Aprendizado de Máquina

Classificação

Eduardo R. Hruschka

Agenda

- Conceitos preliminares e classificador 1R (aquecimento)
- Classificadores Bayesianos
- Avaliação de classificadores
- *k-Nearest Neighbors* (k-NN)

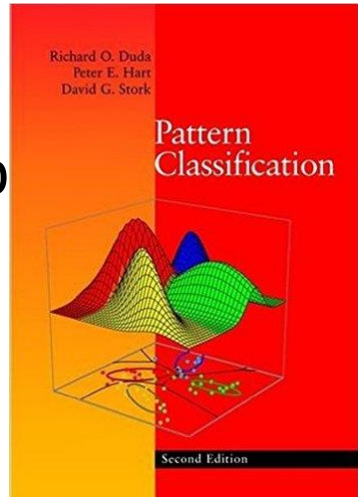
Conceitos preliminares

- Tarefa: dado um conjunto de exemplos pré-classificados/rotulados, induzir um modelo/classificador para novos casos.
- Aprendizado supervisionado: classes são conhecidas para os exemplos usados para construir o modelo/classificador.
- Um classificador pode ser um modelo de regressão logística, um conjunto de regras lógicas, uma árvore de decisão, um modelo Bayesiano, uma rede neural etc.
- Aplicações típicas: aprovação de crédito, marketing direto, detecção de fraude etc.

- Algoritmos simples frequentemente funcionam muito bem na prática. Além disso:
  - Menor tempo de construção do modelo;
  - Combinação (*ensembles*) de algoritmos simples;
  - *Baselines*.

- Sugestão:
  - Usar um único atributo (melhor discriminador – 1R);
  - Usar todos os atributos, assumindo independência condicional;
  - Árvores de Decisão (interpretabilidade) e *Random Forests*;
  - Regressão Logística (LASSO);
  - K-NN;
  - Ensembles, SVMs e redes neurais.

- Sucesso de cada algoritmo depende do domínio de aplicação: ver *No Free Lunch Theorems*.



Outlook	Temperature	Humidity	Windy	**Play**
sunny	85	85	false	**no**
sunny	80	90	true	**no**
overcast	83	86	false	**yes**
rainy	70	96	false	**yes**
rainy	68	80	false	**yes**
rainy	65	70	true	**no**
overcast	64	65	true	**yes**
sunny	72	95	false	**no**
sunny	69	70	false	**yes**
rainy	75	80	false	**yes**
sunny	75	70	true	**yes**
overcast	72	90	true	**yes**
overcast	81	75	false	**yes**
rainy	71	91	true	**no**
rainy	63	84	true	**?**

Weather Data* : Considerando-se dados históricos, construir um modelo para os valores do atributo meta *play.*

* Witten & Frank, 2000

Alternativas de modelagem

- Encontrar uma função discriminadora f(**x**) que mapeia **x** em um rótulo de classe. Ex: f(**x**)=0 para C_1 e f(**x**)=1 para C_2.
- Modelar a distribuição de probabilidades *a posteriori* $P(C_k|\mathbf{x})$ diretamente, usando modelos discriminativos.
- Inicialmente encontrar as densidades condicionais de classe, $P(\mathbf{x}|C_k)$, bem como $P(C_k)$, individualmente para cada classe, e depois usar o Teorema de Bayes:

$$P(C_k|\mathbf{x}) = \frac{P(\mathbf{x}|C_k)P(C_k)}{P(\mathbf{x})}$$

- Equivalente a encontrar $P(\mathbf{x},C_k)$ – modelos geradores.

Modelos geradores:

- Abordagem computacionalmente pesada e, se **x** possui alta dimensionalidade, precisaremos de grandes amostras;
- Permite estimar a densidade marginal dos dados, P(**x**), que é útil para detectar novos dados que possuem baixa probabilidade dado o modelo (*outlier detection, novelty detection*).

Modelos discriminativos:

- Particularmente interessante se estamos interessados apenas em $P(C_k \mid \mathbf{x})$, e não em $P(\mathbf{x}, C_k)$.

Função discriminadora:

- Alternativa mais simples, mas que causa perda considerável de informação (*e.g.*, *reject option*, combinação de modelos etc.)

Aquecimento: 1R (função discriminadora)

Aprende uma árvore de decisão de um nível.

- Todas as regras usam somente um atributo.

