

Aprendizado de Máquina

Classificação

Eduardo R. Hruschka

Agenda

- Conceitos preliminares e classificador 1R (aquecimento)
- Classificadores Bayesianos
- Avaliação de classificadores
- *k-Nearest Neighbors* (k-NN)
- Árvores de Classificação
- Ensembles
- Random Forests
- Noções sobre SVM

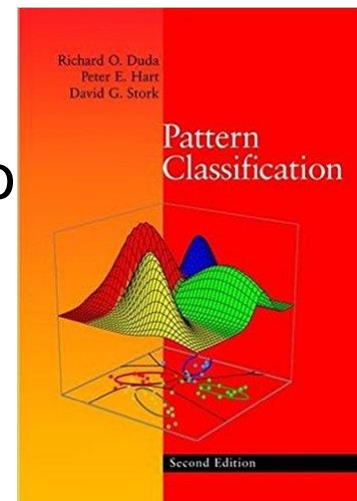
Conceitos preliminares

- 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*.

Alternativas de modelagem

- Encontrar uma função discriminadora $f(\mathbf{x})$ que mapeia \mathbf{x} em um rótulo de classe. Ex: $f(\mathbf{x})=0$ para C_1 e $f(\mathbf{x})=1$ para C_2 .
- Modelar a distribuição de probabilidades *a posteriori* $P(C_k | \mathbf{x})$ diretamente, usando modelos discriminadores.
- 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 \mathbf{x} possui alta dimensionalidade, precisaremos de grandes amostras;
- Permite estimar a densidade marginal dos dados, $P(\mathbf{x})$, 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 $P(C_k | \mathbf{x})$, 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 (\leq , $<$, $>$, \geq).

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.

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:

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

- $P(C_k)$ pode ser estimada a partir da frequência relativa das classes;
- $P(\mathbf{x})$ é a *constante de normalização*:

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

→ Como estimar $P(\mathbf{x} | C_k)$?

→ Presumindo independência condicional temos:

Outlook	Temperature		Humidity		Windy		Play						
	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	?

Verossimilhança para as duas classes:

$$\text{Para "yes"} = 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053$$

$$\text{Para "no"} = 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0206$$

Convertendo para probabilidades por meio de normalização:

$$P(\text{"yes"}) = 0.0053 / (0.0053 + 0.0206) = 0.205$$

$$P(\text{"no"}) = 0.0206 / (0.0053 + 0.0206) = 0.795$$

Mergulhando no detalhe

Probabilidade de um evento H dada a evidência E :

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



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.

$$P[H | E] = \frac{P[E_1 | H]P[E_2 | H] \dots P[E_n | H]P[H]}{P[E]}$$

Recapitulando

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

← *Evidência E*

Probabilidade da classe “yes”

$$P[\text{yes} | E] = P[\text{Outlook} = \text{Sunny} | \text{yes}]$$

$$\times P[\text{Temperature} = \text{Cool} | \text{yes}]$$

$$\times P[\text{Humidity} = \text{High} | \text{yes}]$$

$$\times P[\text{Windy} = \text{True} | \text{yes}]$$

$$\times \frac{P[\text{yes}]}{P[E]}$$

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

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*:

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

Sunny

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

Overcast

$$\frac{3 + \mu/3}{9 + \mu}$$

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

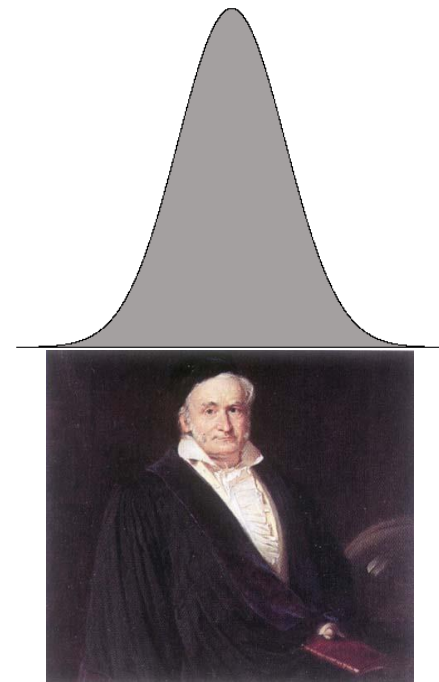
- Por exemplo, pode-se presumir uma distribuição Gaussiana para estimar as probabilidades:

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

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

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

- Misturas de Gaussianas (*clustering*)
- Discretização



Karl Gauss
1777-1855

Outlook			Temperature		Humidity		Windy			Play	
	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>No</i>	
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	$\mu = 73$	$\mu = 75$	$\mu = 79$	$\mu = 86$	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	$\sigma = 6.2$	$\sigma = 7.9$	$\sigma = 10.2$	$\sigma = 9.7$	True	3/9	3/5		
Rainy	3/9	2/5									

- Valor de densidade:

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

Naive Bayes - Discussão

- Naive 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 ($i = 1, \dots, n$). Escolher X_i para o qual o NB apresenta a melhor acurácia e inseri-lo em $A_S = \{\text{atributos selecionados}\}$;
 - 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 , inserir o atributo adicional e repetir 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: $O(n + (n-1) + \dots + 1) = O(n^2)$
 - 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)
- Árvores de Classificação
- Ensembles
- Random Forests
- Noções sobre SVM

Avaliação de classificadores

- Considerando uma medida de avaliação (acurácia, precisão 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** usado 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).

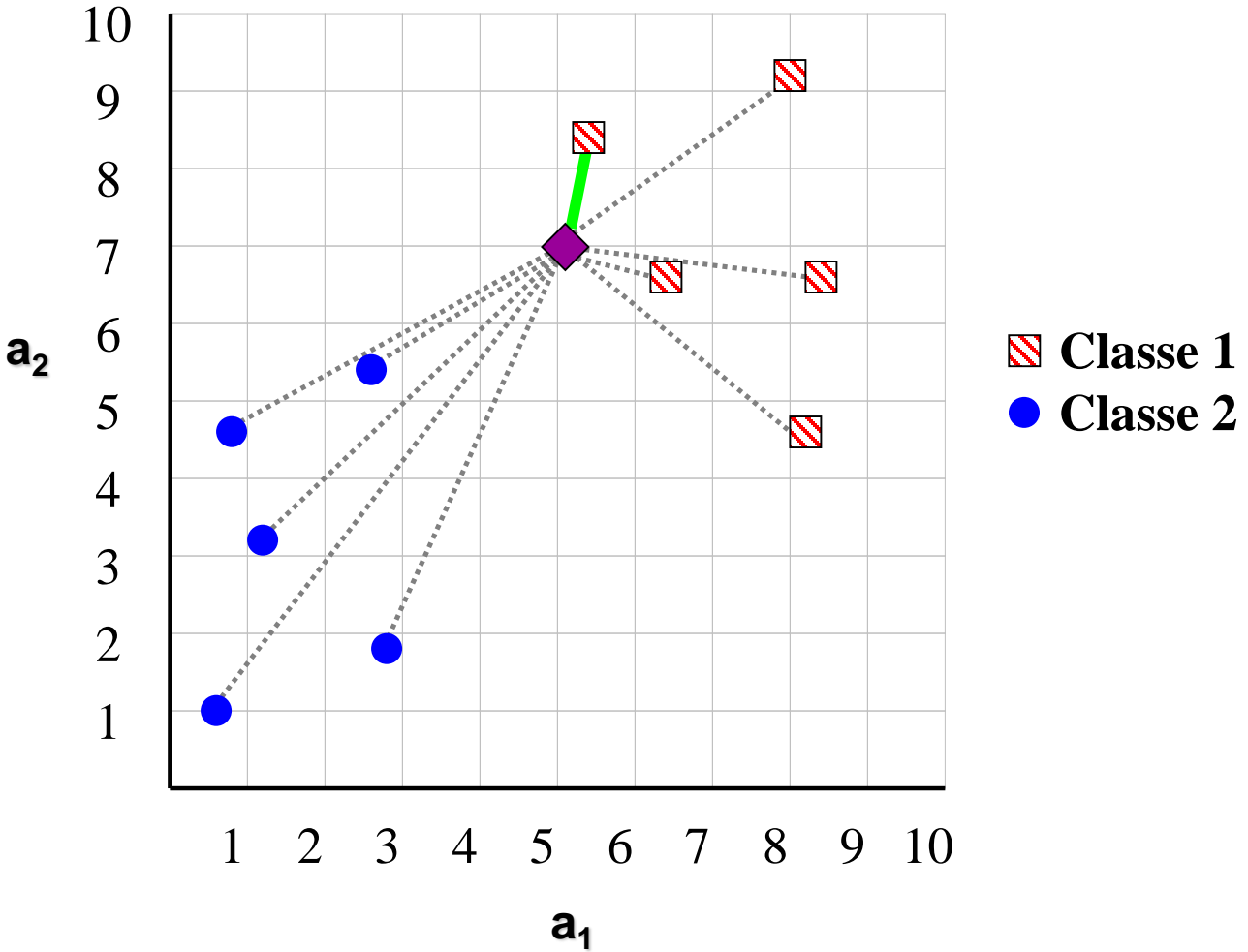
Agenda

- Conceitos preliminares e classificador 1R (aquecimento)
- Classificadores Bayesianos
- Avaliação de classificadores
- *k-Nearest Neighbors* (k-NN)
- Árvores de Classificação
- Ensembles
- Random Forests
- Noções sobre SVM

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, SBB D 2003, Manaus.

