

# APRENDIZAGEM DE MÁQUINA

(usando Python)

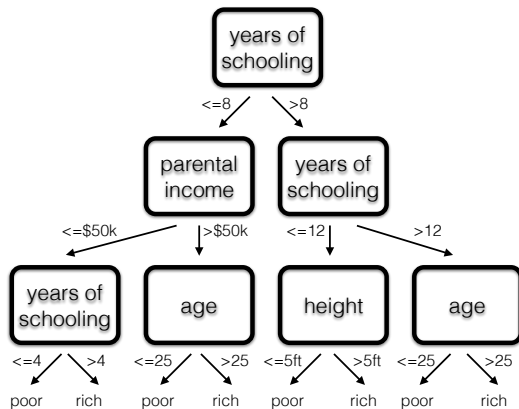
Thiago Marzagão

## ÁRVORE DE DECISÃO & VALIDAÇÃO

# árvore de decisão

- Aulas passadas: queríamos prever variáveis quantitativas.
- Aula de hoje (e seguintes): queremos prever variáveis qualitativas.
- Em outras palavras: queremos prever a *classe*.
- Em outras palavras (2): queremos *classificar*.
- Vários algoritmos de classificação: regressão logística, árvore de decisão, SVM, NN.

# árvore de decisão



# árvore de decisão

**Table 4.1.** The vertebrate data set.

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard	cold-blooded	scales	yes	yes	no	no	no	fish
shark								
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

(Intro to Data Mining, p. 147)

## visão geral

- Objetivo: dividir as amostras recursivamente até obter “folhas” suficientemente homogêneas.
- Começamos com as amostras todas.
- Dividimos as amostras em dois grupos, com base em alguma variável (e ponto de corte, se a variável é quantitativa).
- Mas não arbitrariamente! Dividimos as amostras em dois grupos de tal maneira que os dois grupos sejam tão homogêneos quanto possível com relação a  $y$ .
- Dividimos cada grupo da mesma forma e assim vamos “crescendo” a árvore de cima p/ baixo.
- Diferentes medidas de homogeneidade são usadas na prática. Duas mais comuns: Gini e entropia.

# Gini

- $G = \sum_{c=1}^C p_{gc}(1 - p_{gc})$
- $p_{gc}$  é a porcentagem de amostras no grupo  $g$  que pertencem à classe  $c$ .
- Quanto menor  $G$ , maior a homogeneidade dos grupos.
- Exemplo: 2 classes, um grupo  $c/ 70$  amostras dividido 50/50, um grupo  $c/ 30$  amostras dividido 60/40.
- $G_{g=1} = (0,5 \times (1 - 0,5)) + (0,5 \times (1 - 0,5)) = 0,5$
- $G_{g=2} = (0,6 \times (1 - 0,6)) + (0,4 \times (1 - 0,4)) = 0,48$
- $G = 70/100 \times 0,5 + 30/100 \times 0,48 = 0,494$
- Dividimos as amostras recursivamente. P/ cada divisão escolhemos a variável e ponto de corte que minimizam  $G$ . Paramos quando os grupos forem 100% homogêneos (às vezes um pouco antes, p/ evitar overfitting; mais sobre isso depois).

# Entropia

- $E = - \sum_{c=1}^C p_{gc}(\log(p_{gc}))$
- ( $\log$  geralmente na base 2, mas base não é fundamental)
- $p_{gc}$  é a porcentagem de amostras no grupo  $g$  que pertencem à classe  $c$ .
- Quanto menor  $E$ , maior a homogeneidade dos grupos.
- Exemplo: 2 classes, um grupo  $c/ 70$  amostras dividido 50/50, um grupo  $c/ 30$  amostras dividido 60/40.
- $E_{g=1} = -0,5 \times \log 0,5 - 0,5 \times \log 0,5 \approx 0,693$
- $E_{g=2} = -0,6 \times \log 0,6 - 0,4 \times \log 0,4 \approx 0,673$
- $E = 70/100 \times 0,693 + 30/100 \times 0,673 = 0,687$
- Dividimos as amostras recursivamente. P/ cada divisão escolhemos a variável e ponto de corte que minimizam  $E$ . Paramos quando os grupos forem 100% homogêneos ('as vezes um pouco antes, p/ evitar overfitting; mais sobre isso depois).

## árvore de decisão

- Crescemos a árvore de cima p/ baixo, usando  $G$  ou  $E$ , até que os grupos sejam 100% homogêneos.
- A cada divisão é preciso identificar qual variável maximiza a homogeneidade dos grupos resultantes.
- Isso é feito iterativamente - i.e., na base da tentativa e erro.
- Parece uma tarefa hercúlea mas seu laptop é capaz de fazer isso em uma fração de segundos.
- Depois de pronta, podemos usar a árvore p/ classificar novas amostras.