Versão Básica:

- Um ramo para cada valor do atributo;
- Para cada ramo, atribuir a classe mais frequente;
- Taxa de erro de classificação: proporção de exemplos que não pertencem à classe majoritária do ramo correspondente;
- Escolher o atributo com a menor taxa de erro de classificação;

- Aplicação imediata para atributos nominais/categóricos/binários;

- Para atributos ordinais/contínuos há vários algoritmos de discretização para definir estratégias de corte nos valores dos atributos (<=, <, >, >=).

Algoritmo 1R

Para cada atributo:

Para cada valor do atributo gerar uma regra:

Contar a frequência de cada classe;

Encontrar a classe mais frequente;

Formar uma regra que atribui à classe mais frequente este atributo-valor;

Calcular a taxa de erro de classificação das regras;

Escolher as regras com a menor taxa de erro de classificação.

- Quantas vezes precisamos varrer a base de dados?
- Complexidade computacional?

Exemplo para a base *Weather*

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Atributo	Regras	Erros	Total erros
Outlook	Sunny → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	
Temp	Hot → No*	2/4	5/14
	Mild → Yes	2/6	
	Cool → Yes	1/4	
Humidity	High → No	3/7	4/14
	Normal → Yes	1/7	
Windy	False → Yes	2/8	5/14
	True → No*	3/6	

* empate

Qual seria a capacidade de generalização do modelo?

- 1R foi descrito por Holte (1993):
  - Avaliação experimental em 16 bases de dados;
  - Em muitos *benchmarks*, regras simples não são muito piores do que árvores de decisão mais complexas.
- Fácil implementação;
- Muito usado para análise exploratória de dados;
- Árvores de Decisão estendem essa ideia;
- Outro algoritmo eficaz e eficiente: Naive Bayes.

Holte, Robert C., Very Simple Classification Rules Perform Well on Most Commonly Used Datasets, *Machine Learning* 11 (1), pp. 63-90, 1993.

Classificador Bayesiano

- Contrariamente ao 1R, o Naive Bayes (NB) usa todos os atributos.

- Presume que os atributos são igualmente importantes e condicionalmente independentes.
  - Valor de um atributo não influencia no valor de outro atributo, dada a informação da classe;

- Na prática, tais premissas são frequentemente violadas, mas ainda assim o NB é muito competitivo:
  - Probabilidades estimadas não precisam necessariamente ser corretas, o que importa são as avaliações relativas.

- Parece haver consenso que, na prática, deve ser o primeiro algoritmo a testar.

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Desejamos estimar:

$$P(C_k \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid C_k)P(C_k)}{P(\mathbf{x})}$$

- $P(C_k)$ pode ser estimada a partir da frequência relativa das classes;

- $P(\mathbf{x})$ é a *constante de normalização*:

$$P(\mathbf{x}) = \sum_k P(\mathbf{x} \mid C_k)P(C_k)$$

$\rightarrow$ Como estimar $P(\mathbf{x} \mid C_k)$?

$\rightarrow$ Presumindo independência condicional temos:

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	**Hot**	2	2	**High**	3	4	**False**	6	2	9	5
Overcast	4	0	**Mild**	4	2	**Normal**	6	1	**True**	3	3		
Rainy	3	2	**Cool**	3	1								
Sunny	2/9	3/5	**Hot**	2/9	2/5	**High**	3/9	4/5	**False**	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	**Mild**	4/9	2/5	**Normal**	6/9	1/5	**True**	3/9	3/5		
Rainy	3/9	2/5	**Cool**	3/9	1/5								

Para um novo exemplo:

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

Verosssimilhança para as duas classes:

Para "yes" = 2/9 × 3/9 × 3/9 × 3/9 × 9/14 = 0.0053

Para "no" = 3/5 × 1/5 × 4/5 × 3/5 × 5/14 = 0.0206

Convertendo para probabilidades por meio de normalização:

P("yes") = 0.0053 / (0.0053 + 0.0206) = 0.205

P("no") = 0.0206 / (0.0053 + 0.0206) = 0.795

Exemplo elaborado por Witten & Frank

Mergulhando no detalhe

Probabilidade de um evento *H* dada a evidência *E*:

$$P[H \mid E] = \frac{P[E \mid H]P[H]}{P[E]}$$

Probabilidade *a priori* para H, P[H]:

- Probabilidade de um evento antes de verificar a evidência.

