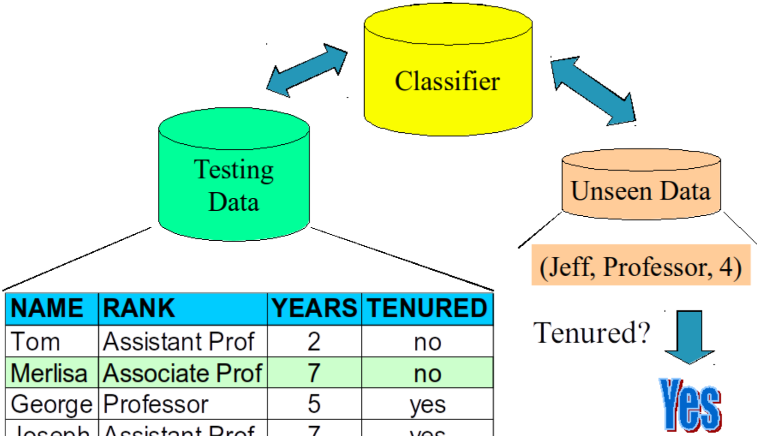


NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Processo de Classificação

- Construção e utilização do modelo
- Utilização do modelo
  - Estimar a acurácia do modelo
  - As classes indicadas pelo modelo para um conjunto de dados de teste podem ser usadas para determinar sua qualidade
  - Acurácia: porcentagem dos dados de teste que são corretamente classificados pelo modelo
  - O conjunto de dados de teste deve ser independente do conjunto de dados de treinamento
  - Se a acurácia (ou outra forma de avaliação de modelos) for aceitável, o modelo pode ser adotado para prever classes para novas instâncias
  - Um subconjunto de dados pode ser usado para selecionar modelos e/ou seus parâmetros e, neste caso, este é chamado de dados de validação

Utilização do Modelo



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Classificação Linear e Regressão Logística

Classificação via Mínimos Quadrados e Regressão Logística

- Método dos Mínimos Quadrados
- Classificação via Mínimos Quadrados
- Regressão Logística

Método dos Mínimos Quadrados

- Seja a matriz $\mathbf{A}$ definida como

$$\mathbf{A} = \begin{bmatrix} \phi_0(x_1) & \phi_1(x_1) & \dots & \phi_n(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \dots & \phi_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_m) & \phi_1(x_m) & \dots & \phi_n(x_m) \end{bmatrix}$$

- E o vetor $\mathbf{y}$ como

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

Método dos Mínimos Quadrados

- Então o Método dos Mínimos Quadrados pode ser usado para determinar os coeficientes lineares do modelo
- Para isso, deve-se resolver o sistema linear

$$\mathbf{A}^T\mathbf{A}\mathbf{c} = \mathbf{A}^T\mathbf{y}$$

Método dos Mínimos Quadrados: Python

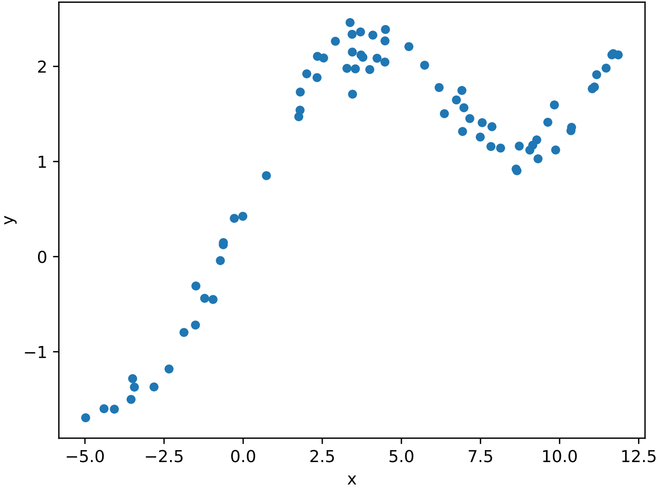
- Arquivo dos dados de entrada: `reg_2_tr_X.dat`
- Arquivo dos valores esperados: `reg_2_tr_Y.dat`
- `http://pandas.pydata.org/pandas-docs/stable/visualization.html`

x	y
4.4970249	2.3881414
1.7572682	1.4709548
2.0093448	1.9221703
-1.5073673	-0.71916443
5.2374846	2.2081153
...	...
-1.4937856	-0.30838877
2.3348046	1.8823554
4.4828742	2.268271
8.7256175	1.1620196

