

# Aprendizado Baseado em Instâncias

Cleber Zanchettin

Ricardo Prudêncio

# Introdução

- Soluções para novos problemas podem ser definidas, adaptando soluções dadas a problemas *similares*
- É comum memorizarmos situações e recuperá-las quando necessário
- Procedimento usado no dia-a-dia

# Aprendizado Baseado em Instâncias

- Problemas resolvidos no passado são representados como *instâncias*
  - E.g., exemplos de treinamento previamente etiquetados
- Instâncias são *recuperadas* e *adaptadas* para resolver novos problemas

# Aprendizado Baseado em Instâncias

- Generaliza informações com base em exemplos de treinamento:
  - Para inferir a classe de *novas instâncias* (ou instâncias de consulta)
  - Cada vez que uma instância é recebida, se computa uma função objetivo com base no conhecimento oferecido pela base de *exemplos de treinamento*
- Estima a classe da nova instância com base em *comportamentos locais*
- É uma técnica *incremental*

# Aprendizado Baseado em Instâncias

- *Lazy learning* ou aprendizado preguiçoso
  - Método de aprendizagem em que a generalização além dos dados de treinamento é adiada até que uma consulta seja feita ao sistema
- Diferente da aprendizagem ansiosa (*eager learning*)
  - Sistema tenta generalizar os dados de treinamento antes de receber uma consulta

# Aprendizado Baseado em Instâncias

- Classe de algoritmos de aprendizado que inclui, por exemplo:
  - *K-Vizinhos Mais Próximos*
  - *Raciocínio Baseado em Casos*

# Algoritmo de K-Vizinhos Mais Próximos

K-Nearest Neighbors (k-NN)

# Algoritmo k-NN

- Todas as instâncias correspondem a *pontos* em um espaço *n-dimensional*
- Vizinhança definida por uma função de *distância*, ou por uma função de *similaridade*
  - Menor distância = maior similaridade
- Classe de um novo exemplo é definida a partir dos *vizinhos mais próximos*



# Algoritmo k-NN

- O atributo de saída é definido na forma:
  - *Discreta* (por exemplo, por maior número de votos)
  - *Contínua* (o atributo de saída é definido pela ponderação das saídas das K instâncias mais próximas)

$$\hat{f}(x_q) = \frac{\sum_{i=1}^k f(x_i)}{k}$$

# Algoritmo k-NN

- Definições:
  - $x_i$  : instância descrita pelo vetor  $\langle a_1(x_i), \dots, a_n(x_i) \rangle$
  - $f(x_i)$  : classe de  $x_i$
- Treinamento básico:
  - Armazenar exemplos de treinamento  $\langle x_i, f(x_i) \rangle$

# Algoritmo k-NN

- Dado exemplo  $x_q$  a ser classificado,
  - Seja  $x_1, \dots, x_k$  as *k instâncias mais similares* a  $x_q$
  - Retorne classe *majoritária* das instâncias recuperadas

$$\hat{f}(x_q) \leftarrow \arg \max_{v \in V} \sum_{i=1}^k \delta(v, f(x_i))$$

onde

$$\delta(v, f(x_i)) = 1 \quad \text{se} \quad v == f(x_i) \quad e$$

$$\delta(v, f(x_i)) = 0, \quad \text{caso contrário}$$

# Algoritmo k-NN

- Algoritmo k-NN usa comumente a ***Distância Euclidiana*** para definição de vizinhança

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^n (a_r(x_i) - a_r(x_j))^2}$$

# Algoritmo k-NN

- Atributos de maior *escala numérica* podem dominar função de distância
  - Usualmente, os atributos são normalizados para intervalo entre 0 e 1

$$a_{NORM}(x) = \frac{a(x) - \min_i(a(x_i))}{\max_i a(x_i) - \min_i(a(x_i))}$$

$$a_{NORM}(x) = \frac{a(x) - \text{mean}_i(a(x_i))}{\text{std}_i(a(x_i))}$$

# Algoritmo k-NN

- Boa prática: incluir a normalização dos dados implicitamente no cálculo da distância

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^n \frac{(a_r(x_i) - a_r(x_j))^2}{(\max_i a_r(x_i) - \min_i a_r(x_j))^2}}$$

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^n \frac{(a_r(x_i) - a_r(x_j))^2}{(std_i(a_r(x_i)))^2}}$$

# Algoritmo k-NN

- Distância de Hamming para *atributos categóricos*:
  - Soma 1 para cada atributo cujo valor não coincide nas instâncias

$$d_{HAMMING}(x_i, x_j) = \sum_{r=1}^n dist(a_r(x_i), a_r(x_j))$$

$$dist(a_r(x_i), a_r(x_j)) = \begin{cases} 0, & \text{se } a_r(x_i) == a_r(x_j) \\ 1, & \text{caso contrário} \end{cases}$$

# Algoritmo k-NN

- Função de distância considerando *missing values* (Witten, Frank (2000, p.115)):
  - Para atributos categóricos: distância é igual a 1 na presença de *missing values*
  - Para atributos numéricos:
    - Se os dois valores comparados são *missing values* então distância igual a 1
    - Se apenas um dos valores é *missing value*, então distância é o maior dentre os seguintes valores:
      - Tamanho normalizado do atributo presente
      - Um (1) menos o tamanho normalizado do atributo presente



# Algoritmo k-NN

- Outras funções de distância:

- Distância L1 Normalizada

$$d(x_i, x_j) = \sum_{r=1}^n \frac{|a_r(x_i) - a_r(x_j)|}{\max_i a_r(x_i) - \min_i a_r(x_j)}$$

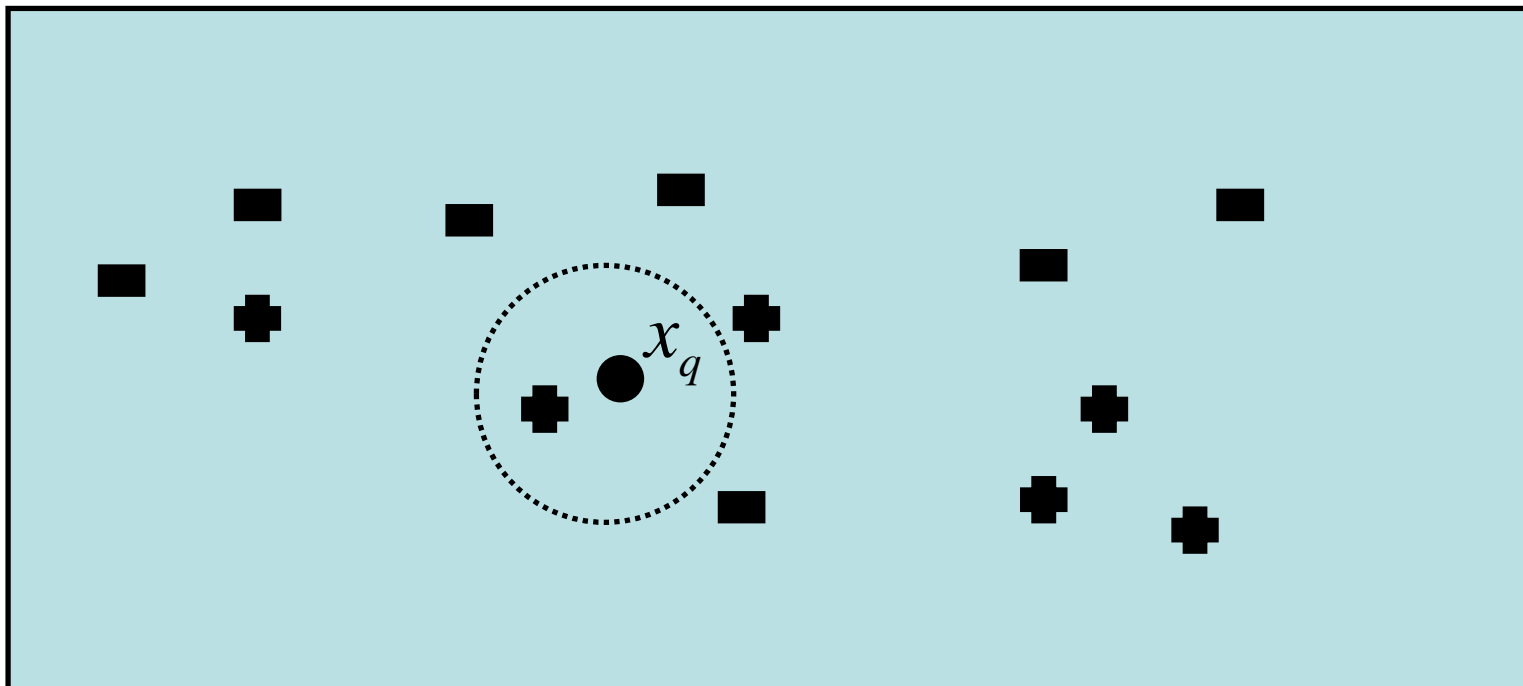
- Distância Cosseno Normalizada

$$d(x_i, x_j) = \frac{\sum_{r=1}^n a_r(x_i) * a_r(x_j)}{\sum_{r=1}^n a_r(x_i)^2 * \sum_{r=1}^n a_r(x_j)^2}$$

# Algoritmo k-NN

## - Exemplo

Espaço de instâncias



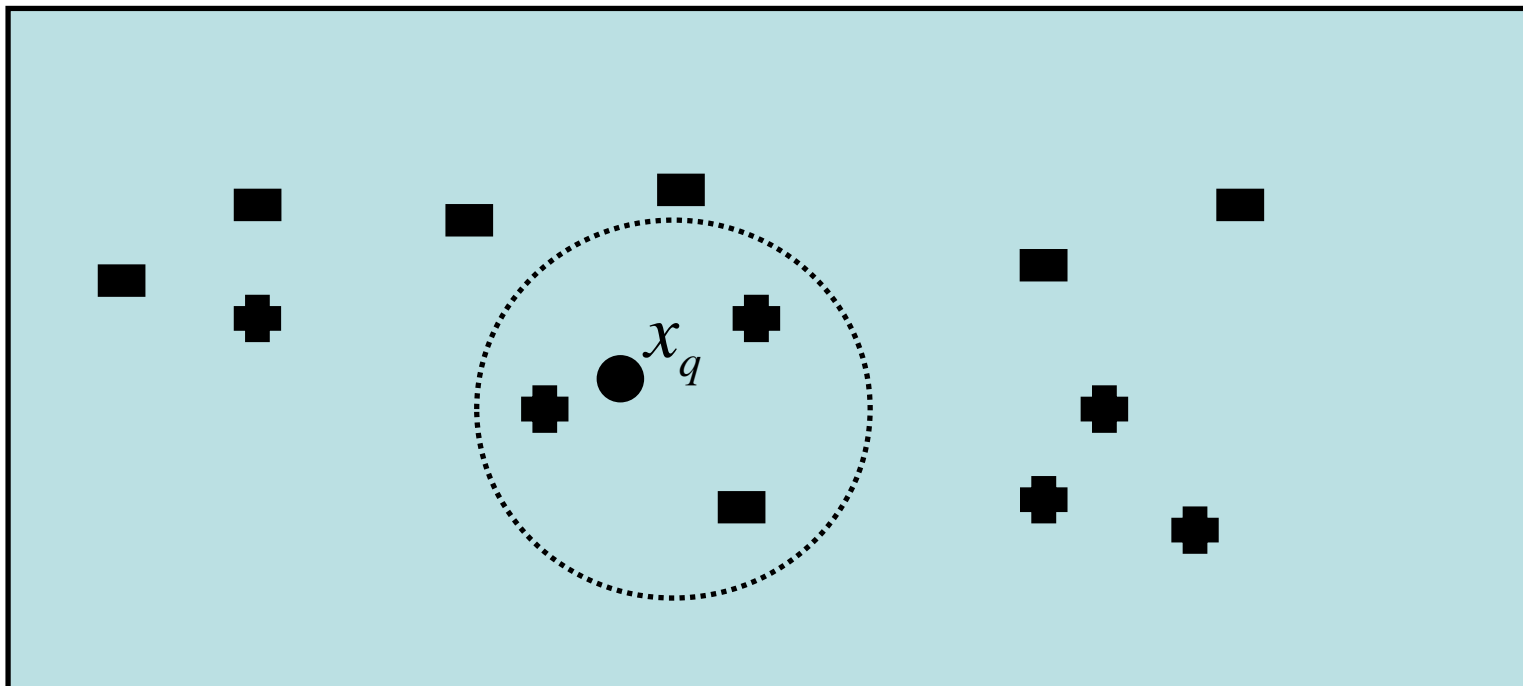
- Exemplos da classe negativa
- ✚ Exemplos da classe positiva
- Exemplos a ser classificado

- Com  $k = 1$ , exemplo  $x_q$  recebe classe positiva

# Algoritmo k-NN

## - Exemplo

Espaço de instâncias



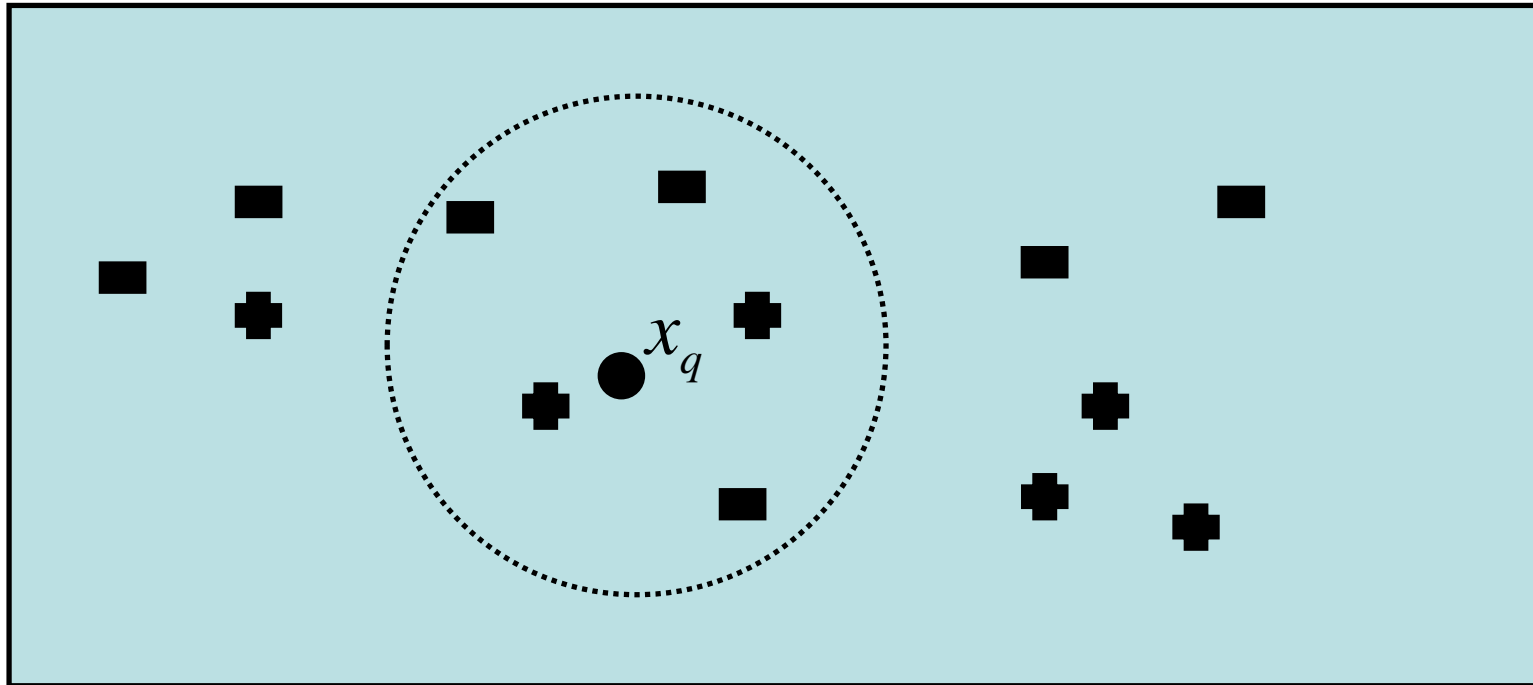
- Exemplos da classe negativa
- ✚ Exemplos da classe positiva
- Exemplos a ser classificado