Conceitos fundamentais

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

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

Algoritmo básico:

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

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

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

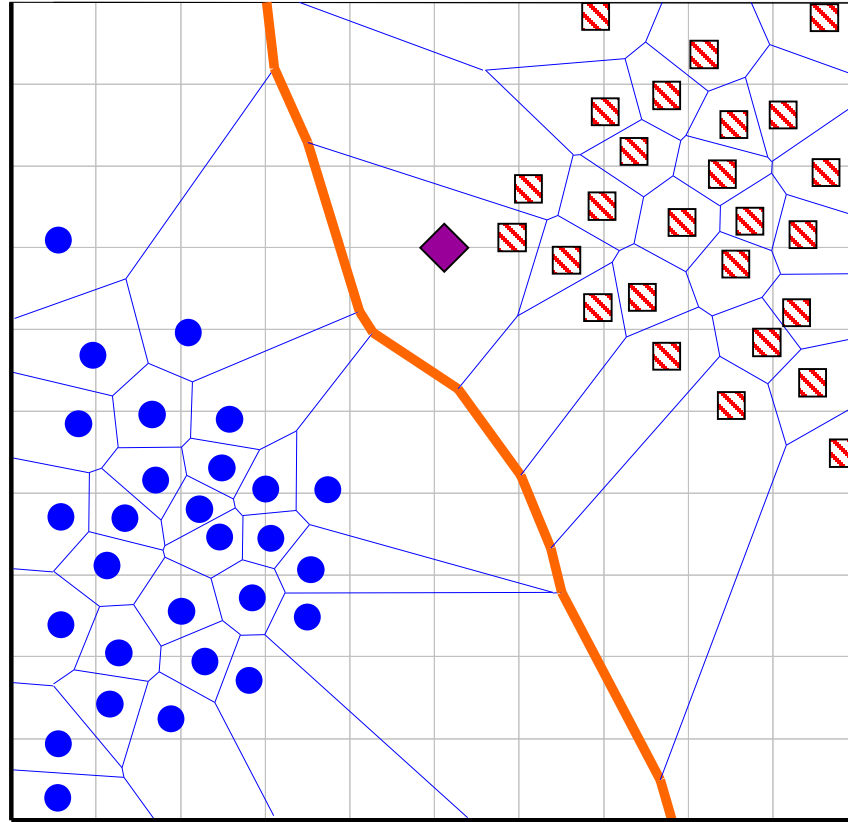
K-NN para regressão

- $f: \mathcal{R}^n \rightarrow \mathcal{R}$
- Algoritmo:
Dado um exemplo \mathbf{x}_q cujo valor da variável dependente (y) se deseja estimar e considerando que $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k$ representam os k exemplos mais próximos de \mathbf{x}_q , retornar:

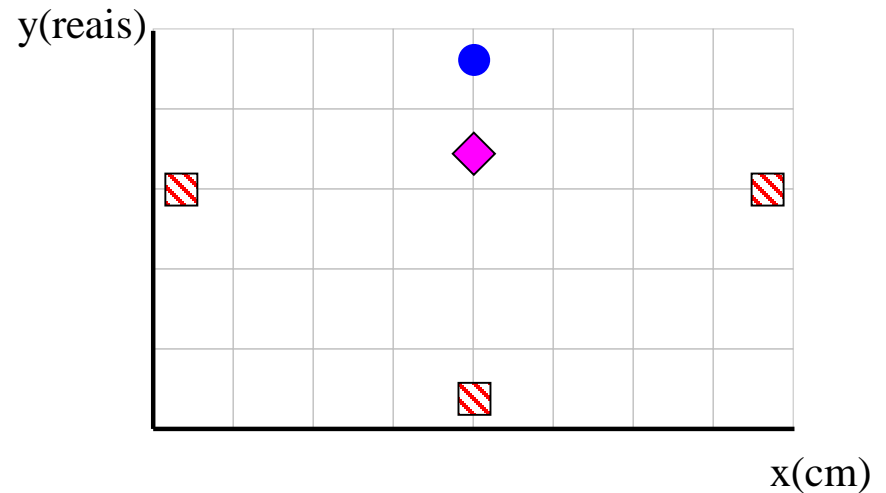
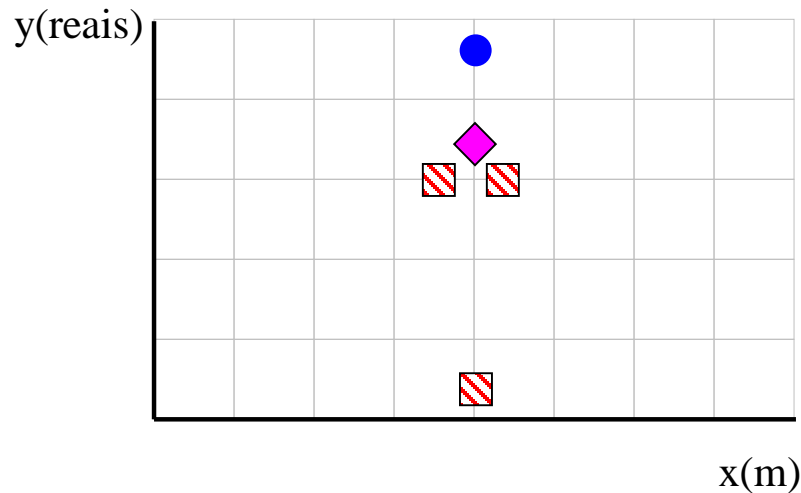
$$y = f(\mathbf{x}_q) = \frac{\sum_{i=1}^k f(\mathbf{x}_i)}{k}$$

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

Superfície de decisão



Sensibilidade em relação à escala



Como diminuir este problema?

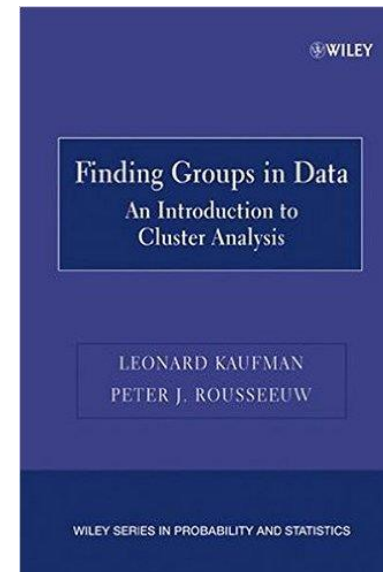
- Normalizações (linear, escore-z)
- Transformações
- Testar via validação-cruzada

Como lidar com atributos nominais?