Método dos Mínimos Quadrados: Python

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
x = pd.read_csv("reg_2_tr_X.dat", header=0)
y = pd.read_csv("reg_2_tr_Y.dat", header=0)
data = pd.concat([x, y], axis=1)
data.plot.scatter(x="x", y="y")
# plt.savefig("grafico.eps", format="eps")
plt.show()
```

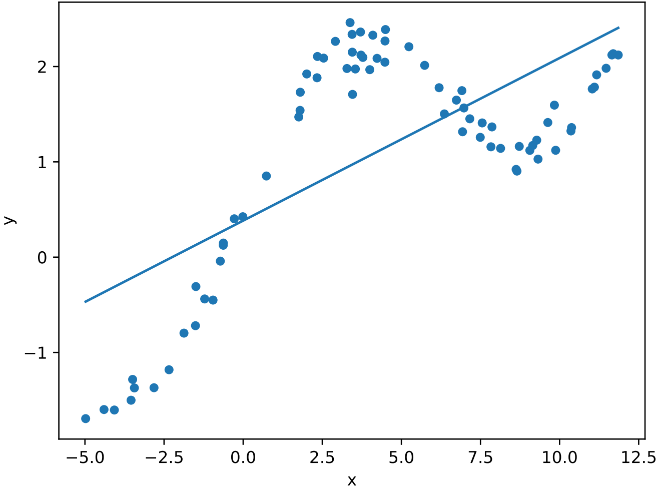
Método dos Mínimos Quadrados: Python



Método dos Mínimos Quadrados: Python

```
A = np.zeros(shape=(len(data), 2))
A[:,0] = 1
A[:,1] = data["x"]
A = np.matrix(A)
At = np.transpose(A)
Y = np.transpose(np.matrix(data["y"]))

x = np.linalg.solve(At*A, At*Y)

xt = np.linspace(min(data["x"]),max(data["x"]),100)
yt = x[0,0] + x[1,0]*xt

ax = data.plot.scatter(x="x", y="y")
plt.plot(xt, yt)
plt.show()
```

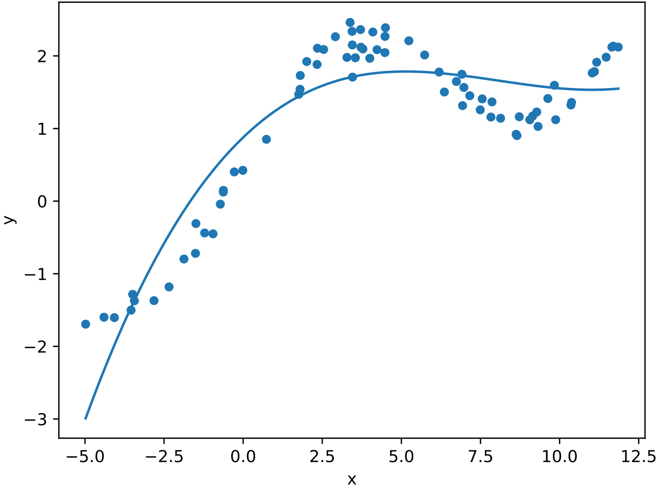
Método dos Mínimos Quadrados: Python



Método dos Mínimos Quadrados: Python