Probabilidade *a posteriori* para P[H|E]:

Probabilidade de um evento após verificar a evidência.

$$P[H \mid E] = \frac{P[E_1 \mid H]P[E_2 \mid H]\ldots P[E_n \mid H]P[H]}{P[E]}$$

Recapitulando

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

← ***Evidência E***

$$P[yes \mid E] = P[Outlook = Sunny \mid yes]$$
$$\times P[Temperature = Cool \mid yes]$$
$$\times P[Humidity = High \mid yes]$$
$$\times P[Windy = True \mid yes]$$
$$\times \frac{P[yes]}{P[E]}$$

Probabilidade da classe "yes"

$$= \frac{\frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14}}{P[E]}$$

Problemas?

- O que acontece se um determinado valor de atributo não aparece na base de treinamento, mas aparece no exemplo de teste? (e.g. “outlook=overcast” para classe “no”)
  - Probabilidade correspondente será zero.
  - Probabilidade *a posteriori* será também zero.

- Possível solução: usar correção/estimador de *Laplace.*

➢ Como resultado, as probabilidades nunca serão *zero.*

- No caso geral, pode-se adicionar uma constante μ.

➢ Exemplo: atributo *outlook* para a classe *yes:*

$$\frac{2+\mu/3}{9+\mu} \qquad \frac{4+\mu/3}{9+\mu} \qquad \frac{3+\mu/3}{9+\mu}$$

Sunny ***Overcast*** ***Rainy***

Problemas?

- Valor ausente no treinamento: excluir exemplo da base;
- Valor ausente na classificação: omitir atributo com valor ausente do cálculo. Exemplo:

Outlook	Temp.	Humidity	Windy	Play
?	Cool	High	True	?

Verossimilhança para "yes" = $3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0238$

Verossimilhança para "no" = $1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0343$

Chance ("yes") = 0.0238 / (0.0238 + 0.0343) = 41%

Chance ("no") = 0.0343 / (0.0238 + 0.0343) = 59%

Atributos contínuos

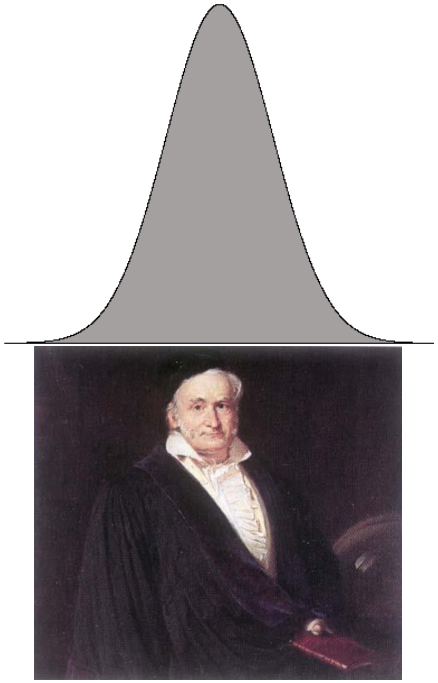
- Por exemplo, pode-se presumir uma distribuição Gaussiana para estimar as probabilidades:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\sigma = \frac{1}{n-1}\sum_{i=1}^{n} (x_i - \mu)^2$$

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- Misturas de Gaussianas (*clustering*)
- Discretização



Karl Gauss
1777-1855

Outlook			Temperature		Humidity		Windy			Play	
	Yes	*No*	*Yes*	*No*	*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	64, 68,	65, 71,	65, 70,	70, 85,	False	6	2	9	5
Overcast	4	0	69, 70,	72, 80,	70, 75,	90, 91,	True	3	3		
Rainy	3	2	72, …	85, …	80, …	95, …					
Sunny	2/9	3/5	μ =73	μ =75	μ =79	μ =86	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	σ =6.2	σ =7.9	σ =10.2	σ =9.7	True	3/9	3/5		
Rainy	3/9	2/5									

- Valor de densidade:

$$f(temperature = 66 \mid yes) = \frac{1}{\sqrt{2\pi}6.2} e^{-\frac{(66-73)^2}{2*6.2^2}} = 0.0340$$

Naive Bayes - Discussão

- Naïve Bayes funciona bem mesmo quando suas premissas são violadas;
- Classificação não requer estimativas precisas da probabilidade, desde que a máxima seja atribuída à classe correta*;
- Entretanto, a existência de muitos atributos redundantes pode causar problemas → selecionar melhores atributos;
- Muitos atributos numéricos não seguem uma distribuição *Gaussiana* (→ GMM, *kernel density estimators* etc.);
- Complexidade computacional & paralelização;
- Redes Bayesianas.