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

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

- 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:

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

Função alvo contínua (regressão):

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

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

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:

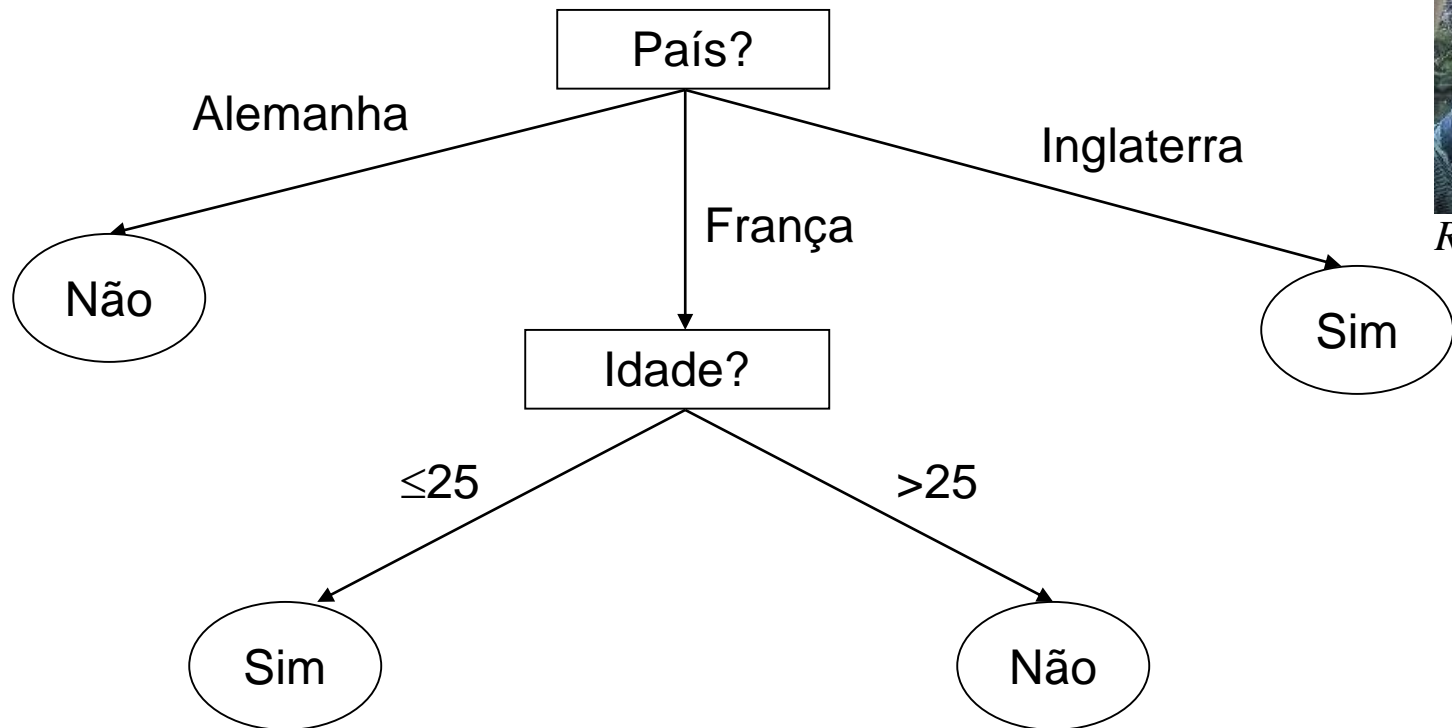
- Qual é a função de distância a ser empregada?
- Classificar o objeto #9 com $k=1,2,3,4,5$ (com e sem ponderação dos votos)
- Como escolher k ?
- Descreva um algoritmo para otimizar os parâmetros do k-NN que leve em conta: dois tipos de normalização, pesos dos atributos entre $[0,1]$ e número de vizinhos k em $\{1,2,\dots,8\}$.

Agenda

- Conceitos preliminares e classificador 1R (aquecimento)
- Classificadores Bayesianos
- Avaliação de classificadores
- *k-Nearest Neighbors* (k-NN)
- Árvores de Classificação
- Ensembles
- Random Forests
- Noções sobre SVM

Árvores de classificação

- Aproximam funções discretas, representadas por uma árvore de decisão;
- Permitem tradução via regras “se...então” → *interpretabilidade*;
 - Nós internos □: teste em atributos *previsores*;
 - Nós externos ○: valor previsto para o atributo meta;



Ross Quinlan

➤ Veremos conceitos do ID3 (Quinlan, 1986) e do C4.5 (Quinlan, 1993).

Ideia geral

1. Árvore é construída de maneira *top-down*, *recursivamente* e usando a ideia de *dividir para conquistar*;
2. Inicialmente, todos os exemplos de treinamento são *posicionados* na raiz da árvore;
3. Exemplos são *particionados* recursivamente com base em atributos selecionados, objetivando-se separar os exemplos por classes;
4. Condições de parada:
 - Todos os exemplos para um dado nó pertencem à mesma classe;
 - Não existem mais atributos para continuar o *particionamento*;
 - Todos os exemplos de treinamento estão classificados;

Ideia geral com medida intuitiva de pureza

- Como obter uma árvore de decisão para a seguinte base de dados?

SEXO	PAÍS	IDADE	COMPRAR
M	França	25	Sim
M	Inglaterra	21	Sim
F	França	23	Sim
F	Inglaterra	34	Sim
F	França	30	Não
M	Alemanha	21	Não
M	Alemanha	20	Não
F	Alemanha	18	Não
F	França	34	Não
M	França	55	Não

➤ Considerando que “COMPRAR” é nosso atributo-meta (classe) ...

Para cada atributo previsor (SEXO, PAÍS, IDADE), montar uma tabela:

- Linhas: atributo previsor (variável independente);
 - Colunas: atributo-meta (variável dependente);
 - Células: nº de tuplas para a combinação de valores atributo-classe.
- Qual atributo discrimina melhor: SEXO ou PAÍS?

Classe

SEXO	Sim	Não
M	2	3
F	2	3

Classe

PAÍS	Sim	Não
França	2	3
Inglaterra	2	0
Alemanha	0	3

Se SEXO=M então Classe=Não; Senão Classe=Sim. \Rightarrow Acurácia (A) = 50%.

➤ Mas *regra default* (atribuir sempre CLASSE=Não) retorna A=60%!

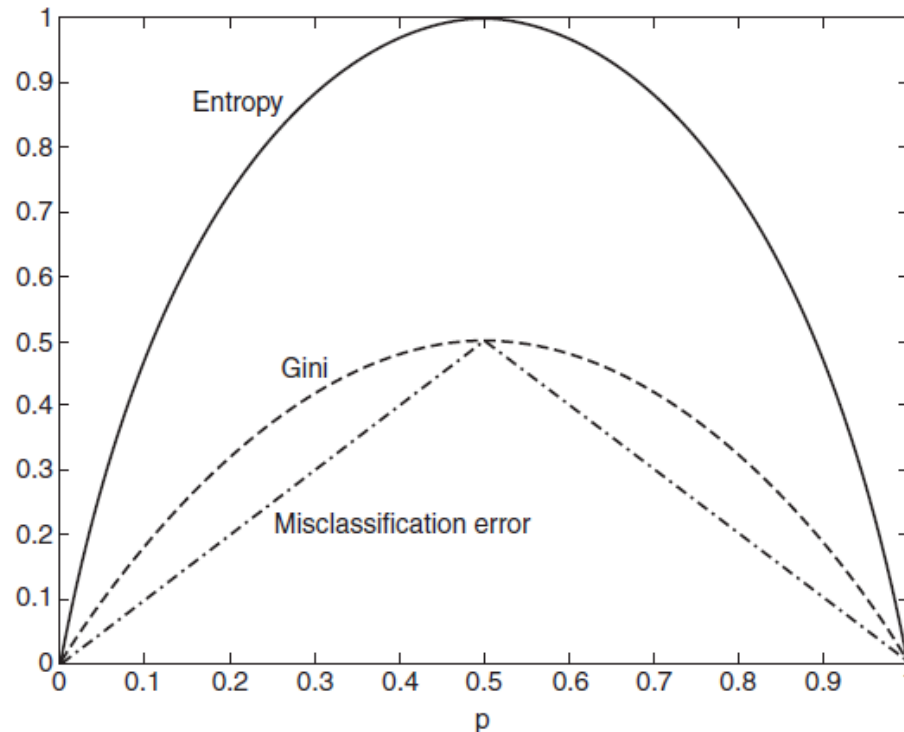
Se PAÍS=Inglaterra então Sim; Senão Não \Rightarrow A = 80%.

\Rightarrow O que dizer sobre o atributo IDADE? Como medir a informação?

Medindo impureza via entropia

- Permite medir a *informação* fornecida por cada atributo;
- Caracteriza a impureza de um conjunto de exemplos. Para um nó da árvore com p exemplos + e n exemplos - temos:

$$entropia = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}$$



Voltemos ao exemplo anterior...

Considerando o atributo meta “comprar” e 10 exemplos no nó raiz (4+,6-) :

- Entropia: **E = 0,97** sendo **P(+)=0,4** e **P(-)=0,6**.
- Para cada atributo teremos um valor de E. Para o atributo SEXO:

	Classe (COMPRAR)	
SEXO	Sim	Não
M	2	3
F	2	3

Ponderando o valor de E pelo número de exemplos que apresentam determinado valor para o atributo temos:

$$E(\text{SEXO}) = \frac{5}{10} (-\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5}) + \frac{5}{10} (-\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5})$$

E(SEXO) = 0,97.