```
A = np.zeros(shape=(len(data), 4))
A[:,0] = 1
A[:,1] = data["x"]
A[:,2] = data["x"]*data["x"]
A[:,3] = data["x"]*data["x"]*data["x"]
A = np.matrix(A)
At = np.transpose(A)

x = np.linalg.solve(At*A, At*Y)

xt = np.linspace(min(data["x"]),max(data["x"]),100)
yt = x[0,0] + x[1,0]*xt + x[2,0]*xt*xt + x[3,0]*xt*xt*xt
ax = data.plot.scatter(x="x", y="y")
plt.plot(xt, yt)
plt.show()
```

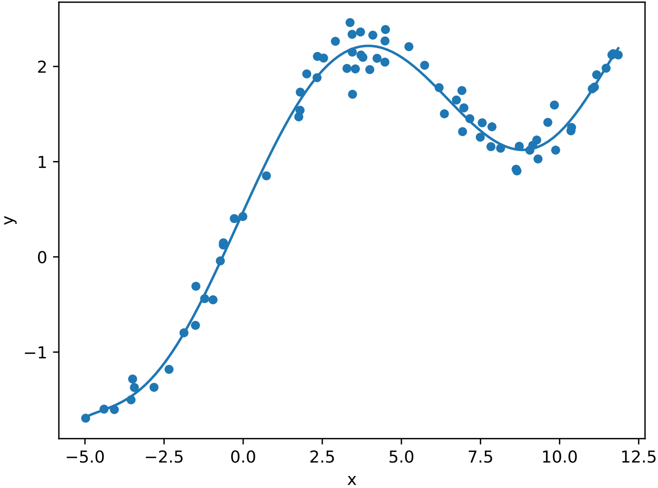
Método dos Mínimos Quadrados: Python



Método dos Mínimos Quadrados: Python

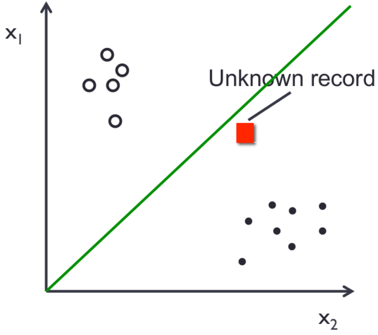
```
componentes = 7
A = np.zeros(shape=(len(data), componentes))
for i in range(0, componentes):
        A[:,i] = pow(data["x"], i)

A = np.matrix(A)
At = np.transpose(A)

x = np.linalg.solve(At*A, At*Y)

xt = np.linspace(min(data["x"]),max(data["x"]),100)
yt = np.zeros(shape=(len(xt)))
for i in range(0, componentes):
        yt += x[i,0] * pow(xt, i)

