Predictive Analytics

L. Torgo

ltorgo@fc.up.pt
Faculdade de Ciências / LIAAD-INESC TEC, LA
Universidade do Porto

Dec, 2014

What is Prediction?

Definition

- Prediction (forecasting) is the ability to anticipate the future.
- Prediction is possible if we assume that there is some regularity in what we observe, i.e. if the observed events are not random.

Example

Medical Diagnosis: given an historical record containing the symptoms observed in several patients and the respective diagnosis, try to forecast the correct diagnosis for a new patient for which we know the symptoms.

Prediction Models

- Are obtained on the basis of the assumption that there is an unknown mechanism that maps the characteristics of the observations into conclusions/diagnoses. The goal of prediction models is to discover this mechanism.
  - Going back to the medical diagnosis what we want is to know how symptoms influence the diagnosis.
- Have access to a data set with "examples" of this mapping, e.g. this patient had symptoms x, y, z and the conclusion was that he had disease p
- Try to obtain, using the available data, an approximation of the unknown function that maps the observation descriptors into the conclusions, i.e. $Prediction = f(Descriptors)$

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

"Entities" involved in Predictive Modelling

- Descriptors of the observation:
  set of variables that describe the properties (features, attributes) of the cases in the data set
- Target variable:
  what we want to predict/conclude regards the observations
- The goal is to obtain an approximation of the function $Y = f(X_1, X_{,2}, \cdots, X_p)$, where Y is the target variable and $X_1, X_{,2}, \cdots, X_p$ the variables describing the characteristics of the cases.
- It is assumed that Y is a variable whose values depend on the values of the variables which describe the cases. We just do not know how!
- The goal of the modelling techniques is thus to obtain a good approximation of the unknown function $f()$

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

How are the Models Used?

Predictive models have two main uses:

1. Prediction
   use the obtained models to make predictions regards the target variable of new cases given their descriptors.
2. Comprehensibility
   use the models to better understand which are the factors that influence the conclusions.

Types of Prediction Problems

- Depending on the type of the target variable (Y) we may be facing two different types of prediction models:
  1. Classification Problems - the target variable Y is nominal
     e.g. medical diagnosis - given the symptoms of a patient try to predict the diagnosis
  2. Regression Problems - the target variable Y is numeric
     e.g. forecast the market value of a certain asset given its characteristics

Examples of Prediction Problems

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.8	3.0	5.5	2.1	virginica
5.7	2.5	5.0	2.0	virginica
...	...	...	...	...

Species = f(Sepal.Length, ...)

Classification Task

Regression Task

season	size	speed	mxPH	mnO2	Cl	Chla	...	a1
winter	small	medium	8.00	9.8	60.800	50.0	...	0.0
spring	small	medium	8.35	8.0	57.750	1.3	...	1.4
autumn	small	medium	8.10	11.4	40.020	15.6	...	3.3
spring	small	medium	8.07	4.8	77.364	1.4	...	3.1
autumn	small	medium	8.06	9.0	55.350	10.5	...	9.2
winter	small	high	8.25	13.1	65.750	28.4	...	15.1
...	...	...	...	...	...	...	...	...

a1 = f(season, size, ...)

Types of Prediction Models

- There are many techniques that can be used to obtain prediction models based on a data set.
- Independently of the pros and cons of each alternative, all have some key characteristics:
  1. They assume a certain functional form for the unknown function $f()$
  2. Given this assumed form the methods try to obtain the best possible model based on:
     1. the given data set
     2. a certain preference criterion that allows comparing the different alternative model variants

Functional Forms of the Models

- There are many variants. Examples include:
  - Mathematical formulae - e.g. linear discriminants
  - Logical approaches - e.g. classification or regression trees, rules
  - Probabilistic approaches - e.g. naive Bayes
  - Other approaches - e.g. neural networks, SVMs, etc.
  - Sets of models (ensembles) - e.g. random forests, adaBoost
- These different approaches entail different compromises in terms of:
  - Assumptions on the unknown form of dependency between the target and the predictors
  - Computational complexity of the search problem
  - Flexibility in terms of being able to approximate different types of functions
  - Interpretability of the resulting model
  - etc.

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Which Models or Model Variants to Use?

- This question is often known as the Model Selection problem
- The answer depends on the goals of the final user - i.e. the Preference Biases of the user
- Establishing which are the preference criteria for a given prediction problem allows to compare and select different models or variants of the same model

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Evaluation Metrics

Classification Problems

The setting

- Given data set $\{< \mathbf{x}_i, y_i >\}_{i=1}^{N}$, where $\mathbf{x}_i$ is a feature vector $< x_1, x_2, \cdots, x_p >$ and $y_i \in \mathcal{Y}$ is the value of the nominal variable Y
- There is an unknown function $Y = f(\mathbf{x})$

The approach

- Assume a functional form $h_\theta(\mathbf{x})$ for the unknown function $f()$, where θ are a set of parameters
- Assume a preference criterion over the space of possible parameterizations of $h()$
- Search for the "optimal" $h()$ according to the criterion and the data set

Classification Error

Error Rate

- Given a set of test cases N_{test} we can obtain the predictions for these cases using some classification model.
- The *Error Rate* ($L_{0/1}$) measures the proportion of these predictions that are incorrect.
- In order to calculate the error rate we need to obtain the information on the true class values of the N_{test} cases.

Classification Error

Error Rate

- Given a test set for which we know the true class the error rate can be calculated as follows,

$$L_{0/1} = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} I(\hat{h}_\theta(\mathbf{x}_i), y_i)$$

where $I()$ is an indicator function such that $I(x, y) = 0$ if $x = y$ and 1 otherwise; and $\hat{h}_\theta(\mathbf{x}_i)$ is the prediction of the model being evaluated for the test case i that has as true class the value y_i.

Confusion Matrices

- A square $nc \times nc$ matrix, where nc is the number of class values of the problem
- The matrix contains the number of times each pair (ObservedClass,PredictedClass) occurred when testing a classification model on a set of cases

		Pred.		
		c_1	c_2	c_3
Obs.	c_1	n_{c_1,c_1}	n_{c_1,c_2}	n_{c_1,c_3}
	c_2	n_{c_2,c_1}	n_{c_2,c_2}	n_{c_2,c_3}
	c_3	n_{c_3,c_1}	n_{c_3,c_2}	n_{c_3,c_3}

- The error rate can be calculated from the information on this table.

An Example in R

```
trueVals <- c("c1","c1","c2","c1","c3","c1","c2","c3","c2","c3")
preds <- c("c1","c2","c1","c3","c3","c1","c1","c3","c1","c2")
confMatrix <- table(trueVals,preds)
confMatrix

##          preds
## trueVals c1 c2 c3
##       c1  2  1  1
##       c2  3  0  0
##       c3  0  1  2

errorRate <- 1-sum(diag(confMatrix))/sum(confMatrix)
errorRate

## [1] 0.6
```

Cost-Sensitive Applications

- In the error rate one assumes that all errors and correct predictions have the same value
- This may not be adequate for some applications

Cost/benefit Matrices

Table where each entry specifies the cost (negative benefit) or benefit of each type of prediction

		Pred.		
		c_1	c_2	c_3
Obs.	c_1	$B_{1,1}$	$C_{1,2}$	$C_{1,3}$
	c_2	$C_{2,1}$	$B_{2,2}$	$C_{2,3}$
	c_3	$C_{3,1}$	$C_{3,2}$	$B_{3,3}$

- Models are then evaluated by the total balance of their predictions, i.e. the sum of the benefits minus the costs.