Ganho de Informação (GI) = 0,97 – 0,97 = 0,00.

➤ Não há ganho de informação ao particionar com base no SEXO.

- Consideremos o atributo PAÍS:

PAÍS	COMPRAR		Total
	Sim (+)	Não (-)	
França	2	3	5
Inglaterra	2	0	2
Alemanha	0	3	3
Total	4	6	10

$$\begin{aligned}
 E(\text{PAÍS}) = & 5/10 \cdot \text{Info}(\text{França}) + 2/10 \cdot \text{Info}(\text{Inglaterra}) + 3/10 \cdot \text{Info}(\text{Alemanha}) = \\
 & 5/10 (-2/5 \log_2 2/5 - 3/5 \log_2 3/5) (\text{França}) + \\
 & 2/10 (-2/2 \log_2 2/2 - 0/2 \log_2 0/2) (\text{Inglaterra}) + \\
 & 3/10 (-0/3 \log_2 0/3 - 3/3 \log_2 3/3) (\text{Alemanha})
 \end{aligned}$$

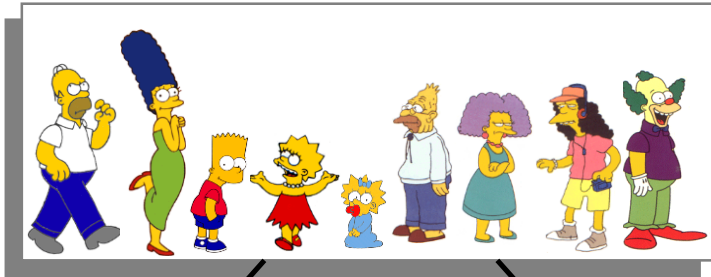
$$E(\text{PAÍS}) = 0,485$$

- Ganho de Informação (GI) / Redução na Entropia: $(0,97 - 0,485) = 0,485$.
- PAÍS (A=80%) é um atributo melhor do que SEXO (A=60%).

Lidando com atributos contínuos

ID Registro	Cabelo	Peso	Idade	Classe
 Homer	0''	250	36	M
 Marge	10''	150	34	F
 Bart	2''	90	10	M
 Lisa	6''	78	8	F
 Maggie	4''	20	1	F
 Abe	1''	170	70	M
 Selma	8''	160	41	F
 Otto	10''	180	38	M
 Krusty	6''	200	45	M

- **Considerar “ID”?**
- **Como lidar com demais atributos?**

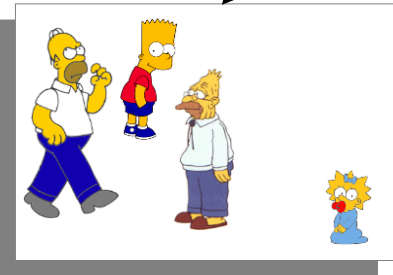


$$Entropia(4\mathbf{F},5\mathbf{M}) = -(4/9)\log_2(4/9) - (5/9)\log_2(5/9) = \mathbf{0.9911}$$

sim

não

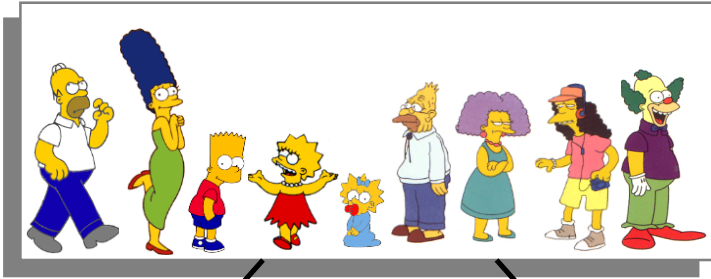
Cabelo < 5?



$$Entropia(1\mathbf{F},3\mathbf{M}) = -(1/4)\log_2(1/4) - (3/4)\log_2(3/4) = \mathbf{0.8113}$$

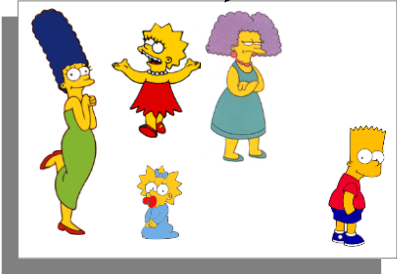
$$Entropia(3\mathbf{F},2\mathbf{M}) = -(3/5)\log_2(3/5) - (2/5)\log_2(2/5) = \mathbf{0.9710}$$

$$GI(\text{Cabelo} \leq 5) = \mathbf{0.9911} - (4/9 * \mathbf{0.8113} + 5/9 * \mathbf{0.9710}) = \mathbf{0.0911}$$



$$Entropia(4F,5M) = -(4/9)\log_2(4/9) - (5/9)\log_2(5/9) = 0.9911$$

sim
peso <= 160? não



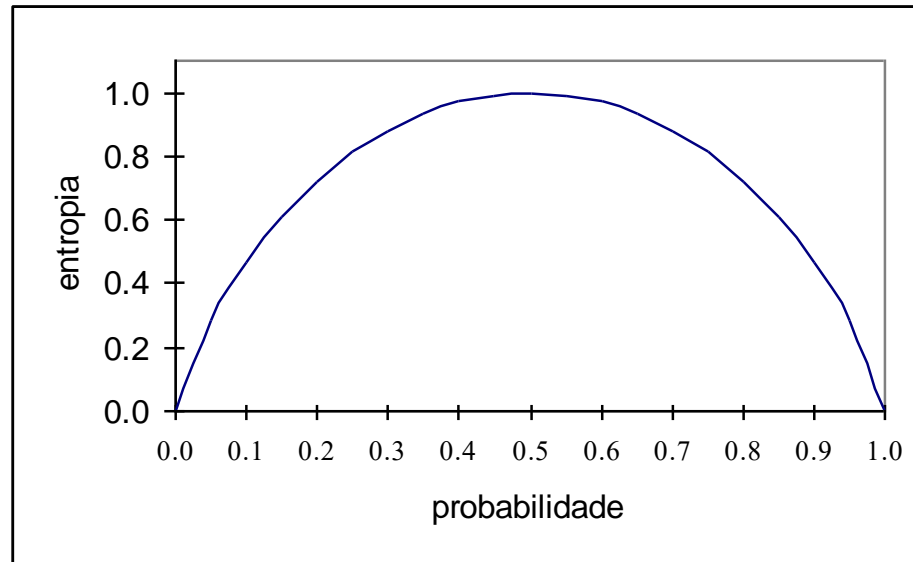
$$Entropia(4F,1M) = -(4/5)\log_2(4/5) - (1/5)\log_2(1/5) = 0.7219$$

$$Entropia(0F,4M) = -(0/4)\log_2(0/4) - (4/4)\log_2(4/4) = 0$$

$$GI(\text{peso} \leq 160) = 0.9911 - (5/9 * 0.7219 + 4/9 * 0) = 0.5900$$

➤ “Peso” discrimina melhor do que “Cabelo”...

Árvores de classificação - discussão

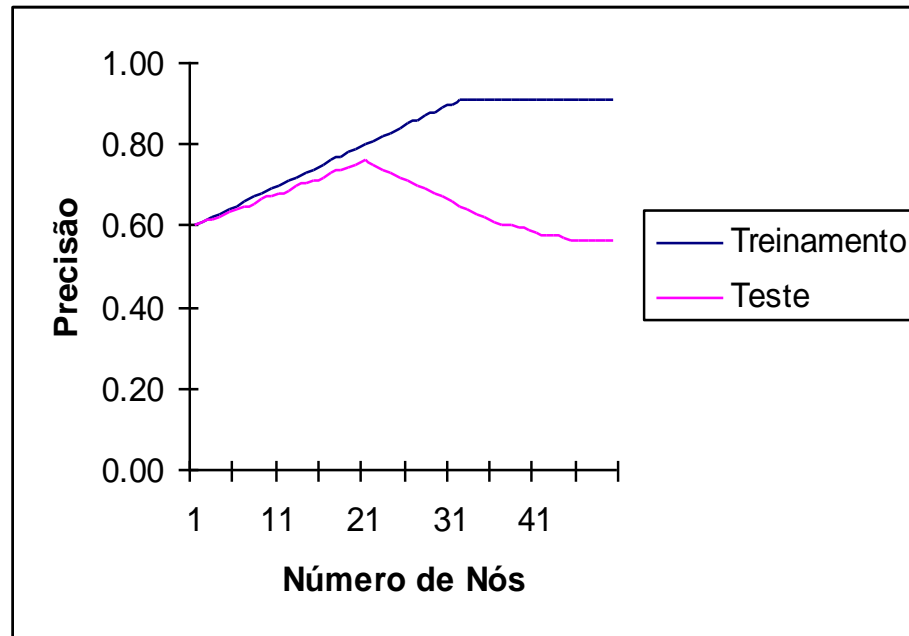


- Lembrando que $\log_2 1=0$ e definindo $\log_2 0=0$;
- Para problemas em que há “c” classes temos:

$$entropia = \sum_{i=1}^c (-p_i \log_2 p_i)$$

Onde p_i denota a probabilidade da classe “i”.

