MÉTODOS ESTATíSTICOS EM DATA MINING
- Folha 5: Métodos Modernos de Classificação e Regressão -

47. Para segmentar um nó de uma árvore de decisão, considere as medidas de impureza de Gini e a entropia de Shannon.

    (a) Mostre que ambas as medidas são estritamente côncavas. (Sugestão: calcule a derivada da medida em ordem a $p = (p_1, p_2, ..., p_K)$ e verifique que se trata de uma matriz definida negativa).

    (b) Mostre que para cada uma destas duas medidas de impureza, a distribuição uniforme é a mais impura. (Sugestão: use a desigualdade de Cauchy-Schwartz, $\left(\sum_{i=1}^{K} a_i b_i\right)^2 \leq \sum_{i=1}^{K} a_i^2 \sum_{i=1}^{K} b_i^2$ e a desigualdade $log(x) \leq x - 1$)

    (c) Para o caso de duas classes, faça um gráfico de ambas as medidas como função de p_1 e mostre assim que são semelhantes.

48. Considere a divisão de um conjunto de dados em K grupos, e atribua a cada grupo (e ao conjunto total) a classe mais frequente. Seja N o cardinal do conjunto total e n_k o cardinal do grupo k.

    (a) Mostre que o erro aparente não é aumentado pela divisão. (Sugestão: $e_{aparente} = \frac{N - max(n_c)}{N}$

    (b) Mostre que esse erro será sempre menor após a divisão, a não ser que a classe mais frequente seja a mesma para todos os grupos.

    (c) Deduza que durante o crescimento de uma árvore, o erro aparente é sempre reduzido em cada passo.

49. Descreva o método de poda de árvores de decisão de Breiman & al (1984).

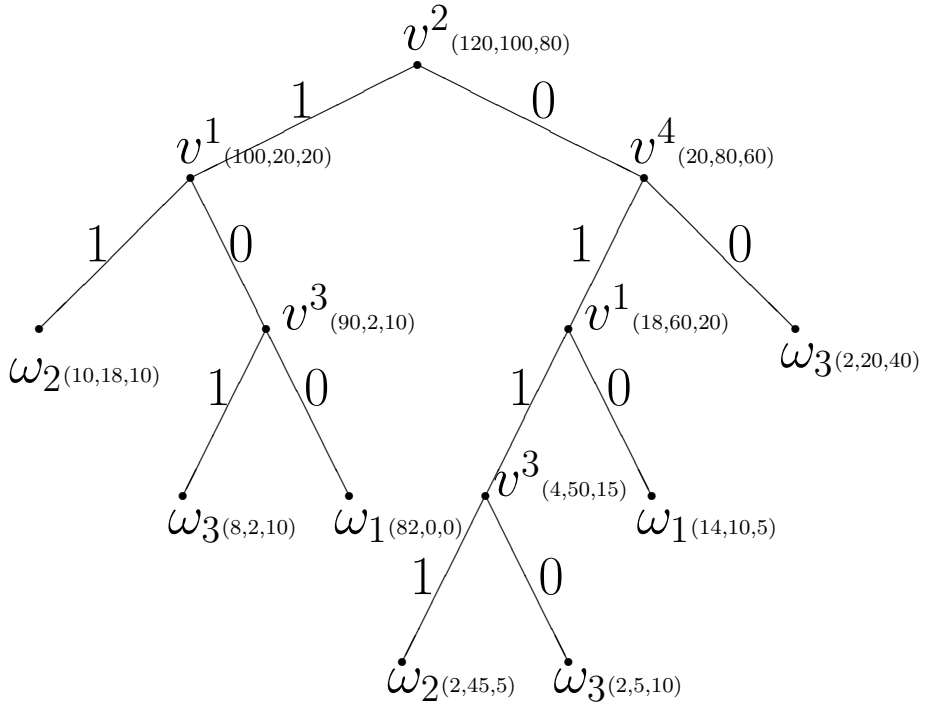
50. Aplique o algoritmo de poda de custo-complexidade de Breiman & al. (1984) à árvore que se segue. $\mathcal{V} = \{\ v^m\ /\ 1 \leq m \leq 4\ \}$ designa o conjunto dos atributos descritivos e $\Omega = \{\ \omega_j\ /\ 1 \leq j \leq 3\}$ o conjunto das classes a discriminar.