An Example in R

```
trueVals <- c("c1","c1","c2","c1","c3","c1","c2","c3","c2","c3")
preds <- c("c1","c2","c1","c3","c3","c1","c1","c3","c1","c2")
confMatrix <- table(trueVals,preds)
costMatrix <- matrix(c(10,-2,-4,-2,30,-3,-5,-6,12),ncol=3)
colnames(costMatrix) <- c("predC1","predC2","predC3")
rownames(costMatrix) <- c("obsC1","obsC2","obsC3")
costMatrix

##       predC1 predC2 predC3
## obsC1     10     -2     -5
## obsC2     -2     30     -6
## obsC3     -4     -3     12

utilityPreds <- sum(confMatrix*costMatrix)
utilityPreds

## [1] 28
```

Predicting a Rare Class

E.g. predicting outliers

- Problems with two classes
- One of the classes is much less frequent and it is also the most relevant

		Preds.	
		Pos	Neg
Obs.	Pos	True Positives (TP)	False Negatives (FN))
	Neg	False Positives (FP)	True Negatives (TN)

Precision and *Recall*

		Preds.	
		P	N
Obs.	P	TP	FN
	N	FP	TN

- *Precision* - proportion of the signals (events) of the model that are correct

$$Prec = \frac{TP}{TP + FP}$$

- *Recall* - proportion of the real events that are captured by the model

$$Rec = \frac{TP}{TP + FN}$$

Precision and *Recall*

Examples

		Prevs. P	Prevs. N
Obs.	P	2	2
	N	1	1

$$Precision = \frac{TP}{TP + FP} = \frac{2}{2+1} = 0.667$$

$$Recall = \frac{TP}{TP + FN} = \frac{2}{2+2} = 0.5$$

$$ErrorRate = \frac{2+1}{2+2+1+1} = 0.5$$

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

The F-Measure

Combining Precision and Recall into a single measure

- Sometimes it is useful to have a single measure - e.g. optimization within a search procedure
- Maximizing one of them is easy at the cost of the other (it is easy to have 100% recall - always predict "P").
- What is difficult is to have both of them with high values
- The F-measure is a statistic that is based on the values of precision and recall and allows establishing a trade-off between the two using a user-defined parameter (β),

$$F_\beta = \frac{(\beta^2 + 1) \cdot Prec \cdot Rec}{\beta^2 \cdot Prec + Rec}$$

where β controls the relative importance of *Prec* and *Rec*. If $\beta = 1$ then *F* is the harmonic mean between *Prec* and *Rec*; When $\beta \to 0$ the weight of *Rec* decreases. When $\beta \to \infty$ the weight of *Prec* decreases.

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Regression Problems

The setting

- Given data set $\{< \mathbf{x}_i, y_i >\}_{i=1}^{N}$, where $\mathbf{x}_i$ is a feature vector $< x_1, x_2, \cdots, x_p >$ and $y_i \in \Re$ is the value of the numeric variable Y
- There is an unknown function $Y = f(\mathbf{x})$

The approach

- Assume a functional form $h_\theta(\mathbf{x})$ for the unknown function $f()$, where θ are a set of parameters
- Assume a preference criterion over the space of possible parameterizations of $h()$
- Search for the "optimal" $h()$ according to the criterion and the data set

Measuring Regression Error

Mean Squared Error

- Given a set of test cases N_{test} we can obtain the predictions for these cases using some regression model.
- The *Mean Squared Error* (*MSE*) measures the average squared deviation between the predictions and the true values.
- In order to calculate the value of *MSE* we need to have both the predicitons and the true values of the N_{test} cases.

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Measuring Regression Error

Mean Squared Error (cont.)

- If we have such information the *MSE* can be calculated as follows,

$$MSE = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} (\hat{y}_i - y_i)^2$$

where $\hat{y}_i$ is the prediction of the model under evaluation for the case i and y_i the respective true target variable value.

- Note that the *MSE* is measured in a unit that is squared of the original variable scale. Because of the this is sometimes common to use the *Root Mean Squared Error* (*RMSE*), defined as $RMSE = \sqrt{MSE}$

Measuring Regression Error

Mean Absolute Error

- The *Mean Absolute Error* (*MAE*) measures the average absolute deviation between the predictions and the true values.
- The value of the *MAE* can be calculated as follows,

$$MAE = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} |\hat{y}_i - y_i|$$

where $\hat{y}_i$ is the prediction of the model under evaluation for the case i and y_i the respective true target variable value.

- Note that the *MAE* is measured in the same unit as the original variable scale.

Relative Error Metrics

- Relative error metrics are unit less which means that their scores can be compared across different domains.
- They are calculated by comparing the scores of the model under evaluation against the scores of some baseline model.
- The relative score is expected to be a value between 0 and 1, with values nearer (or even above) 1 representing performances as bad as the baseline model, which is usually chosen as something too naive.

Relative Error Metrics (cont.)

- The most common baseline model is the constant model consisting of predicting for all test cases the average target variable value calculated in the training data.
- The *Normalized Mean Squared Error* (*NMSE*) is given by,

$$NMSE = \frac{\sum_{i=1}^{N_{test}} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{N_{test}} (\bar{y} - y_i)^2}$$

- The *Normalized Mean Absolute Error* (*NMAE*) is given by,

$$NMAE = \frac{\sum_{i=1}^{N_{test}} |\hat{y}_i - y_i|}{\sum_{i=1}^{N_{test}} |\bar{y} - y_i|}$$

Relative Error Metrics (cont.)

- The *Mean Average Percentage Error* (*MAPE*) is given by,

$$MAPE = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \frac{|\hat{y}_i - y_i|}{y_i}$$

- The *Correlation* between the predictions and the true values ($\rho_{\hat{y},y}$) is given by,

$$\rho_{\hat{y},y} = \frac{\sum_{i=1}^{N_{test}} (\hat{y}_i - \bar{\hat{y}})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N_{test}} (\hat{y}_i - \bar{\hat{y}})^2 \sum_{i=1}^{N_{test}} (y_i - \bar{y})^2}}$$

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

An Example in R

```
trueVals <- c(10.2,-3,5.4,3,-43,21,
              32.4,10.4,-65,23)
preds <-  c(13.1,-6,0.4,-1.3,-30,1.6,
            3.9,16.2,-6,20.4)
mse <- mean((trueVals-preds)^2)
mse
```

```
## [1] 494
```

```
rmse <- sqrt(mse)
rmse
```

```
## [1] 22.23
```

```
mae <- mean(abs(trueVals-preds))
mae
```

```
## [1] 14.35
```

```
nmse <- sum((trueVals-preds)^2) /
        sum((trueVals-mean(trueVals))^2)
nmse
```

```
## [1] 0.5916
```

```
nmae <- sum(abs(trueVals-preds)) /
        sum(abs(trueVals-mean(trueVals)))
nmae
```

```
## [1] 0.6563
```

```
mape <- mean(abs(trueVals-preds)/trueVals)
mape
```

```
## [1] 0.2908
```

```
corr <- cor(trueVals,preds)
corr
```

```
## [1] 0.6745
```

Linear Discriminant

The Linear Discriminant

The Idea

Search for linear combinations of the variables that better separate between the objects of the classes

The formalism for two classes - Fisher linear discriminant

- Let $\hat{C}$ the pooled sample covariance matrix

$$\hat{C} = \frac{1}{n_1 + n_2}\left(n_1 \hat{C}_1 + n_2 \hat{C}_2\right)$$

where n_i is the number of training cases per class and $\hat{C}_i$ are the $p \times p$ sample covariance matrices for each class. The sample covariance between two variables is given by

$$Cov(X, Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

The Linear Discriminant (cont.)

The formalism (cont.)