- a) Compreensibilidade / facilidade para gerar regras;
- b) Possibilidade de *super-ajuste* (erros, ruído, poucos dados):



Definição: Uma hipótese $h \in H$ *super-ajusta* os dados de treinamento se existe uma alternativa $h' \in H$ tal que h apresenta um erro menor do que h' no conjunto de treinamento mas um erro maior na distribuição completa de exemplos.

c) Procedimentos de poda:

- conjunto de validação;
- eliminar antecedentes das regras obtidas a partir da árvore;

d) GI tem um *bias* (tendência, preferência) que favorece a escolha de atributos com muitos valores;

e) Para minimizar/superar limitações:

- procedimentos de poda;
- outros critérios de escolha de atributos;
- seleção de atributos *a priori*.

Exercício

Para responder às perguntas 1-5, considere a seguinte base de dados:

A_1	A_2	A_3	A_4	Classe
S	H	H	W	N
S	H	H	S	N
O	H	H	W	Y
R	M	H	W	Y
R	C	N	W	Y
R	C	N	S	N
O	C	N	S	Y
S	M	H	W	N
S	C	N	W	Y
R	M	N	W	Y
S	M	N	S	Y
O	M	H	S	Y
O	H	N	W	Y
R	M	H	S	N

- 1) Obter a árvore de classificação pelo ganho de informação;
- 2) Quais são as regras de classificação obtidas por meio desta árvore?
- 3) Qual é a acurácia deste conjunto de regras para o conjunto de treinamento? Esta acurácia é uma estimativa adequada para a capacidade de generalização do classificador (ao se considerar dados não vistos)?
- 4) Supondo-se que não se pode dispor adicionalmente de mais dados, descreva um procedimento que permita estimar melhor a acurácia do classificador em questão para dados *novos* (e.g., ainda não observados e que serão classificados pela árvore de decisão).

Agenda

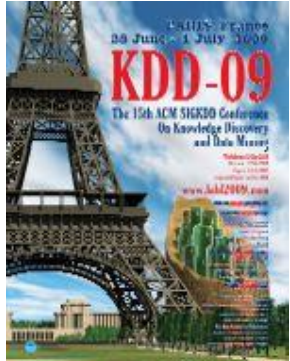
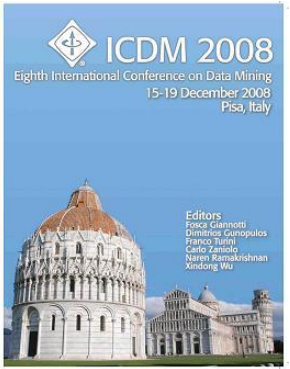
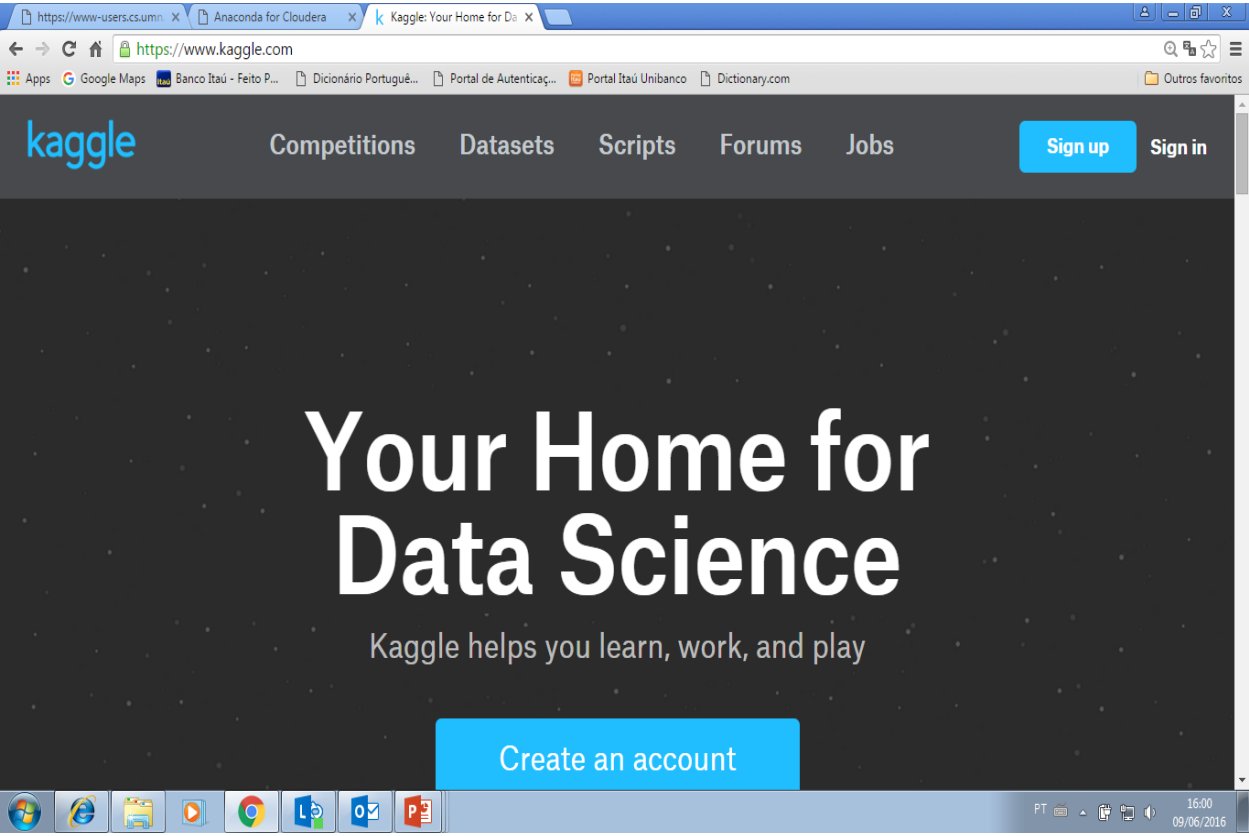
- Conceitos preliminares e classificador 1R (aquecimento)
- Classificadores Bayesianos
- Avaliação de classificadores
- *k-Nearest Neighbors* (k-NN)
- Árvores de Classificação
- Ensembles
- Random Forests
- Noções sobre SVM

Ensembles - Intuição

Para a previsão do tempo, considere que cada coluna representa um dia da semana:

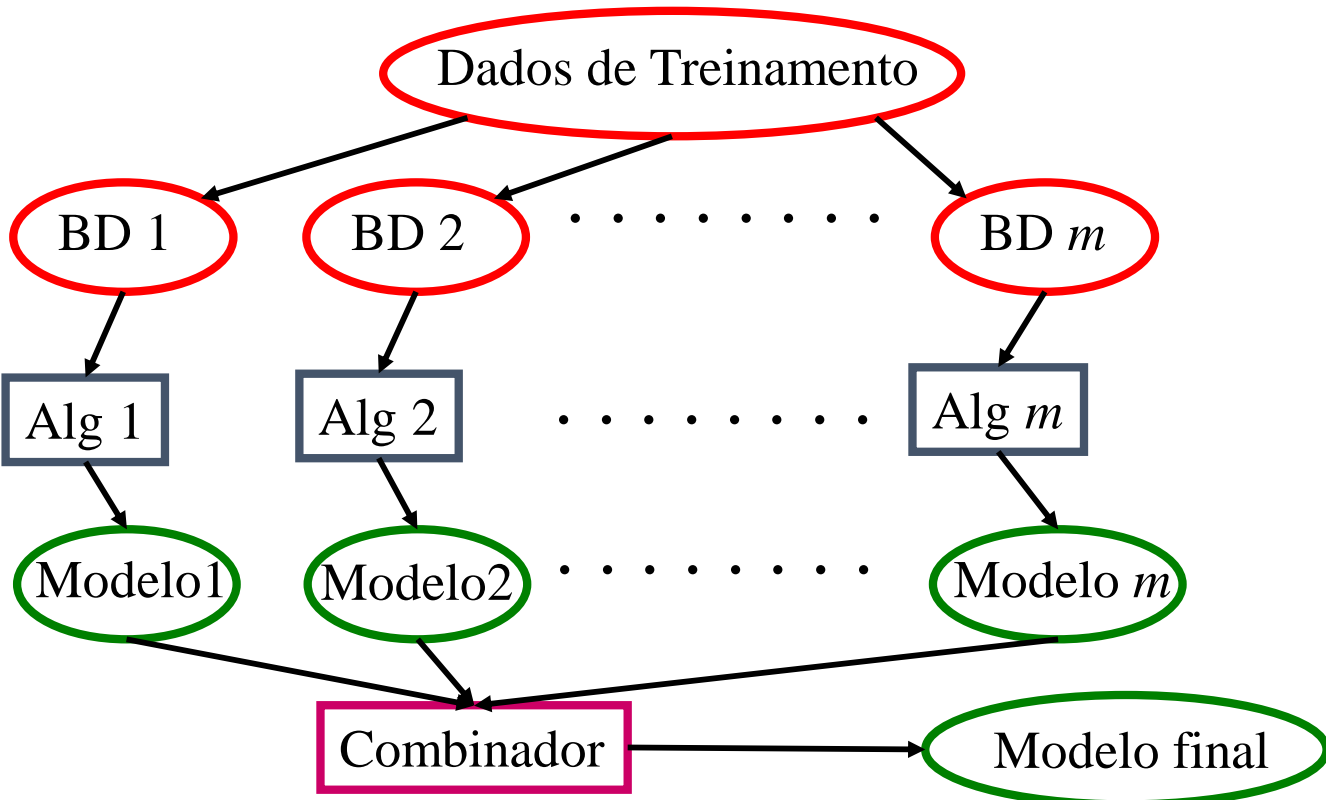
Realidade							
Modelo 1							
Modelo 2							
Modelo 3							
Modelo 4							
Modelo 5							
Combinação							

Grande sucesso na prática



Ideia geral

- Usar vários modelos para melhorar a acurácia preditiva;
- A combinação de modelos diversificados e independentes nos permite tomar melhores decisões;



Nota: BDs podem ser iguais ou diferentes.

O que esperar?

Suponha que:

- Erros de classificadores não são correlacionados;
- Temos 5 classificadores;
- Classificação final via “voto majoritário”;
- Cada classificador-componente tenha acurácia de 70%;

➤ Qual é a acurácia esperada do ensemble?

➤ $10(.7^3)(.3^2) + 5(.7^4)(.3) + (.7^5) = 83.7\%$.

➤ Para 101 classificadores temos uma acurácia de 99,9%.

O que fazer na prática?

- Diferentes algoritmos de aprendizado;
- Mesmos algoritmos com parâmetros diferentes (e.g., k-NN, redes neurais etc.);
- Bases de dados com diferentes atributos;
- Subconjuntos da mesma base de dados (*bagging*);
- Reponderar a base de treinamento (*boosting*);
- *Stacking* (meta-aprendizado a partir de previsões de um conjunto de classificadores);

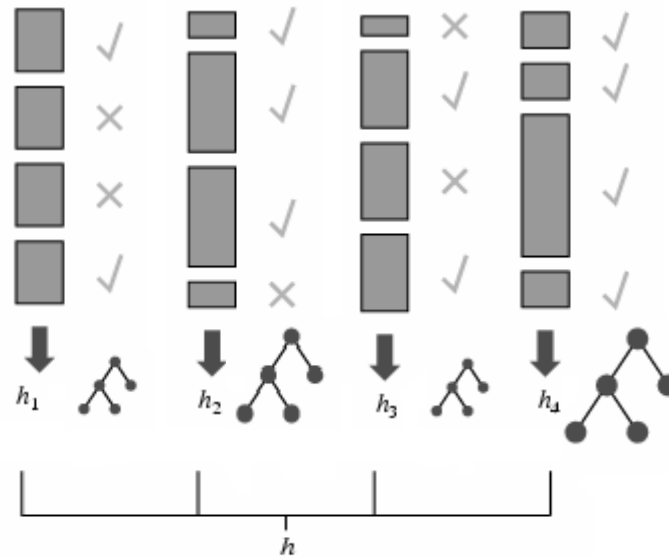
Bagging

Algoritmo de Bootstrap Aggregation (Brieman, 1996):

- 1) Amostrar, M vezes, N exemplos da base de dados (com reposição);
 - 2) Treinar M classificadores (um para cada amostra);
 - 3) Combinar os classificadores via voto majoritário.
- Passo 1) insere variância nas bases de treinamento dos componentes, aumentando a estabilidade do *ensemble*.
 - Espera-se que em cada amostra existam 63,2% de tuplas não repetidas;
 - *Random Forests* são baseadas nessa ideia.

Boosting

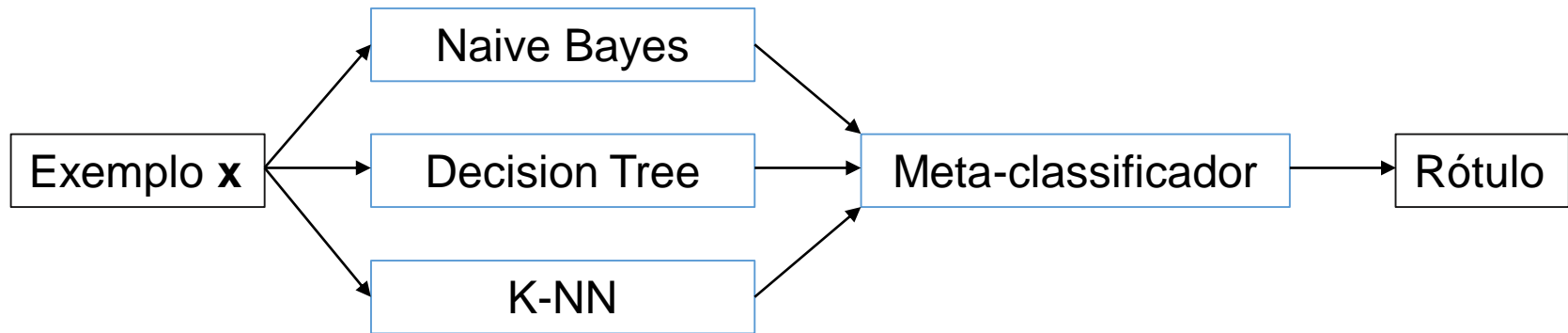
- Em vez de reamostrar, repondera exemplos;
- Cada iteração induz um classificador e repondera exemplos;
- Ensemble é baseado no voto ponderado (acurácia) dos componentes.



- Cada retângulo representa um exemplo;
- Tamanho da árvore reflete a acurácia e indica seu peso no ensemble.

Stacking

- Saída dos classificadores componentes gera uma base de dados para um meta classificador;
- Usualmente implementado via validação cruzada *leave-one-out*;
- Considere um problema de classificação binária, $Y \in \{0,1\}$:



- Cada passo do LOO gera um vetor de rótulos, e.g., [0 0 1];
- Tarefa do meta-classificador é aprender a classificar baseando-se nos vetores de rótulos, e.g., [0 0 1] → {0}

Agenda

- Conceitos preliminares e classificador 1R (aquecimento)
- Classificadores Bayesianos
- Avaliação de classificadores
- *k-Nearest Neighbors* (k-NN)
- Árvores de Classificação
- Ensembles
- Random Forests
- Noções sobre SVM

Indução do classificador:

Considere que temos N exemplos de treinamento.

Para cada uma das t iterações faça:

- 1 - Amostrar N exemplos com reposição (*bagging*).
- 2 - Induzir uma árvore usando apenas um subconjunto aleatoriamente escolhido dos atributos (e.g., 20%).
- 3 - Armazenar a árvore obtida.

Classificação

Para cada uma das t árvores de classificação:

Predizer o rótulo de classe do exemplo do conjunto-alvo.

Retornar a classe predita com maior frequência.

Leo
Breiman



Agenda

- Conceitos preliminares e classificador 1R (aquecimento)
- Classificadores Bayesianos
- Avaliação de classificadores
- *k-Nearest Neighbors* (k-NN)
- Árvores de Classificação
- Ensembles
- Random Forests
- **Noções sobre SVM**

Support Vector Machine (SVM) - Noções

O que é uma SVM?