* Domingos & Pazzani, On the Optimality of the Simple Bayesian Classifier under Zero-One Loss, Machine Learning 29, 103-130, 1997.

- Atributos irrelevantes e redundantes podem comprometer acurácia de classificação;
- Selecionar atributos com base no desempenho do classificador NB. Informalmente pode-se sumarizar o NBW como segue:

1) Construir um classificador NB para cada atributo X_i (i = 1,..., n). Escolher X_i para o qual o NB apresenta a melhor acurácia e inseri-lo em A_S = {atributos selecionados};
2) Para todo $X_i \notin A_S$ construir um NB formado por $\{X_i\} \cup A_S$. Escolher o melhor classificador dentre os disponíveis e verificar se é melhor do que o obtido anteriormente:
   a) SE sim, ENTÃO atualizar A_S, inserindo o atributo adicional e repetindo o passo 2);
   b) SE não, ENTÃO parar e usar o classificador obtido anteriormente.

- NB possui complexidade de tempo linear com o número de exemplos e de atributos;
- Constante de tempo do NB também é baixa (computar frequências relativas e/ou densidades);
- Algoritmo NB é facilmente paralelizável;
- O que dizer sobre o NBW?
  - Teoria: $O\ (2^n)$, onde n é o número de atributos;
  - Busca gulosa *poda* o espaço de busca do problema de otimização combinatória: $O\ (n + (n-1) + ... + 1) = O\ (n^2)$
  - Por exemplo, para n=100 temos: 1.2×10^{30} versus 10^4 avaliações de classificadores diferentes para escolher o melhor.

Agenda

- Conceitos preliminares e classificador 1R (aquecimento)
- Classificadores Bayesianos
- Avaliação de classificadores
- *k-Nearest Neighbors* (k-NN)

Avaliação de classificadores

- Considerando uma medida de avaliação (acurácia, precisão, gini etc.), o procedimento padrão é fazer validação cruzada (*k-fold cross-validation*):

- *Leave-one out* (LOO):

- **Custo computacional** da validação cruzada com *k* pastas:
  - k treinamentos do classificador em N(k-1)/k exemplos.
  - k validações em n/k exemplos.
- **Output**: média das avaliações obtidas nas k validações.

❖ Qual **classificador** uso em produção, para classificar dados não vistos na base de treinamento?

- Classificador induzido com a maior quantidade de dados disponível (e.g., para NB toda a base de treinamento).
- k classificadores (ensemble).

Nota: não há consenso sobre o uso dos termos **teste** e **validação** (ambos podem significar a mesma coisa).