- The following is the score of the separation provided by a p-dimensional vector $\mathbf{w}$,

$$S_{\mathbf{w}} = \frac{\mathbf{w}^T \hat{\mu}_1 - \mathbf{w}^T \hat{\mu}_2}{\mathbf{w}^T \hat{C} \mathbf{w}}$$

- Given this score the goal is to find the vector $\mathbf{w}$ that maximizes it. There is a solution for this maximization problem given by,

$$\hat{\mathbf{w}}_{lda} = \hat{C}^{-1}(\hat{\mu}_1 - \hat{\mu}_2)$$

- Canonical discriminant functions extend the idea for more than two classes

Canonical Discriminant Functions

Example

```
library(MASS)
data(iris)
lda(Species ~ ., iris)


## Call:
## lda(Species ~ ., data = iris)
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##     0.3333     0.3333     0.3333
##
## Group means:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa            5.006       3.428        1.462       0.246
## versicolor        5.936       2.770        4.260       1.326
## virginica         6.588       2.974        5.552       2.026
##
## Coefficients of linear discriminants:
##                    LD1     LD2
## Sepal.Length  0.8294  0.0241
## Sepal.Width   1.5345  2.1645
## Petal.Length -2.2012 -0.9319
## Petal.Width  -2.8105  2.8392
##
## Proportion of trace:
##    LD1    LD2
```

Using LDA for prediction in R

```
sp <- sample(1:150,100)
tr <- iris[sp,]
ts <- iris[-sp,]
l <- lda(Species ~ ., tr)
preds <- predict(l,ts)
(mtrx <- table(preds$class,ts$Species))

##
##              setosa versicolor virginica
##   setosa         16          0         0
##   versicolor      0         18         0
##   virginica       0          1        15

(err <- 1-sum(diag(mtrx))/sum(mtrx))

## [1] 0.02
```


Hands on LDAs - the Vehicle data set

The data set `Vehicle` is available in package **mlbench**. Load it and explore its help page to grab a minimal understanding of the data and then answer the following questions:

1. Obtain a random split of the data into two sub-sets using the proportion 80%-20%.
2. Obtain a linear discriminant using the larger set.
3. Obtain the predictions of the obtained model on the smaller set.
4. Obtain a confusion matrix of the predictions and calculate the respective accuracy.

Multiple Linear Regression

Multiple Linear Regression

- Multiple linear regression is probably the most used statistical method
- It is one of the many possible approaches to the multiple regression problem where given a training data set $\mathbf{D} = \{\langle \mathbf{x}_i, y_i \rangle\}_{i=1}^{n}$ we want to obtain an approximation of the unknown regression function $f()$ that maps the predictors values into a target continuous variable value.
- In matrix notation we have $\mathbf{D} = \mathbf{X}|\mathbf{Y}$, where $\mathbf{X}$ is a matrix $n \times p$, and $\mathbf{Y}$ is a matrix $n \times 1$.

Multiple Linear Regression (cont.)

- A regression model $r_D(.)$ can be seen as a function that transforms a vector of values of the predictors, $\mathbf{x}$, into a real number, Y. This model is an approximation of the unknown $f()$ function.
- Regression models assume the following relationship, $y_i = r(\beta, \mathbf{x}_i) + \epsilon_i$, where $r(\beta, \mathbf{x}_i)$ is a regression model with parameters β and ϵ_i are observation errors.
- The goal of a learning method is to obtain the model parameters β that minimize a certain preference criterion.

Multiple Linear Regression (cont.)

- In the case of multiple linear regression the functional form that is assumed is the following:

$$Y = \beta_0 + \beta_1 \cdot X_1 + \cdots + \beta_p \cdot X_p$$

- The goal is to find the vector of parameters β that minimizes the sum of the squared errors $\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 \cdot X_1 + \cdots + \beta_p \cdot X_p))^2$

Multiple Linear Regression

Pros and Cons

- Well-known and over-studied topic with many variants of this simple methodology (e.g. Drapper and Smith, 1981)
- Simple and effective approach when the “linearity” assumption is adequate to the data.
- Form of the model is intuitive - a set of additive effects of each variable towards the prediction
- Computationally very efficient
- Too strong assumptions on the shape of the unknown regression function

Drapper and Smith (1981): Applied Regression Analysis, 2nd edition. Wiley Series in Probability and Mathematical Statistics.

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Obtaining Multiple Linear Regression Models in R

```
library(DMwR)
data(algae)
algae <- algae[-c(62, 199), ]  # the 2 incomplete samples
clean.algae <- knnImputation(algae)  # lm() does not handle NAs!
la1 <- lm(a1 ~ ., clean.algae[, 1:12])
la1

##
## Call:
## lm(formula = a1 ~ ., data = clean.algae[, 1:12])
##
## Coefficients:
##  (Intercept)  seasonspring  seasonsummer  seasonwinter    sizemedium
##     42.94206       3.72698       0.74760       3.69295       3.26373
##    sizesmall      speedlow   speedmedium           mxPH          mnO2
##      9.68214       3.92208       0.24676      -3.58912       1.05264
##           Cl           NO3           NH4          oPO4           PO4
##     -0.04017      -1.51124       0.00163      -0.00543      -0.05224
##          Chla
##     -0.08802
```

Obtaining Multiple Linear Regression Models in R (cont.)

```
summary(la1)
```

```
##
## Call:
## lm(formula = a1 ~ ., data = clean.algae[, 1:12])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -37.68 -11.89  -2.57   7.41  62.19
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  42.94206   24.01088    1.79   0.0754 .
## seasonspring  3.72698    4.13774    0.90   0.3689
## seasonsummer  0.74760    4.02071    0.19   0.8527
## seasonwinter  3.69295    3.86539    0.96   0.3406
## sizemedium    3.26373    3.80205    0.86   0.3918
## sizesmall     9.68214    4.17997    2.32   0.0217 *
## speedlow      3.92208    4.70631    0.83   0.4057
## speedmedium   0.24676    3.24187    0.08   0.9394
## mxPH         -3.58912    2.70353   -1.33   0.1860
## mnO2          1.05264    0.70502    1.49   0.1372
## Cl           -0.04017    0.03366   -1.19   0.2343
## NO3          -1.51124    0.55134   -2.74   0.0067 **
## NH4           0.00163    0.00100    1.63   0.1052
## oPO4         -0.00543    0.03988   -0.14   0.8918
## PO4          -0.05224    0.03075   -1.70   0.0911 .
## Chla          0.08802    0.08000    1.10   0.2727
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.6 on 182 degrees of freedom
```


The Diagnostic Information of the summary() Call

- Distribution of the residuals (errors) of the model - should have mean zero and should be normally distributed and as small as possible
- Estimate of each coefficient and respective standard error
- Test of the hypothesis that each coefficient is null, i.e. $H0 : \beta_i = 0$
  - Uses a t-test
  - Calculates a t-value as β_i / s_{β_i}
  - Presents a column (Pr(>t)) with the level at which the hypothesis is rejected. A value of 0.0001 would mean that we are 99.99% confident that the coefficient is not null
  - Coefficients for which we can reject the hypothesis are tagged with a symbol

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

The Diagnostic Information of the summary() Call (cont.)

- We are also given the R^2 coefficients (multiple and adjusted). These coefficients indicate the degree of fit of the model to the data, i.e. the proportion of variance explained by the model. Values near 1 (100%) are better. The adjusted coefficient is more demanding as it takes into account the size of the model
- Finally, there is also a test of the hypothesis that there is no dependence of the target variable on the predictors, i.e. $H0 : \beta_1 = \beta_2 = \cdots = \beta_p = 0$. The F-statistic is used with this purpose. R provides the confidence level at which we are sure to reject this hypothesis. A p-level of 0.0001 means that we are 99.99% confident that the hypothesis is not true.

