Visual Pattern Recognition: desining pattern classifiers

Image Processing — scc0251

www.icmc.usp.br/∼moacir — moacir@icmc.usp.br

ICMC/USP — São Carlos, SP, Brazil

2011

Agenda

1. Um problema de classificação
2. Classificação: terminologia
3. Como construir um classificador
4. Classificador de Regressão Logística
   - Função de custo
5. Classificador dos k-vizinhos mais próximos

Agenda

Um problema de classificação

Problema — dadas duas classes de imagens:

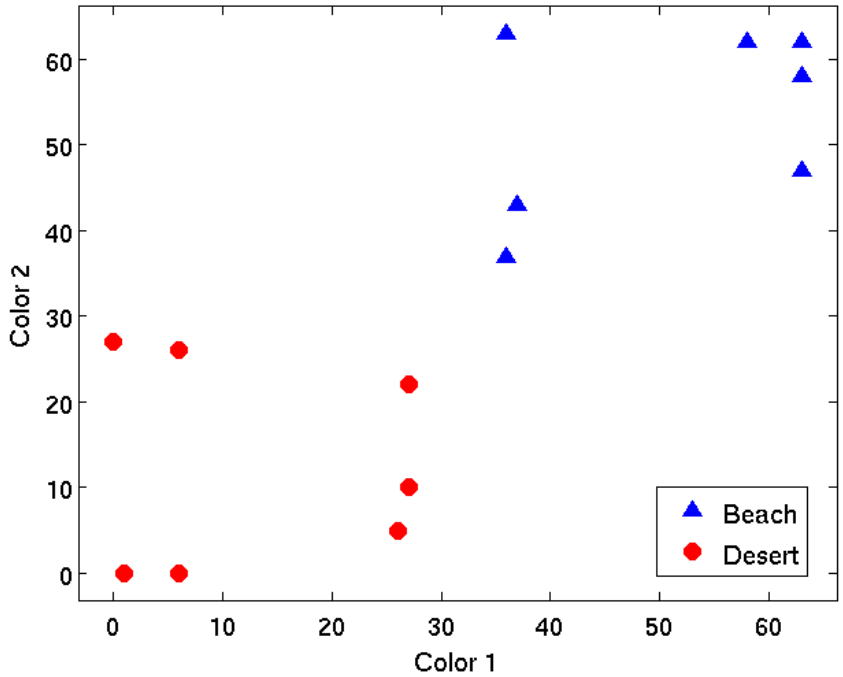
- classe 1: fotos tiradas de **desertos**, e
- classe 2: fotos tiradas de **praias**,

e um conjunto de 7 imagens de cada classe, desenvolva um programa que seja capaz de classificar uma nova, e desconhecida, imagem, em uma dessas duas classes.

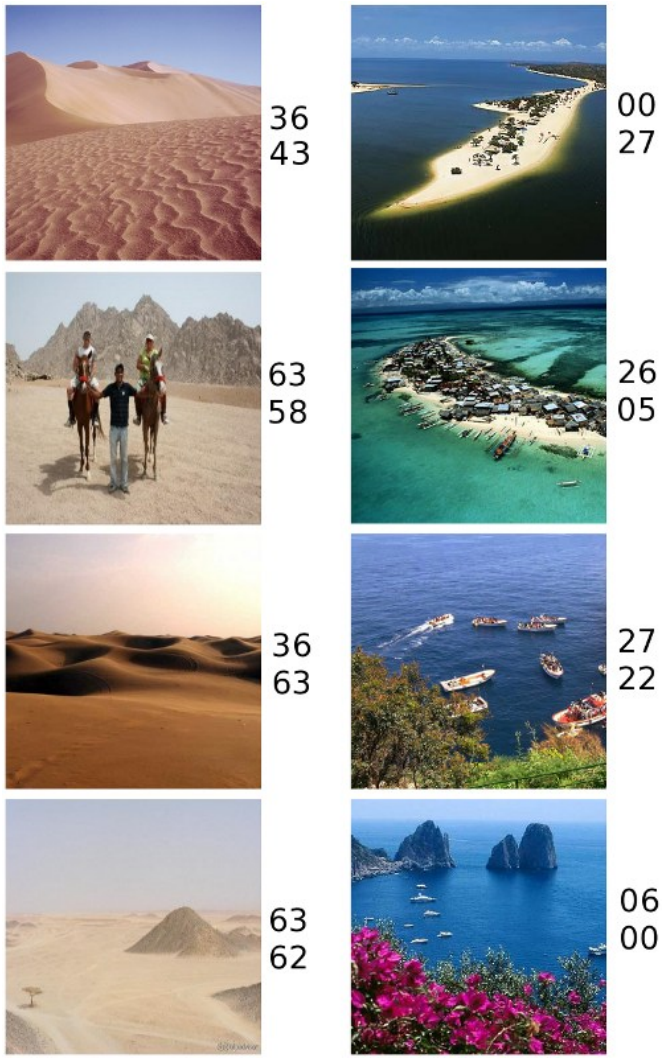
- **Objeto do problema**: cada imagem.

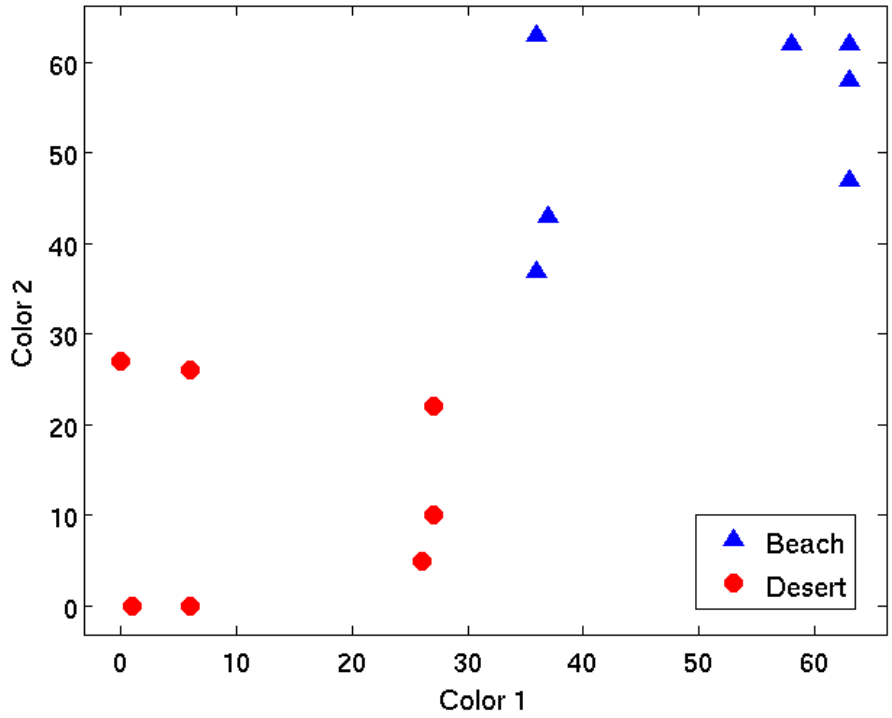
Um problema de classificação

- **Atributo** (característica): valores extraídos das imagens que possam ser usados para medir uma (dis)similaridade entre duas imagens.
  Alguma sugestão?
  - Reduza o número de cores da imagem para 64, obtenha o histograma e use dois atributos: as duas cores mais frequentes.
  - Cada imagem é representada por 2 valores: espaço de atributos 2D.