- Com  $k = 3$ , exemplo  $x_q$  recebe classe positiva

# Algoritmo k-NN

## - Exemplo

Espaço de instâncias

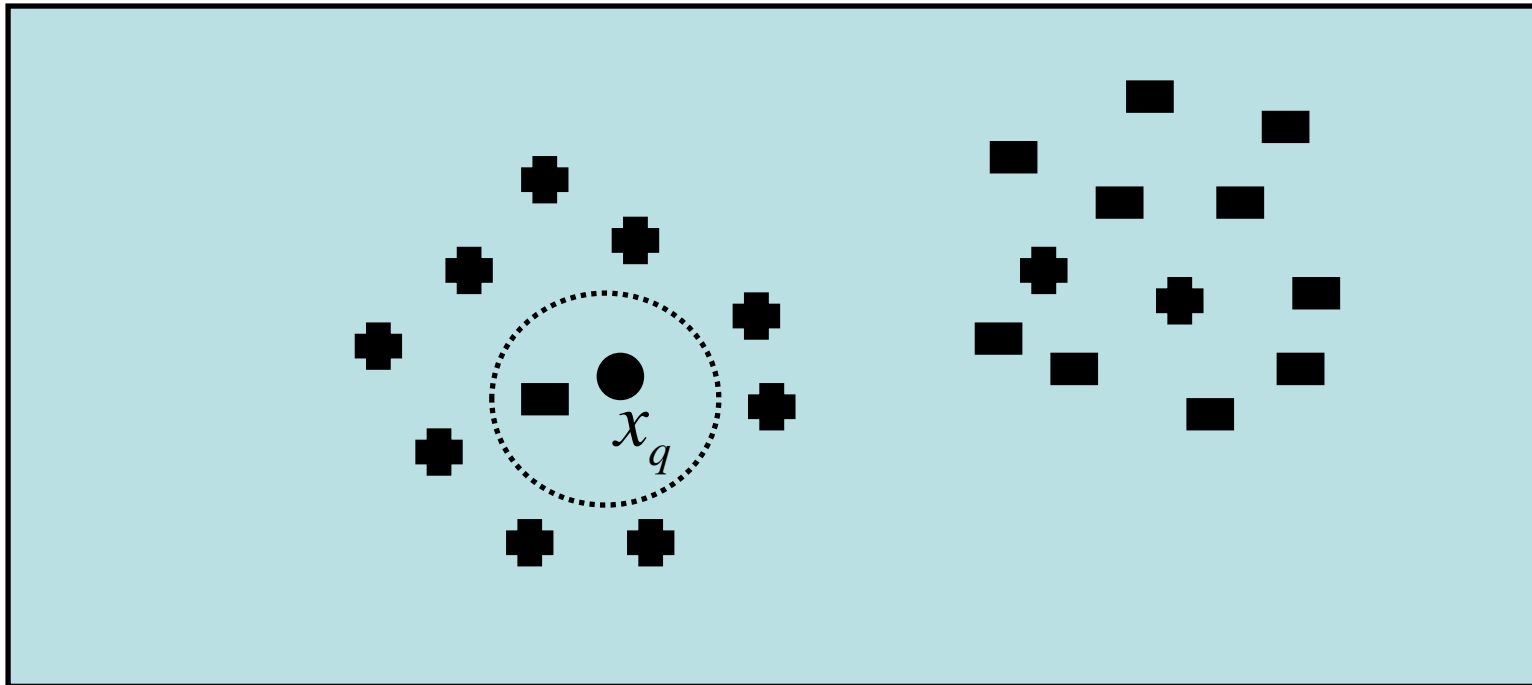


- Exemplos da classe negativa
- ⊕ Exemplos da classe positiva
- Exemplos a ser classificado

- Com  $k = 5$ , exemplo  $x_q$  recebe classe negativa

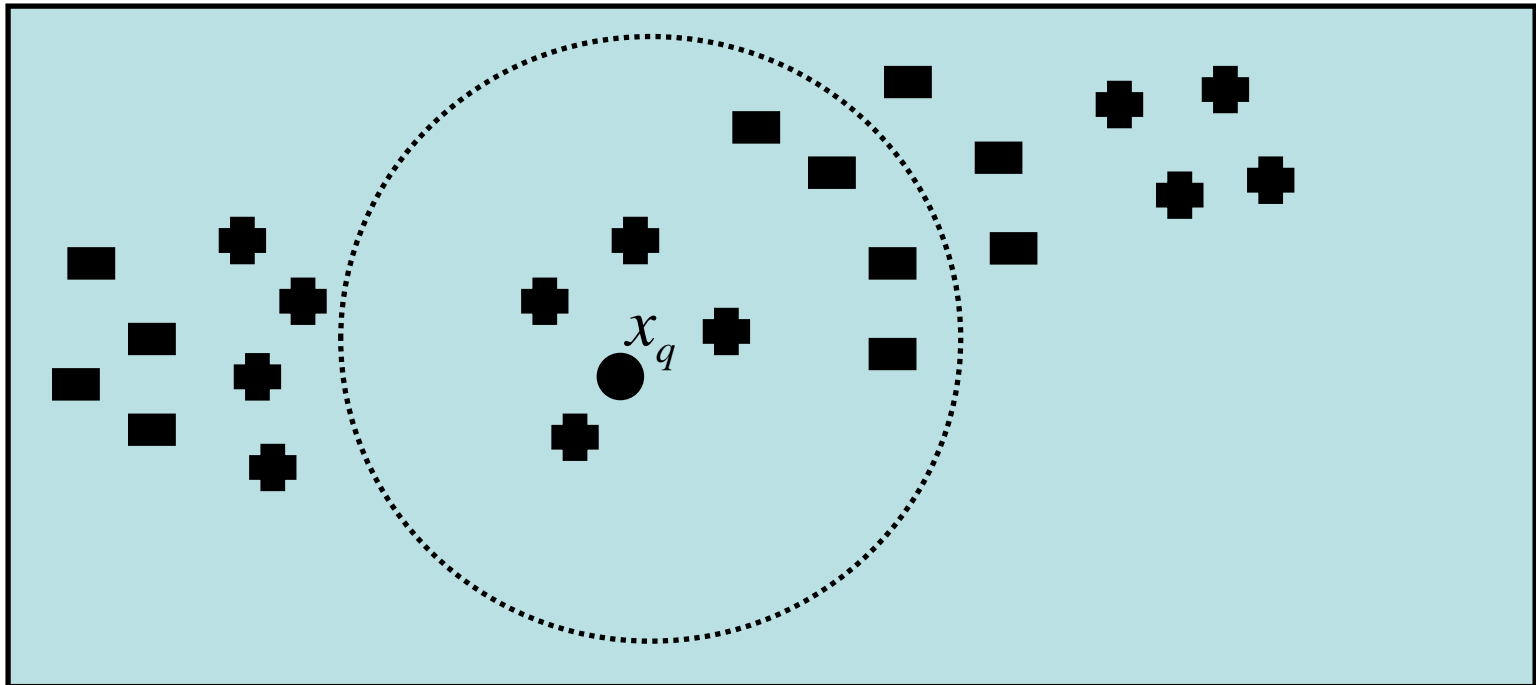
# Algoritmo k-NN

- O *dilema* da escolha do parâmetro k
  - O algoritmo se torna mais flexível
  - Valores muito baixos podem aumentar a contribuição de *exemplos ruidosos*
  - *Menor gasto computacional*



# Algoritmo k-NN

- O *dilema* da escolha do parâmetro k
  - Valores muito altos podem aumentar a contribuição de exemplos *pouco similares*, e assim, *menos relevantes*
  - *Mais robusto a ruído*
  - *Menor flexibilidade*

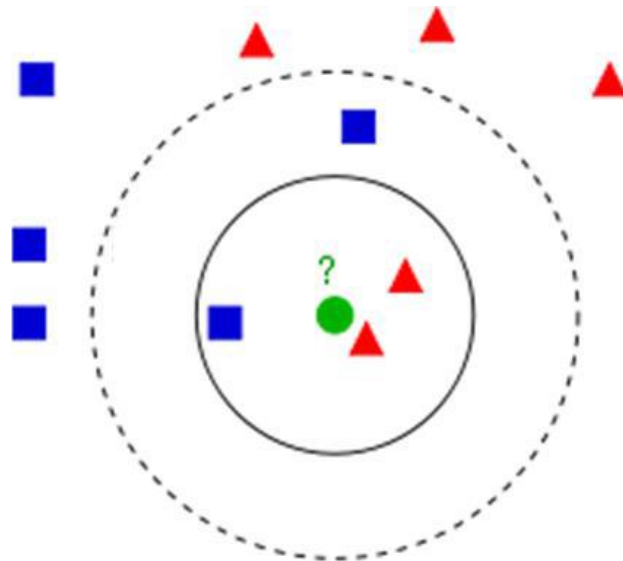


# Algoritmo k-NN

- O que fazer em caso de **empate** entre duas ou mais classes?
  - Considerar apenas os  **$k-1$**  vizinhos mais próximos
  - Em caso de novo empate, **repetir** esse processo
  - Esse processo para quando uma classe for **unânime**

k = 4: empate

k = 3: triângulo vermelho



# Algoritmo k-NN

- O valor do parâmetro  $k$  é escolhido comumente através de *tentativa-e-erro*
  - Avaliação empírica com diferentes valores de  $k$
  - *Validação cruzada*



# Algoritmo k-NN com Ponderação pela Distância

- A contribuição de cada vizinho pode ser ponderada pela distância com a instância a ser classificada

$$\hat{f}(x_q) \leftarrow \arg \max_{v \in V} \sum_{i=1}^k w_i * \delta(v, f(x_i))$$

$$w_i = \frac{1}{d(x_q, x_i)^2} \quad w_i = \frac{1}{d(x_q, x_i)} \quad w_i = 1 - d(x_q, x_i)$$

# Algoritmo k-NN com Ponderação pela Distância

- Com ponderação, a escolha adequada de  $k$  se tornaria menos importante?
  - Note que instâncias muito distantes teriam pouca contribuição na predição
- “There is no harm in allowing all training examples to have an influence on the classification...” – T. Mitchell (1997, p. 234)
- *Método de Shepard*: k-NN ponderado usando todos os exemplos de treinamento como vizinhos

# Algoritmo k-NN

## - Discussão

- Se somente os k mais próximos forem considerados:
  - Algoritmo é denominado *local*
- Se todas as instâncias de treinamento forem consideradas:
  - Algoritmo é denominado *global*

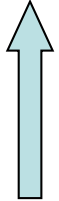
# Algoritmo k-NN

## - Discussão

- Computam a saída em função dos atributos de entrada
- Se os atributos de entrada não forem bem definidos o resultado pode ser ruim:
  - Por exemplo, instâncias são representadas por 10 atributos de entrada e 1 de saída
  - Porém, somente 2 atributos de entrada são relevantes
  - Os 8 outros podem influenciar a saída, mas nem deveriam existir na base de aprendizado!
- Uma abordagem para resolver esse problema é dar um peso para cada atributo:
  - Assim os mais relevantes serão mais considerados!
- Outra é remover os atributos menos significativos

# Algoritmo k-NN para Regressão

- Algoritmo pode ser usado para estimar valores de funções contínuas

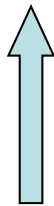
$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^k f(x_i)}{k}$$


- Predição é a *média simples* dos valores alvo armazenados nas instâncias recuperadas

# Algoritmo k-NN para Regressão

- Regressão com Ponderação pela Distância

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^k w_i f(x_i)}{\sum_{i=1}^k w_i}$$



- Predição é a *média ponderada* dos valores alvo armazenados nas instâncias recuperadas

# Algoritmo k-NN

## - Discussão

- **Vantagens**

- Fácil de implementar
- Não requer uma etapa de treinamento
- Ideal para conjuntos de dados pequenos ou médios
  - Algoritmos, como árvores de decisão, precisam de mais dados para gerar um bom modelo
- Usa informação local, podendo ser implementado com comportamentos adaptativos
- Pode ser paralelizado

# Algoritmo k-NN

## - Discussão

- **Desvantagens**

- É muito sensível a presença de atributos irrelevantes e/ou redundantes
  - *Curse of Dimensionality*
- Custo computacional e armazenamento em alguns contextos é impraticável
  - Reduzir o número de exemplos de treinamento pode amenizar esse problema
  - *Algoritmos baseados em protótipos* também podem ajudar



## □ Exemplo

- ▣ Qual a classe da amostra abaixo, dado o conjunto de treinamento ao lado?

Altura	Peso	Sexo
1,75	52,0	?

Altura	Peso	Sexo
1,87	76,1	0
1,65	75,2	1
1,80	60,0	1
1,81	55,9	0
1,90	93,3	1
1,74	65,2	1
1,49	45,1	0
1,56	53,2	0
1,73	55,1	0
1,76	63,1	1

- Primeiro passo
  - ▣ Normalizar os valores
    - z-score

Média Altura	Média Peso
1,73	64,22

Desvio Altura	Desvio Peso
0,13	14,01

$$z = \frac{x - \mu}{\sigma}$$

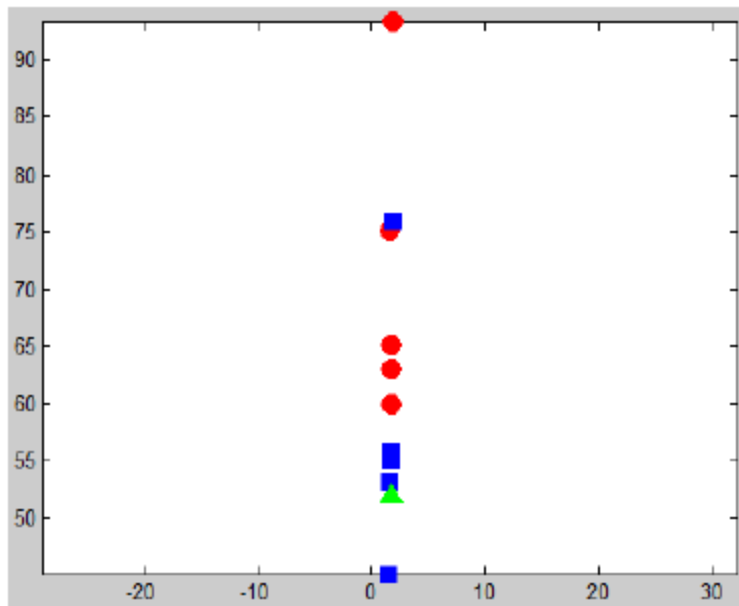
Altura	Peso	Sexo
1,87	76,1	0
1,65	75,2	1
1,80	60,0	1
1,81	55,9	0
1,90	93,3	1
1,74	65,2	1
1,49	45,1	0
1,56	53,2	0
1,73	55,1	0
1,76	63,1	1

- Primeiro passo
  - ▣ Normalizar os valores
    - Z-score

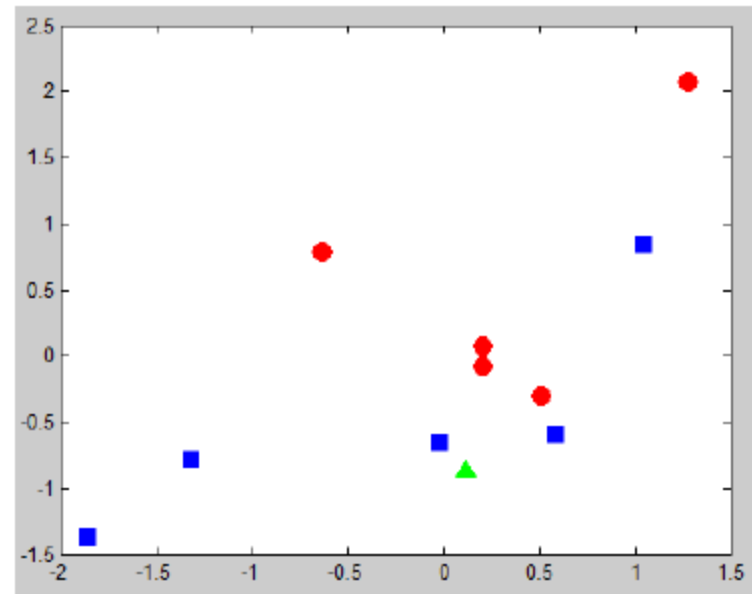
Altura	Peso	Sexo
0,12	-0,87	?

Altura	Peso	Sexo
1,04	0,84	0
-0,63	0,78	1
0,51	-0,30	1
0,58	-0,59	0
1,27	2,07	1
0,20	0,06	1
-1,85	-1,36	0
-1,32	-0,78	0
-0,02	-0,65	0
0,20	-0,07	1

## Sem ajuste de escala



## zscore



□ Segundo passo

- ▣ Calcular as distâncias da amostra desconhecida para as conhecidas

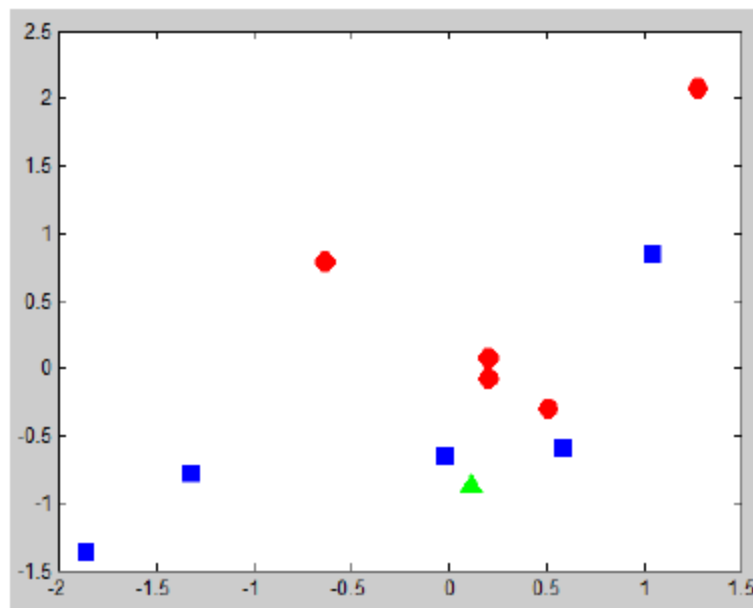
Altura	Peso	Sexo
0,12	-0,87	?