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Simplifying the Linear Model

```
final.la1 <- step(la1)
```

```
summary(final.la1)
```

```
##
## Call:
## lm(formula = a1 ~ size + mxPH + Cl + NO3 + PO4, data = clean.algae[,
##     1:12])
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -28.87 -12.73  -3.74   8.42  62.93
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  57.2855    20.9613    2.73   0.0069 **
## sizemedium    2.8005     3.4019    0.82   0.4114
## sizesmall    10.4064     3.8224    2.72   0.0071 **
## mxPH         -3.9708     2.4820   -1.60   0.1113
## Cl           -0.0523     0.0317   -1.65   0.1003
## NO3          -0.8953     0.3515   -2.55   0.0116 *
## PO4          -0.0591     0.0112   -5.29  3.3e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.5 on 191 degrees of freedom
## Multiple R-squared:  0.353, Adjusted R-squared:  0.332
## F-statistic: 17.3 on 6 and 191 DF,  p-value: 5.55e-16
```

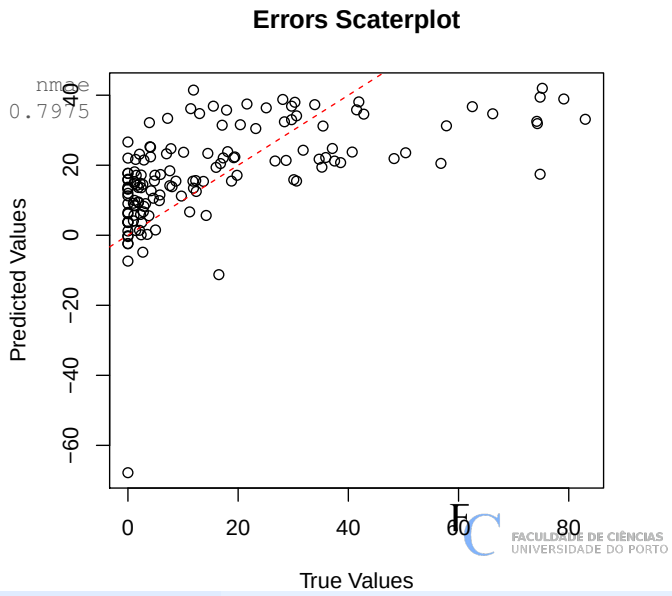
Using the Models for Prediction

```
data(testAlgae)
data(algaeSols)
clean.test.algae <- knnImputation(test.algae,
    k = 10, distData = clean.algae[, 1:11])
preds <- predict(final.la1,clean.test.algae)
regr.eval(algae.sols$a1,preds,
          train.y=clean.algae$a1)
```

```
##       mae       mse      rmse      mape      nmse
##  12.7843 296.0934  17.2074       Inf    0.7048
```

But there are no negative algae frequencies!...

```
plot(algae.sols$a1,preds,main='Errors Scaterplot',
     ylab='Predicted Values',xlab='True Values')
abline(0,1,col='red',lty=2)
```



Hands on Linear Regression - the Boston data set

The data set `Boston` is available in package **MASS**. Load it and explore its help page to grab a minimal understanding of the data and then answer the following questions:

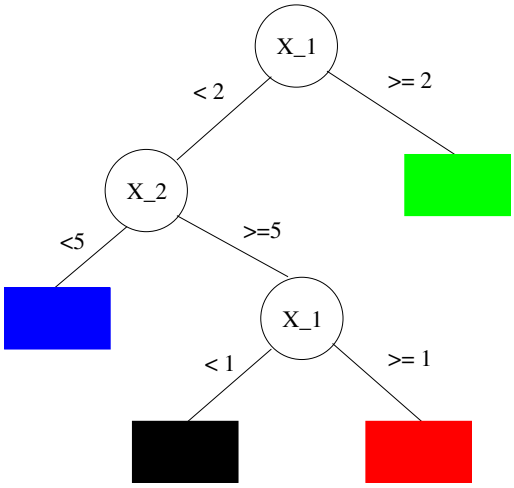
1. Obtain a random split of the data into two sub-sets using the proportion 70%-30%.
2. Obtain a multiple linear regression model using the larger set.
3. Check the diagnostic information provided for the model.
4. Obtain the predictions of the obtained model on the smaller set.
5. Obtain the mean squared error of these predictions and also an error scatter plot.

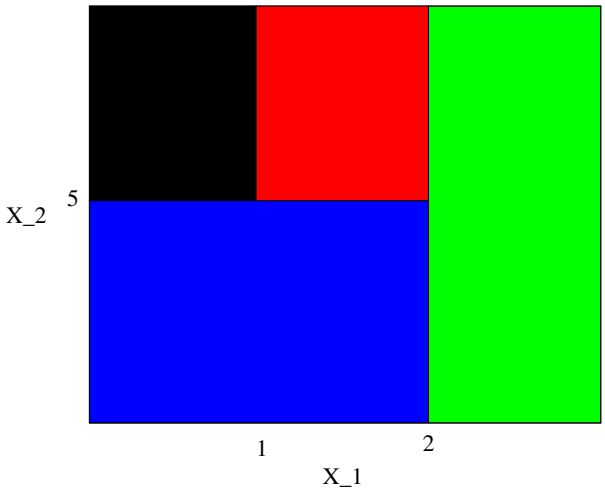
Tree-based Models

Tree-based Models

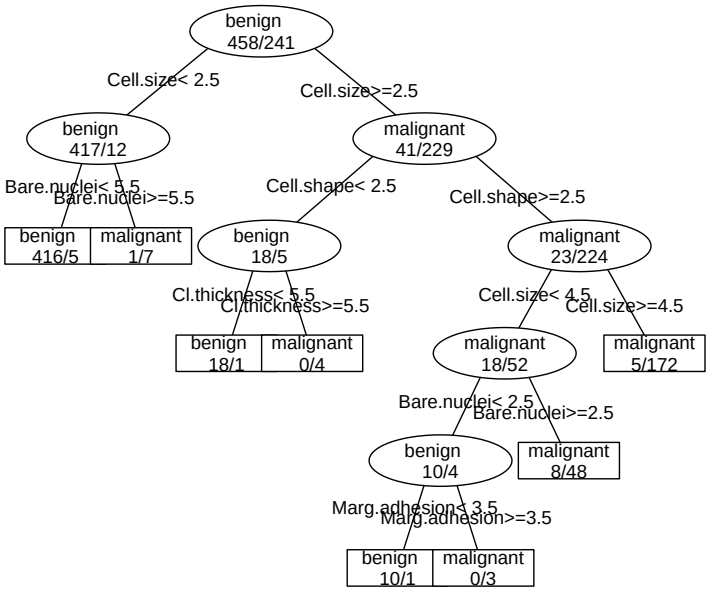
- Tree-based models (both classification and regression trees) are models that provide as result a model based on logical tests on the input variables
- These models can be seen as a partitioning of the input space defined by the input variables
- This partitioning is defined based on carefully chosen logical tests on these variables
- Within each partition all cases are assigned the same prediction (either a class label or a numeric value)
- Tree-based models are known by their (i) computational efficiency; (ii) interpretable models; (iii) embedded variable selection; (iv) embedded handling of unknown variable values and (v) few assumptions on the unknown function being approximated