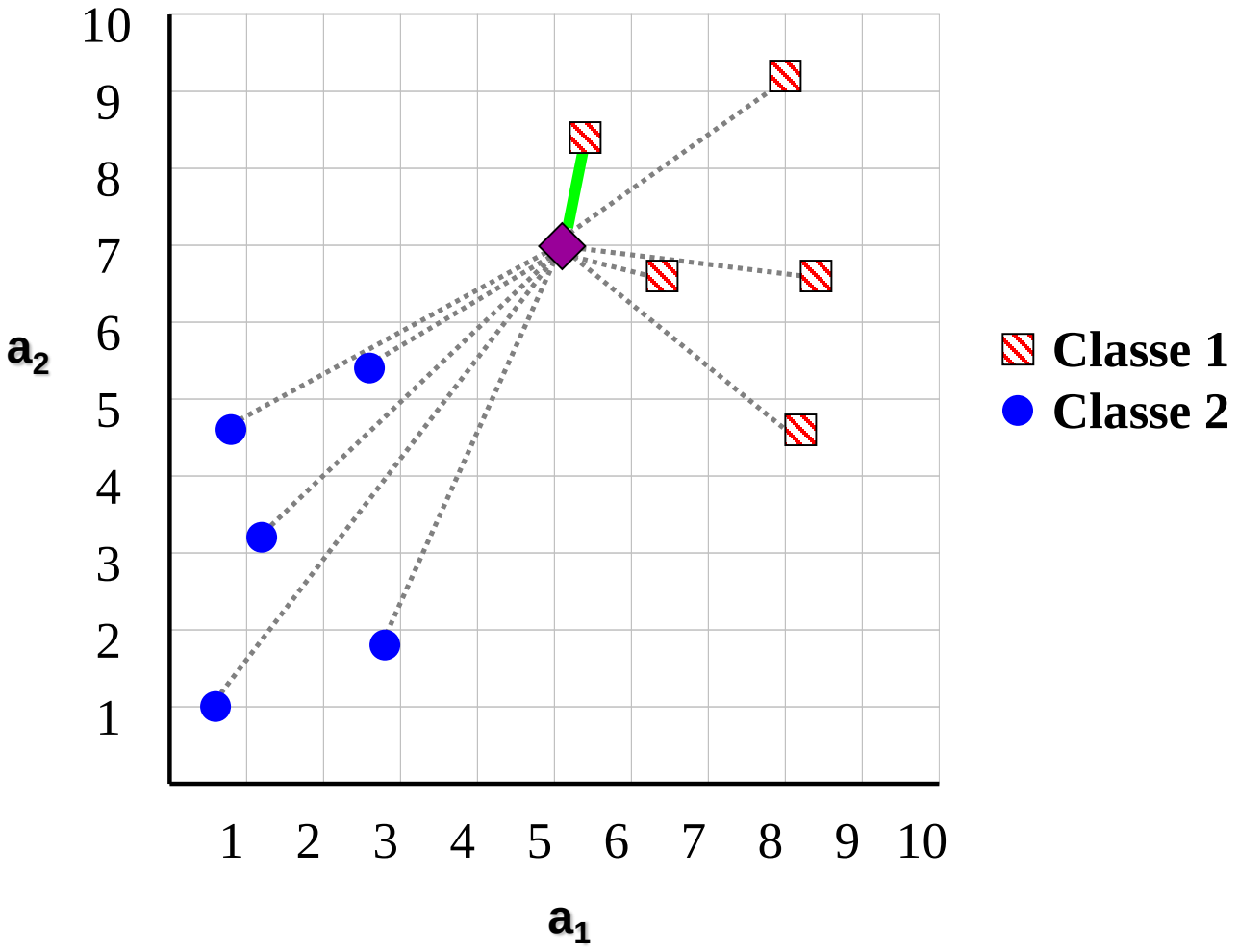
Agenda

- Conceitos preliminares e classificador 1R (aquecimento)
- Classificadores Bayesianos
- Avaliação de classificadores
- *k-Nearest Neighbors* (k-NN)

Lazy algorithms

- Não constroem descrições gerais e explícitas (função alvo) a partir dos exemplos de treinamento;
- Generalização é *adiada* até o momento da classificação;
- Armazena-se uma base de exemplos (*instances*) que é usada para realizar a classificação de uma nova *query* (exemplo *não visto*);
- Inclui técnicas como KNN, CBR, métodos de regressão;
- Em muitos casos apresenta um alto custo computacional (por conta do cálculo de distâncias).

Noção Intuitiva



Fonte: Keogh, E. A Gentle Introduction to Machine Learning and Data Mining for the Database Community, SBBD 2003, Manaus.

Conceitos fundamentais

- Exemplos correspondem a pontos no $\Re^n$;
- Vizinhos definidos em função de uma medida de distância;
- Por exemplo, considerando-se dois vetores $\mathbf{x}=[x_1,x_2,...,x_n]$ e $\mathbf{y}=[y_1,y_2,...,y_n]$, a distância Euclidiana é:

$$d_E(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- A distância Euclidiana é uma medida de dissimilaridade. Como obter, a partir desta, uma medida de similaridade?

- *f:* $\Re^n \rightarrow V$, $V=\{v_1, v_2,..., v_s\}$ /* *s* classes */

Algoritmo básico:

Dado um exemplo $\mathbf{x}_q$ a ser classificado e considerando que $\{\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_k\}$ representam os *k* exemplos mais próximos de $\mathbf{x}_q$, retornar:

$$f(\mathbf{x}_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(\mathbf{x}_i)) \qquad \begin{cases} (a = b) \Rightarrow \delta(a,b) = 1 \\ (a \neq b) \Rightarrow \delta(a,b) = 0 \end{cases}$$

