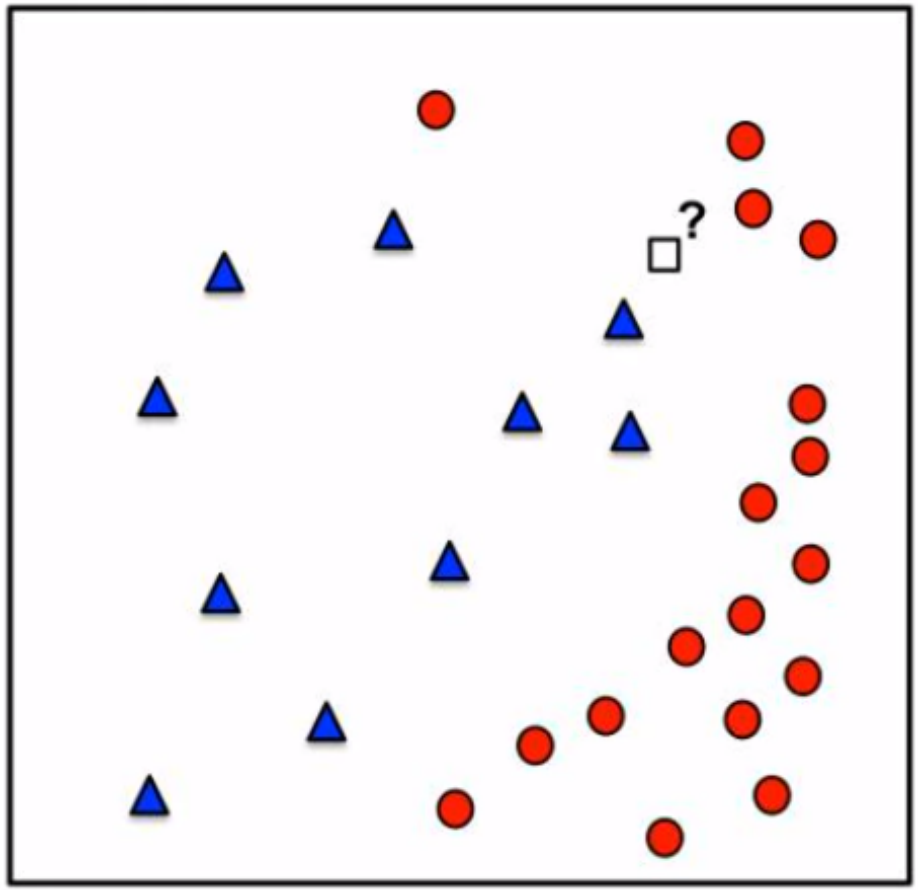
Reconhecimento de Padrões em Imagens
Métodos não-paramétricos

Dainf - UTFPR

Leyza Dorini - Rodrigo Minetto

K *Nearest-Neighbours* (kNN)

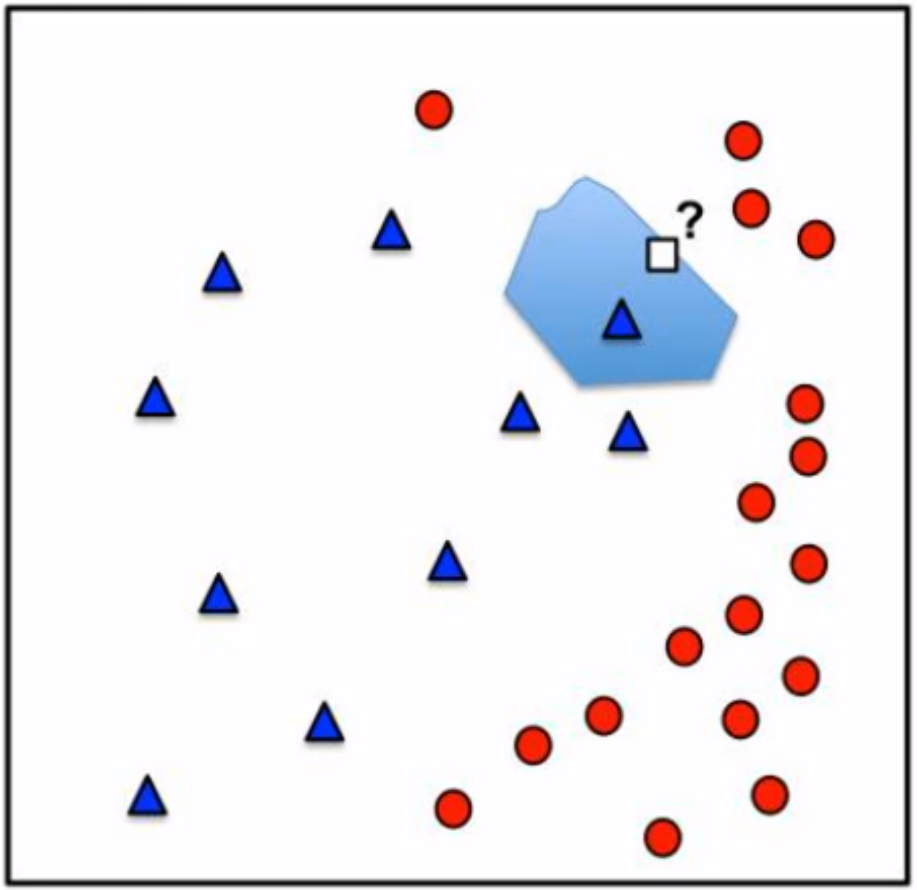
Como você classificaria a nova amostra, representada por um quadrado?