ax = data.plot.scatter(x="x", y="y")
plt.plot(xt, yt)
plt.show()
```

Método dos Mínimos Quadrados: Python

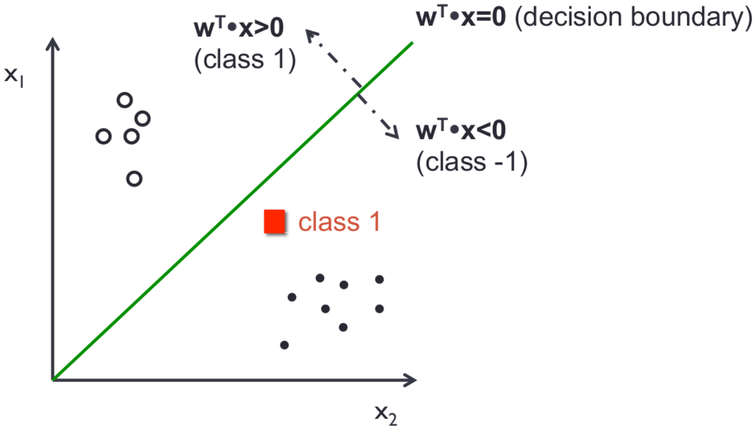


Classificação via Mínimos Quadrados

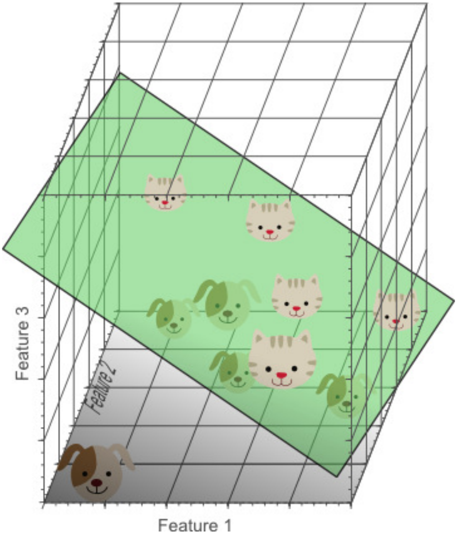
- O mesmo método pode ser usado para classificação
- O modelo gera uma superfície de separação



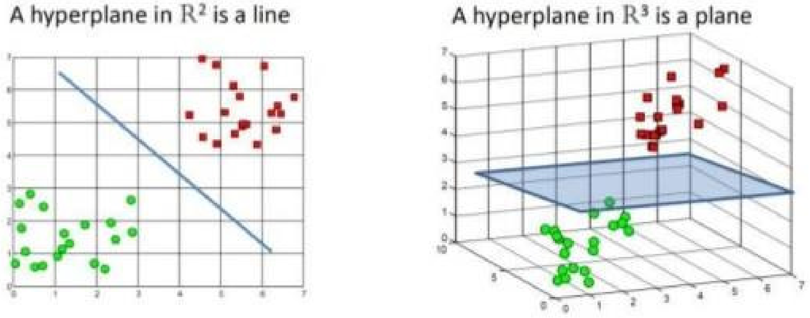
Classificação via Mínimos Quadrados



Classificação via Mínimos Quadrados



Classificação via Mínimos Quadrados



A hyperplane in $\mathbb{R}^n$ is an n-1 dimensional subspace

Método dos Mínimos Quadrados: Python

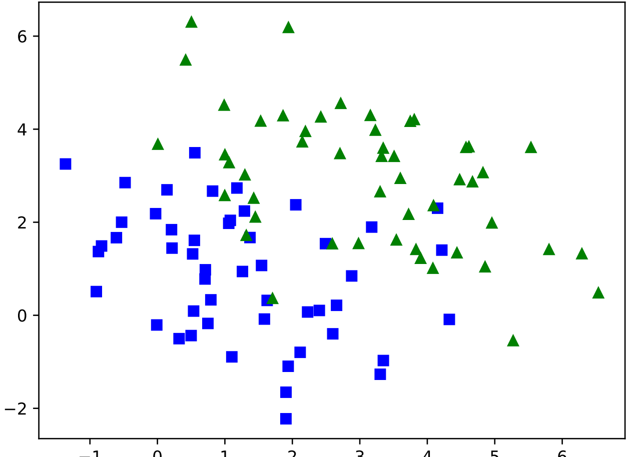
- Arquivo dos dados de entrada: `class_1_tr_X.dat`
- Arquivo dos valores esperados: `class_1_tr_Y.dat`

x1,x2	y
4.15252672368477,2.30191850430311	-1
1.37166852444448,1.67089522310923	-1
1.62651299618563,0.320022670251675	-1
1.90556026445907,-1.65342465419423	-1
2.11557633795811,-0.795657656851138	-1
...	...
1.42725086615869,2.51795284362126	1
5.80426651901857,1.41610146547005	1
5.2727032012369,-0.544831984744078	1
3.30064702289753,2.65651491151454	1

Método dos Mínimos Quadrados: Python

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

x = pd.read_csv("class_1_tr_X.dat", header=0)
y = pd.read_csv("class_1_tr_Y.dat", header=0)
data = pd.concat([x, y], axis=1)
x0 = data.loc[ data["y"] == -1 ]
x1 = data.loc[ data["y"] == 1 ]
plt.plot(x0["x1"],x0["x2"], 'bs', x1["x1"],x1["x2"], 'g^')
# plt.savefig("grafico.eps", format="eps")
plt.show()
```