51. Considere o conjunto de dados Boston sobre os preços de casas nos subúrbios de Boston (package MASS do R ou página da disciplina). Construa uma árvore de regressão para prever o preço das casas e desenhe essa árvore. Faça variar o parâmetro de complexidade e veja o que acontece à árvore.

52. Mostre que uma rede neuronal com apenas duas camadas (de entrada e de saída) só consegue criar fronteiras lineares entre as classes. (Sugestão: a fronteira entre duas classes é dada por $g_l(x) = g_j(x)$ em que $g_l(x) = f_o(\alpha_l + \sum_{i \to l} w_{il} x_i))$

53. Mostre que se a função de activação das unidades escondidas é linear, uma rede neuronal com 3 camadas é equivalente a uma rede neuronal com duas camadas. (Sugestão: $g_l(x) = f_o(\alpha_l + \sum_{j \to h} w_{jh} f_h(\alpha_j + \sum_{i \to j} w_{ij} x_i))$ em que $f_h(x) = ax + b$)

54. Dada a amostra 1.0, 1.1, 1.3, 1.6, 1.9, 2.0, 2.3, 2.7, 3.0, estime a f.d.p. pelo método do histograma e pelo método do núcleo, $\hat{p}(x) = \frac{1}{n.h} \sum_{i=1}^{n} \mathcal{K}(\frac{x - X_i}{h})$, usando uma função núcleo uniforme e $h = 0.4$.



55. Considere o conjunto de dados sobre o tipo de sindroma de Cushing, doença relacionada com hipertensão (ver package MASS do R ou página da disciplina). Comeşe por eliminar da tabela de dados as observações cujo tipo é desconhecido, faça um histograma das duas variáveis predictivas e depois aplique logaritmos a essas variáveis. De seguida coloque cerca de 30% das observações no conjunto teste e as restantes no treino. Corra agora o método do vizinho mais próximo e compare os valores reais do teste com os previstos.

56. Considere o conjunto de dados Singh sobre a expressão de 150 genes em 52 amostras com tumor da próstata e 50 normais (ver página da disciplina). Usando o método dos K-vizinhos mais próximos, calcule o erro no teste para $K = 1$ e $K = 3$. Adeque agora uma árvore de decisão aos dados usando a função rpart do R e estime o erro no teste. De seguida corra uma rede neuronal nos dados com 5 unidades na camada escondida e um "weight decay" de 0.05 e calcule o erro no treino e no teste. Faça o mesmo usando agora uma máquina de suporte vectorial com núcleo radial, linear e sigmoid.

57. * Considere o estimador da f.d.p. do método do núcleo, $\hat{p}(x) = \frac{1}{n.h^p} \sum_{i=1}^{n} \mathcal{K}(\frac{x-X_i}{h})$. Mostre que o estimador K-NN de $p(x)$ é um caso particular do de cima para uma escolha apropriada de $\mathcal{K}$ e h (que varia com x).

58. * Uma máquina de suporte vectorial é como um método do núcleo pesado e consiste em fazer um mapeamento dos dados num espaço de maiores dimensões e depois a considerar o hiperplano de margem máxima nesse novo espaço. Como um exemplo bastante simples de um classificador, considere um baseado na média mais próxima no espaço de maiores dimensões. Verifique que este classificador coincide com o método do núcleo usual.

59. * O método do vizinho mais próximo atribui um novo indivíduo com características x à classe do seu vizinho mais próximo do conjunto de treino onde no caso mais simples se usa a distância euclidiana usual ou, de forma equivalente, o seu quadrado: $||x - x_n||^2$ em que x_n é o vizinho mais próximo.

    (a) Exprima esta regra usando produtos escalares.

(b) Formule a regra do vizinho mais próximo num espaço de dimensão superior, usando para o efeito uma função núcleo tal como é feito no classificador subjacente às máquinas de suporte vectorial.

(c) Por exemplo para o caso de um núcleo radial, $\mathcal{K}(x_i, x_j) = \exp(- \parallel \mathrm{x_i} - \mathrm{x_j} \parallel^2 /2\sigma^2)$, qual das duas regras (alíneas (a) e (b)) acha que dará melhores resultados em termos de erro de previsão? Justifique a sua resposta.