O uso das amostras mais próximas é a base para o algoritmo de aprendizado kNN

K *Nearest-Neighbours* (kNN)

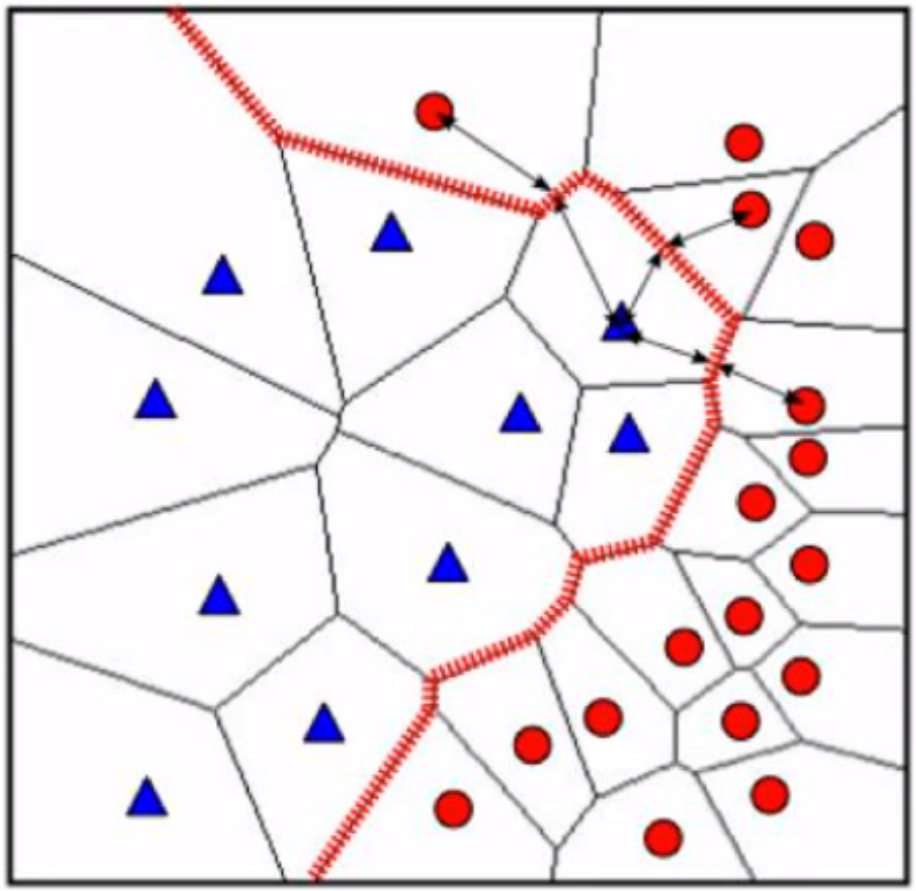
Portanto, para classificar um ponto, encontramos a amostra de treinamento mais próxima e fazemos a predição com base no seu rótulo.



Esse processo cria uma célula de Voronoi.

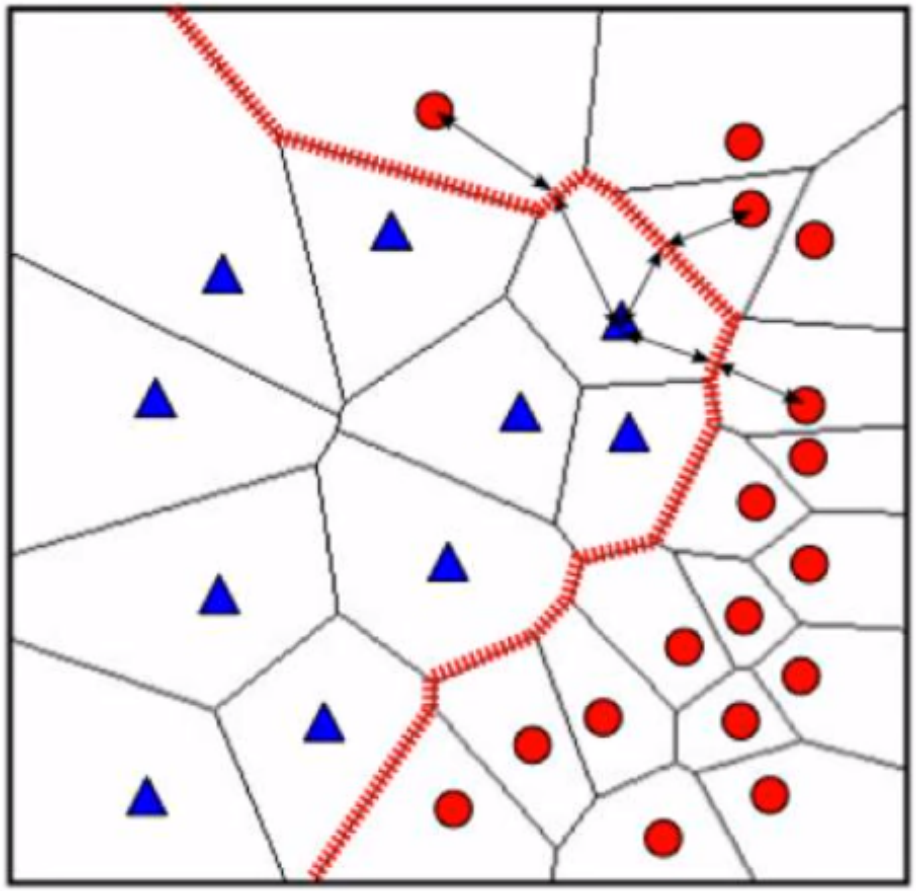
K *Nearest-Neighbours* (kNN)

Ao fazer isso, temos um Diagrama de Voronoi: fronteiras estão a uma mesma distância de diferentes amostras de treinamento.