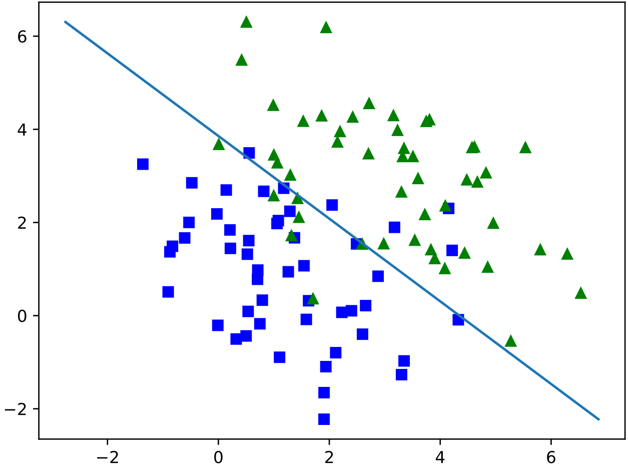
Classificação via Mínimos Quadrados



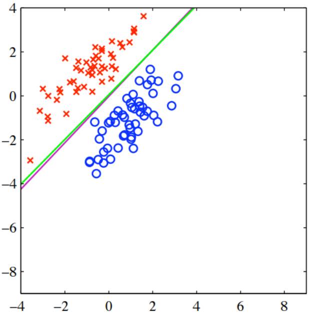
Método dos Mínimos Quadrados: Python

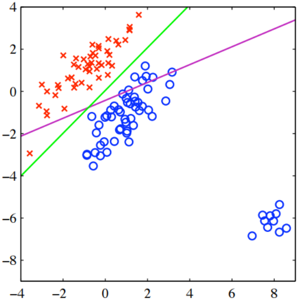
```
A = np.zeros(shape=(len(data), 3))
A[:,0] = 1
A[:,1] = data["x1"]
A[:,2] = data["x2"]

A = np.matrix(A)
At = np.transpose(A)
Y = np.transpose(np.matrix(data["y"]))

x = np.linalg.solve(At*A, At*Y)

x2t = np.linspace(min(data["x2"]),max(data["x2"]),100)
# equacao da reta em g(x1,x2) = 0
x1t = ( - x[0,0] - x[2,0]*x2t ) / x[1,0]

plt.plot(x0["x1"],x0["x2"], 'bs', x1["x1"],x1["x2"], 'g^', x1t, x2t, "-")
plt.show()
```

Classificação via Mínimos Quadrados



Classificação via Mínimos Quadrados



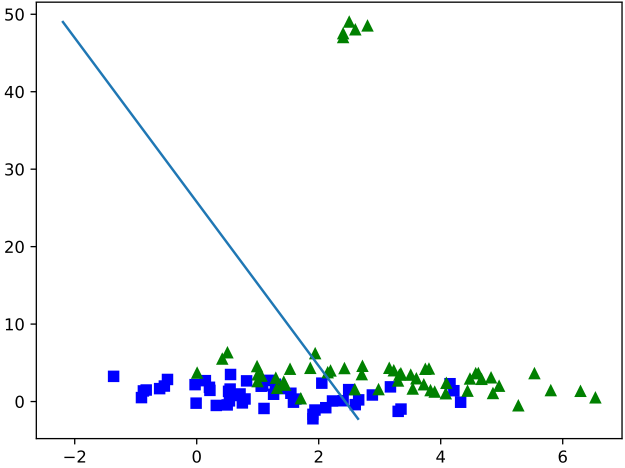


Método dos Mínimos Quadrados: Python

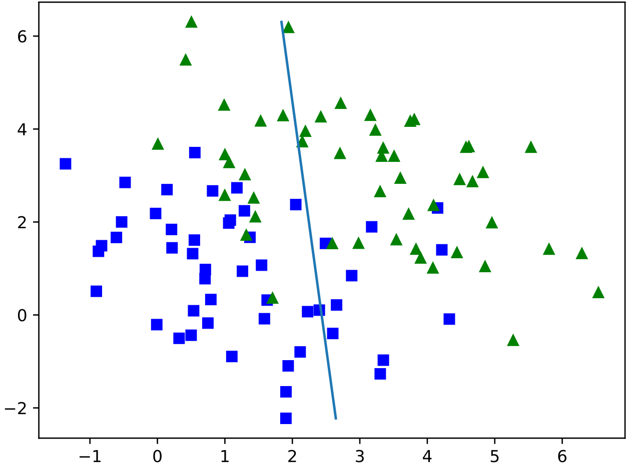
- Acrescentando os pontos que seguem nos arquivos de dados