An Example of Trees Partitioning

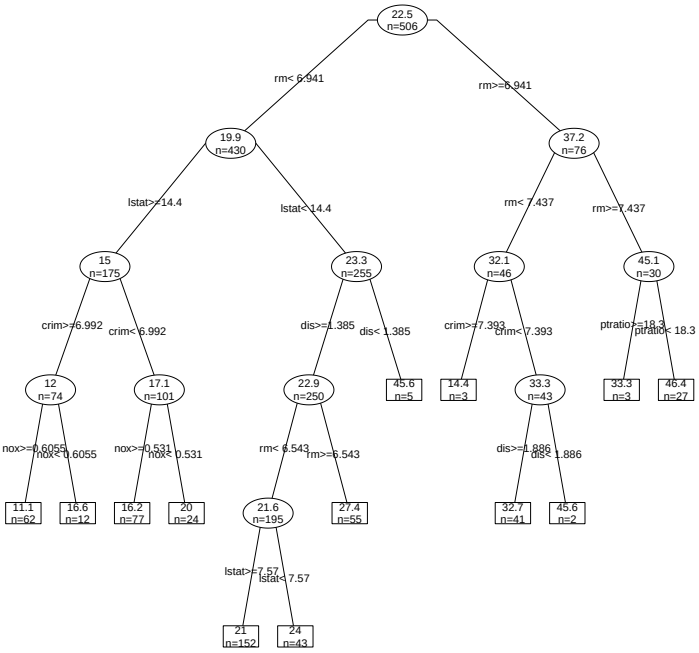




An Example of a Classification Tree



An Example of a Regression Tree



FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Tree-based Models

- Most tree-based models are binary trees with logical tests on each node
- Tests on numerical predictors take the form $x_i < \alpha$, with $\alpha \in \Re$
- Tests on nominal predictors take the form $x_j \in \{v_1, \cdots, v_m\}$
- Each path from the top (root) node till a leaf can be seen as a logical condition defining a region of the predictors space.
- All observations "falling" on a leaf will get the same prediction
  - the majority class of the training cases in that leaf for classification trees
  - the average value of the target variable for regression trees
- The prediction for a new test case is easily obtained by following a path from the root till a leaf according to the case predictors values

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

The Recursive Partitioning Algorithm

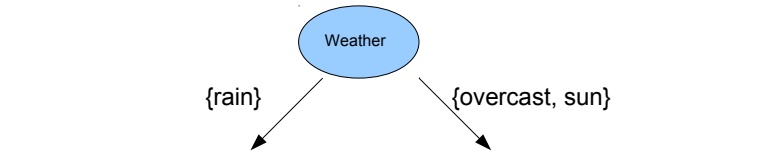
1: **function** RECURSIVEPARTITIONING(D)
Input : D, a sample of cases, $\{\langle x_{i,1}, \cdots, x_{i,p}, y_i \rangle\}_{i=1}^{N_{train}}$
Output : t, a tree node

2: **if** <TERMINATION CRITERION> **then**
3: **Return** a leaf node with the majority class in D
4: **else**
5: $t \leftarrow$ new tree node
6: $t.split \leftarrow$ <FIND THE BEST PREDICTORS TEST>
7: $t.leftNode \leftarrow$ RecursivePartitioning($\mathbf{x} \in D : \mathbf{x} \vDash t.split$)
8: $t.rightNode \leftarrow$ RecursivePartitioning($\mathbf{x} \in D : \mathbf{x} \nvDash t.split$)
9: **Return** the node t
10: **end if**
11: **end function**

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

The Recursive Partitioning Algorithm - an example

Weather	Temp.	Humidity	Wind	Decision
rain	26	high	15	dontPlay
rain	35	normal	102	dontPlay
overcast	27	high	99	Play
overcast	26	normal	97	Play
rain	12	high	120	Play
overcast	21	normal	74	Play
sun	30	normal	89	dontPlay
sun	19	high	111	dontPlay
sun	14	normal	81	Play
overcast	10	normal	70	Play
rain	11	normal	95	Play
rain	15	high	94	Play
sun	19	high	41	dontPlay
sun	35	normal	38	dontPlay
rain	29	high	79	dontPlay
rain	26	normal	75	dontPlay
overcast	30	high	108	Play
overcast	30	normal	16	Play
rain	33	high	96	Play
overcast	30	normal	13	Play
sun	32	normal	55	dontPlay
sun	11	high	108	dontPlay
sun	33	normal	103	Play
overcast	14	normal	32	Play
rain	28	normal	44	Play
rain	21	high	84	Play
sun	29	high	105	dontPlay



Weather	Temp	Humidity	Wind	Decision
rain	26	high	15	dontPlay
rain	35	normal	102	dontPlay
rain	12	high	120	Play
rain	11	normal	95	Play
rain	15	high	94	Play
rain	29	high	79	dontPlay
rain	26	normal	75	dontPlay
rain	33	high	96	Play
rain	28	normal	44	Play
rain	21	high	84	Play

Weather	Temp	Humidity	Wind	Decision
overcast	27	high	99	Play
overcast	26	normal	97	Play
overcast	21	normal	74	Play
overcast	10	normal	70	Play
overcast	30	high	108	Play
overcast	30	normal	16	Play
overcast	30	normal	13	Play
overcast	14	normal	32	Play
sun	30	normal	89	dontPlay
sun	19	high	111	dontPlay
sun	14	normal	81	Play
sun	19	high	41	dontPlay
sun	35	normal	38	dontPlay
sun	32	normal	55	dontPlay
sun	11	high	108	dontPlay
sun	33	normal	103	Play
sun	29	high	105	dontPlay
sun	15	normal	63	dontPlay

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

The Recursive Partitioning Algorithm (cont.)

Key Issues of the RP Algorithm

- When to stop growing the tree - termination criterion
- Which value to put on the leaves
- How to find the best split test

The Recursive Partitioning Algorithm (cont.)

When to Stop?

Too large trees tend to overfit the training data and will perform badly on new data - a question of reliability of error estimates

Which value?

Should be the value that better represents the cases in the leaves

What are the good tests?

A test is good if it is able to split the cases of sample in such a way that they form partitions that are “purer” than the parent node

Classification vs Regression Trees

- They are both grown using the Recursive Partitioning algorithm
- The main difference lies on the used preference criterion
- This criterion has impact on:
  - The way the best test for each node is selected
  - The way the tree avoids over fitting the training sample
- Classification trees typically use criteria related to error rate (e.g. the Gini index, the Gain ratio, entropy, etc.)
- Regression trees typically use the least squares error criterion

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

How to Evaluate a Test in Classification Trees?

Gini Impurity

- The Gini index of a data set D where each example belongs to one of c classes is given by,

$$Gini(D) = 1 - \sum_{i=1}^{c} p_i^2$$

  where p_i is the probability of class i usually estimated with the observed frequency on the training data
- If the data set is split on a logical test T then the resulting Gini index is given by,

$$Gini_T(D) = \frac{|D_T|}{|D|} Gini(D_T) + \frac{|D_{\neg T}|}{|D|} Gini(D_{\neg T})$$

- In this context the reduction in impurity given by T is,

$$\Delta Gini_T(D) = Gini(D) - Gini_T(D)$$

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Gini Impurity - an example

Tempo	Temp	Humidade	Vento	Decisão
chuva	26	alta	15	nãoJogar
chuva	35	normal	102	nãoJogar
chuva	12	alta	120	jogar
chuva	11	normal	95	jogar
chuva	15	alta	94	jogar
chuva	29	alta	79	nãoJogar
chuva	26	normal	75	nãoJogar
chuva	33	alta	96	jogar
chuva	28	normal	44	jogar
chuva	21	alta	84	jogar
enublado	27	alta	99	jogar
enublado	26	normal	97	jogar
enublado	21	normal	74	jogar
enublado	10	normal	70	jogar
enublado	30	alta	108	jogar
enublado	30	normal	16	jogar
enublado	30	normal	13	jogar
enublado	14	normal	32	jogar
sol	30	normal	89	nãoJogar
sol	19	alta	111	nãoJogar
sol	14	normal	81	jogar
sol	19	alta	41	nãoJogar
sol	35	normal	38	nãoJogar
sol	32	normal	55	nãoJogar
sol	11	alta	108	nãoJogar
sol	33	normal	103	jogar
sol	29	alta	105	nãoJogar
sol	15	normal	63	nãoJogar