K *Nearest-Neighbours* (kNN)

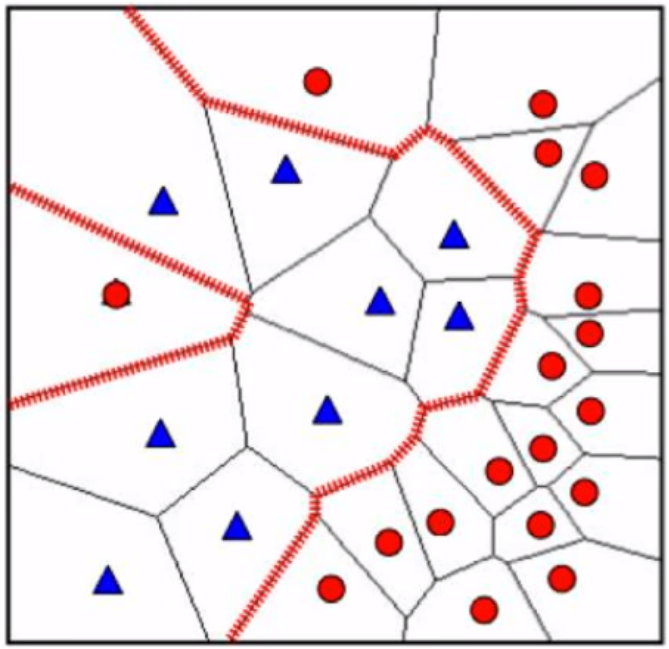
A fronteira de decisão é determinada com base nesse diagrama.



Qual a diferença para as fronteiras de decisão das árvores de decisão e de Naive-Bayes?

K *Nearest-Neighbours* (kNN)

Toda essa simplicidade cobra o seu preço: sensibilidade a *outliers.* Uma única amostra afeta consideravelmente, dado que a abordagem não considera o prior da classe!



Como minimizar este problema? Usar mais de um vizinho nessa análise!

kNN: Algoritmo de Classificação

Dadas as amostras de treinamento, $\{x_i, y_i\}$, e a nova amostra que se deseja classificar, x:

- calcule a distância, $D(x, x_i)$, entre x e cada amostra de treinamento x_i;
- selecione as k amostras que possuem a menor distância, $x_{i1}, \ldots, x_{ik}$, e seus respectivos rótulos, $y_{i1}, \ldots, y_{ik}$;
- atribua a classe y^*, que corresponde àquela com maior frequência em $y_{i1}, \ldots, y_{ik}$.

Exemplo de aplicação

- 16x16 bitmaps
- 8-bit grayscale
- Euclidian distance
  - over raw pixels

$$D(A,B)=\sqrt{\sum_r \sum_c \left(A_{r,c}-B_{r,c}\right)^2}$$

- Accuracy:
  - 7-NN ~ 95.2%
  - SVM ~ 95.8%
  - humans ~ 97.5%

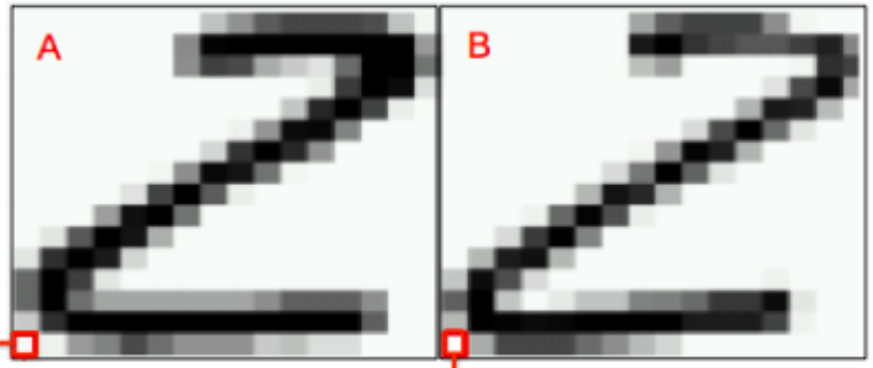


Figure © Sam Roweis, 2006

Retirado de [1].

Seleção da distância

Fundamental para o algoritmo.

Minkowski (p-norm):

$$d(a, b) = \sqrt[p]{\sum_k |a_k - b_k|^p}$$

$p = 2$ é Euclidiana, $p = 1$ Manhattan, etc..