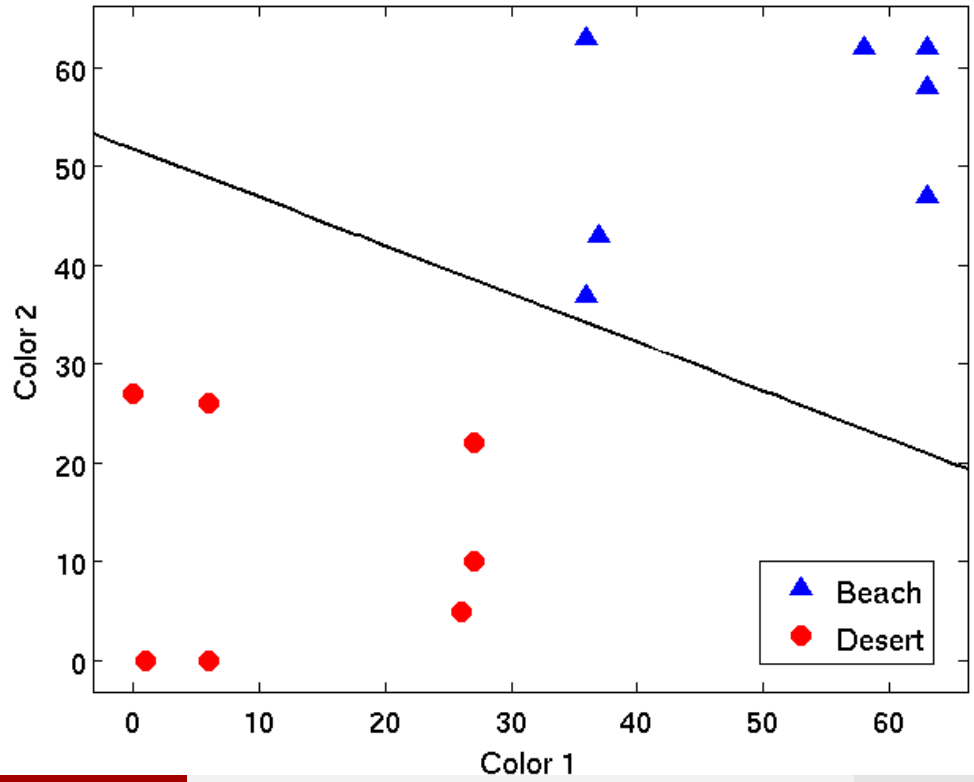
Um problema de classificação





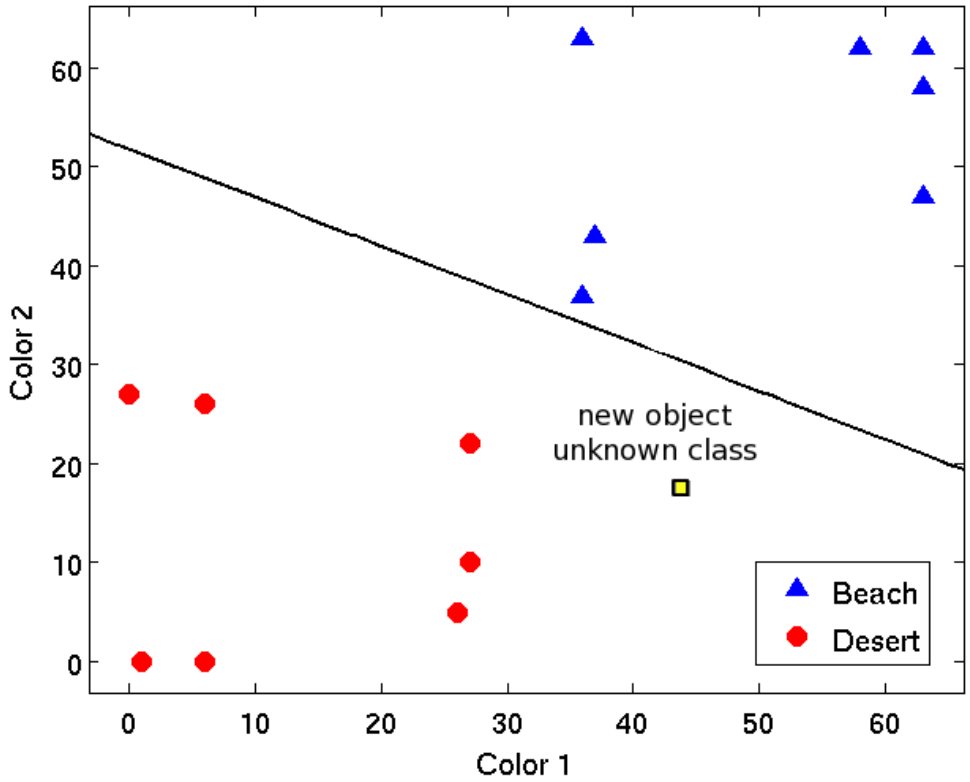
Um problema de classificação

- **Classificador**: um modelo construído usando um conjunto de atributos extraídos de imagens cuja classe é conhecida. Esse modelo deve ser capaz de predizer a classe de uma imagem de classe desconhecida. **Alguma sugestão?**
  - Particionar o espaço de atributos usando a distribuição dos objetos.