Altura	Peso	Sexo	D
1,04	0,84	0	2,64
-0,63	0,78	1	0,90
0,51	-0,30	1	0,96
0,58	-0,59	0	0,74
1,27	2,07	1	4,10
0,20	0,06	1	1,03
-1,85	-1,36	0	2,47
-1,32	-0,78	0	1,36
-0,02	-0,65	0	0,08
0,20	-0,07	1	0,88

## Terceiro passo

### Classificação: $k = 3$

Altura	Peso	Sexo
0,12	-0,87	? = 0

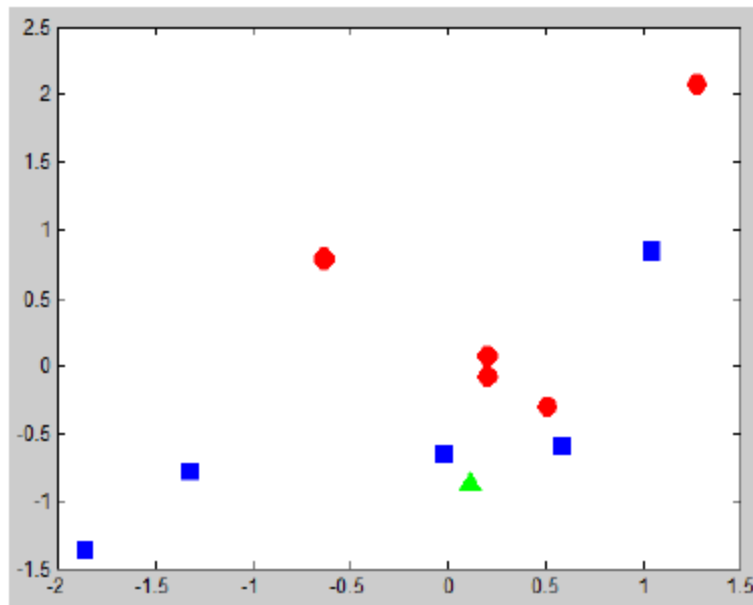


Altura	Peso	Sexo	D
1,04	0,84	0	2,64
-0,63	0,78	1	0,90
0,51	-0,30	1	0,96
<b>0,58</b>	<b>-0,59</b>	<b>0</b>	<b>0,74</b>
1,27	2,07	1	4,10
0,20	0,06	1	1,03
-1,85	-1,36	0	2,47
-1,32	-0,78	0	1,36
<b>-0,02</b>	<b>-0,65</b>	<b>0</b>	<b>0,08</b>
<b>0,20</b>	<b>-0,07</b>	<b>1</b>	<b>0,88</b>

## Terceiro passo

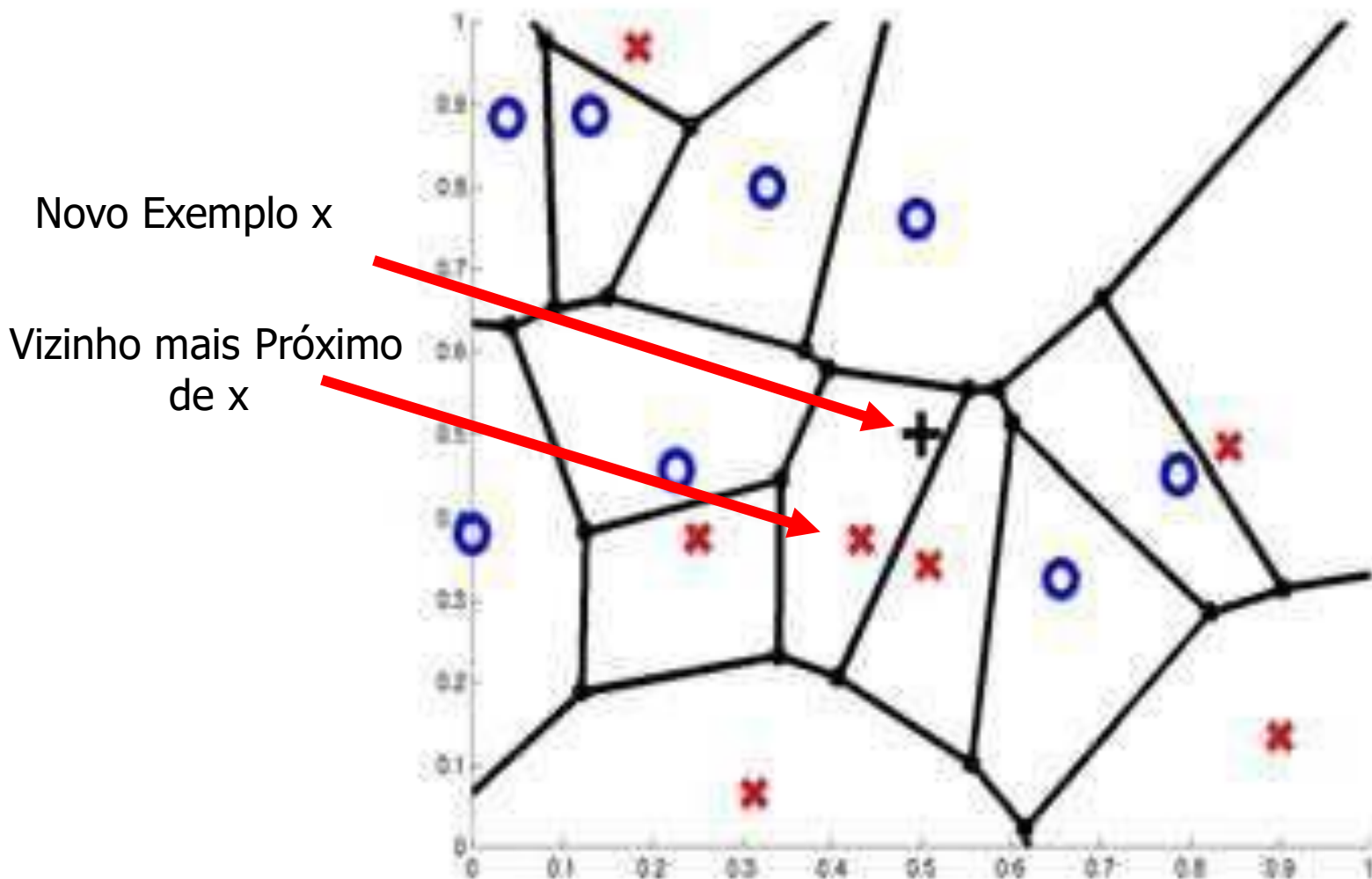
### Classificação: $k = 5$

Altura	Peso	Sexo
0,12	-0,87	? = 1



Altura	Peso	Sexo	D
1,04	0,84	0	2,64
-0,63	0,78	1	0,90
0,51	-0,30	1	0,96
0,58	-0,59	0	0,74
1,27	2,07	1	4,10
0,20	0,06	1	1,03
-1,85	-1,36	0	2,47
-1,32	-0,78	0	1,36
-0,02	-0,65	0	0,08
0,20	-0,07	1	0,88

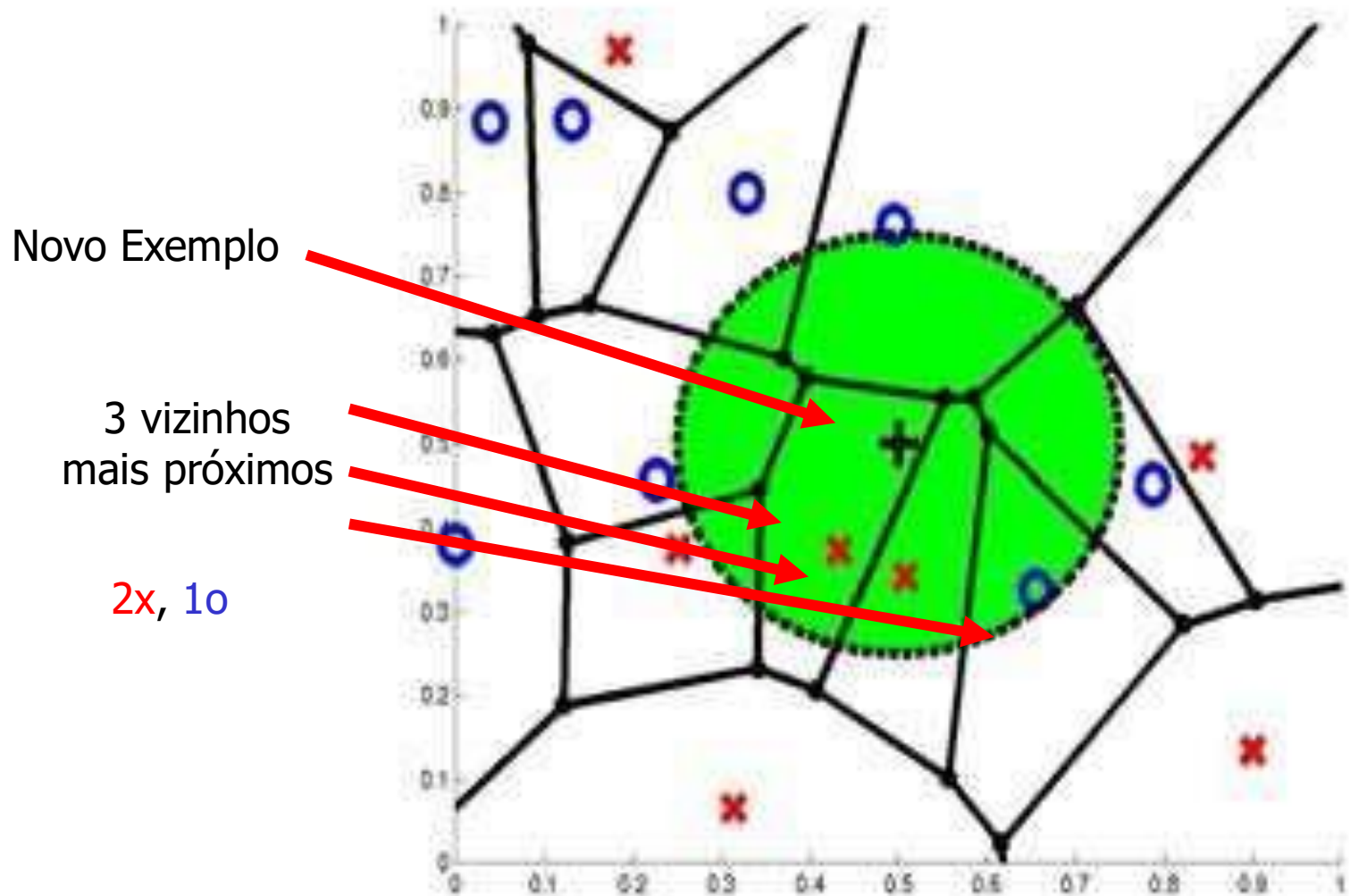
# Diagrama de Voronoi



Exemplo: problema de 2 classes (x, o) e 2 atributos representados nos eixos x e y