Seleção da distância

A distância mais comumente utilizada é a Euclidiana:

$$d(a,b) = \sqrt{\sum_k (a_k - b_k)^2}$$

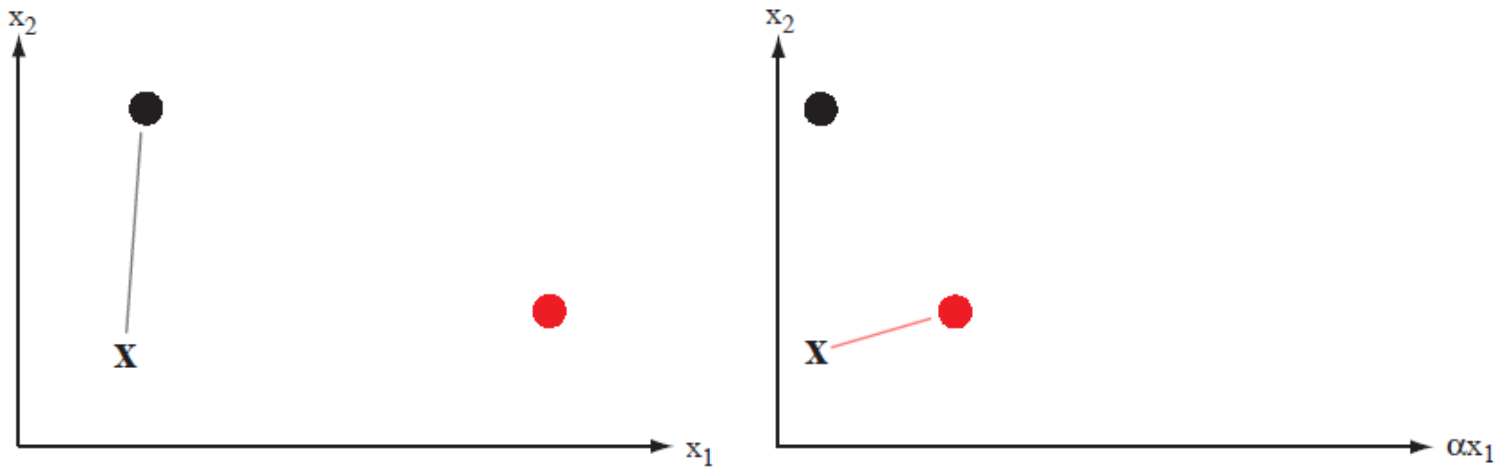
Entretanto, é importante considerar que ela trata todas as características como igualmente importantes. Se isso não for adequado ao problema, pode-se considerar pesos, por exemplo:

$$d(a,b) = \sqrt{\sum_k w_i (a_k - b_k)^2}$$

sendo que, quando maior o valor de w_i, mais importante é a característica.