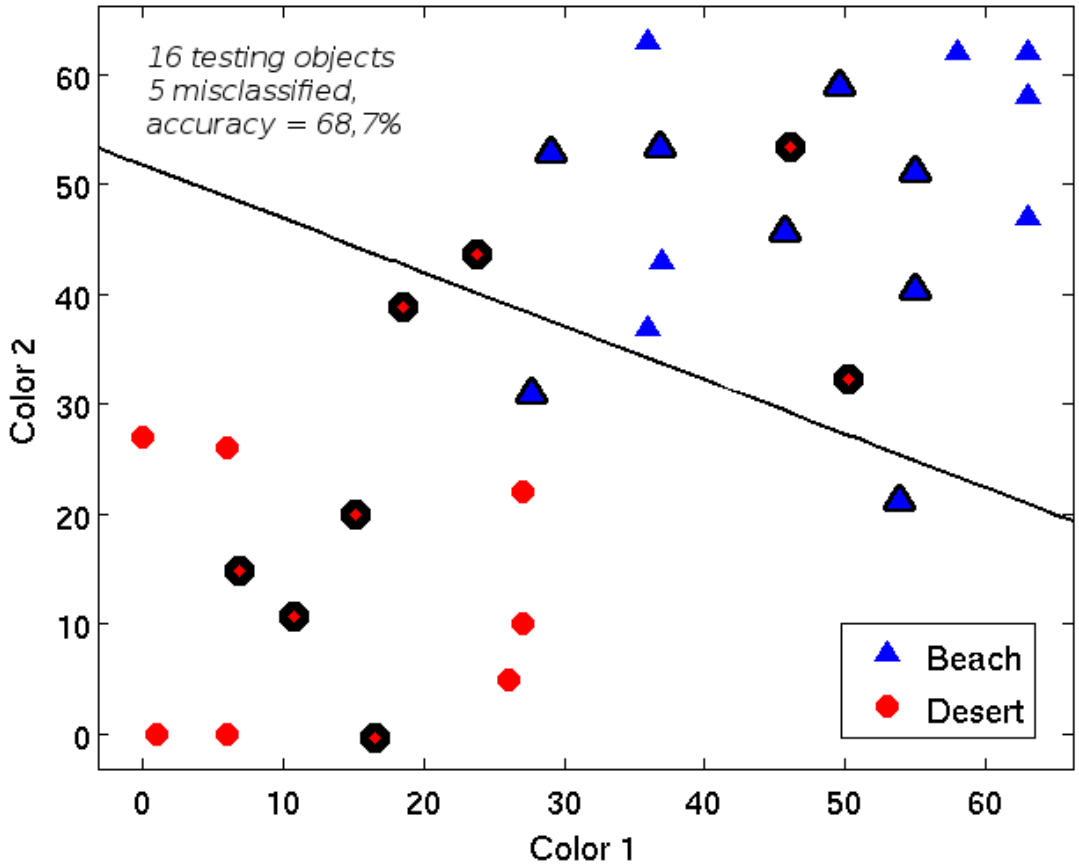
Um problema de classificação

- O conjunto dos objetos usados para construir o classificador é chamado de **conjunto de treinamento**.
- O **classificador treinado** é usado para predizer a classe de um **novo objeto**. Essa capacidade é chamada de generalização.



Um problema de classificação

- Quando o classificador está pronto, é possível testar sua precisão usando objetos adicionais (não usados para treinamento).
- Esse conjunto de objetos de classe conhecida é chamado **conjunto de testes**.



Agenda

Terminologia/notação

Classes: grupos de objetos similares, $\Omega = \{\omega_1, \omega_2, ..., \omega_c\}$

Conjunto de dados (dataset): $\mathbf{X} = \{x_1, x_2, ..., x_N\}$, for $x_i \in \mathbb{R}^M$

$\mathbf{x}_i \in \mathbb{R}^M$ an **object** in the feature space: the *feature vector*

$I(\mathbf{x}_i) = y_i \in \Omega$ the **labels** assigned to the object

the matrix N objects $\times$ M features:

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,M} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,M} \\ \cdots & \cdots & & \cdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,M} \end{bmatrix}, \text{ labels} = \begin{bmatrix} I(\mathbf{x}_1) = y_1 \\ I(\mathbf{x}_2) = y_2 \\ \cdots \\ I(\mathbf{x}_N) = y_N \end{bmatrix}$$

Agenda

Como construir um classificador

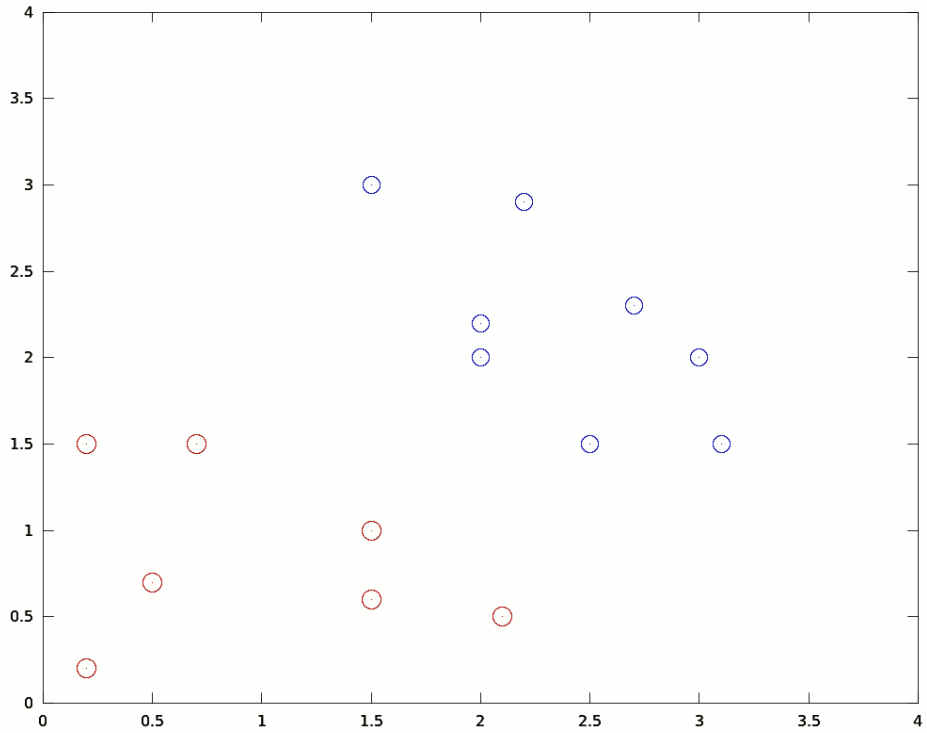
Classificador: função: $H : \mathbb{R}^M \rightarrow \Omega$, for $\Omega = \{\omega_1, \omega_2, ..., \omega_c\}$

Hipótese parametrizada: $h_\theta(\mathbf{x}) = \theta^T \mathbf{x}$.

- usa parâmetros $\theta_0, \theta_1, ...\theta_M$, onde M é o número de atributos.
- para 2 atributos: $h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$.
- a função que separa as classes no espaço de atributos é obtida encontrando os valores de θ, de forma que um dado $\mathbf{x}$ possa ser classificado (considerando duas classes):

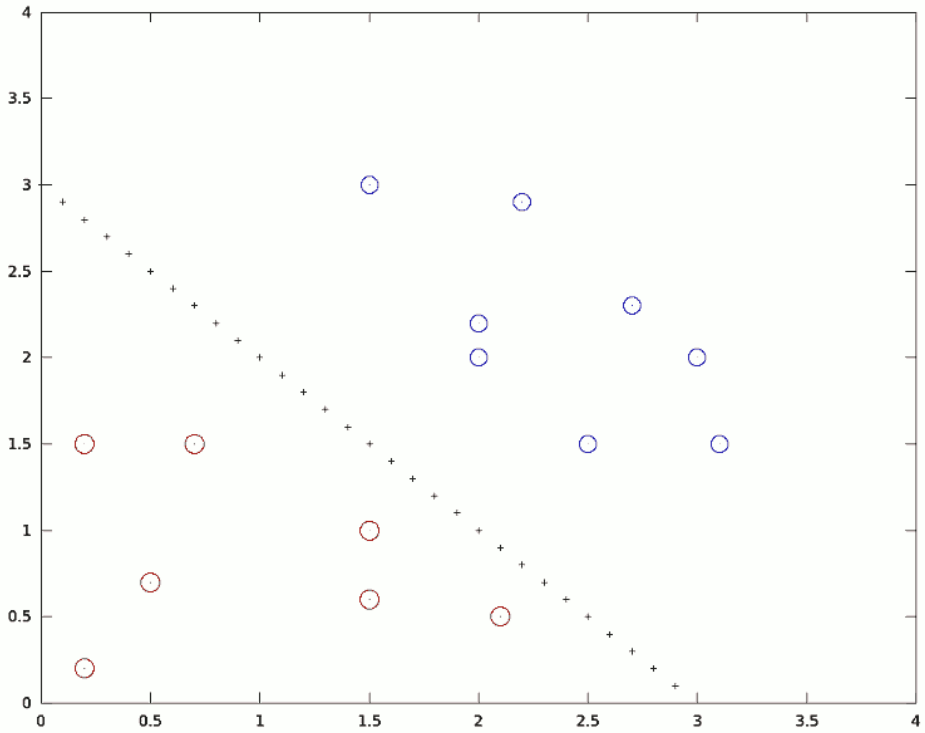
$$y_i = I(x_i) = \begin{cases} 1 & \text{s } h_\theta(\mathbf{x}) \geq 0 \\ 0 & \text{se } h_\theta(\mathbf{x}) < 0 \end{cases}$$