- Um subconjunto dos exemplos de treinamento (vetores de suporte – \mathbf{x}_s);
- Pesos para cada um dos vetores de suporte (w_s);
- Uma função de similaridade $K(\mathbf{x}_q, \mathbf{x}_s)$ – *kernel*;



Vladimir
Vapnik

Considerando que $y \in \{-1, +1\}$, para classificar um exemplo do conjunto alvo \mathbf{x}_q fazemos:

$$f(\mathbf{x}_q) = \text{sign} \sum_s w_s y_s K(\mathbf{x}_q, \mathbf{x}_s)$$

- SVM é um caso particular de k-NN, com custo computacional tipicamente de $O(N^3)$.

Tsang et al., Core Vector Machines: Fast SVM Training on Very Large Data Sets, JMLR 6, 363-392 (2005)

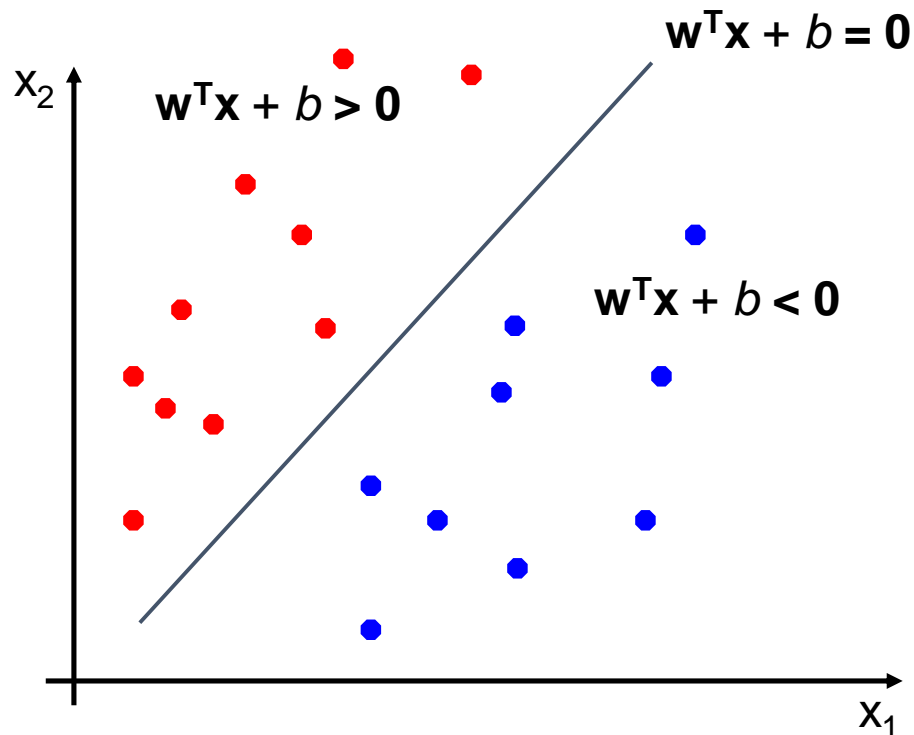
Como obter os vetores, pesos e kernel?

Algumas opções de kernels:

- Linear: $K(\mathbf{x}_s, \mathbf{x}_q) = \mathbf{x}_s \cdot \mathbf{x}_q$
 - Polinomial: $K(\mathbf{x}_s, \mathbf{x}_q) = (\mathbf{x}_s \cdot \mathbf{x}_q)^d$
 - Gaussiano etc.
-
- Escolher aquele que se ajusta melhor aos dados: fronteiras de decisão podem requerer kernels complexos;
 - Como escolher os vetores e os pesos?
 - Maximizar a margem \rightarrow pesos \rightarrow vetores

Classificação com máxima margem

- Classificação binária pode ser vista como uma tarefa de separar classes num espaço de características:



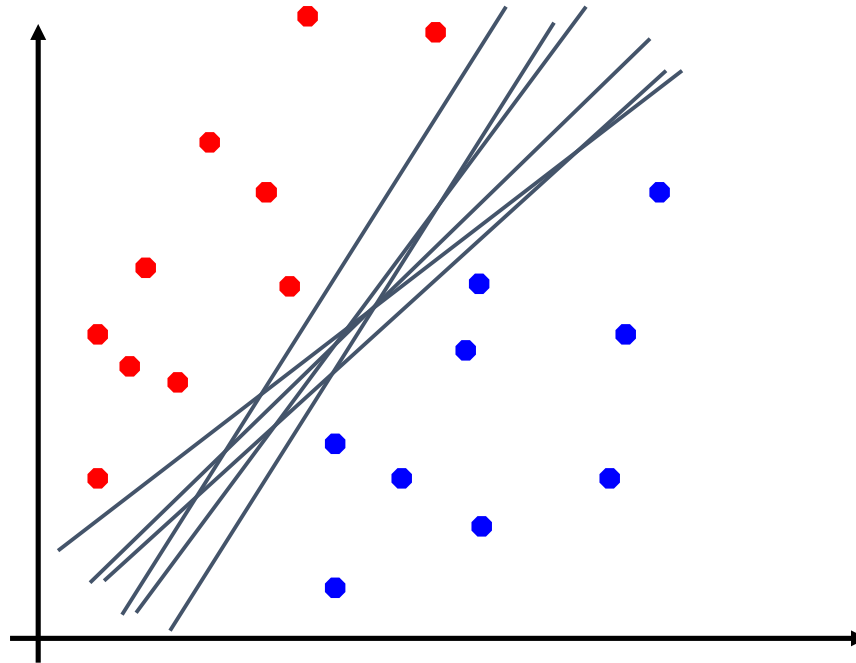
$$f(\mathbf{x}) = \text{sign}(w^T \mathbf{x} + b)$$

Exemplo:

$$w^T = [-1 \ 1], \quad \mathbf{x} = [x_1 \ x_2]^T, \quad b=0.$$

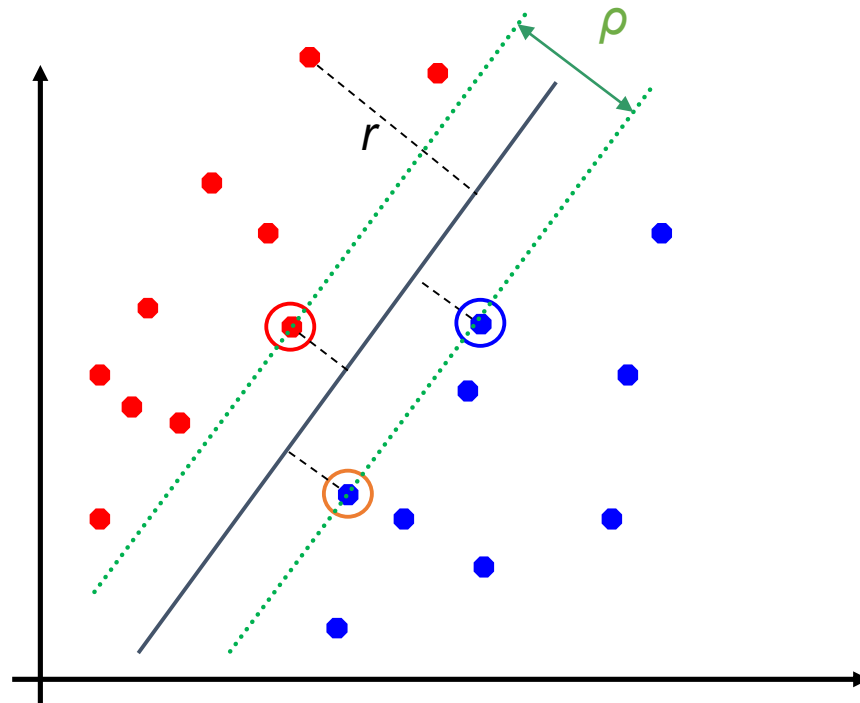
$$f(\mathbf{x}) = \text{sign}(x_2 - x_1)$$

Qual é o separador ótimo?