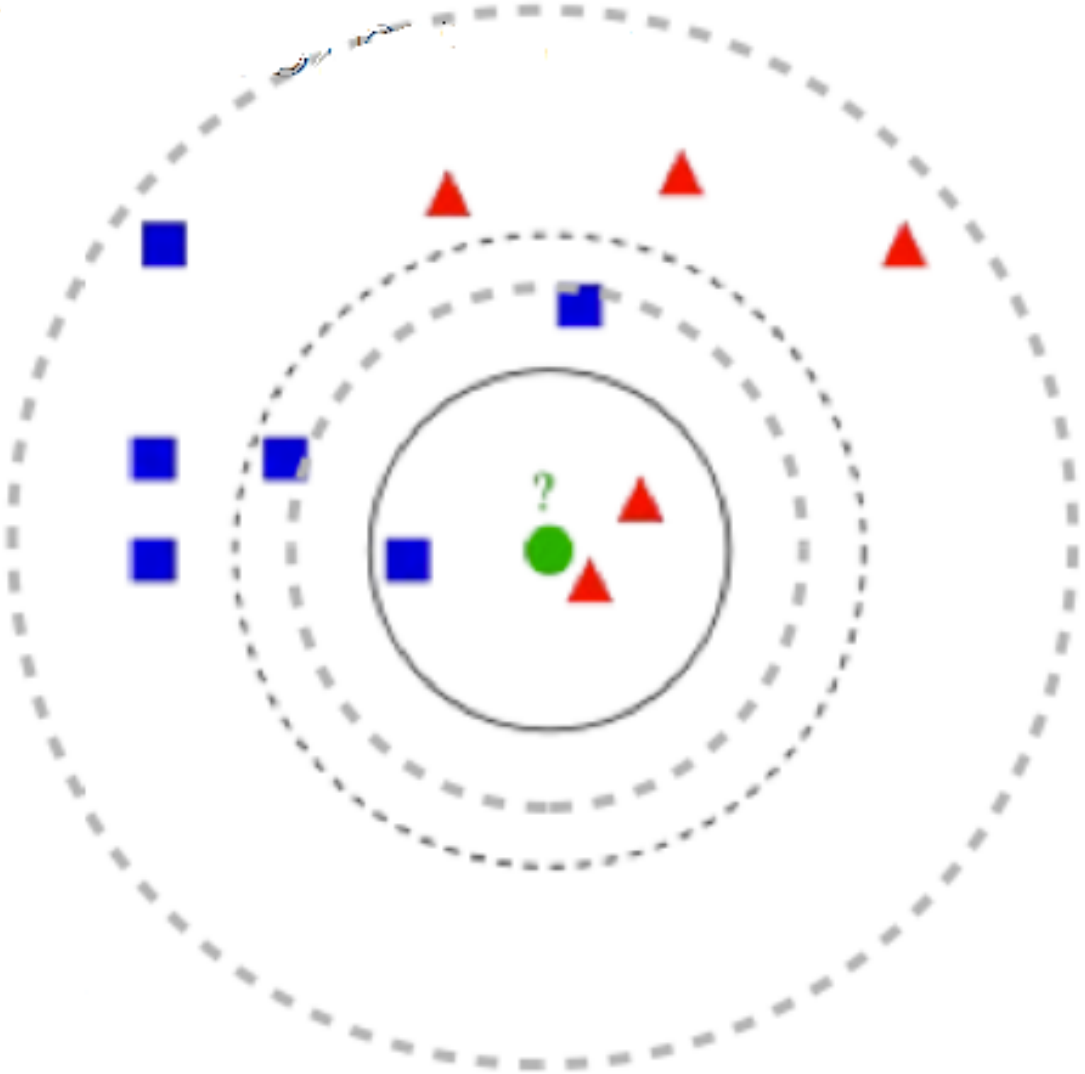
Seleção da distância

Além disso, é preciso considerar algumas limitações da distância Euclidiana (o que pode demandar pré-processamento ou a escolha de outra distância). Por exemplo, ao se alterar a escala de uma característica, a medida pode ser influenciada.



OBS. Diversas outras funções de distância podem ser exploradas no `scikit-learn`.

Como escolher o valor de k?



Como escolher o valor de k?

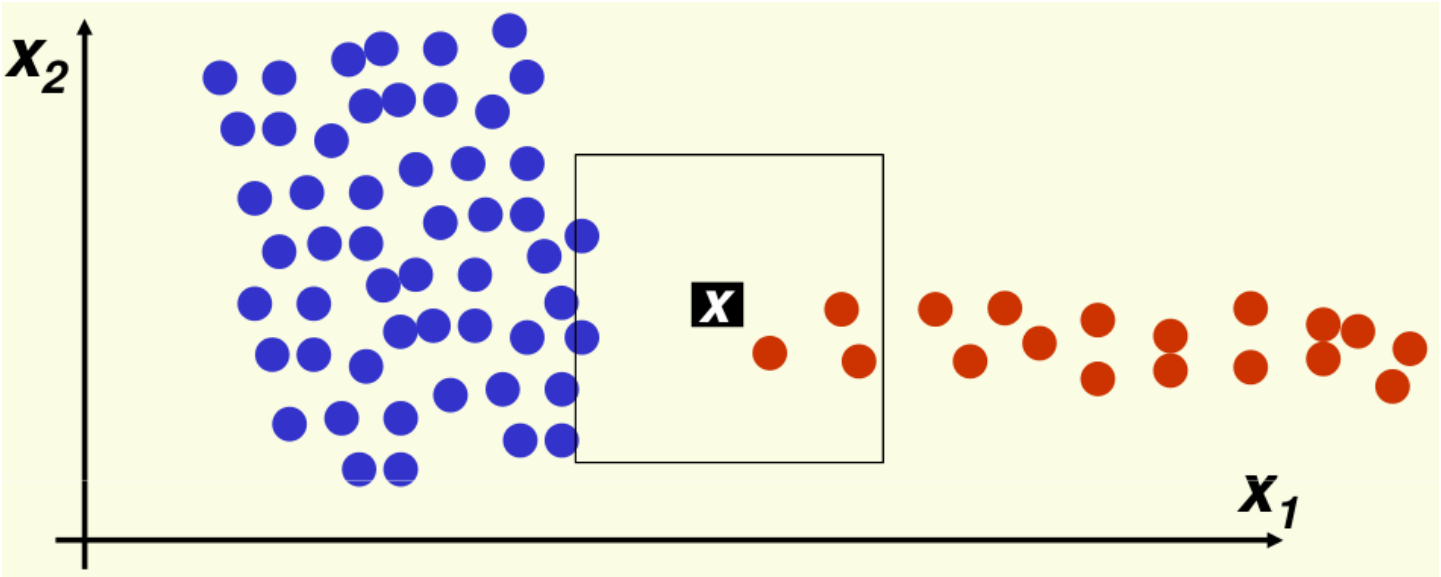
Na teoria, se um número infinito de amostras estiver disponível, quanto maior o k, melhor a classificação (ou seja, mais próximo ao erro do classificador ótimo de Bayes).

Na prática:

- k deve ser suficientemente grande para minimizar a taxa de erro - k pequeno introduz ruído nas fronteiras de decisão.
- k deve ser suficientemente pequeno para que somente amostras próximas sejam consideradas - k grande conduz a fronteiras de decisão muito suavizadas.

Encontrar o melhor *trade-off* não é trivial. Dica: validação cruzada (usando diferentes valores de k, escolha o que tem melhor generalização).

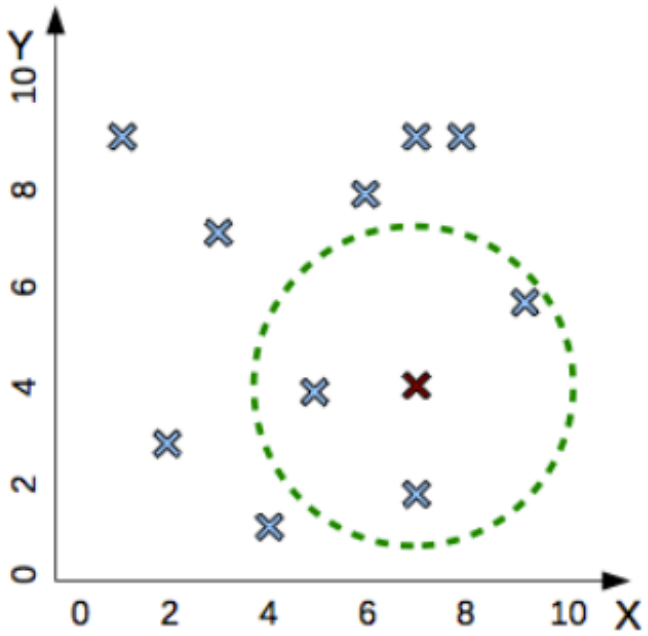
Como escolher o valor de k?