Como construir um classificador



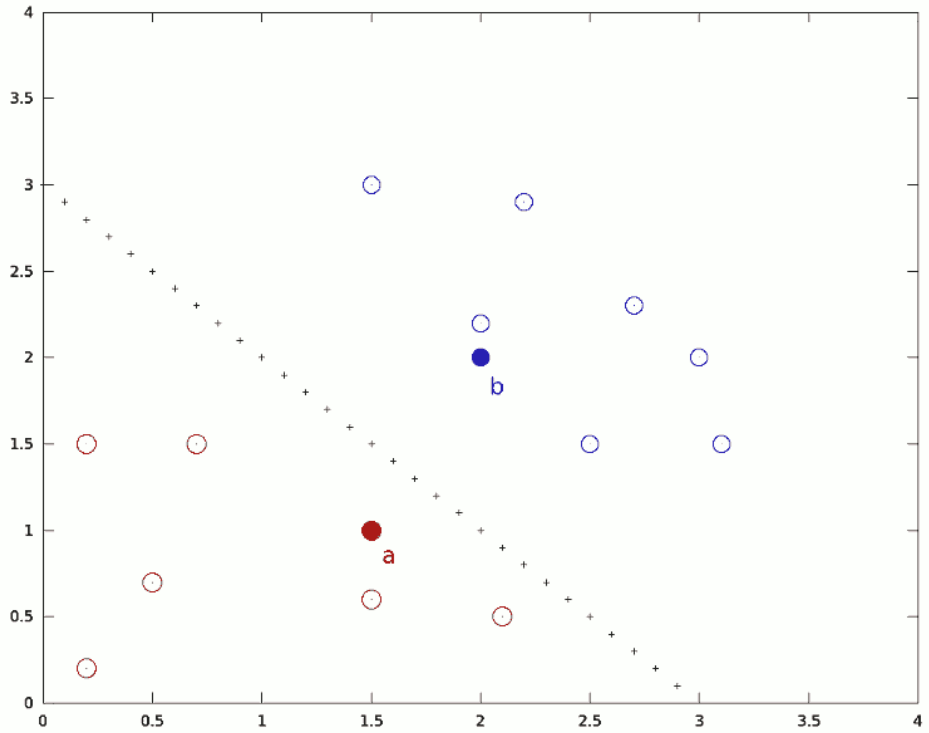
Quais parâmetros devemos usar para separar as duas classes?