# Diagrama de Voronoi, K=3

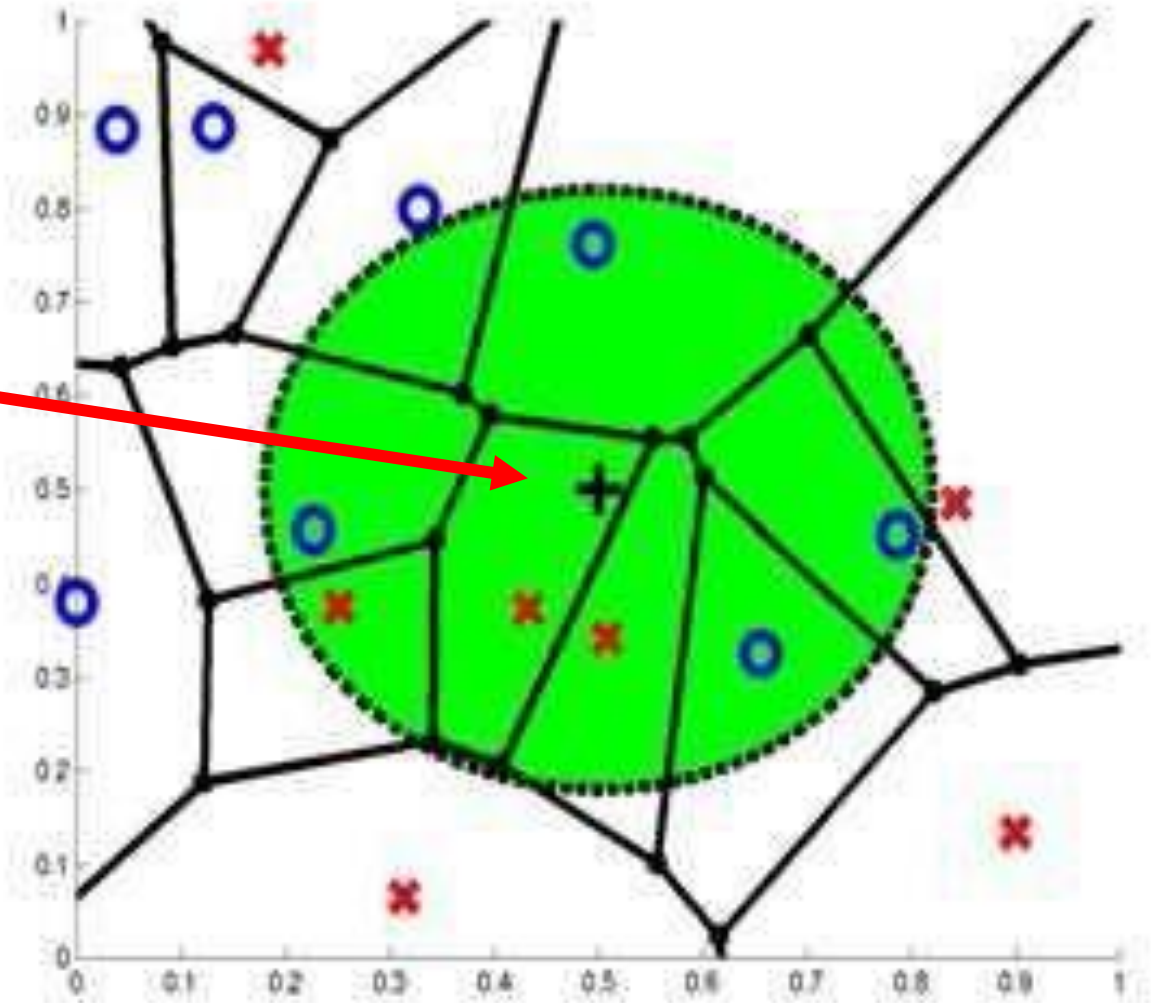


# Diagrama de Voronoi, K=7

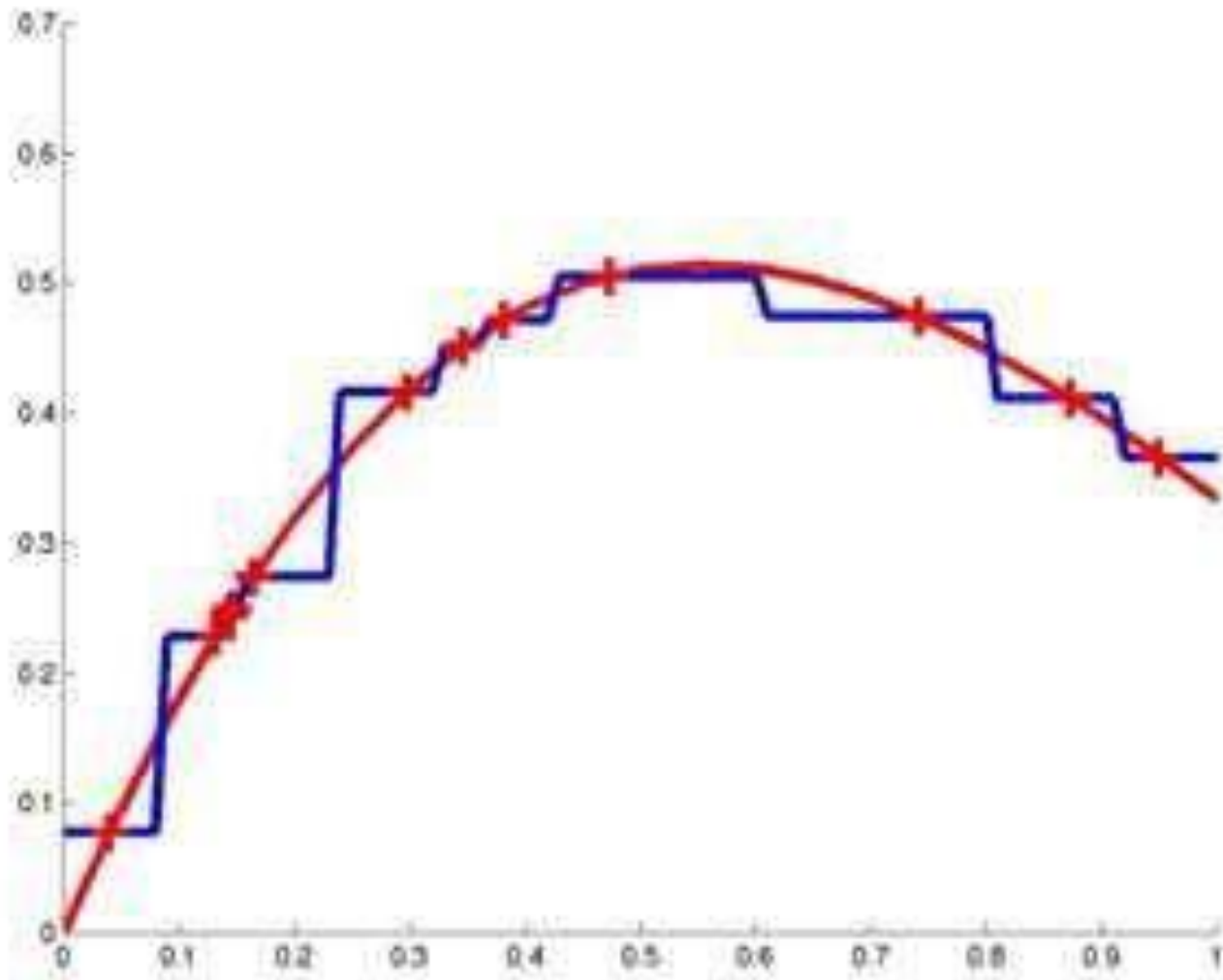
Novo Exemplo

7 vizinhos  
mais próximos

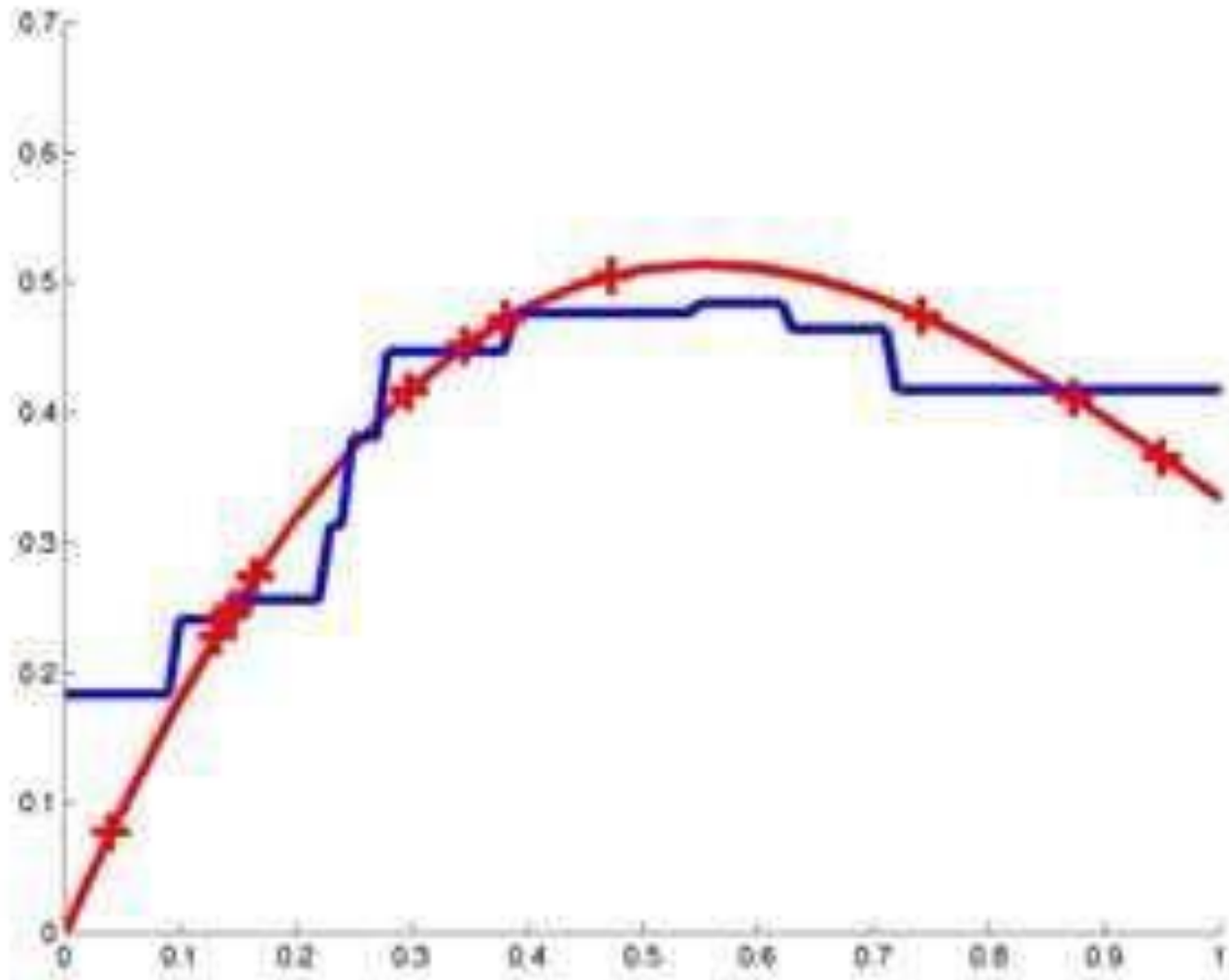
3x, 4o



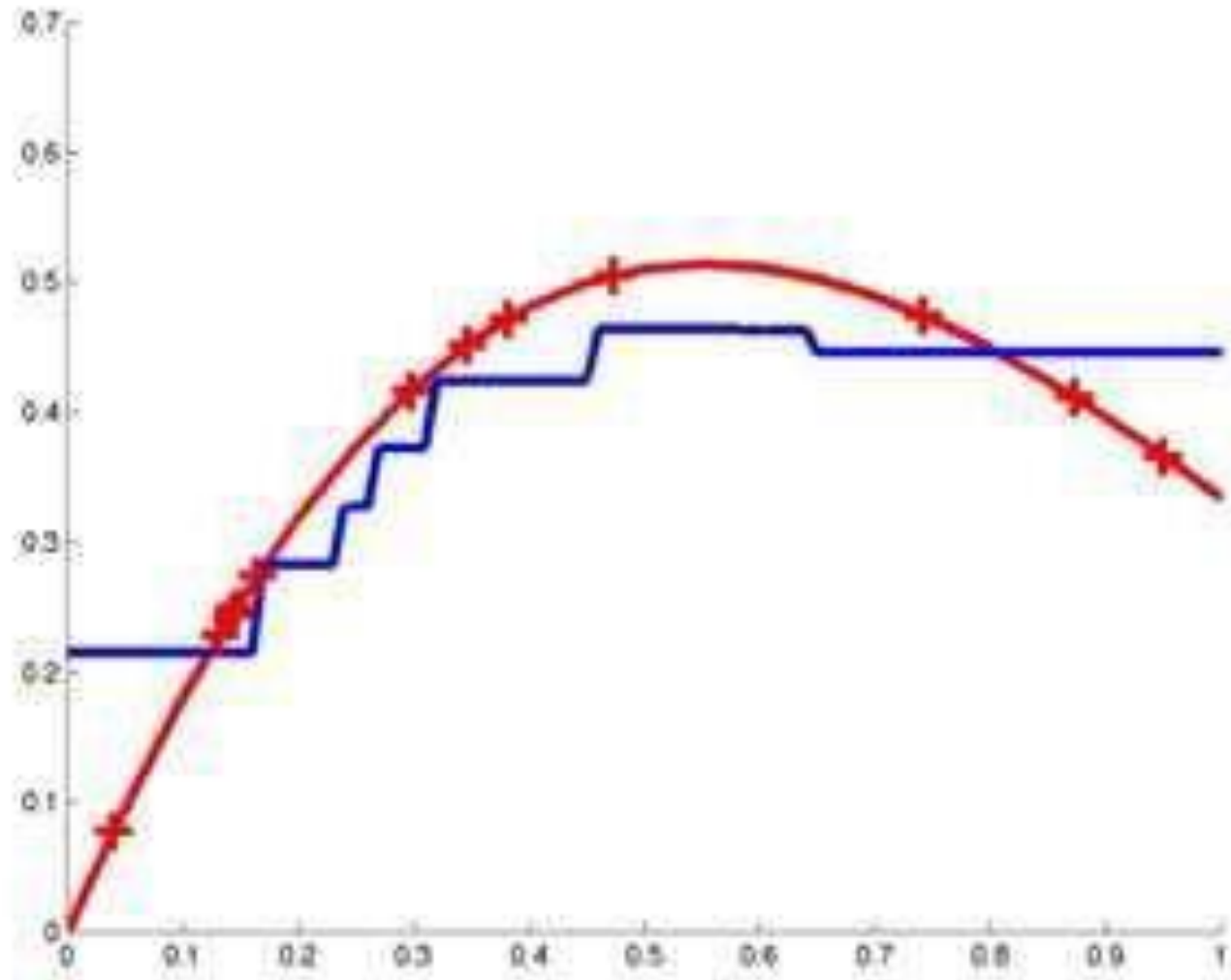
# KNN: Regressão, $K=1$



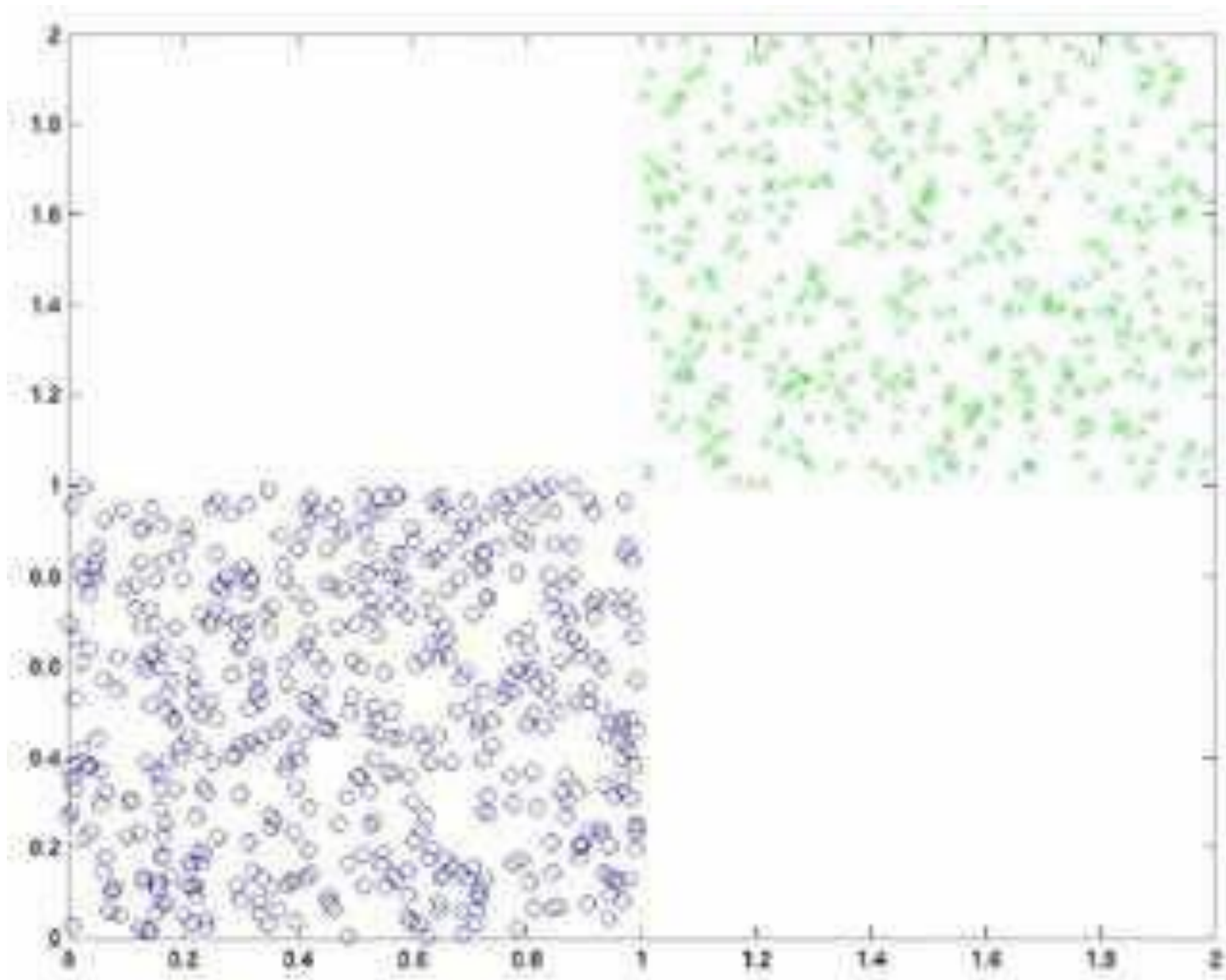
# KNN: Regressão, $K=3$



# KNN: Regressão, $K=5$

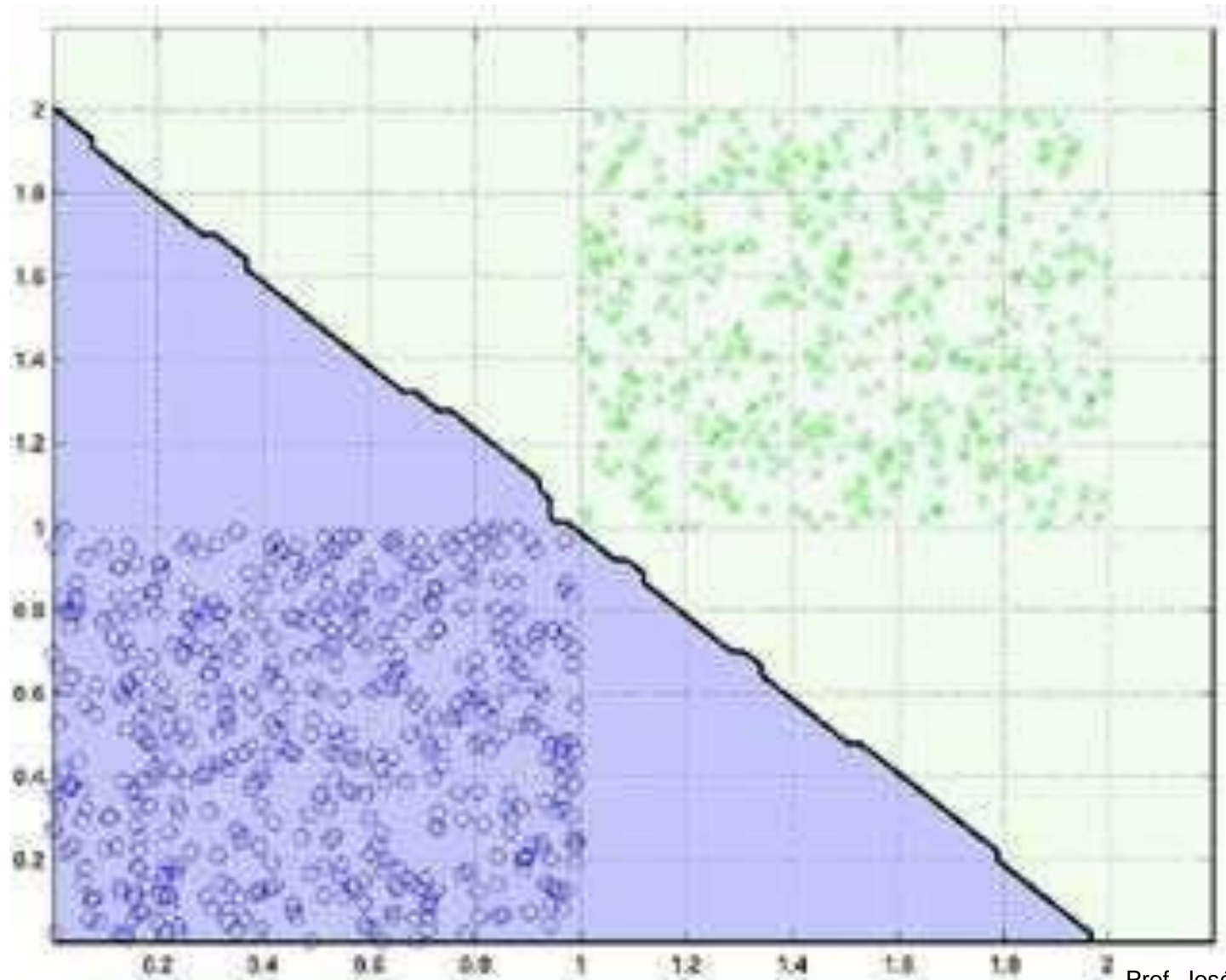


# Fronteiras

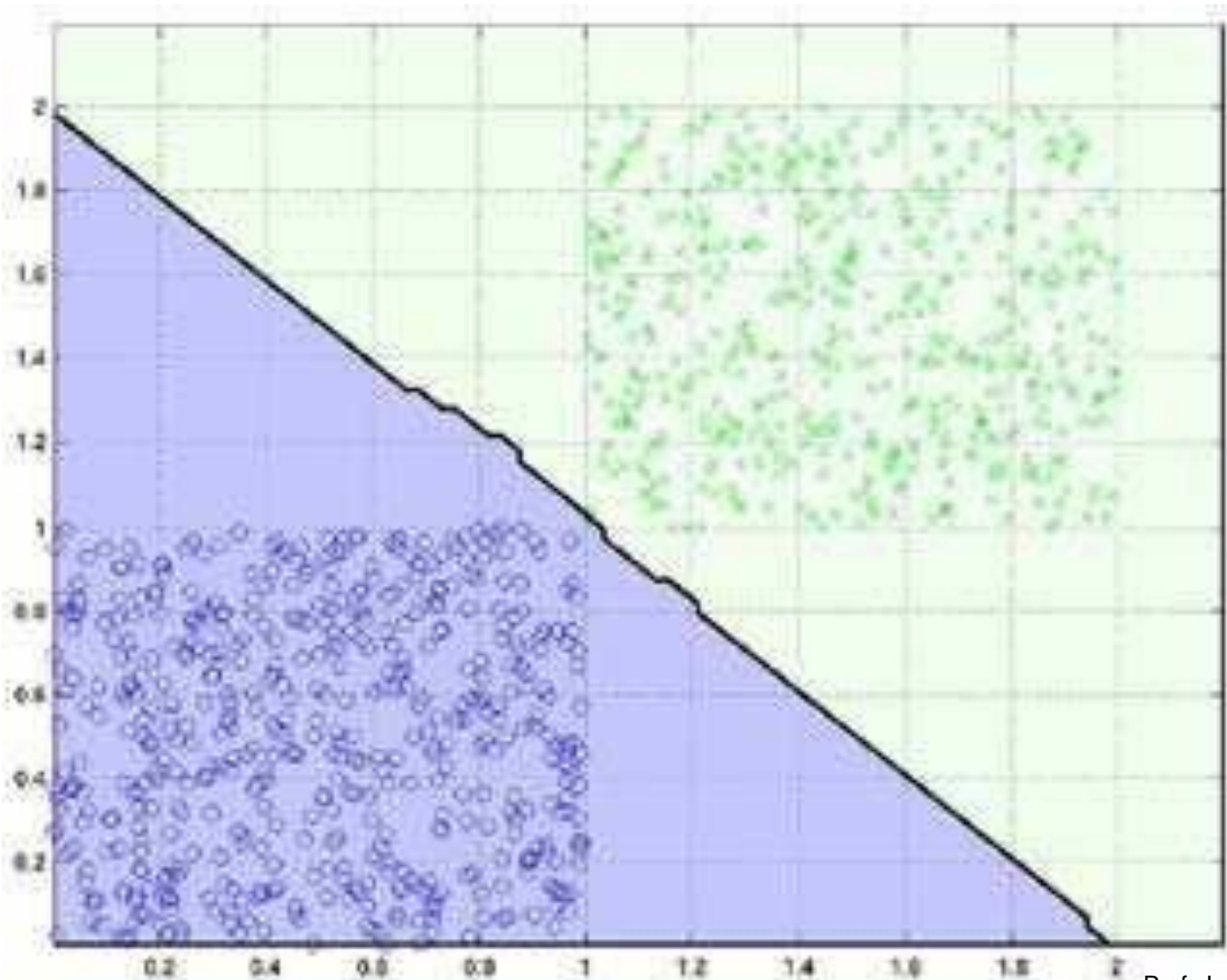




# Fronteiras, K=1

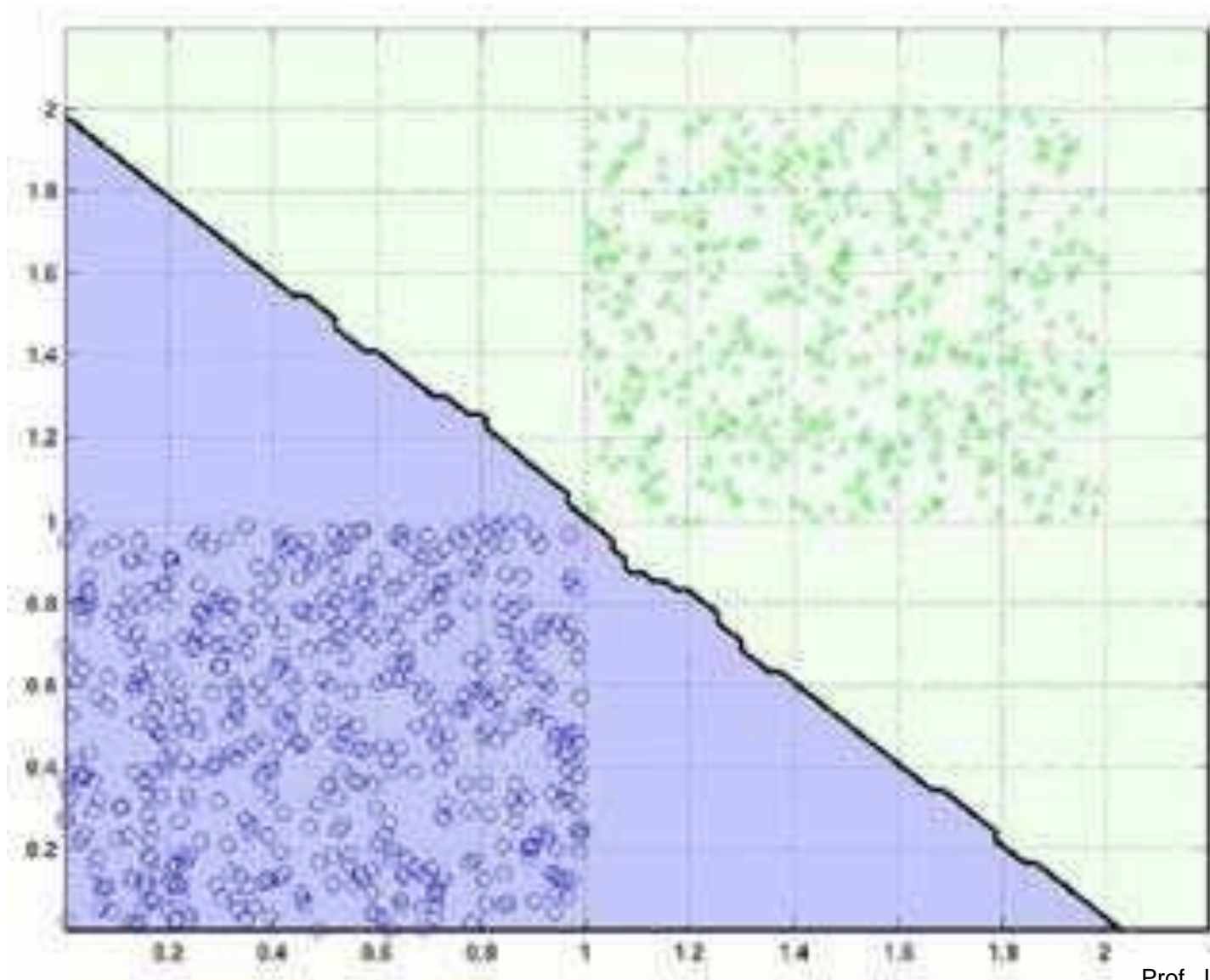


# Fronteiras, K=5

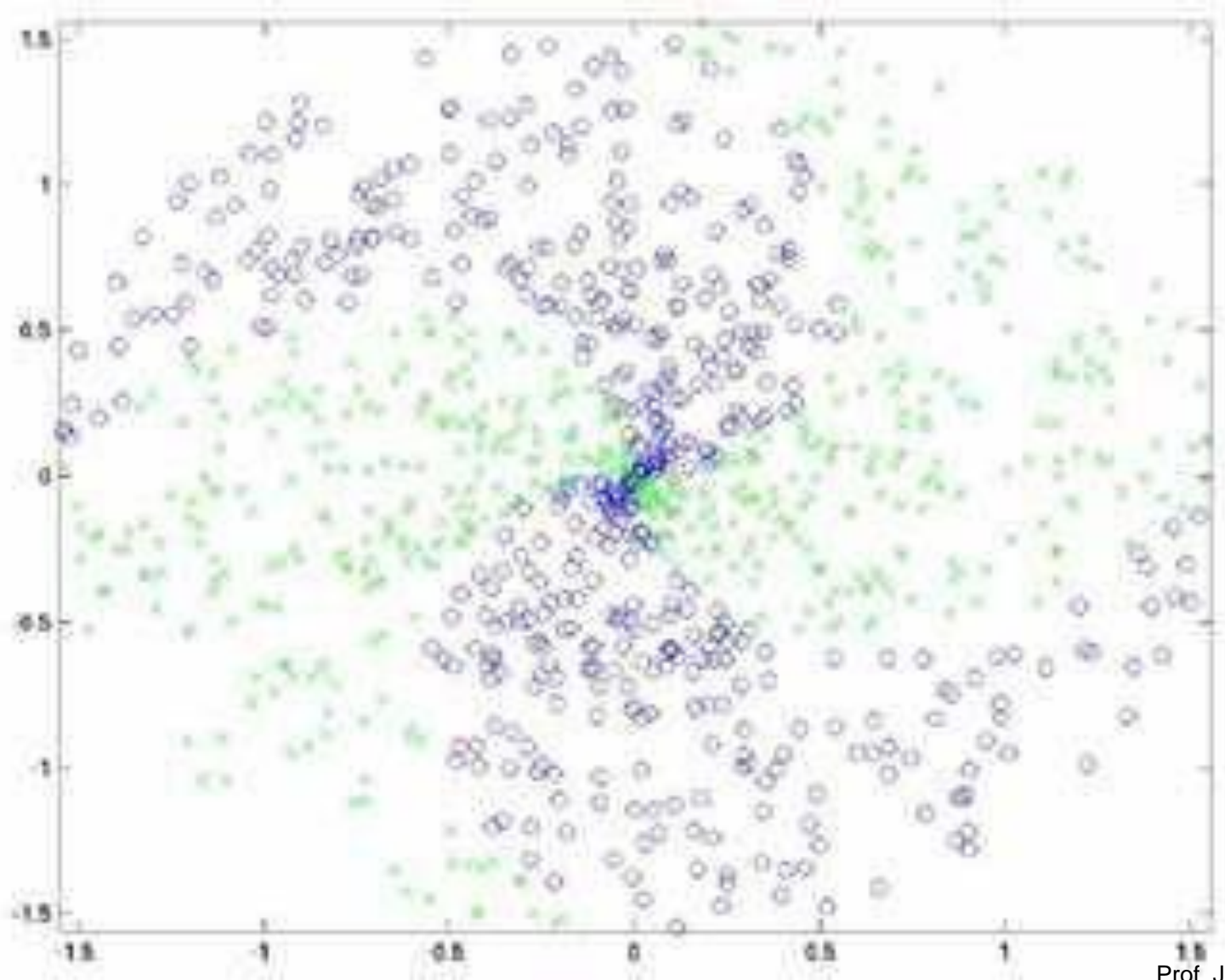




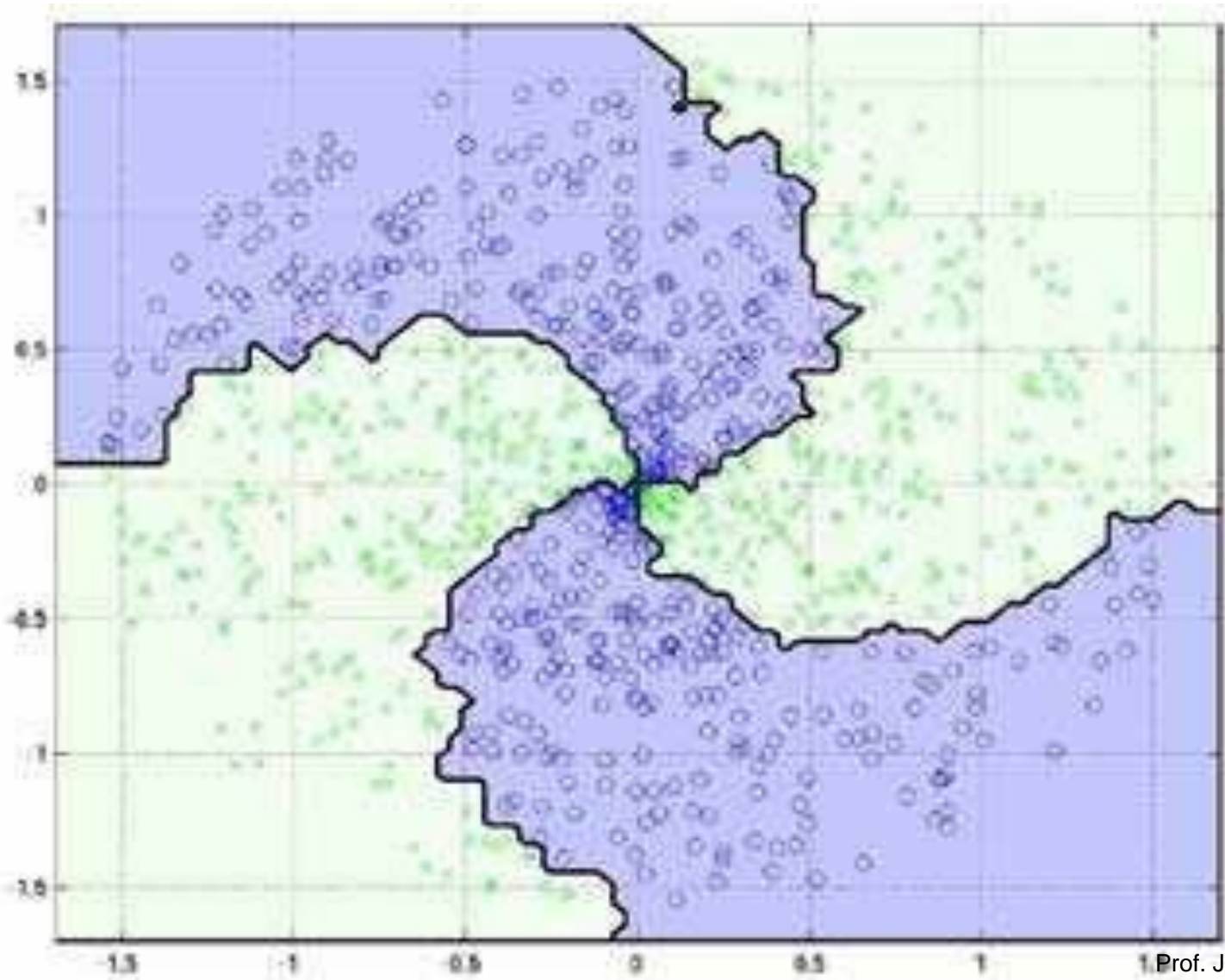