Neste exemplo, a classificação é correta apenas para $k = 1, \ldots, 5$.

Complexidade computacional

What you see



Find nearest neighbors
of the testing point (red)

What algorithm sees

- Training set:

  {(1,9), (2,3), (4,1),
  (3,7), (5,4), (6,8),
  (7,2), (8,8), (7,9), (9,6)}

- Testing instance:

  (7,4)

- Nearest neighbors?

  compare one-by-one
  to each training instance

- n comparisons
- each takes d operations

Complexidade computacional

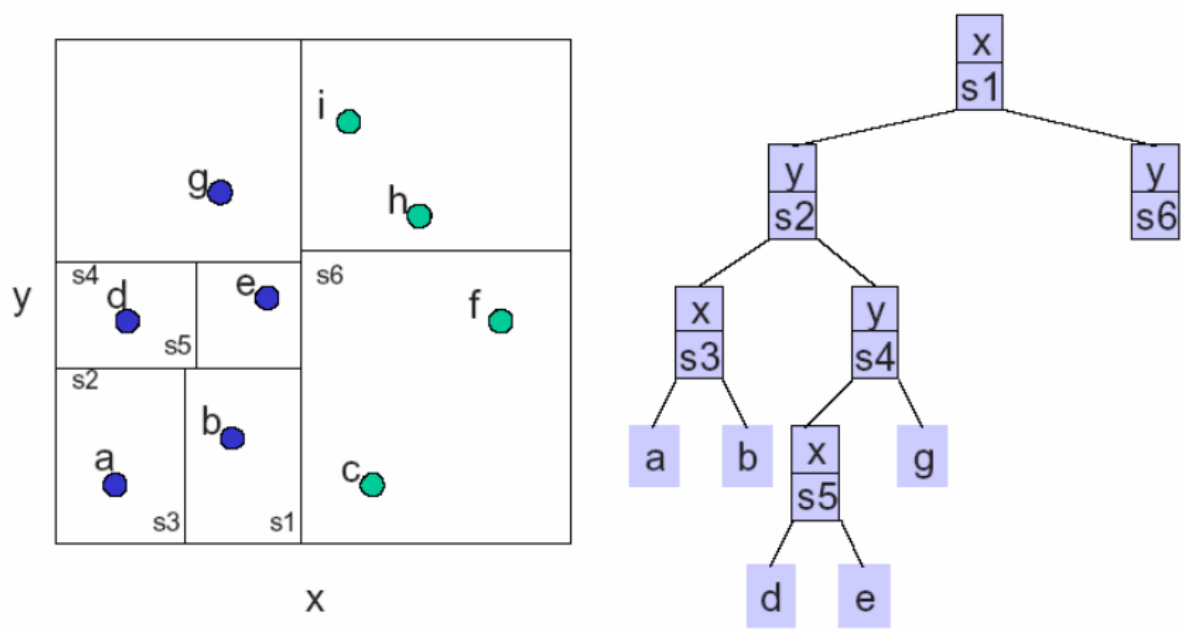
Em algoritmos básicos, todas as amostras são armazenadas.
Suponha que tenhamos n exemplos com dimensão d. Temos:

- $\mathcal{O}(d)$ para calcular a distância de uma amostra.
- $\mathcal{O}(nd)$ para calcular a distância ao vizinho mais próximo.
- $\mathcal{O}(knd)$ para calcular as k amostras mais próximas.

Isso é muito caro para um grande número de amostras (que vimos que são necessárias para um bom desempenho do algoritmo).

Redução da complexidade

A ideia básica consiste em particionar recursivamente o espaço e procurar por vizinhos mais próximos apenas perto do ponto de teste. Uma possível implementação se dá pelas kd-trees, uma generalização de árvores binárias de busca que permite armazenar pontos em um espaço k-dimensional.