Como construir um classificador



$$\theta = [-3, 1, 1], \quad h_\theta(\mathbf{x}) = -3 + 1x_1 + 1x_2.$$

Como construir um classificador

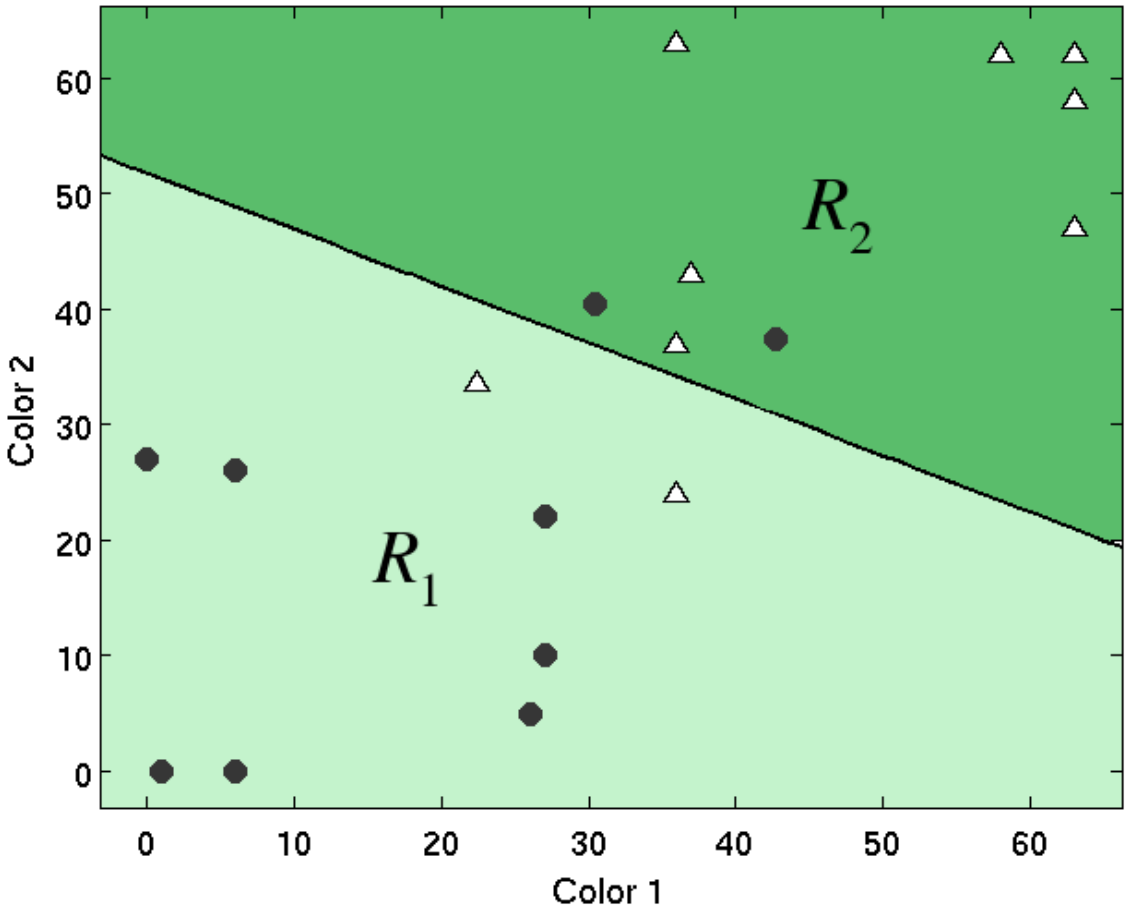


$a = (1.5, 1)$
$h_\theta(a) = -3 + 1.5 + 1 = -0.5$

$b = (2, 2.2)$
$h_\theta(b) = -3 + 2 + 2.2 = 1.2$

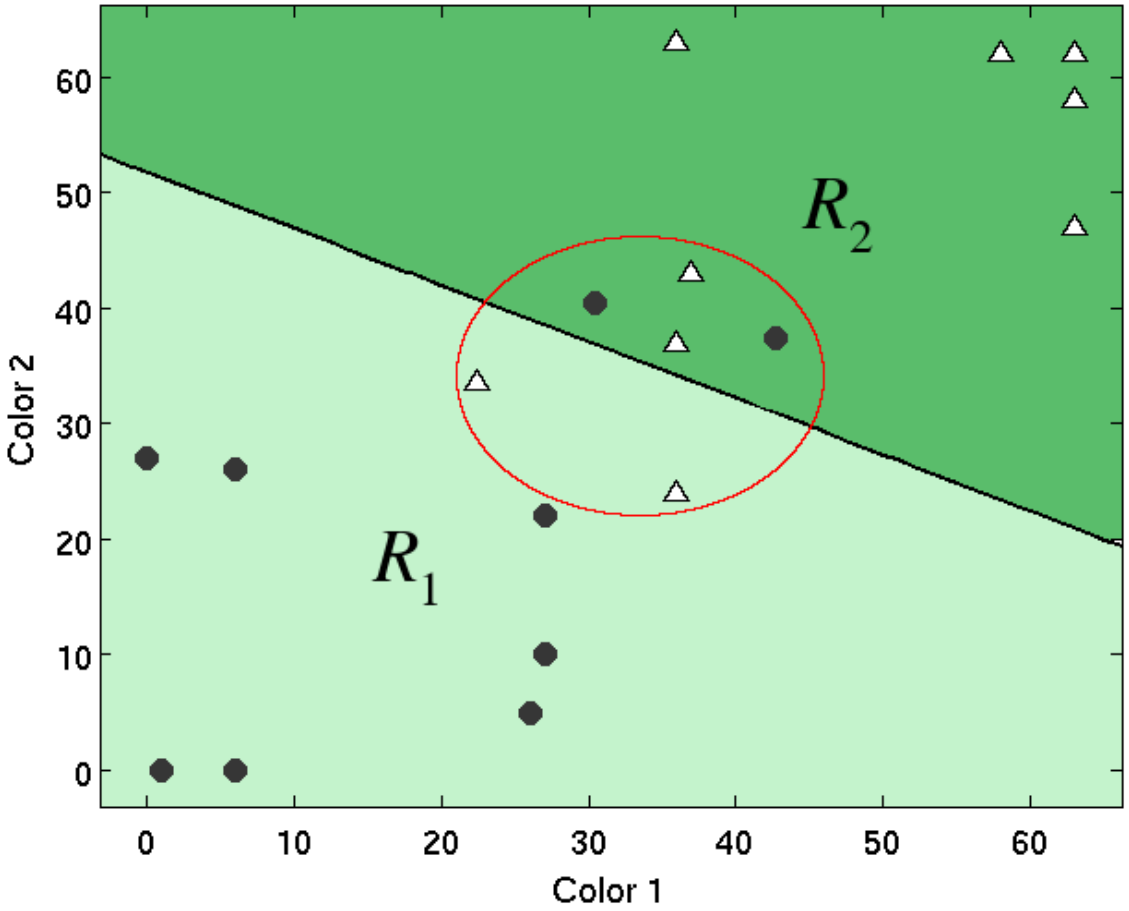
Como construir um classificador

A função particiona o espaço de atributos em regiões de decisão: $R_1, R_2, ..., R_c$, formando a **superfície de decisão**.



Como construir um classificador

Em alguns casos, o classificador apresenta erros no conjunto de treinamento (classes não separáveis linearmente).



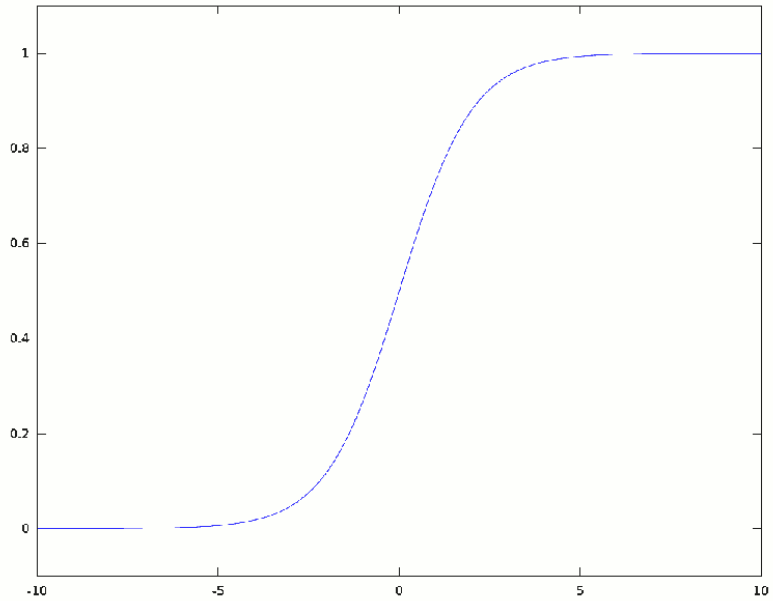
Agenda

The Classificador de Regressão Logística

Para computar valores na faixa $[0, 1]$ e ter melhor controle dos limites, uma função logística (ou sigmoidal) pode ser usada para avaliar a hipótese

$$h_\theta(\mathbf{x}) = g(\theta^T \mathbf{x})$$

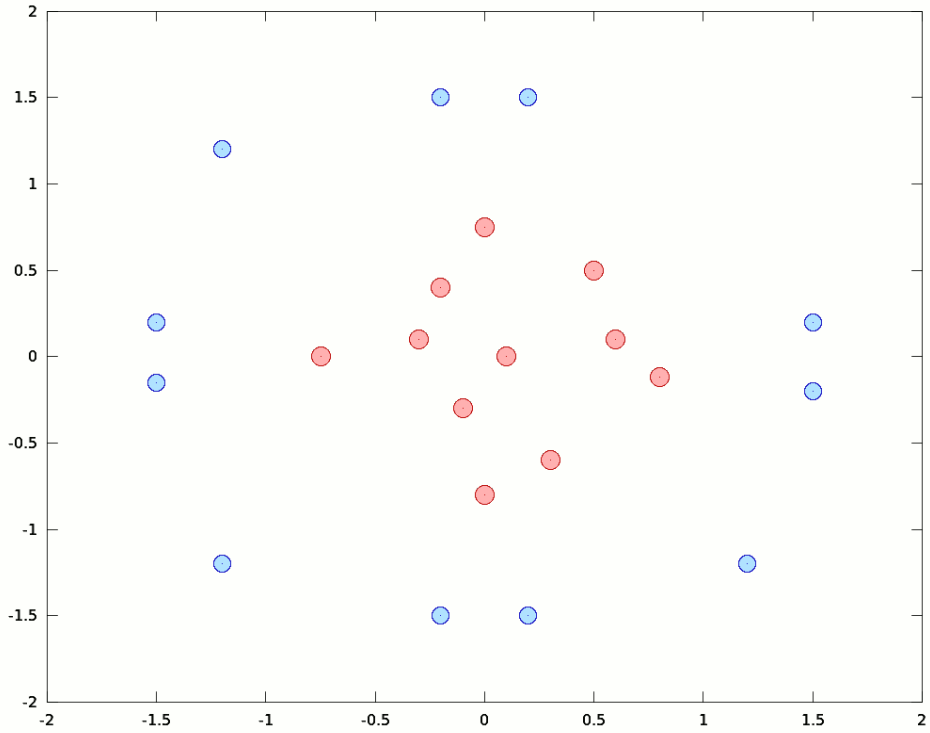
$$g(z) = \frac{1}{1 + e^{-z}}$$



$$y_i = I(x_i) = \begin{cases} 1 & \text{se } h_\theta(\mathbf{x}) \geq 0.5 \\ 0 & \text{se } h_\theta(\mathbf{x}) < 0.5 \end{cases}$$