➢ **Classificação por meio da classe majoritária da vizinhança.**

K-NN para regressão

- $f: \Re^n \rightarrow \Re$

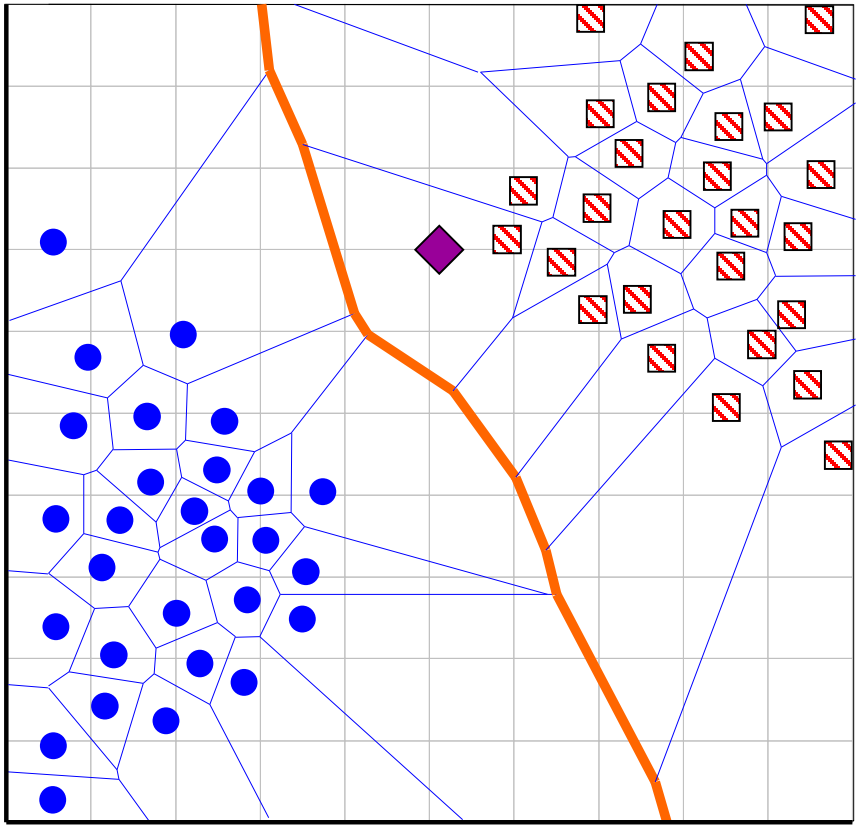
- Algoritmo:

  Dado um exemplo $\mathbf{x}_q$ cujo valor da variável dependente (y) se deseja estimar e considerando que $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ representam os *k* exemplos mais próximos de $\mathbf{x}_q$, retornar:

$$y = f(\mathbf{x}_q) = \frac{\sum_{i=1}^{k} f(\mathbf{x}_i)}{k}$$

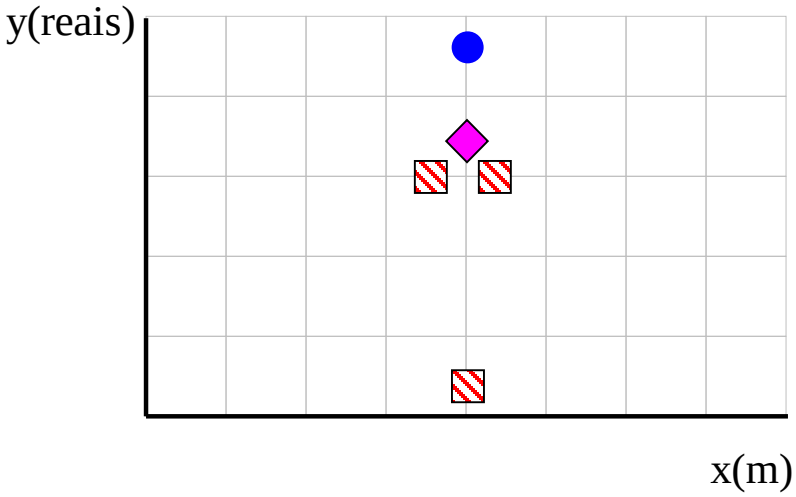
➢ **Predição por meio da média da vizinhança.**