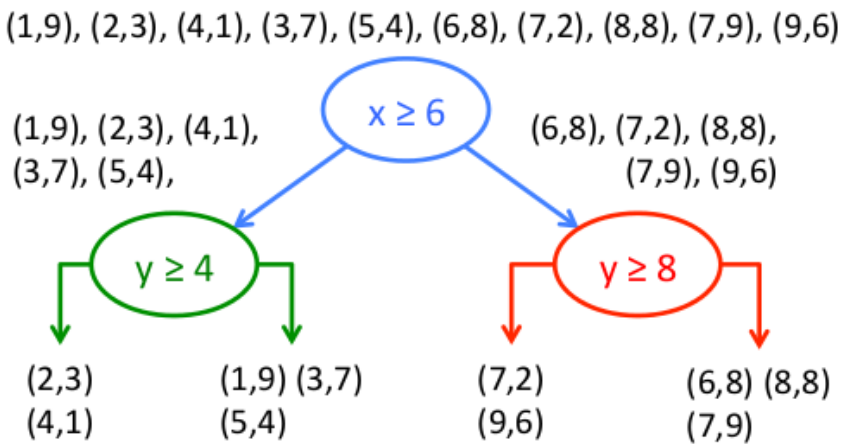


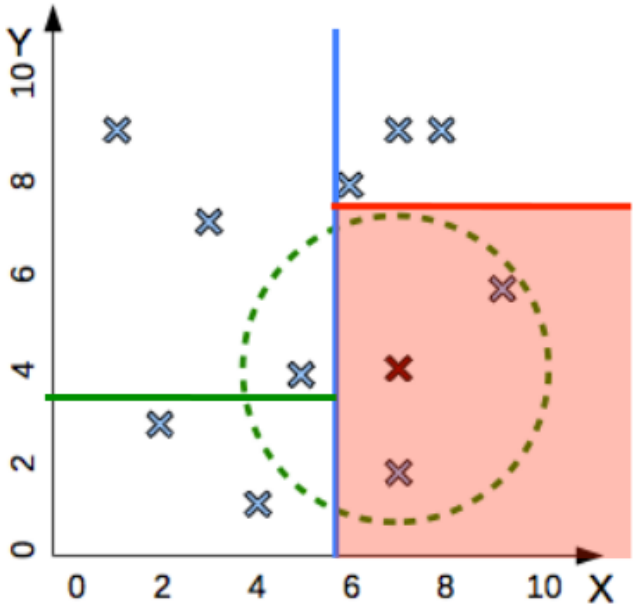
Redução da complexidade

Contruir uma kd-tree a partir dos dados de treinamento: escolher uma dimensão, encontrar a média, separar os dados, repetir.

Exemplo: encontrar os pontos mais próximos a (7,4).

- encontrar a região contendo (7,4)
- comparar a distância para todos os pontos nesta região





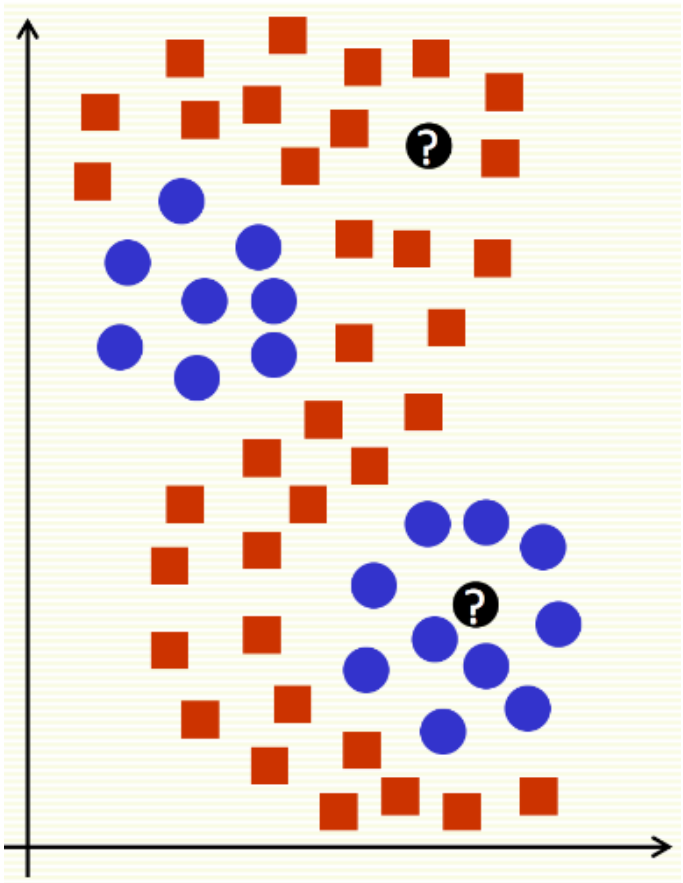
Redução da complexidade

Utilizando kd-trees, temos a seguinte complexidade:

- contrução da kd-tree: $\mathcal{O}(n \log n)$.
- busca do vizinho mais próximo: aproximadamente $\mathcal{O}(\log n)$.
- busca dos m vizinhos mais próximos: aproximadamente $\mathcal{O}(m \log n)$.

Dados multimodais

A maioria dos métodos paramétricos não consegue lidar com a distribuição do exemplo abaixo. O k-NN tem um desempenho satisfatório, considerando que um número adequado de amostras está disponível.