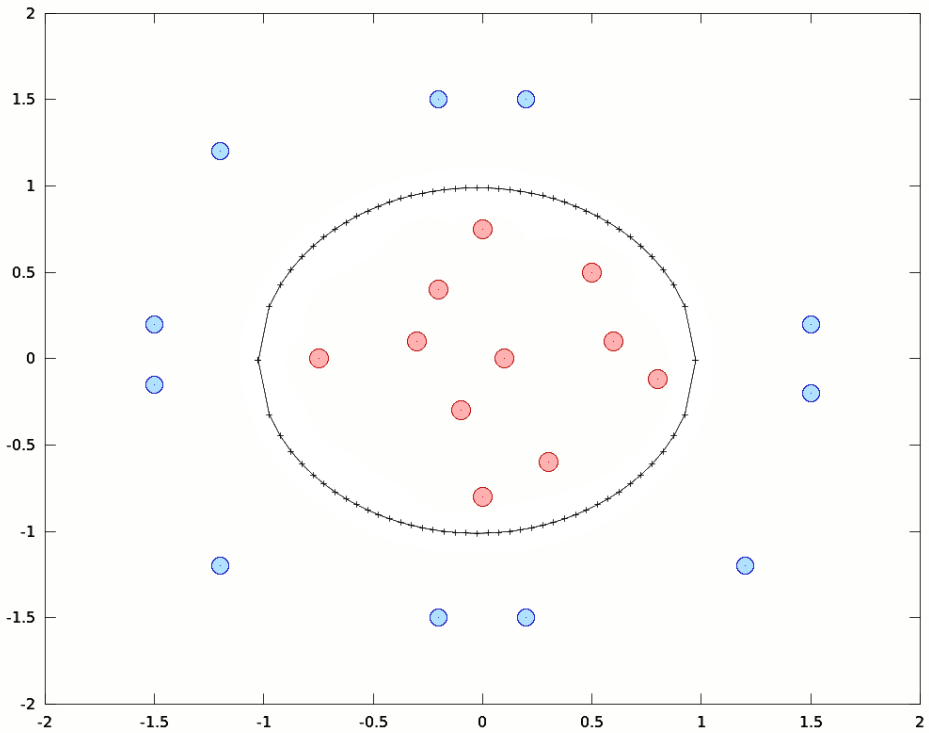
Dessa forma é mais fácil interpretar a **confiança** acerca da classificação: valores ao redor de 0.5 significam que o objeto está sobre o limiar da partição

Como construir um classificador: não linear



Como separar essas classes? Adicionando atributos polinomiais Exemplo:
$h_\theta(\mathbf{x}) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$.

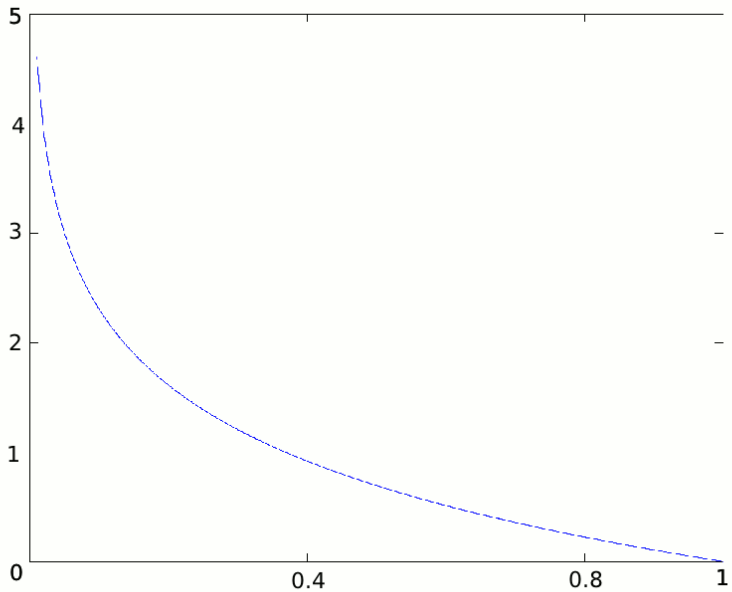
Como construir um classificador: non-linear



Usando: $\theta = [-1, 0, 0, 1, 1]$, $\quad h_\theta(\mathbf{x}) = g(-1 + x_1^2 + x_2^2)$.

Função de custo da Regressão Logística

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

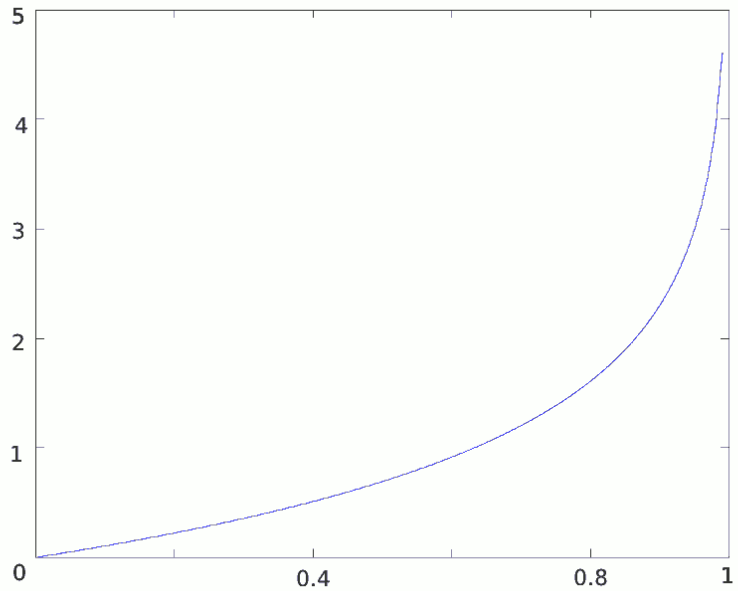


$Cost = 0$ se $y = 1$ e $h_\theta(\mathbf{x}) = 1$

Idéia: penalizar o algoritmo de aprendizado quando ele erra.
Ex: $Cost = 2.3$ se $y = 1$ e $h_\theta(\mathbf{x}) = 0.1$

Função de custo da Regressão Logística

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



$Cost = 0$ se $y = 0$ e $h_\theta(\mathbf{x}) = 0$

Ex: $Cost = 4.6$ se $y = 0$ e $h_\theta(\mathbf{x}) = 0.99$

Função de custo da Regressão Logística

Se somarmos o custo de cada amostra no conjunto de treinamento, temos:

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} Cost(h_\theta(\mathbf{x_i}, y_i)$$

- Reescrevendo a função numa só equação:

$$Cost(h_\theta(\mathbf{x}), y) = -y \log(h_\theta(\mathbf{x})) - (1 - y) \log(1 - h_\theta(\mathbf{x}))$$