# Fronteiras, K=10



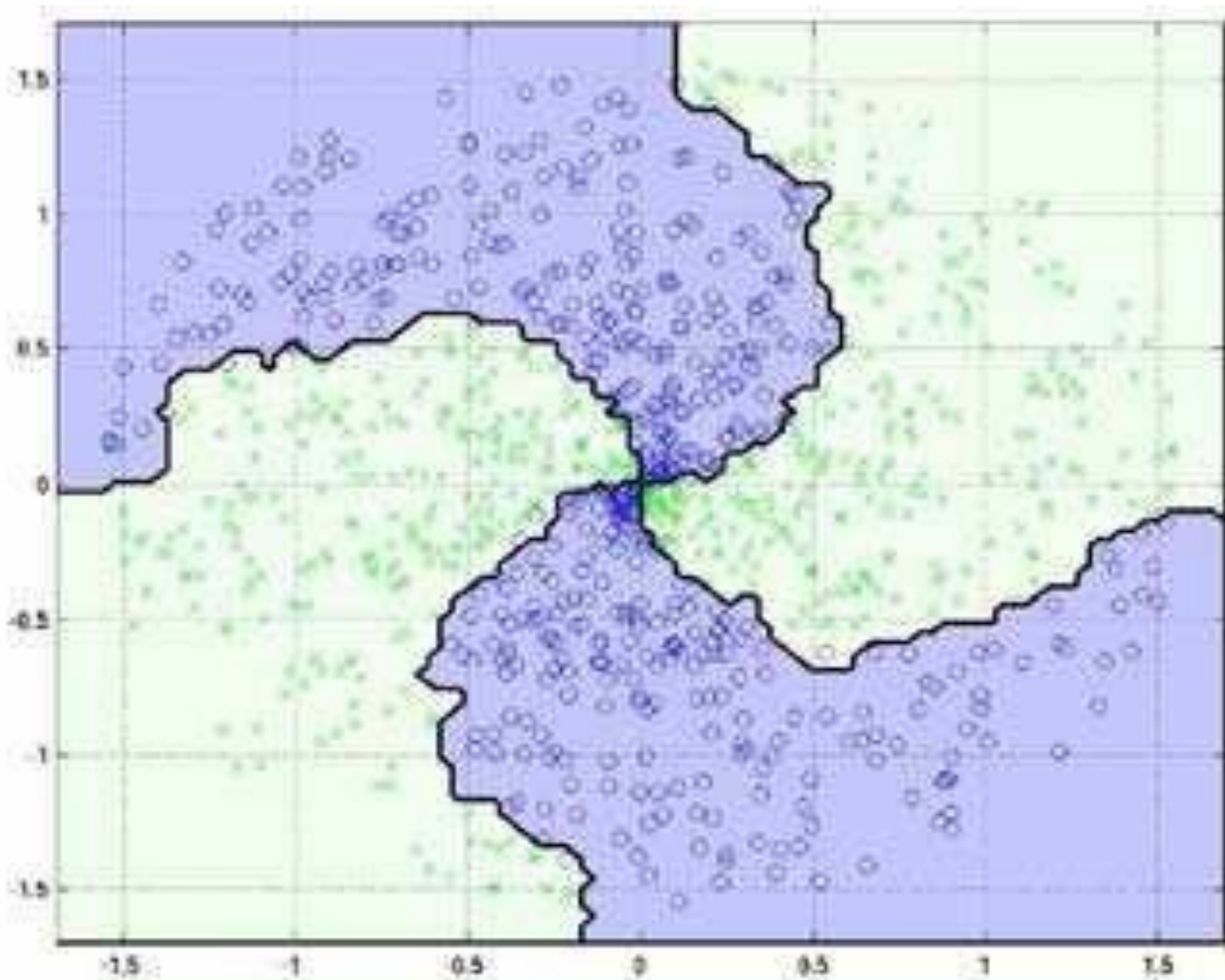
# Fronteiras



# Fronteiras, $K=1$

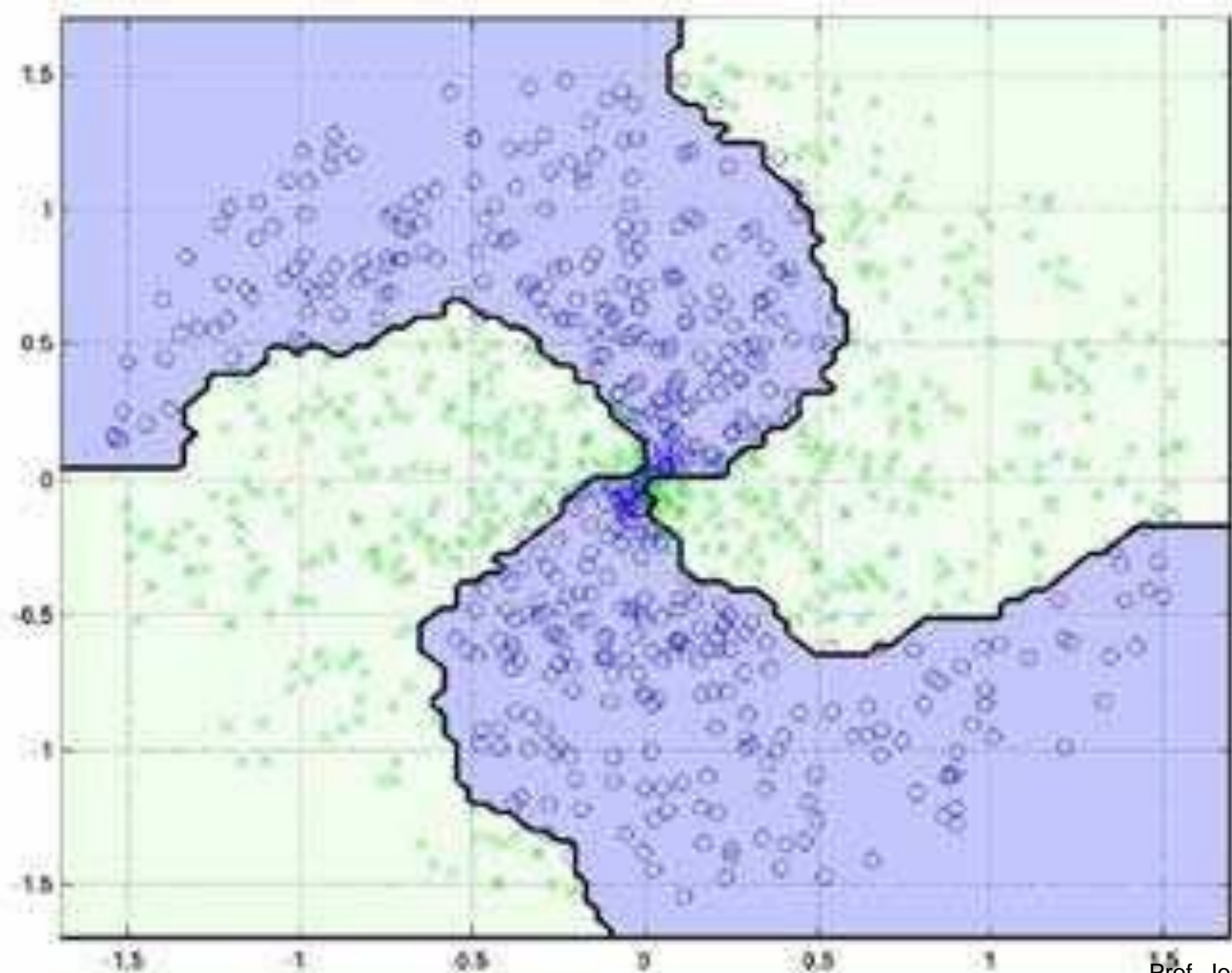


# Fronteiras, K=5

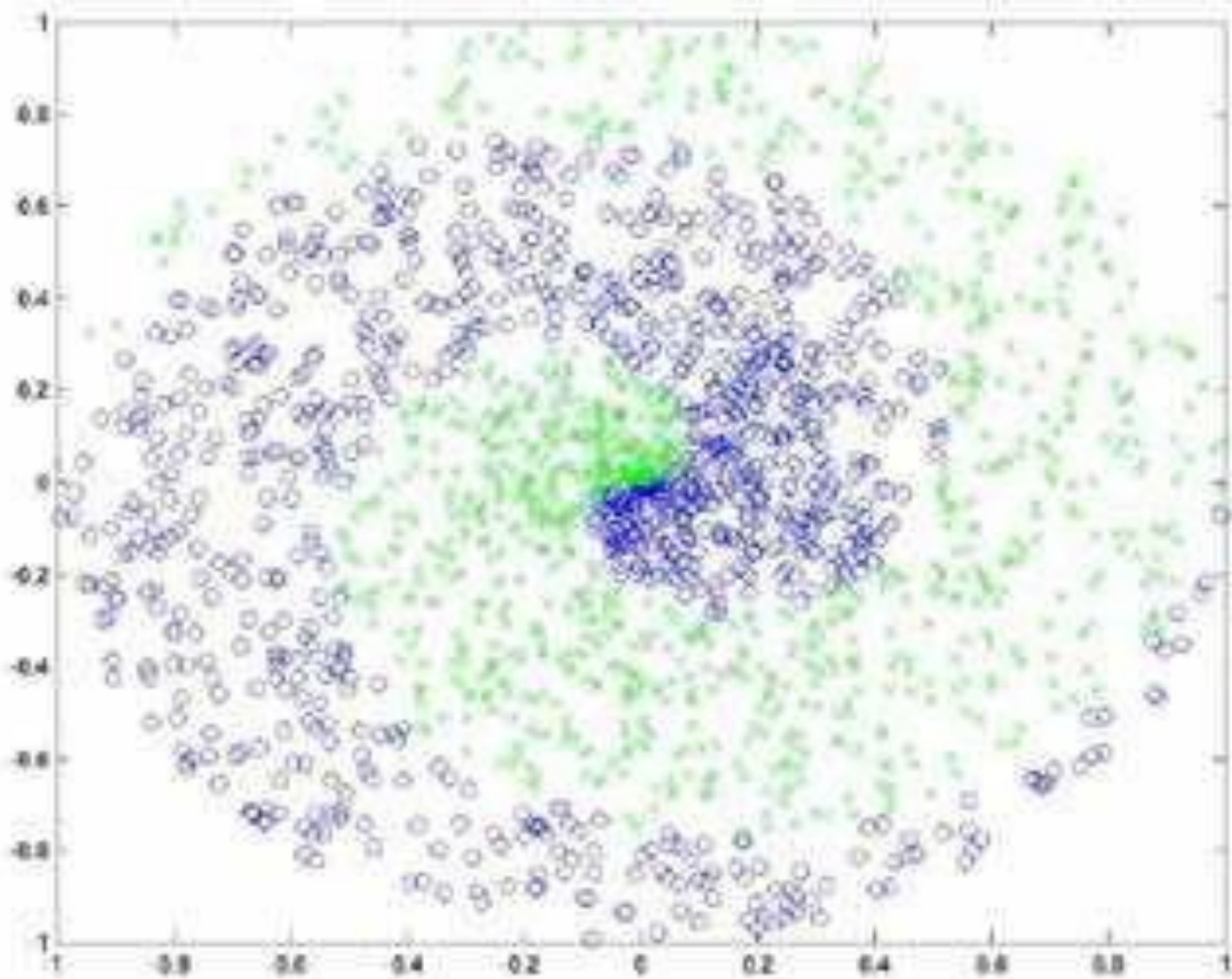




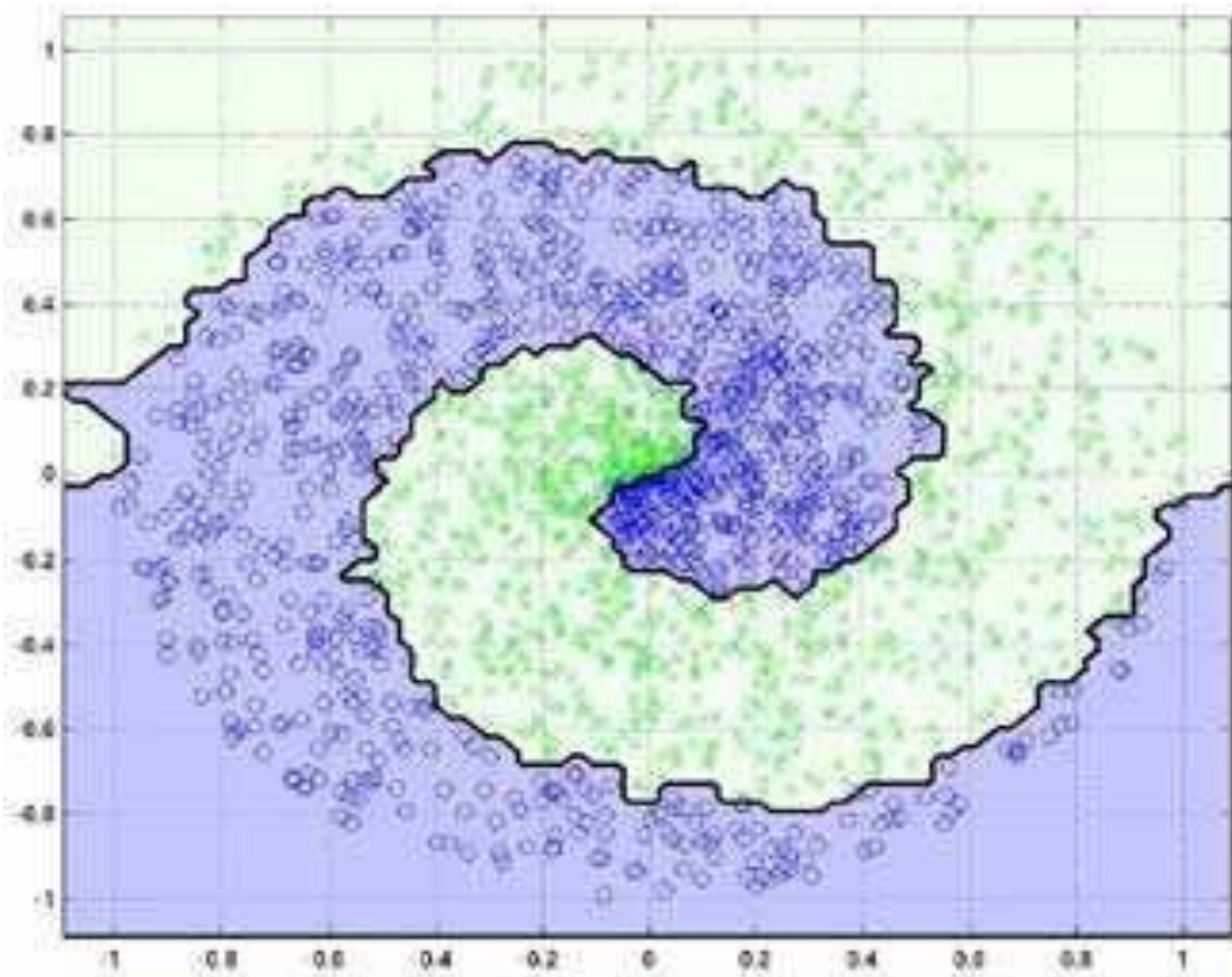
# Fronteiras, K=10



# Fronteiras

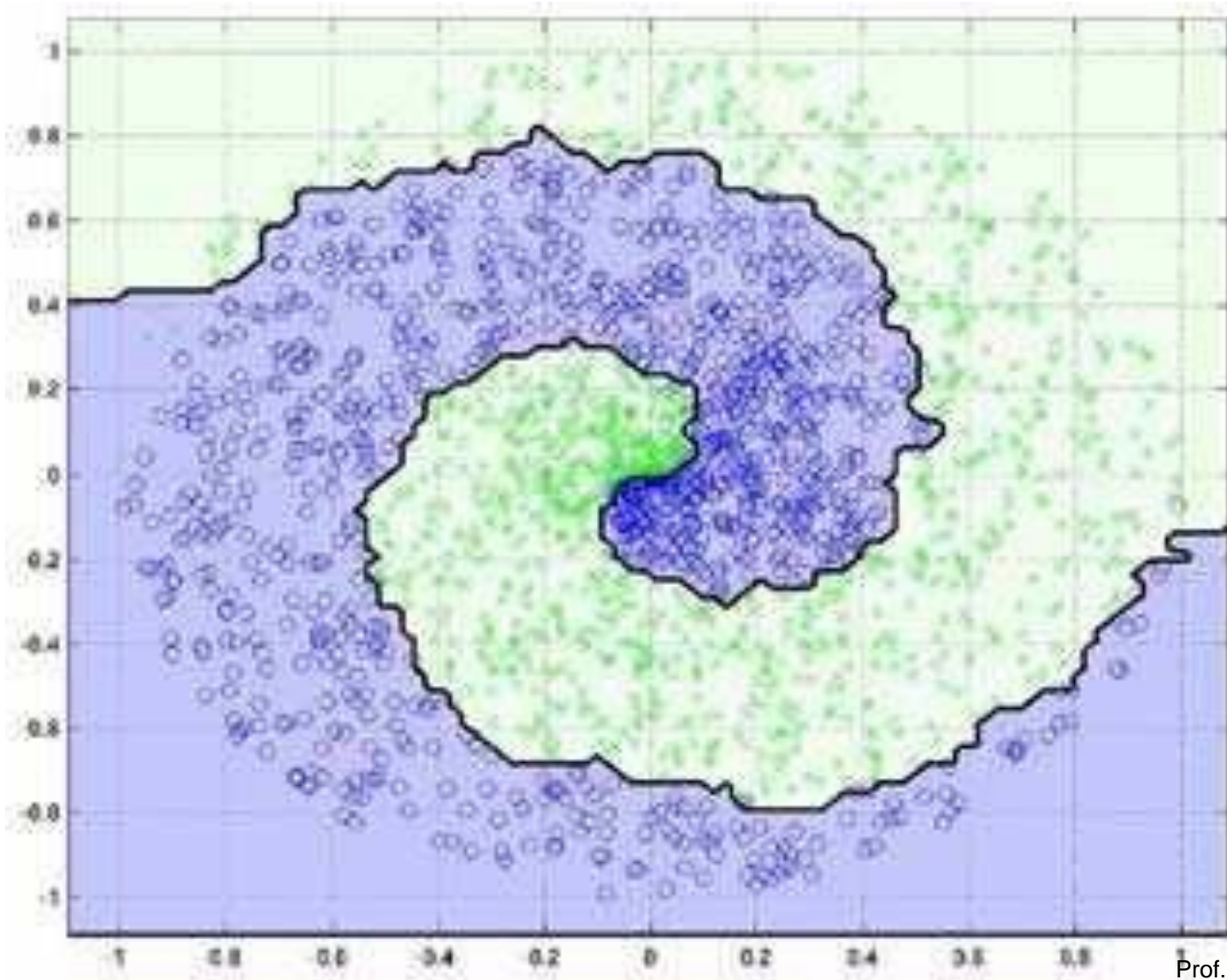


# Fronteiras, $K=1$



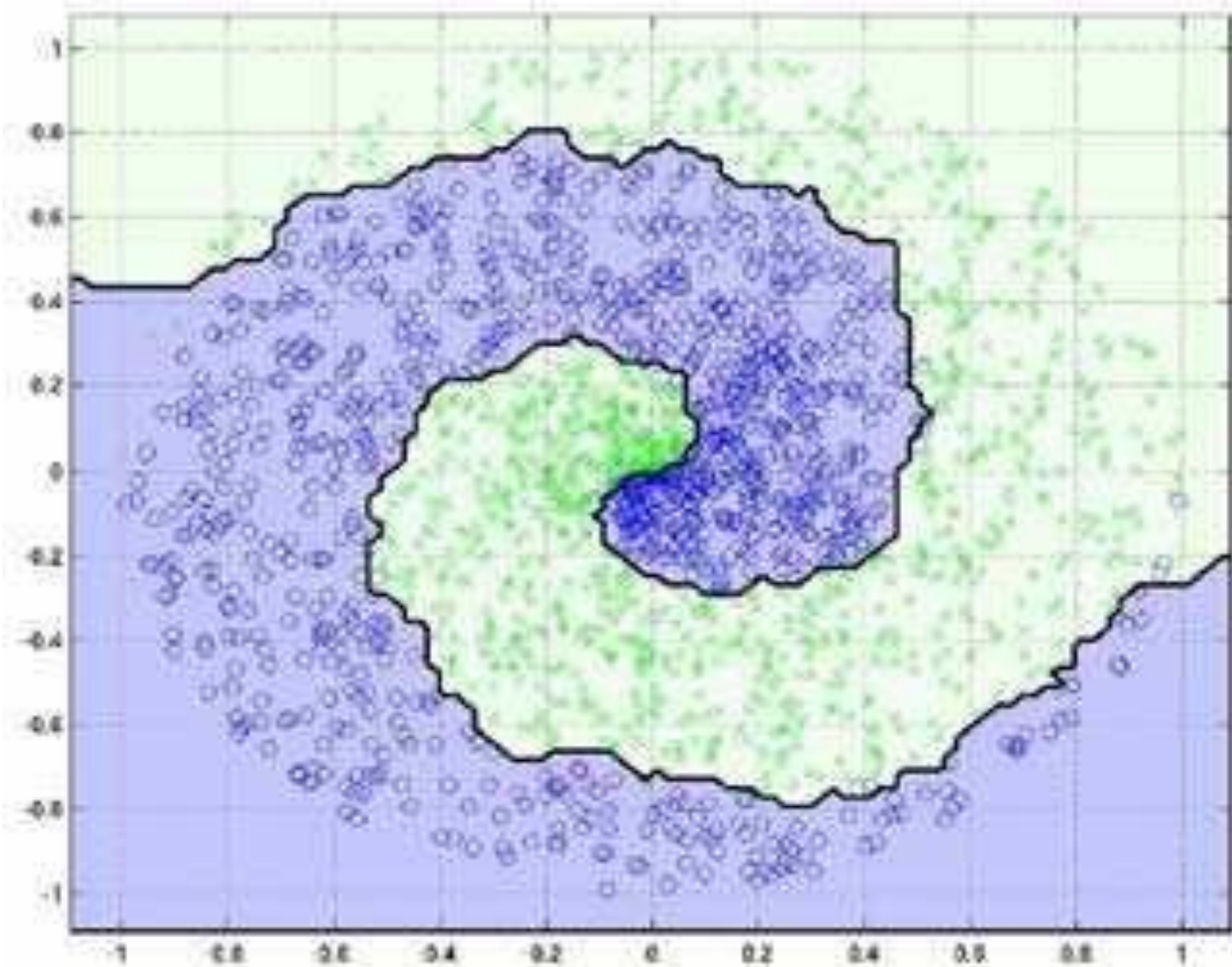


# Fronteiras, K=5

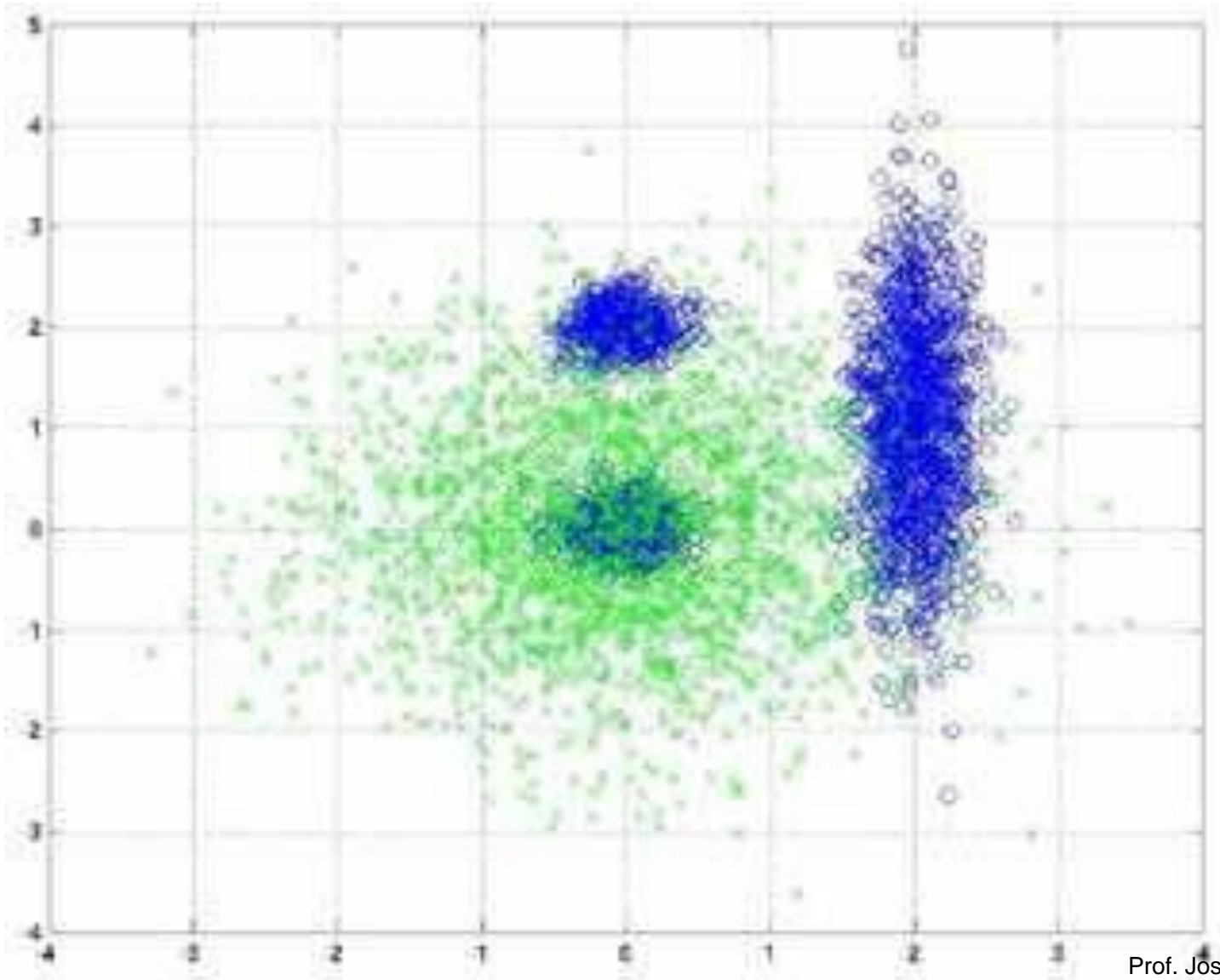




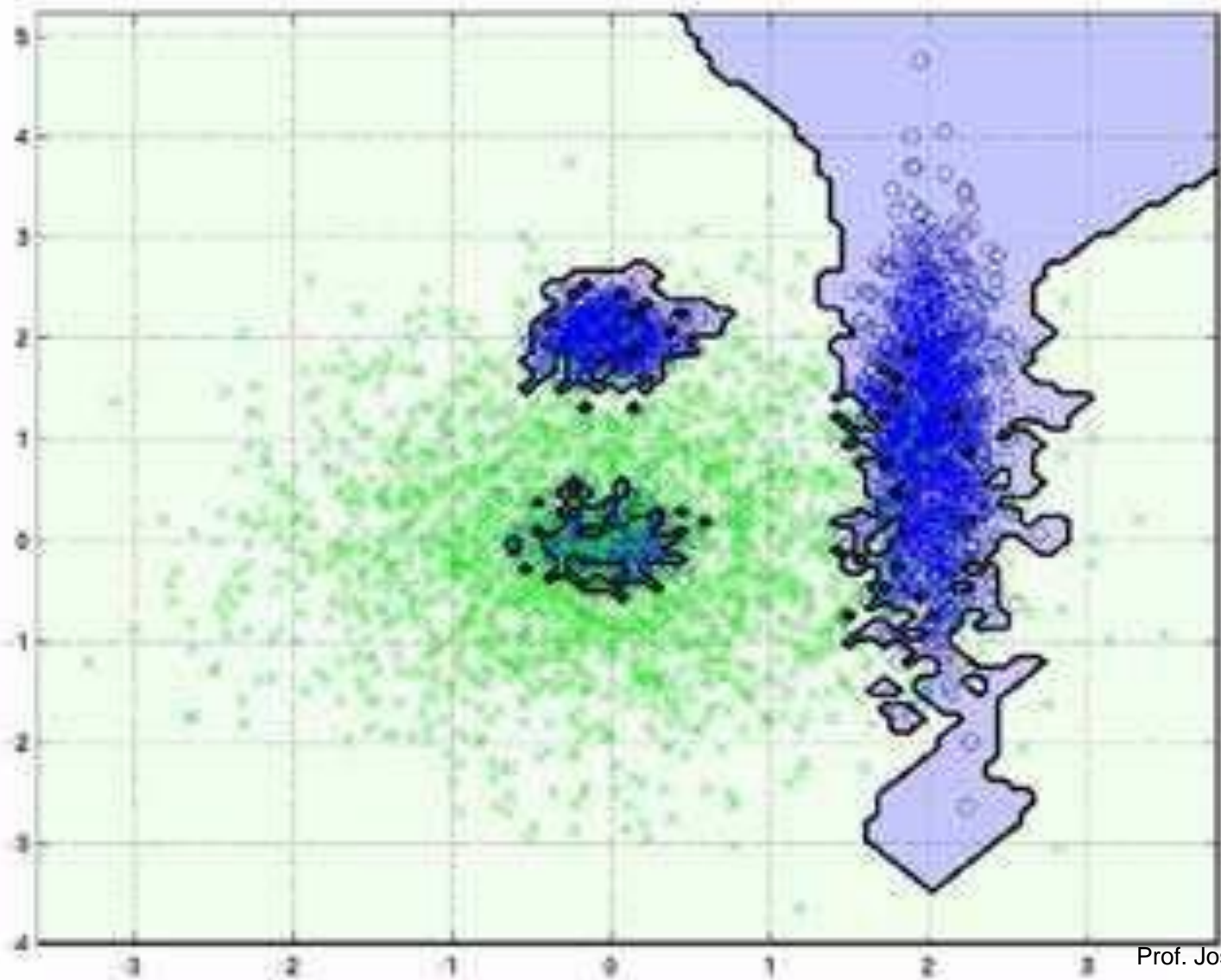
# Fronteiras, K=10



# Fronteiras

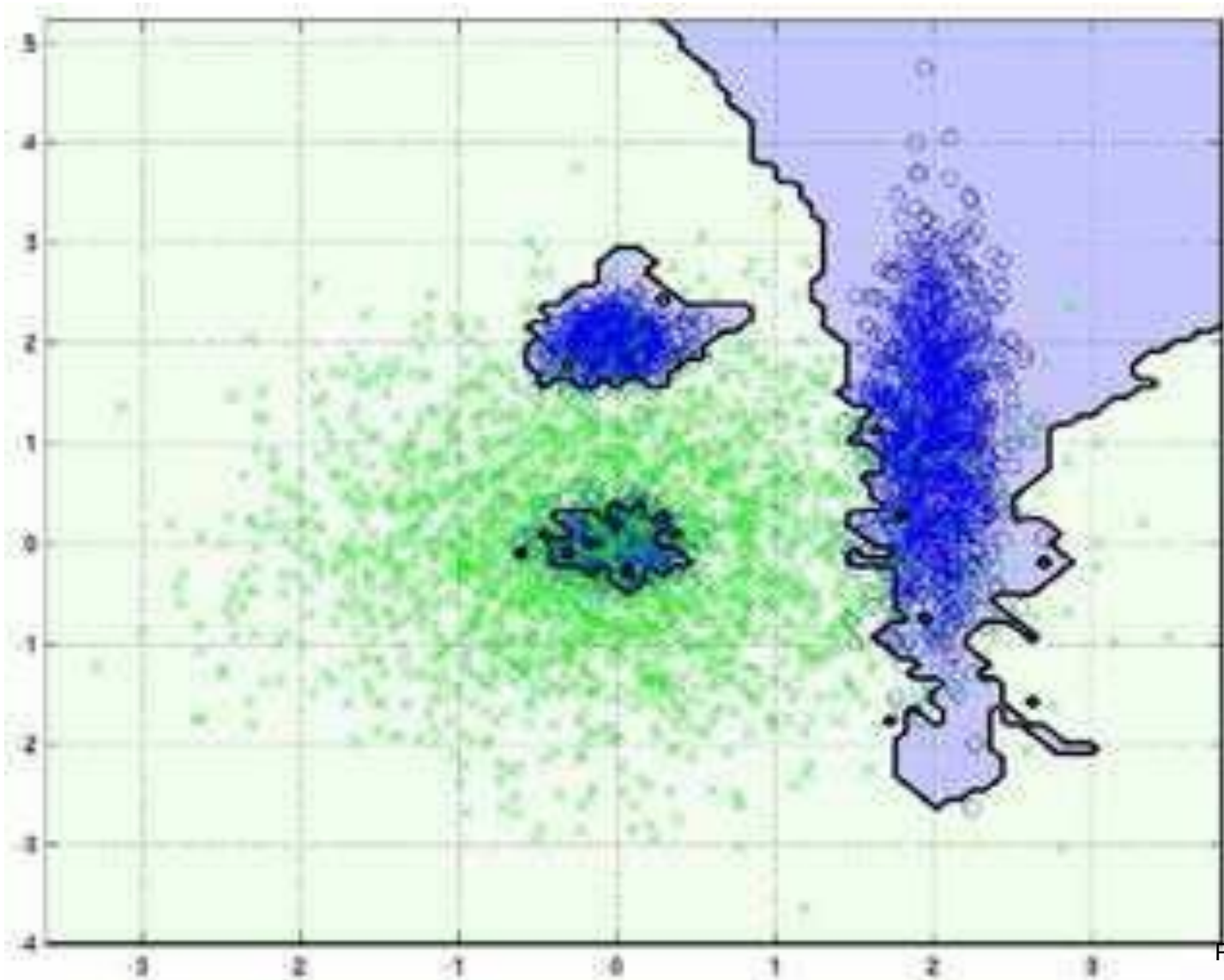


# Fronteiras, $K=1$

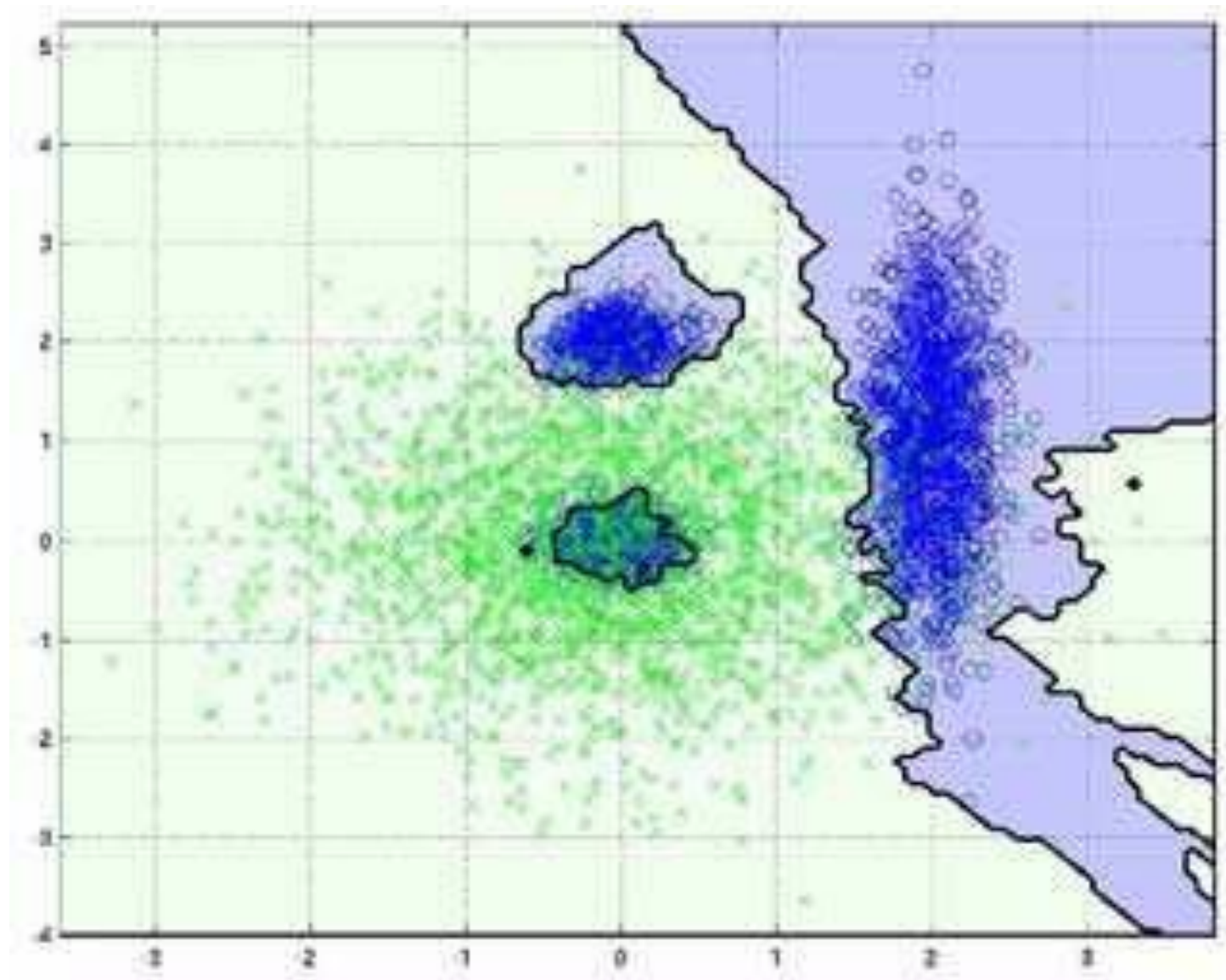




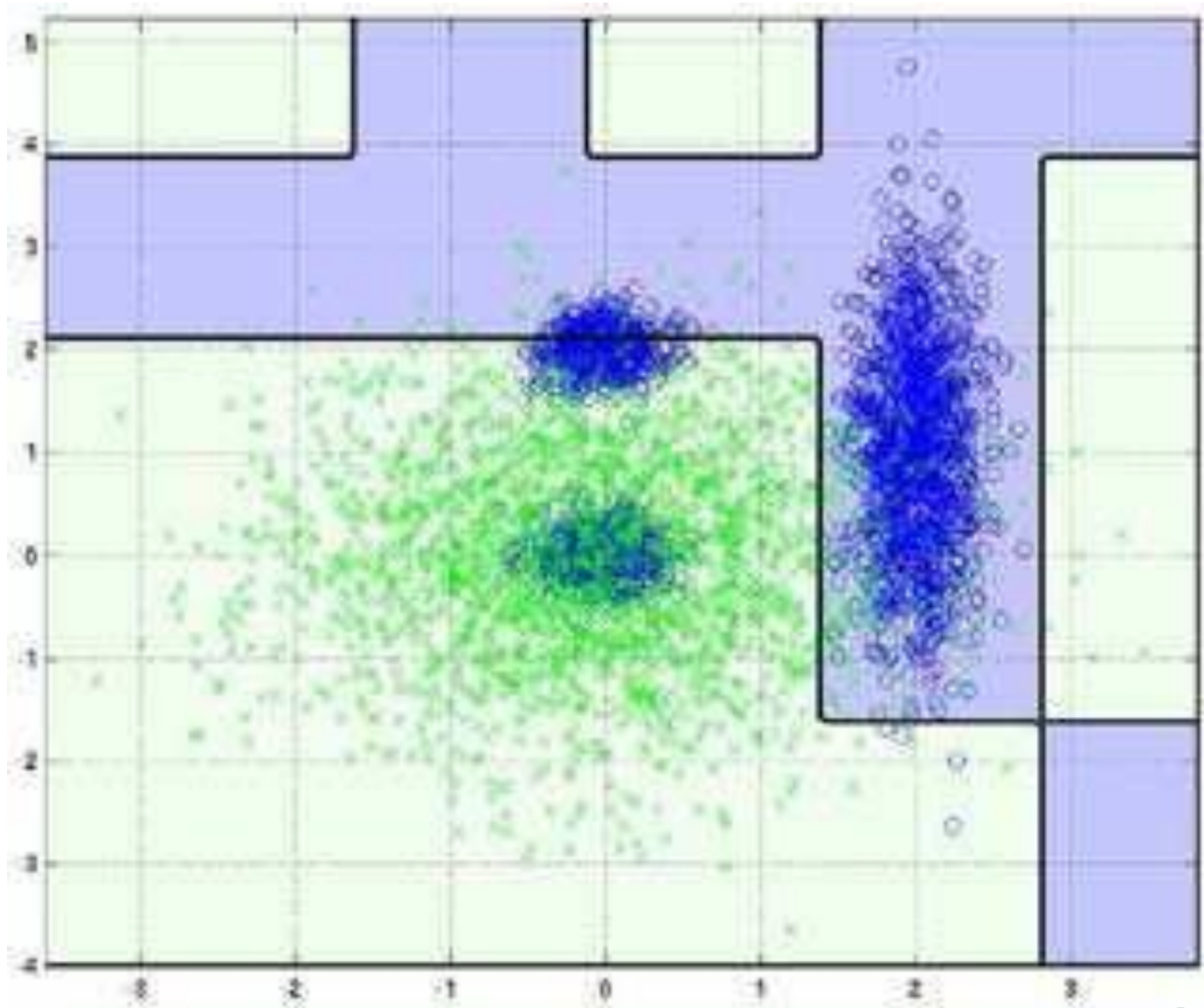