$$Gini(D) = 1 - \left(\left(\frac{16}{16+12}\right)^2 + \left(\frac{12}{16+12}\right)^2\right) = 0.49$$

$$\begin{aligned} Gini_{Tempo \in \{chuva\}}(D) =& \frac{10}{28} \cdot Gini(D_{Tempo \in \{chuva\}}) \\ &+ \frac{18}{28} \cdot Gini(D_{Tempo \notin \{chuva\}}) \\ =& 0.489 \end{aligned}$$

$$Gini(D_{Tempo \in \{chuva\}}) = 1 - \left(\left(\frac{4}{4+6}\right)^2 + \left(\frac{6}{4+6}\right)^2\right) = 0.48$$

$$Gini(D_{Tempo \notin \{chuva\}}) = 1 - \left(\left(\frac{8}{8+10}\right)^2 + \left(\frac{10}{8+10}\right)^2\right) = 0.49$$

$$\Delta Gini_{Tempo \in \{chuva\}}(D) = 0.49 - 0.489 = 0.001$$

Calcule o valor de $\Delta Gini_{Tempo \in \{enublado\}}(D)$

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Which Tests are Tried?

Numeric Predictors

- Given a set of data D and a continuous variable A let $V_{A,D}$ be the set of values of A occurring in D
- Start by ordering the set $V_{A,D}$
- Evaluate all tests $A < x$ where x takes as values all mid-points between every successive value in the ordered set

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Numeric Predictors - an example

- Given the unsorted values of `Temp`:
  26 35 27 26 12 21 30 19 14 10 11 15 19 35 29 26 30 30 33 30 32 11 33 14 28 21 29 15
- Start by ordering them:
  10 11 11 12 14 14 15 15 19 19 21 21 26 26 26 27 28 29 29 30 30 30 30 32 33 33 35 35
- Then try (i.e. evaluate) all tests in between each value:
  - $Temp < 10.5$
  - $Temp < 11.5$
  - $Temp < 13$
  - $Temp < 14.5$
  - etc.
- Choose the test with the best score to be the best test in variable *Temp*

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Which Tests are Tried?

Nominal Predictors

- Given a set of data D and a nominal variable A let $V_{A,D}$ be the set of values of A occurring in D
- Evaluate all possible combinations of subset of values in $V_{A,D}$
- Note that there are some optimizations that reduce the computational complexity of this search

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Nominal Predictors - an example

- Given the values of `Tempo`:
  chuva, enublado, sol
- Try (i.e. evaluate) all subsets of these values:
  - *Tempo* $\in \{chuva\}$
  - *Temp* $\in \{enublado\}$
  - *Temp* $\in \{sol\}$
- Choose the test with the best score to be the best test in variable *Tempo*

How to Evaluate a Test in Regression Trees?

Least Squares Regression Trees

- Regression trees are usually grown by trying to **minimize the sum of the squared errors**, leading to *Least Squares Regression Trees*
- According to the LS (Least Squares) criterion the error in a node of the tree is given by,

$$Err(t) = \frac{1}{n_t} \sum_{\langle \mathbf{x}_i, y_i \rangle \in D_t} (y_i - k_t)^2$$

where D_t is the sample of cases in node t, n_t is the cardinality of this set and k_t is the constant used in the node

- It can be easily proven that the constant k that minimizes this error is the average target variable value of the cases in the leaf.

Least Squares Regression Trees

- Any logical test s divides the cases in D_t in two partitions, D_{t_L} and D_{t_R}. The resulting pooled error is given by,

$$Err(t,s) = \frac{n_{t_L}}{n_t} \times Err(t_L) + \frac{n_{t_R}}{n_t} \times Err(t_R)$$

  where n_{t_L}/n_t (n_{t_R}/n_t) is the proportion of cases going to the left (right) branch of t
- We can estimate the value of the split s by the respective error reduction,

$$\Delta(s,t) = Err(t) - Err(t,s)$$

- Finding the best split test for a node t involves evaluating all possible tests for this node using the above equations

Least Squares Regression Trees (cont.)

- For continuous variables this requires a sorting operation on the values of this variable occurring in the node
- After this sorting, a fast incremental algorithm (Torgo, 1999) can be used to find the best cut-point value for the test
- With respect to nominal variables, Breiman and colleagues (1984) have proved a theorem that avoids trying all possible combinations of values, reducing the computational complexity of this task from $O(2^{v-1}-1)$ to $O(v-1)$, where v is the number of values of the nominal variable

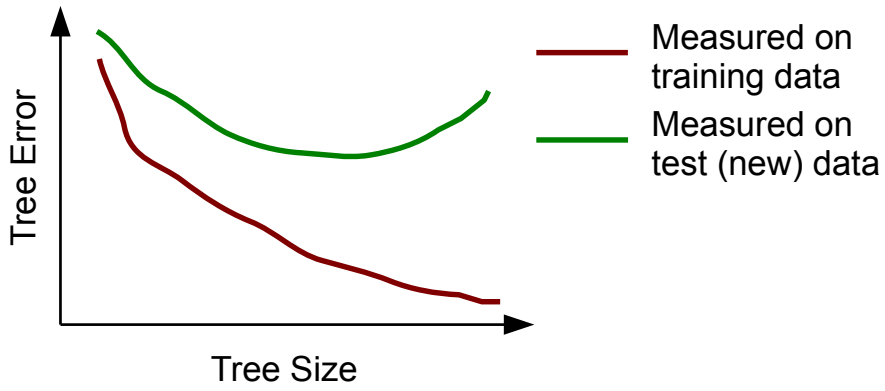
Breiman et al. (1984): Classification and Regression Trees

Torgo L. (1999): Inductive learning of tree-based regression models. PhD thesis, Department of Computer Science, Faculty of Sciences, University of Porto.

Torgo,L. (2011) : Regression Trees. In Encyclopedia of Machine Learning, C.Sammut and G.I.Webb (Eds.). Pages 842–845, Springer.

Deciding when to stop growing the trees

- The scores discussed before keep improving as we grow the tree
- At an extreme, an overly large tree, will perfectly fit the given training data (i.e. all cases are correctly predicted by the tree)
- Such huge trees are said to be overfitting the training data and will most probably perform badly on a new set of data (a test set), as they have captured spurious characteristics of the training data



Deciding when to stop growing the trees - 2

- As we go down in the tree the decisions on the tests are made on smaller and smaller sets, and thus potentially less reliable decisions are made
- The standard procedure in tree learning is to grow an overly large tree and then use some statistical procedure to prune unreliable branches from this tree. The goal of this procedure is to try to obtain reliable estimates of the error of the tree. This procedure is usually called post-prunning a tree.
- An alternative procedure (not so frequently used) is to decide during tree growth when to stop. This is usually called pre-prunning.