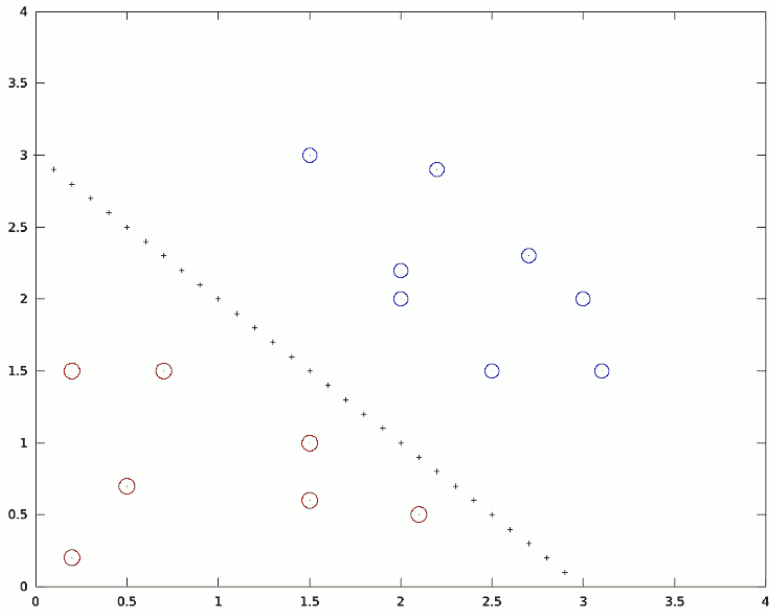
- A função de custo total fica:

$$J(\theta) = \frac{1}{N} \left[\sum_{i=1}^{N} y_i \log(h_\theta(x_i)) + (1 - y_i) \log(1 - h_\theta(x_i)) \right]$$

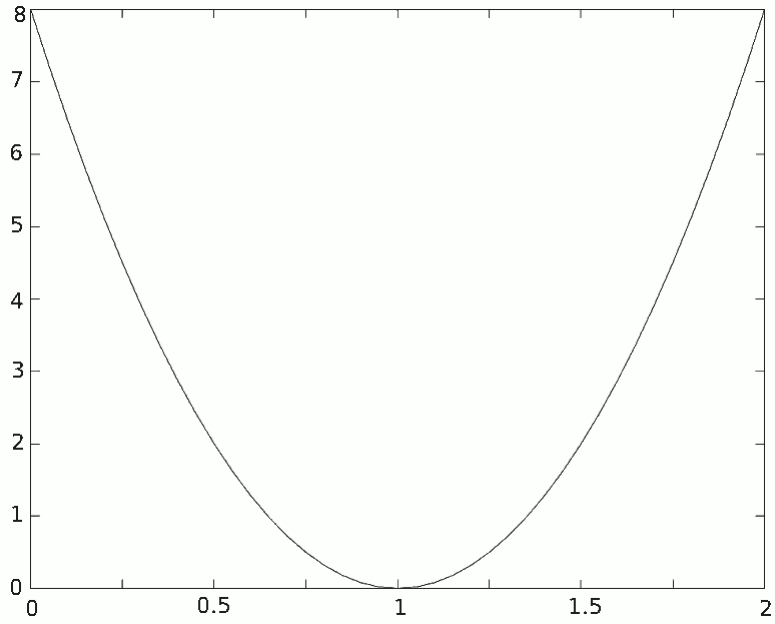
Função de custo da Regressão Logística

Agora o problema é o de minimizar $J(\theta)$ em função de θ:

$$\min_{\theta} [J(\theta)]$$



$\theta = [-3, 1, 1]$
$h_\theta(\mathbf{x}) = -3 + x_1 + x_2.$



$J(\theta)$ em função de θ_1.

Função de custo da Regressão Logística

$$\min_{\theta} [J(\theta)]$$

- Existem muitos algoritmos para otimização, como o do Gradiente Conjugado.
- Um algoritmo simples (mas não tão rápido) é o Gradiente Descendente, que busca a direção descendente da derivada e atualiza os parâmetros de forma a reduzir o custo a cada passo:
  - inicia com algum valor de θ
  - modifica θ de forma a reduzir $J(\theta)$
  - esperando alcançar o mínimo global da função
- Um parâmetro α controla o tamanho de cada passo. Valores grandes podem levar o algoritmo a divergir, valores pequenos demais tornam o algoritmo lento.

Função de custo da Regressão Logística

- O algoritmo repete até a convergência:

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta), \ \forall j$$

- Ou seja, a cada passo ele atualiza de forma simultânea todos os parâmetros:

```
repeat until convergence (or a fixed number of iterations) {
  for each theta (i) {
      temp(i) = theta(i) - alpha (d / d theta(i)) J(theta)
  }
  for each theta (i) {
      theta(i) = temp(i)
  }
}
```

- O gradiente para a regressão logística foi encontrado:

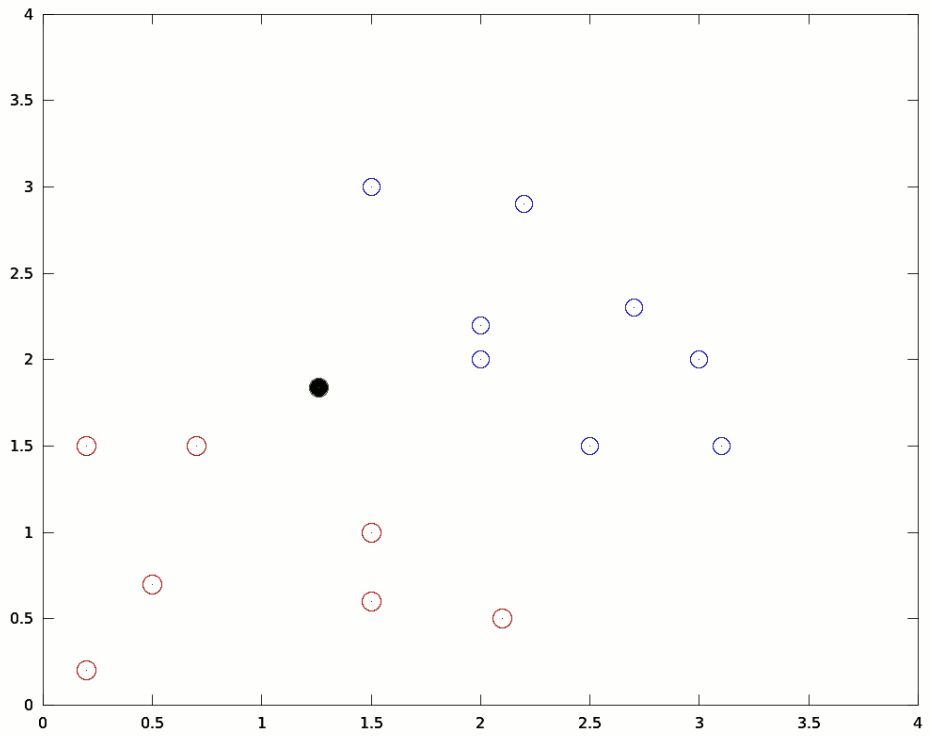
$$\theta_j = \theta_j - \alpha \sum_{i=1}^{N} (h_\theta(x_i) - y_i) x_{i,j}$$