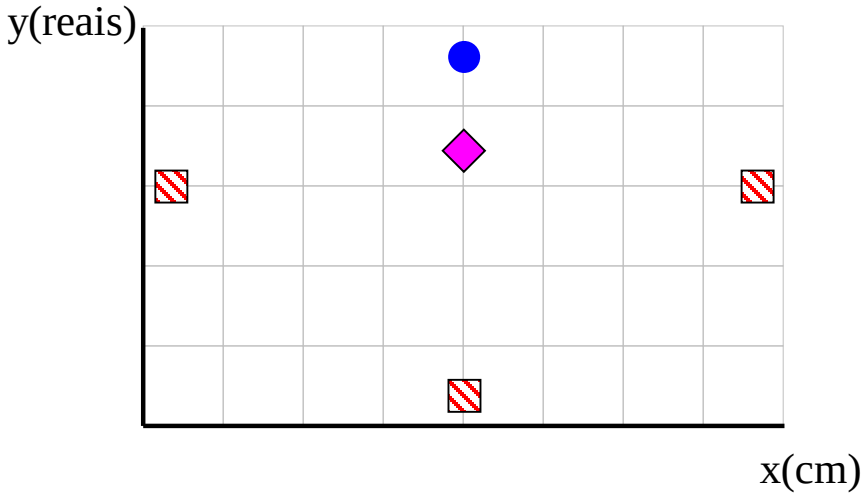
Superfície de decisão



Fonte: Keogh, E. A Gentle Introduction to Machine Learning and Data Mining for the Database Community, SBBD 2003, Manaus.

Sensibilidade em relação à escala





Como diminuir este problema?

- Normalizações (linear, escore-z)
- Transformações
- Testar via validação-cruzada

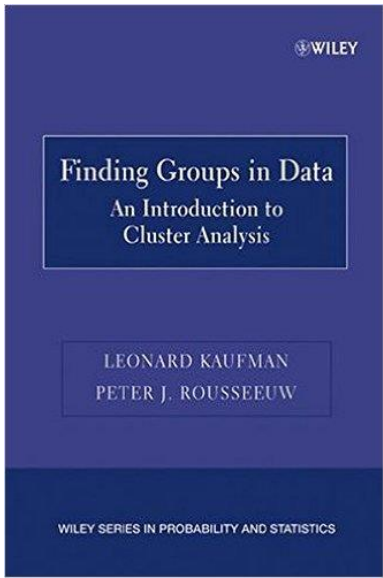
Fonte: Keogh, E. A Gentle Introduction to Machine Learning and Data Mining for the Database Community, SBBD 2003, Manaus.

Como lidar com atributos nominais?

Mudar a função de distância – e.g., usando coeficiente de casamento simples (*simple matching*):

$$d_{SM}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{i=n} s_i \qquad \begin{cases} (x_i = y_i) \Rightarrow s_i = 0; \\ (x_i \neq y_i) \Rightarrow s_i = 1; \end{cases}$$

- Há várias outras medidas de distância (e.g., ver Kaufman & Rousseeuw, Finding Groups in Data, 1990);
- Como lidar com bases de dados formadas por diferentes tipos de atributos (ordinais, contínuos, nominais, binários)?



Chega de democracia: ponderando os votos

Função alvo discreta:

$$\mathrm{f}(\mathbf{x}_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(\mathbf{x}_i))$$

$$\begin{cases} (a = b) \Rightarrow \delta(a,b) = 1 \\ (a \neq b) \Rightarrow \delta(a,b) = 0 \end{cases}$$

Função alvo contínua (regressão):

$$y = f(\mathbf{x}_q) = \frac{\sum_{i=1}^{k} w_i f(\mathbf{x}_i)}{\sum_{i=1}^{k} w_i}$$

Ponderação: $w_i = \frac{1}{d(\mathbf{x}_q, \mathbf{x}_i)}$

Exercício

Considere a seguinte base de dados:

Instância	a_1	a_2	a_3	Classe
1	0	250	36	A
2	10	150	34	B
3	2	90	10	A
4	6	78	8	B
5	4	20	1	A
6	1	170	70	B
7	8	160	41	A
8	10	180	38	B
9	6	200	45	?

Perguntas:

a) Qual é a função de distância a ser empregada?

b) Classificar o objeto #9 com *k*=1,2,3,4,5 (com e sem ponderação dos votos)

c) Como escolher *k*?

d) Descreva um algoritmo para otimizar os parâmetros do k-NN que leve em conta: dois tipos de normalização, pesos dos atributos entre [0,1] e número de vizinhos k em {1,2,...8}.