# Fronteiras, K=5



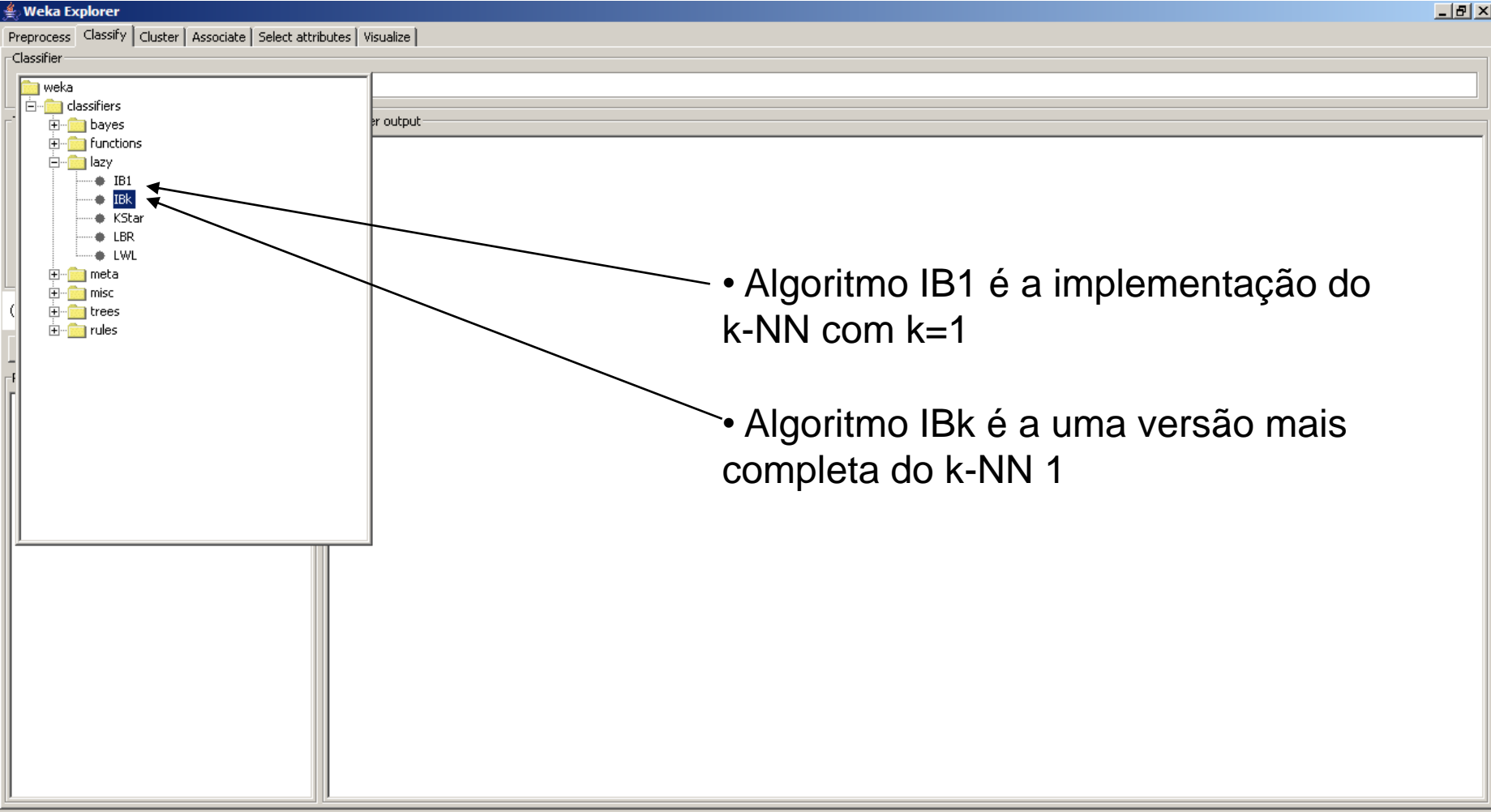
# Fronteiras, K=10



# Fronteiras (Árvore de Decisão)



# Algoritmo k-NN no WEKA

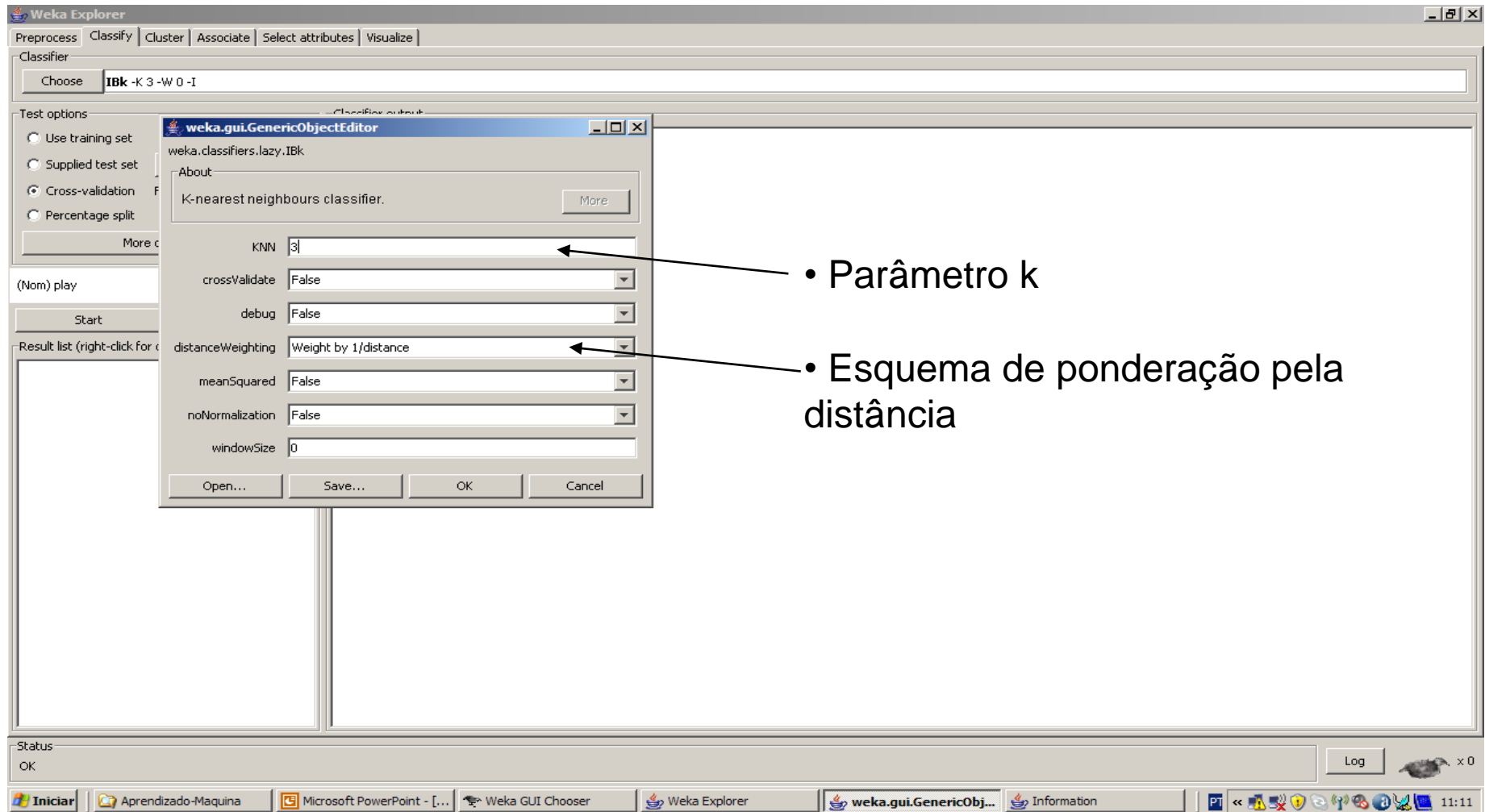


The screenshot shows the Weka Explorer application window. The 'Classifier' tab is selected, and the 'weka' tree structure is expanded. The 'lazy' folder is also expanded, showing a list of algorithms: IB1, IBk, KStar, LBR, and LWL. Two arrows originate from the right side of the image and point to the 'IB1' and 'IBk' entries in the list.

- Algoritmo IB1 é a implementação do k-NN com  $k=1$
- Algoritmo IBk é a uma versão mais completa do k-NN 1