(Post-)Pruning a Tree

Cost-complexity and Error-complexity Pruning

- Grown and overly large tree
- Generate a sequence of sub-trees
  - Error-complexity criterion for regression trees
  - Cost-complexity criterion for regression trees
- Use cross validation to estimate the error of these trees
- Use the x-SE rule to select the best sub-tree

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Classification and Regression Trees in R

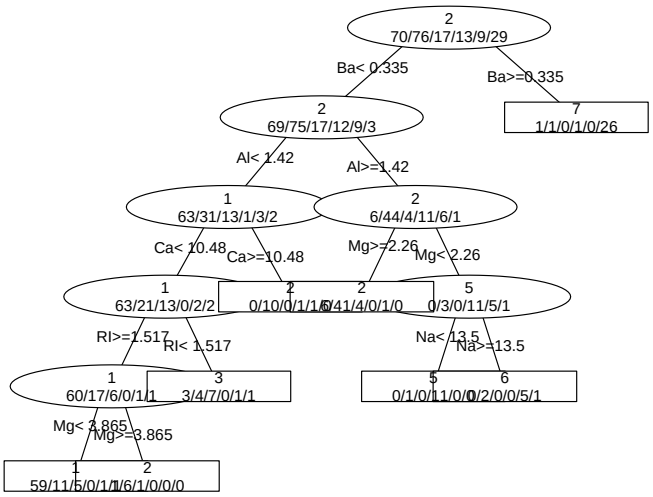
The package `rpart`

- Package `rpart` implements most of the ideas of the system CART that was described in the book “Classification and Regression Trees” by Breiman and colleagues
- This system is able to obtain classification and regression trees.
- For classification trees it uses the Gini score to grow the trees and it uses Cost-Complexity post-pruning to avoid over fitting
- For regression trees it uses the least squares error criterion and it uses Error-Complexity post-pruning to avoid over fitting
- On package `DMwR` you may find function `rpartXse()` that grows and prunes a tree in a way similar to CART using the above infra-structure

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Illustration using a classification task - *Glass*

```
library(DMwR)
data(Glass,package='mlbench')
ac <- rpartXse(Type  ~ .,Glass)
prettyTree(ac)
```



How to use the trees for Predicting?

```
tr <- Glass[1:200,]
ts <- Glass[201:214,]
ac <- rpartXse(Type  ~ .,tr)
predict(ac,ts)
```

```
##       1      2      3       5       6      7
## 201 0.0 0.0000 0.0000 0.09091 0.00000 0.9091
## 202 0.2 0.2667 0.4667 0.00000 0.06667 0.0000
## 203 0.0 0.0000 0.0000 0.09091 0.00000 0.9091
## 204 0.0 0.0000 0.0000 0.09091 0.00000 0.9091
## 205 0.0 0.0000 0.0000 0.09091 0.00000 0.9091
## 206 0.0 0.0000 0.0000 0.09091 0.00000 0.9091
## 207 0.0 0.0000 0.0000 0.09091 0.00000 0.9091
## 208 0.0 0.0000 0.0000 0.09091 0.00000 0.9091
## 209 0.0 0.0000 0.0000 0.09091 0.00000 0.9091
## 210 0.0 0.0000 0.0000 0.09091 0.00000 0.9091
## 211 0.0 0.0000 0.0000 0.09091 0.00000 0.9091
## 212 0.0 0.0000 0.0000 0.09091 0.00000 0.9091
## 213 0.0 0.0000 0.0000 0.09091 0.00000 0.9091
## 214 0.0 0.0000 0.0000 0.09091 0.00000 0.9091
```

How to use the trees for Predicting? (cont.)

```
predict(ac,ts,type='class')
```

```
## 201 202 203 204 205 206 207 208 209 210 211 212 213 214
##   7   3   7   7   7   7   7   7   7   7   7   7   7   7
## Levels: 1 2 3 5 6 7
```

```
ps <- predict(ac,ts,type='class')
table(ps,ts$Type)
```

```
##
## ps   1  2  3  5  6  7
##   1  0  0  0  0  0  0
##   2  0  0  0  0  0  0
##   3  0  0  0  0  0  1
##   5  0  0  0  0  0  0
##   6  0  0  0  0  0  0
##   7  0  0  0  0  0 13
```

```
mc <- table(ps,ts$Type)
err <- 100*(1-sum(diag(mc))/sum(mc))
err
```

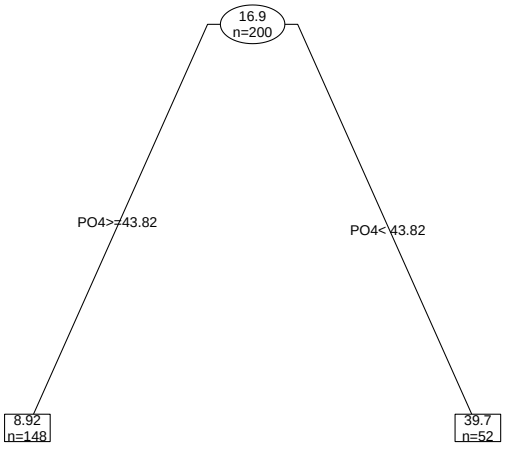
```
## [1] 7.143
```


Illustration using a regression task

Forecasting Algae a1

```
library(DMwR)
data(algae)
d <- algae[,1:12]
ar <- rpartXse(a1 ~ .,d)
```

```
prettyTree(ar)
```



How to use the trees for Predicting?

```
tr <- d[1:150,]
ts <- d[151:200,]
ar <- rpartXse(a1 ~ .,tr)
preds <- predict(ar,ts)
mae <- mean(abs(preds-ts$a1))
mae

## [1] 12.28

cr <- cor(preds,ts$a1)
cr

## [1] 0.5124
```


Hands on Tree-based Models - the Wines data

File `Wine.Rdata` contains two data frames with data on green wine quality: (i) `redWine` and (ii) `whiteWine`. Each of these data sets contains a series of tests with green wines (red and white). For each of these tests the values of several physicochemical variables together with a quality score assigned by wine experts (column `quality`).

1. Build a regression tree for the white wines data set
2. Obtain a graph of the obtained regression tree
3. Apply the tree to the data used to obtain the model and calculate the mean squared error of the predictions
4. Split the data set in two parts: 70% of the tests and the remaining 30%. Using the larger part to obtain a regression tree and apply it to the other part. Calculate again the mean squared error. Compare with the previous scores and comment.

Bayesian Classification
Naive Bayes

Bayesian Classification

- Bayesian classifiers are statistical classifiers - they predict the probability that a case belongs to a certain class
- Bayesian classification is based on the Bayes' Theorem (next slide)
- A particular class of Bayesian classifiers - the Naive Bayes Classifier - has shown rather competitive performance on several problems even when compared to more "sophisticated" methods
- Naive Bayes is available in R on package **e1071**, through function `naiveBayes()`

The Bayes' Theorem - 1

- Let D be a data set formed by n cases $\{\langle \mathbf{x}, y \rangle\}_{i=1}^{n}$, where $\mathbf{x}$ is a vector of p variable values and y is the value on a target nominal variable $Y \in \mathcal{Y}$
- Let H be a hypothesis that states that a certain test cases belongs to a class $c \in \mathcal{Y}$
- Given a new test case $\mathbf{x}$ the goal of classification is to estimate $P(H|\mathbf{x})$, i.e. the probability that H holds given the evidence $\mathbf{x}$
- More specifically, if $\mathcal{Y}$ is the domain of the target variable Y we want to estimate the probability of each of the possible values given the test case (evidence) $\mathbf{x}$

The Bayes' Theorem - 2

- $P(H|\mathbf{x})$ is called the **posterior probability**, or *a posteriori probability*, of H conditioned on $\mathbf{x}$
- We can also talk about $P(H)$, the **prior probability**, or *a priori probability*, of the hypothesis H
- Notice that $P(H|\mathbf{x})$ is based on more information than $P(H)$, which is independent of the observation $\mathbf{x}$
- Finally, we can also talk about $P(\mathbf{x}|H)$ as the posterior probability of $\mathbf{x}$ conditioned on H

Bayes' Theorem

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

The Naive Bayes Classifier

How it works?

- We have a data set D with cases belonging to one of m classes $c_1, c_2, \cdots, c_m$
- Given a new test case $\mathbf{x}$ this classifier produces as prediction the class that has the highest estimated probability, i.e. $\max_{i \in \{1,2,\cdots,m\}} P(c_i|\mathbf{x})$
- Given that $P(\mathbf{x})$ is constant for all classes, according to the Bayes Theorem the class with the highest probability is the one maximizing the quantity $P(\mathbf{x}|c_i)P(c_i)$

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

The Naive Bayes Classifier - 2

How it works? (cont.)