x1,x2	y
2.5,49	1
2.6,48	1
2.4,47	1
2.8,48.5	1
2.4,47.5	1

Classificação via Mínimos Quadrados

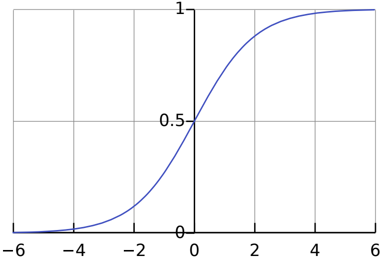


Classificação via Mínimos Quadrados

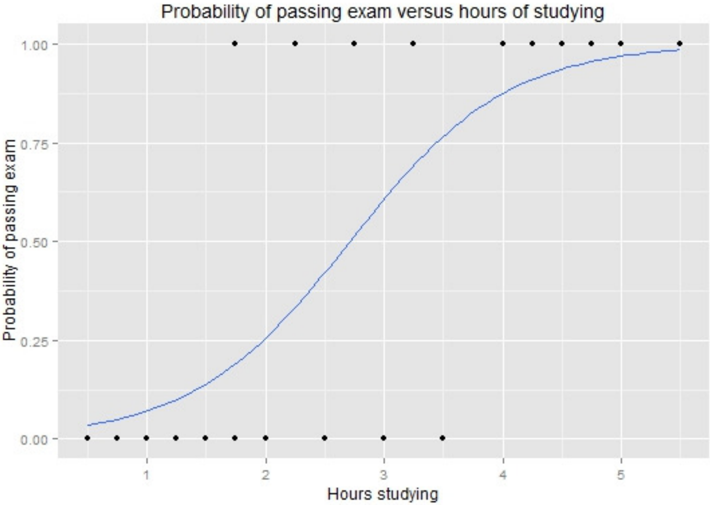


Regressão Logística

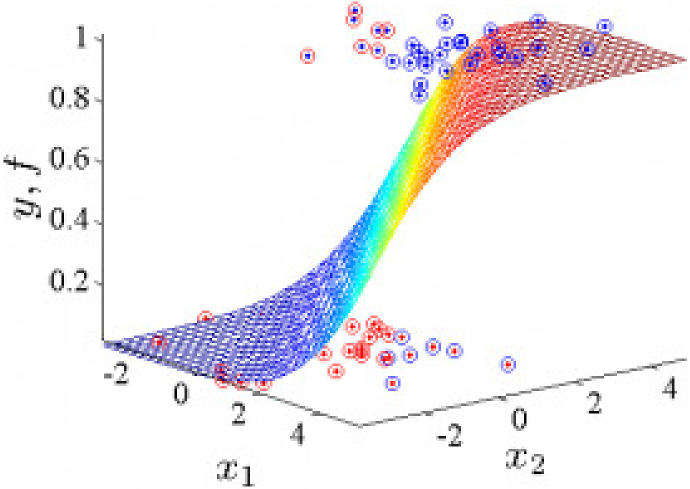
- $g(\mathbf{x}) = \dfrac{1}{1 + e^{-\mathbf{c}^\mathsf{T}\mathbf{x}}}$
- $g(\mathbf{x})$ pode ser interpretada como a probabilidade de x estar associado a um valor esperado 1 ($P(Y = 1|X)$)
- Função logística inversa: $\ln\left(\dfrac{g(\mathbf{x})}{1 - g(\mathbf{x})}\right) = \mathbf{c}^\mathsf{T}\mathbf{x}$