# árvore de decisão

- Principais vantagens:
- Matematicamente simples: método não-paramétrico. Não existem parâmetros  $p$ / estimar ( $\neq$  regressão; não existe  $\mathbf{b}$ ).
- Não exige especificar forma funcional a priori.
- Principal desvantagem:
- Pouca robustez. Pequenas perturbações nos dados podem criar uma árvore completamente diferente. E um corte ótimo num determinado ponto pode resultar em cortes subótimos depois.
- Como resolver isso?

## random forest

- Em vez de criar uma árvore, criamos centenas ou milhares. P/ novas amostras, vale a predição modal das árvores. Isso aumenta a robustez das predições.
- P/ “crescer” cada árvore usamos bootstrapping: pegamos  $N$  amostras, com reposição, com  $N$  sendo o número total de amostras no dataset.
- A cada partição usamos não as variáveis todas, mas um subset aleatório delas, geralmente  $\sqrt{K}$ .
- Quantas árvores? O necessário p/ que as predições fiquem estáveis.
- Individualmente cada árvore tem desempenho ruim, mas a predição modal tende a ser boa.
- Variante: extreme random forest, em que o ponto de corte das variáveis contínuas também é aleatório.
- Na prática: quase ninguém usa uma única árvore de decisão; é random forest ou boosting.

# validação

- Como avaliar o poder preditivo de uma árvore ou random forest?
- Existem vários métodos de validação.
- Método mais simples: dividir as amostras em  $2/3$  treinamento e  $1/3$  teste.
- “Crescemos” a árvore usando apenas os  $2/3$  de treinamento e depois usamos a árvore p/ prever a classe dos  $1/3$  de teste.
- Na verdade isso serve p/ qualquer algoritmo de classificação, não apenas árvores de decisão. Serve inclusive p/ compararmos o desempenho de diferentes algoritmos (ex.: árvore de classificação vs SVM, usando o mesmo dataset).

**Table 4.2.** Confusion matrix for a 2-class problem.

		Predicted Class	
		<i>Class = 1</i>	<i>Class = 0</i>
Actual Class	<i>Class = 1</i>	$f_{11}$	$f_{10}$
	<i>Class = 0</i>	$f_{01}$	$f_{00}$

(Intro to Data Mining, p. 149)

## medidas de desempenho: acurácia

- Acurácia = predições corretas / predições totais
- $$= \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}$$
- Diagonal contém as predições corretas.

## medidas de desempenho: taxa de erros

- Taxa de erros = predições incorretas / predições totais
- $$= \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}$$
- $= 1 - \text{acurácia}$

## medidas de desempenho: precisão

- Precisão = % de vezes em que a classe correta é 0 quando o modelo prevê 0
- $$= \frac{f_{00}}{f_{10} + f_{00}}$$
- P/ mais de 2 classes: média ponderada.

## medidas de desempenho: recall

- Recall = % de vezes em que o modelo prevê 0 quando a classe correta é 0
- $$= \frac{f_{00}}{f_{01} + f_{00}}$$
- P/ mais de 2 classes: média ponderada.



# medidas de desempenho: F1

- $F1 = 2 \frac{\text{precisão} \times \text{recall}}{\text{precisão} + \text{recall}}$
- Melhor: 1. Pior: 0.
- P/ mais de 2 classes: média ponderada.

## medidas de desempenho

- Ok, mas como sei se um modelo é suficientemente bom?
- R: Não existe um valor “mágico” p/ acurácia, etc.
- Depende do caso concreto em análise.
- Ex.: acurácia de 70% p/ qualidade do tomador de crédito. Bom? Ruim?
- Importante: classes desbalanceadas podem ser um problema.
- Ex.: exame médico que detecta um determinado tipo de câncer.
- A cada 100 exames, apenas 1 efetivamente é câncer.
- Classificador que sempre dá não-câncer vai estar correto 99% das vezes!
- Nesse caso a acurácia não é uma boa métrica. É preciso aceitar mais falsos positivos. (Ou usar correções p/ rebalancear classes - e.g., bootstrapping.)

- Uma alternativa comum à divisão  $2/3 - 1/3$  é a chamada validação cruzada:
- ... divide-se o dataset em  $n$  grupos; freqüentemente  $n = 10$
- ... treina-se o algoritmo classificador usando  $n - 1$  grupos
- ... afere-se o desempenho do classificador usando o grupo deixado de fora como teste
- ... repete-se o procedimento  $p/$  cada grupo (i.e., cada grupo será usado como teste uma vez)
- ... afere-se o desempenho médio do classificador

# overfitting

- Todo algoritmo de classificação está sujeito a overfitting.
- Overfitting: o classificador (árvore, SVM, o que seja) responde a idiosincrasias do dataset e assim não desempenha bem com novas amostras. Ex.: por uma razão qualquer todos os candidatos a uma determinada linha de crédito que tinham altura superior a 1,90 foram bons pagadores. Mas poucas pessoas têm mais que 1,90, então esse achado é provavelmente inócuo. Mas o classificador pode “aprender” erroneamente que candidatos mais altos são bons pagadores. Overfitting = classificador confunde ruído com sinal.
- Importante validar o classificador: é preciso avaliar seu desempenho com amostras ainda não vistas.