51) EM R:

```
library(MASS)
data(Boston)
?Boston
str(Boston)
library(rpart)
arv=rpart(Boston[,14]~ ., data=Boston[,-14])
plot(arv)
text(arv)
arv=rpart(Boston[,14]~ ., data=Boston[,-14],cp=0.05)
plot(arv)
text(arv)
```

55) EM R:

```
library(MASS)
cush = Cushings[Cushings$Type!='u',]
cush[,3] = factor(cush[,3],levels=c('a','b','c'))
cush.type = cush[,3]
hist(cush[,1]) hist(cush[,2])
cush[,1] = log(cush[,1])
cush[,2] = log(cush[,2])
```

```
library(class)
number.obs = nrow(cush)
train.vec = runif(number.obs)¿0.3
test.vec = !train.vec
cush.train = cush[train.vec,]
cush.test = cush[test.vec,]
```

```
cush.knn = knn(train=cush.train[,1:2], test=cush.test[,1:2], cl=cush.train[,3], k=1)
cush.knn
cush.test[,3]
```

56) EM R:

```
library(class)
singh.train=read.table("singh.train.data",sep=",")
singh.test=read.table("singh.test.data",sep=",")
singh.knn.1 = knn(singh.train[,-1], singh.test[,-1], singh.train[,1], k=1)
misclass = sum(singh.test$outcome != singh.knn.1)
test.error.knn.1 = misclass/nrow(singh.test)
test.error.knn.1
singh.knn.3 = knn(singh.train[,-1], singh.test[,-1],singh.train[,1], k=3)
misclass = sum(singh.test$outcome != singh.knn.3)
test.error.knn.3 = misclass/nrow(singh.test)
test.error.knn.3
```

```
    library(rpart)
tree = rpart( outcome~ . ,data=singh.train)
plot(tree)
text(tree,use.n=TRUE,pretty=0)
tree.class = predict(tree,newdata=singh.test,type="class")
misclass = singh.test$outcome!=tree.class
tree.test.error = sum(misclass)/nrow(singh.test)


    library(nnet)
singh.nnet = nnet(outcome~ ., data=singh.train, size=5,decay=0.05, maxit=200)
singh.class = predict(singh.nnet, newdata=singh.train,type="class")
singh.class
misclass = singh.train$outcome!=singh.class
nnet.train.error = sum(misclass)/nrow(singh.train)
singh.class = predict(singh.nnet, newdata=singh.test,type="class")
misclass = singh.test$outcome!=singh.class
nnet.test.error = sum(misclass)/nrow(singh.test)


    library(e1071)
model1 = svm(singh.train[,1]~ ., data = singh.train[,-1],kernel="radial")
summary(model1)
# visualize (classes by color, SV by crosses):
plot(cmdscale(dist(singh.train[,-1])), col = as.integer(singh.train[,1]),pch = c("o","+")[1:569
%in% model1$index + 1],xlab="x",ylab="y")
predr1 = predict(model1, singh.test[,-1])
table(predr1, singh.test[,1])
erro.svmr1=sum(predr1!=singh.test[,1])/nrow(singh.test)
erro.svmr1
#Agora com kernel=linear
model2 = svm(singh.train[,1]~ ., data = singh.train[,-1],kernel="linear")
summary(model2)
plot(cmdscale(dist(singh.train[,-1])), col = as.integer(singh.train[,1]),pch = c("o","+")[1:569
%in% model2$index + 1],xlab="x",ylab="y")
predr2 = predict(model2, singh.test[,-1])
table(predr2, singh.test[,1])
erro.svmr2=sum(predr2!=singh.test[,1])/nrow(singh.test)
erro.svmr2
#Agora com kernel=sigmoid
model3 = svm(singh.train[,1]~ ., data = singh.train[,-1],kernel="sigmoid")
summary(model3)
plot(cmdscale(dist(singh.train[,-1])), col = as.integer(singh.train[,1]),pch = c("o","+")[1:569
%in% model3$index + 1],xlab="x",ylab="y")
predr3 = predict(model3, singh.test[,-1])
table(predr3, singh.test[,1])
erro.svmr3=sum(predr3!=singh.test[,1])/nrow(singh.test)
erro.svmr3
```

Algumas rotinas R (por exemplo para fazer o tuning):

CURVAS ROC; roc(); plot(roc()); train() da library(caret)
ÁRVORES: rpart() e depois printcp() (para ver os vários valores do custo-complexidade);

cptable. Há ainda outras: rpartXse, ctree, J48 (implementação do c4.5 no weka); RWeka (para fazer a ligação entre o weka e o R)

KNN; tuning(); tune.knn() para os vários valores de k.

REDES NEURONAIS: tune.nnet(), tunning.nnet() para variar o sdize e o decay; plot.nnet

SVM: e1071 e kernlab (mais populares); tune.svm()