The status bar at the bottom shows 'Status OK' and a 'Log' button. The taskbar at the very bottom includes icons for 'Iniciar', 'Aprendizado-Maquina', 'Microsoft PowerPoint - [I...', 'Weka GUI Chooser', and 'Weka Explorer'. The system clock in the bottom right corner shows '11:04'.

# Algoritmo k-NN no WEKA





# Vantagens

- São algoritmos de aprendizado supervisionado nos quais instâncias são usadas (incrementalmente) para classificar objetos
- Custo de atualização das instâncias é baixo
- Custo ou velocidade do aprendizado é (treinamento) é baixo
- É possível adaptar os algoritmos para obter descrição dos conceitos

# Desvantagens

- São computacionalmente custosos, desde que considerem todas as instâncias
- Não tratam bem atributos ruidosos
- Não são robustos a atributos irrelevantes
- A performance depende muito da escolha da função de similaridade para computer as distâncias
- Não existe forma simples de tratar atributos nominais e faltantes
- Não fornecem uma boa descrição da estrutura dos dados

# Exemplo

- Applet

[http://techlab.bu.edu/classer/artmap\\_applet](http://techlab.bu.edu/classer/artmap_applet)

# Referências

- T. Mitchell, 1997. *Machine Learning*.
- I. Witten, E. Frank, 2000. *Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations*.
- D. Aha, D. Kibler, M. Albert, ,1991. Instance-based learning algorithms. *Machine Learning*, 6:37--66.