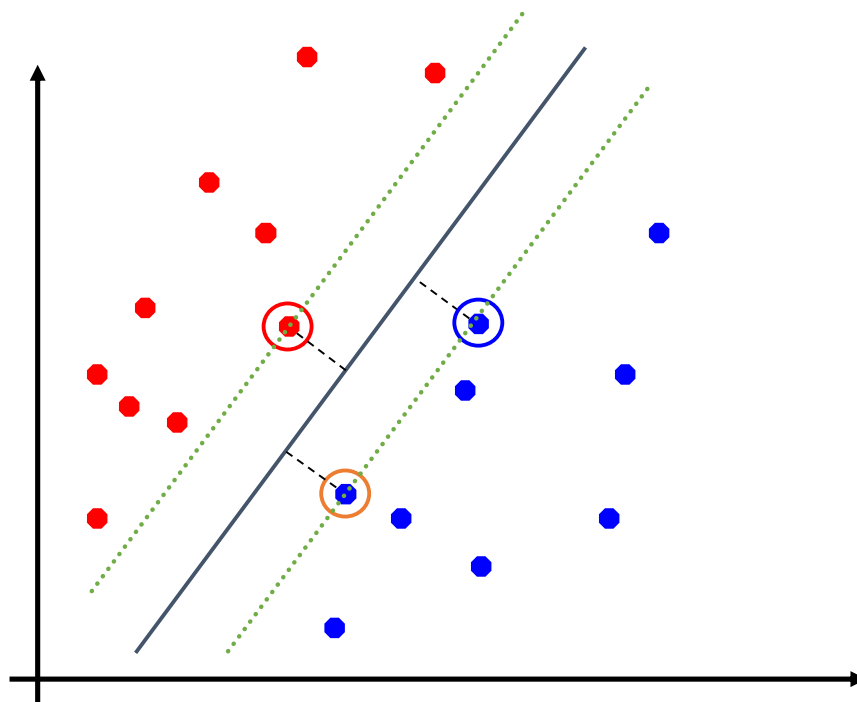


Otimizar a margem de classificação

- Distância de \mathbf{x}_i para o separador é r .
- Exemplos mais próximos ao hiperplano são os **vetores de suporte**.
- **Margem** ρ do separador é a distância entre os vetores de suporte.



- Maximizar a margem é intuitivo e está de acordo com a teoria do aprendizado PAC (*Probably Approximately Correct*, 1984).
- Implica que apenas os **vetores de suporte** importam; demais podem ser ignorados.



Leslie Valiant
(Turing Award)

Formulação matemática

Considere que o conjunto de treinamento $\{(\mathbf{x}_i, y_i)\}_{i=1..N}$, $\mathbf{x}_i \in \mathbf{R}^d$, $y_i \in \{-1, 1\}$, seja separado por um hiperplano de margem ρ . Então para cada (\mathbf{x}_i, y_i) temos:

$$\begin{aligned} \mathbf{w}^T \mathbf{x}_i + b \leq -\rho/2 & \text{ se } y_i = -1 \\ \mathbf{w}^T \mathbf{x}_i + b \geq \rho/2 & \text{ se } y_i = 1 \end{aligned} \iff y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq \rho/2$$

Para cada \mathbf{x}_s a inequação acima se torna uma equação. Após reescalar \mathbf{w} e b por $\rho/2$ na equação, a distância entre cada \mathbf{x}_s e o hiperplano é:

$$r = \frac{y_s(\mathbf{w}^T \mathbf{x}_s + b)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

Então a margem pode ser expressa por meio de \mathbf{w} e b como:

$$\rho = 2r = \frac{2}{\|\mathbf{w}\|}$$

Então podemos formular o problema de otimização quadrática:

Encontrar \mathbf{w} e b tal que

$$\rho = \frac{2}{\|\mathbf{w}\|} \text{ seja maximizado}$$

e para todos os $(\mathbf{x}_i, y_i), i=1..N: y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

Este pode ser reformulado como:

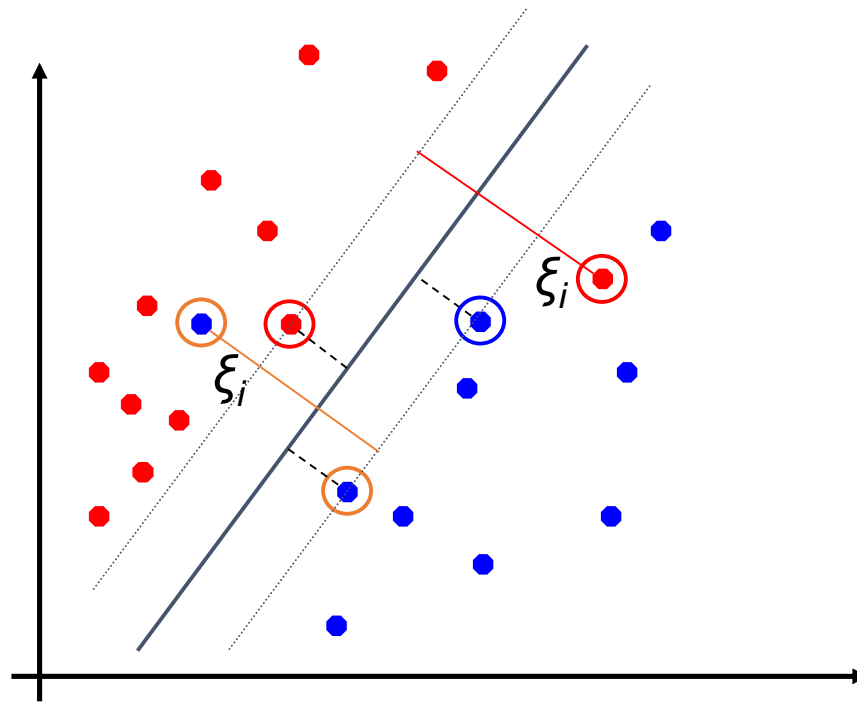
Encontrar \mathbf{w} e b tal que

$$\Phi(\mathbf{w}) = \|\mathbf{w}\|^2 = \mathbf{w}^T \mathbf{w} \text{ seja minimizado}$$

E para todo $(\mathbf{x}_i, y_i), i=1..N: y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

Classificação com margem flexível

- E se o conjunto de treinamento não é linearmente separável?
- Variáveis *frouxas* (*slack*), ξ_i , podem ser adicionadas para permitir erros de classificação em exemplos ruidosos ou difíceis, resultando numa margem flexível:



- Problema de otimização passa então a ser:

Encontrar \mathbf{w} e b tal que

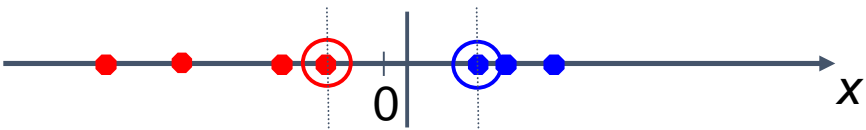
$\Phi(\mathbf{w}) = \mathbf{w}^T \mathbf{w} + C \sum \xi_i$ é minimizado

e para todo $(\mathbf{x}_i, y_i), i=1..N: y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0$

- O parâmetro C é útil para controlar o superajuste (*overfitting*), balanceando a importância relativa de maximizar a margem e ajustá-la aos dados de treinamento.
- Escolher C via validação cruzada.

SVM não linear

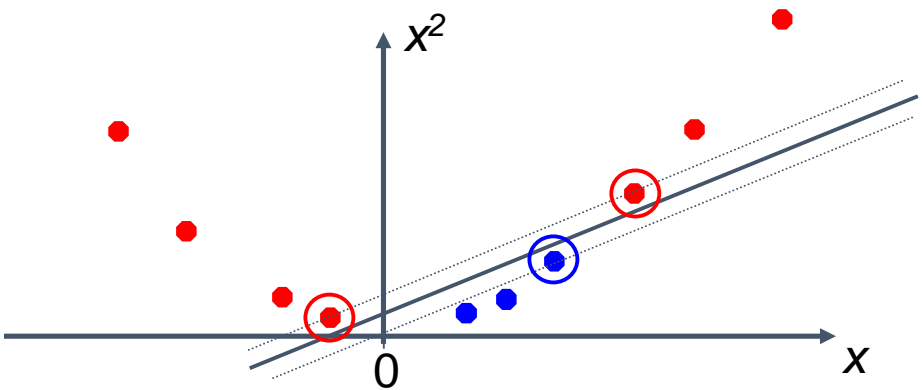
Para dados que são linearmente separáveis podemos usar uma SVM linear (equivalente a um perceptron simples):



E se o problema é muito difícil?



Mapear dados para um espaço de dimensionalidade maior:



Ideia geral: mapear o espaço original para um espaço de maior dimensionalidade onde o conjunto de treinamento é separável
(kernel trick):