Agenda

Classificador dos k-vizinhos mais próximos

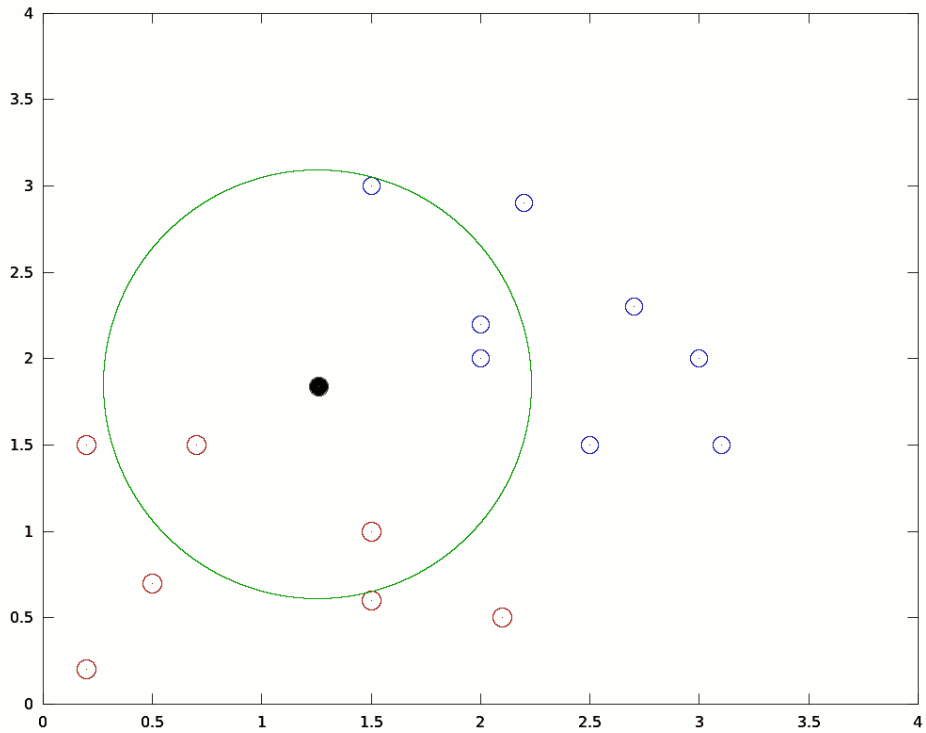
É chamado "*lazy learning algorithm*", por ser baseado apenas nos objetos.

- Regra de decisão simples: um novo objeto é atribuído à classe mais presente considerando os k vizinhos mais próximos no conjunto de treinamento



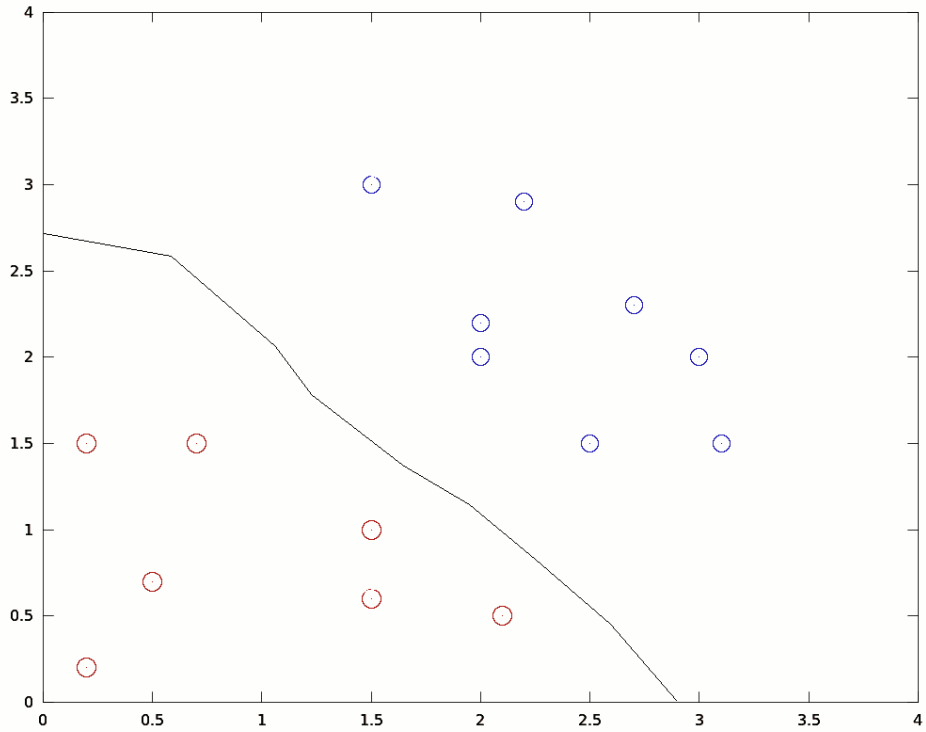
Classificador dos k-vizinhos mais próximos

Para $k > 1$ é um classificador baseado na densidade local. Por exemplo, $k = 5$.



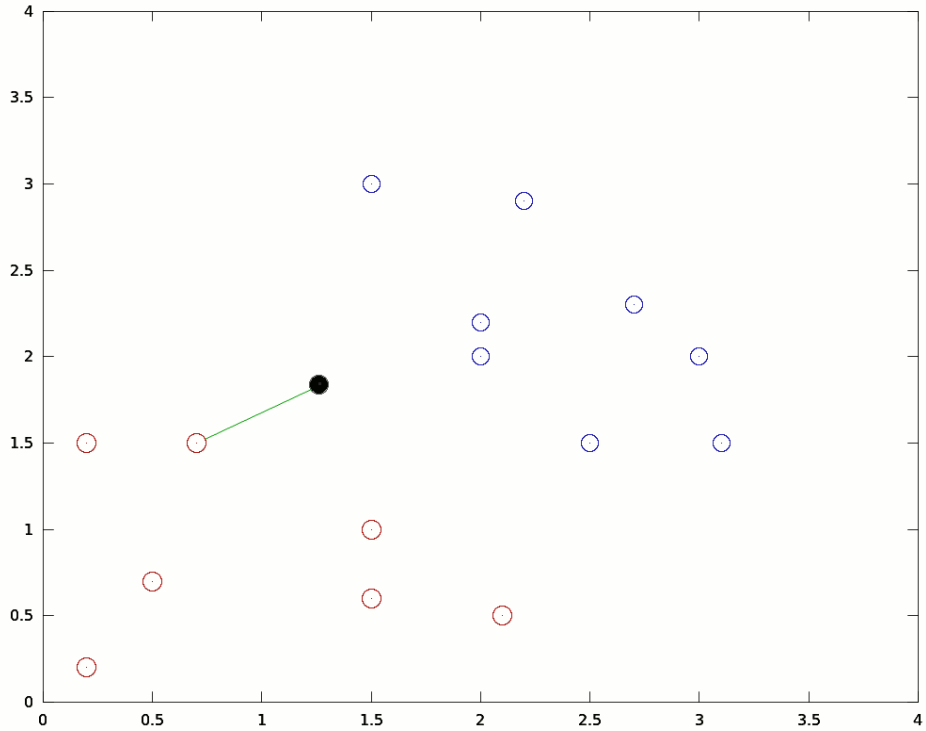
Classificador dos k-vizinhos mais próximos

A superfície de decisão pode ser não linear e as vezes muito complexa.



Classificador dos k-vizinhos mais próximos

Para $k = 1$ é baseado em distâncias.



Classificador dos k-vizinhos mais próximos

O classificador KNN (k-Nearest Neighbors) é simples mas pode ter dificuldade em espaços complexos

- Um valor alto de k pode causar *overfitting*.
- Seu comportamento pode ser difícil de prever.

Classificador dos k-vizinhos mais próximos

Algoritmo:

1. Para cada novo objeto a ser classificado, calcule a distância entre ele e todos os objetos no conjunto de treinamento.
2. Ordene as distâncias.
3. Recupere a classe dos k objetos menos distantes.
4. Atribua ao novo objeto a classe mais presente entre as k recuperadas.

References

- Duda, R.; Hart, P.; Stork, D. Pattern Classification, 2.ed, 2000.
- Bishop, C. Pattern Recognition and Machine Learning, 2.ed, 2006.