Sumário

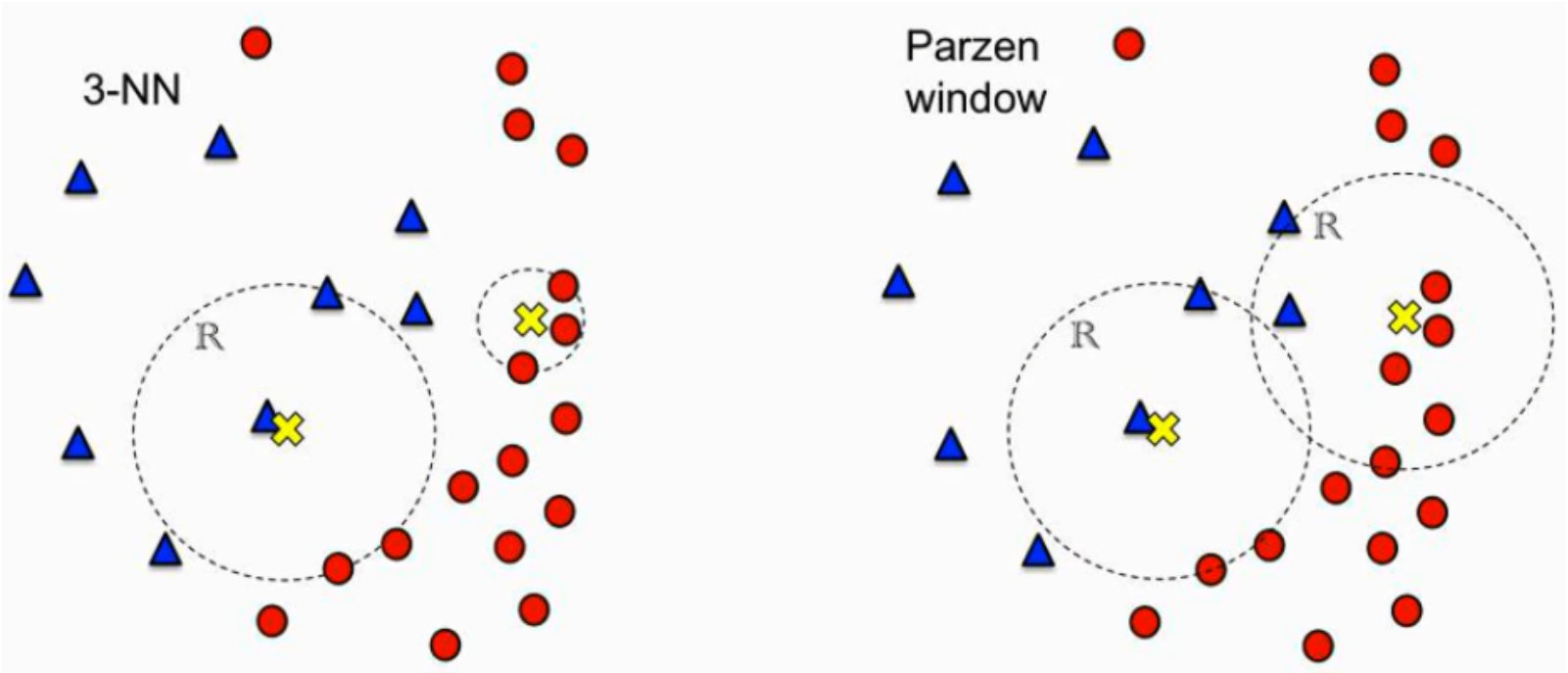
Vantagens

- Pode ser aplicado em dados com qualquer distribuição.
- Simples e intuitivo.
- A classificação é boa se a quantidade de amostras é grande o suficiente.

Desvantagens

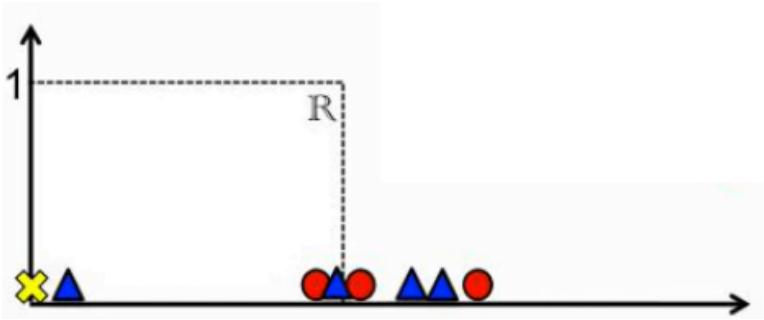
- Escolher o k pode ser difícil.
- Tem um custo computacional associado, mas isso pode ser melhorado.
- Precisa de um grande número de amostras para ter um bom desempenho.
- *Curse of Dimensionality*: a quantidade de dados necessária para treinamento cresce exponencialmente com o aumento da dimensão! Com isso, o curso computacional também aumenta.

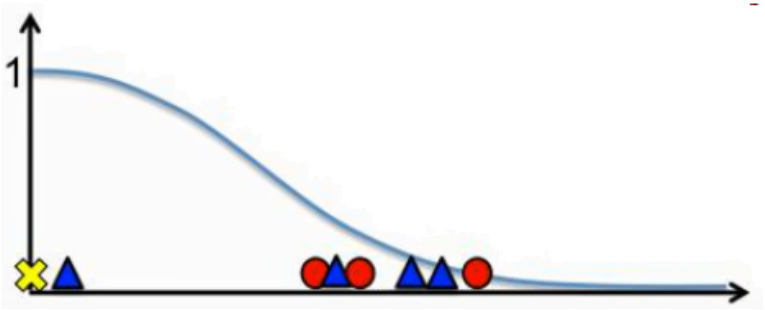
Parzen windows e kernels



$$P(y|x) = \frac{1}{|R(x)|} \sum_{x_i \in R(x)} 1_{y_i=y} = \frac{\sum_i 1_{y_i=y}.1_{x_i \in R(x)}}{\sum_i 1_{x_i \in R(x)}} = \frac{\sum_i 1_{y_i=y}.K(x_i,x)}{\sum_i K(x_i,x)}$$

Parzen windows e *kernels*





Exemplo: *Where on Earth is this Photo From?*

Onde as fotos abaixo foram tiradas[1]?



[1]Paper: James Hays, Alexei A. Efros. im2gps: estimating geographic information from a single image. CVPR08. Project page: http://graphics.cs.cmu.edu/projects/im2gps/

Exemplo: *Where on Earth is this Photo From?*

A partir de 6M de imagens do Flickr com informação de GPS (amostragem densa pelo mundo). Representa as imagens com características significativas e aplica k-NN (no artigo k=120).