Regressão Logística



Regressão Logística



Regressão Logística

- O cálculo dos coeficientes do modelo pode ser feito via Gradiente Descendente (Estocástico)
- Sendo a verossimilhança dada por

$$L(\mathbf{c}) = \prod_{i=1}^{n} P(x_i)^{y_i}(1 - P(x_i))^{1-y_i}$$

- o objetivo é maximizar o $\log$ desta função, dada por

$$l(\mathbf{c}) = \sum_{i=1}^{n} y_i \log(P(x_i)) + (1 - y_1) \log(1 - P(x_1))$$

- A forma como os coeficientes $\mathbf{c}$ são calculados não será detalhada aqui

Regressão Logística: Python

- Arquivo dos dados de entrada: `class_1_tr_X.dat`
- Arquivo dos valores esperados: `class_1_tr_Y.dat`

x1,x2	y
4.15252672368477,2.30191850430311	0
1.37166852444448,1.67089522310923	0
1.62651299618563,0.320022670251675	0
1.90556026445907,-1.65342465419423	0
2.11557633795811,-0.795657656851138	0
...	...
1.42725086615869,2.51795284362126	1
5.80426651901857,1.41610146547005	1
5.2727032012369,-0.544831984744078	1
3.30064702289753,2.65651491151454	1

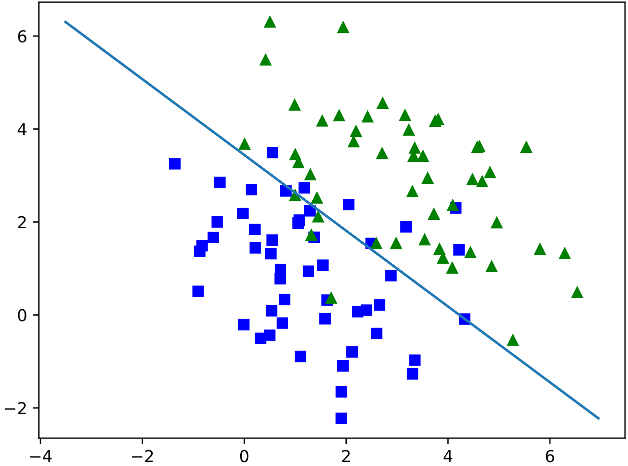
Regressão Logística: Python

```
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

x = pd.read_csv("class_1_tr_X.dat", header=0)
y = pd.read_csv("class_1_tr_Y.dat", header=0)
data = pd.concat([x, y], axis=1)
x0 = data.loc[ data["y"] == 0 ]
x1 = data.loc[ data["y"] == 1 ]
y = np.ravel(y)

model = LogisticRegression()
model = model.fit(x, y)

#model.coef_ indica os coeficientes do modelo
#model.intercept_ indica o vies (bias)
#model.score(x, y) indica a acuracia do modelo
```

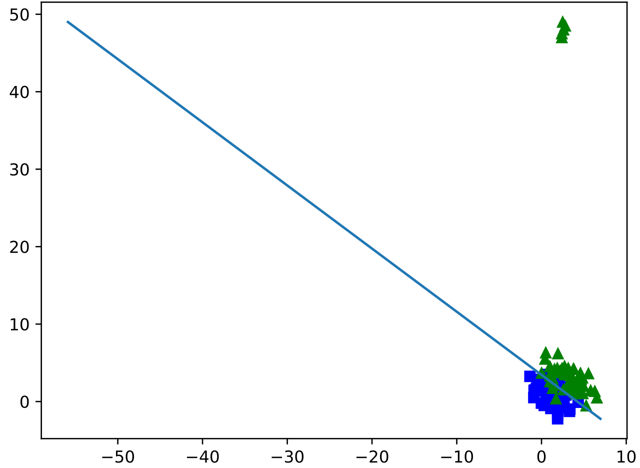
Regressão Logística: Python

```
x2t = np.linspace(min(data["x2"]),max(data["x2"]),100)
x1t = ( - model.intercept_[0] - model.coef_[0][1]*x2t ) / model.coef_[0][0]
plt.plot(x0["x1"],x0["x2"], 'bs', x1["x1"],x1["x2"], 'g^', x1t, x2t, "-")
plt.show()
```

Regressão Logística: Python



Regressão Logística: com dados discrepantes



Regressão Logística: com dados discrepantes