- The class priors $P(c_i)$'s are usually estimated from the training data as $|D_{c_i}|/|D|$, where $|D_{c_i}|$ is the number of cases in D that belong to class c_i
- Regards the quantities $P(\mathbf{x}|c_i)$'s the correct computation would be computationally very demanding. The Naive Bayes classifier simplifies this task by naively assuming *class condition independence*. This essentially resumes to assuming that there is no dependence relationship among the predictors of the problem. This independence allows us to use,

$$P(\mathbf{x}|c_i) = \prod_{k=1}^{p} P(x_k|c_i)$$

- Note that the quantities $P(x_1|c_i), P(x_2|c_i), \cdots, P(x_p|c_i)$ can be easily estimated from the training data

The Naive Bayes Classifier - 3

How to estimate $P(x_1|c_i), P(x_2|c_i), \cdots, P(x_p|c_i)$

- If X_k is a nominal variable then $P(x_k|c_i)$ is the number of values in D of class c_i that have the value x_k in variable (predictor) X_k, divided by the number of cases in D of class c_i
- If X_k is a numeric variable things are slightly more complex. We typically assume that the variable follows a normal distribution with a mean μ and standard deviation σ, defined by,

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

- Thus we estimate $P(x_k|c_i) = f(x_k, \mu_{c_i}, \sigma_{c_i})$, where $\mu_{c_i}(\sigma_{c_i})$ is the mean (standard deviation) of the values of variable X_k for the cases belonging to class c_i

An Illustrative Example

Weather	Temp	Humidity	Wind	Decision
rain	26	high	15	dontPlay
rain	35	normal	102	dontPlay
rain	12	high	120	play
rain	11	normal	95	play
rain	15	high	94	play
rain	29	high	79	dontPlay
rain	26	normal	75	dontPlay
rain	33	high	96	play
rain	28	normal	44	play
rain	21	high	84	play
overcast	27	high	99	play
overcast	26	normal	97	play
overcast	21	normal	74	play
overcast	10	normal	70	play
overcast	30	high	108	play
overcast	30	normal	16	play
overcast	30	normal	13	play
overcast	14	normal	32	play
sun	30	normal	89	dontPlay
sun	19	high	111	dontPlay
sun	14	normal	81	play
sun	19	high	41	dontPlay
sun	35	normal	38	dontPlay
sun	32	normal	55	dontPlay
sun	11	high	108	dontPlay
sun	33	normal	103	play
sun	29	high	105	dontPlay
sun	15	normal	63	dontPlay

How to classify $\langle sun, 16, normal, 70\rangle$?

P(play) = 16/28 = 0.57 P(dontPlay) = 12/28 = 0.43

$$\begin{aligned}
P(\langle sun, 16, normal, 70\rangle|play) &= P(sun|play) \times P(16|play) \times \cdots \\
P(sun|play) &= 2/16 = 0.125 \\
P(16|play) &= f(16, \mu(Temp)_{play}, \sigma(Temp)_{play}) \\
&= f(16, 22.18, 8.38) = 0.23 \\
P(normal|play) &= 10/16 = 0.625 \\
P(70|play) &= f(70, \mu(Wind)_{play}, \sigma(Wind)_{play}) \\
&= f(70, 76.625, 33.06) = 0.42 \\
P(\langle sun, 16, normal, 70\rangle|play) &= 0.125 \times 0.23 \times 0.625 \times 0.42 \\
&= 0.0075 \\
P(\langle sun, 16, normal, 70\rangle|play) \times P(play) &= 0.0075 \times 0.57 \\
&= 0.004275
\end{aligned}$$

Repeat for

P($\langle sun, 16, normal, 70\rangle$ | dontPlay)× P(dontPlay)

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Naive Bayes in R

```
library(e1071)
set.seed(1234)
sp <- sample(1:150,100)
tr <- iris[sp,]
ts <- iris[-sp,]
nb <- naiveBayes(Species ~ ., tr)
(mtrx <- table(predict(nb,ts),ts$Species))
```

```
##
##              setosa versicolor virginica
##   setosa         12          0         0
##   versicolor      0         21         1
##   virginica       0          0        16
```

```
(err <- 1-sum(diag(mtrx))/sum(mtrx))
```

```
## [1] 0.02
```

```
head(predict(nb,ts,type='raw'))
```

```
##      setosa versicolor virginica
## [1,]      1  1.742e-19 6.014e-34
## [2,]      1  1.969e-19 3.725e-35
## [3,]      1  2.991e-18 2.648e-34
## [4,]      1  5.136e-19 2.007e-32
## [5,]      1  3.718e-18 2.193e-32
## [6,]      1  2.880e-20 3.149e-37
```


Laplace Correction

- What if one of the $P(x_k|c_i)$'s is equal to zero? This can easily happen in nominal variables if one of the values does not occur in a class.
- This would make the product $P(\mathbf{x}|c_i) = \prod_{k=1}^{p} P(x_k|c_i) = 0$
- This zero probability would cancel the effects of all other $P(x_k|c_i)$'s

- The Laplace correction or Laplace estimator is a technique for probability estimation that tries to overcome these issues
- It consist in estimating $P(x_k|c_i)$ by $\frac{|D_{x_k,c_i}|+q}{|D_{c_i}|+q}$, where q is an integer greater than zero (typically 1)

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Laplace Correction in R

```
library(e1071)
set.seed(1234)
sp <- sample(1:150,100)
tr <- iris[sp,]
ts <- iris[-sp,]
nb <- naiveBayes(Species ~ ., tr,laplace=1)
(mtrx <- table(predict(nb,ts),ts$Species))
```

```
##
##              setosa versicolor virginica
##   setosa         12          0         0
##   versicolor      0         21         1
##   virginica       0          0        16
```

```
(err <- 1-sum(diag(mtrx))/sum(mtrx))
```

```
## [1] 0.02
```

```
head(predict(nb,ts,type='raw'))
```

```
##      setosa versicolor virginica
## [1,]      1  1.742e-19 6.014e-34
## [2,]      1  1.969e-19 3.725e-35
## [3,]      1  2.991e-18 2.648e-34
## [4,]      1  5.136e-19 2.007e-32
## [5,]      1  3.718e-18 2.193e-32
## [6,]      1  2.880e-20 3.149e-37
```


k-Nearest Neighbors

k Nearest Neighbors

- The k-nearest neighbor method was first described in the early 1950s.
- This method is computationally intensive with large data sets and it did not enjoy lots of popularity because of this.
- With the advent of cheap computing power its popularity has increased a lot because it is a very simple and effective method that can easily handle both classification and regression problems.
- k-nearest neighbors can be seen as methods that learn by analogy - i.e. they are based on the notion of similarity between cases.

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

k Nearest Neighbors (cont.)

The Basic Idea

- If we are given a new test case **x** for which we want a prediction
  1. search in the training set for the most similar cases (the nearest neighbors) to **x**
  2. use the outcomes of these cases to obtain the prediction for **x**

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

k Nearest Neighbors

Main Characteristics

- The k-nearest neighbors are known as lazy learners as they do not learn any model of the data
- Learning in k-nearest neighbors consists simply in storing the training data
  - Variants here include storing in data structures that provide efficient querying of the nearest neighbors
- They do not make any assumption on the unknown functional form we are trying to approximate, which means that with sufficient data they are applicable to any problem
- They usually achieve good results but...
  - They require a proper distance metric to be defined - issues like normalization, irrelevant variables, unknown values, etc., may have a strong impact on their performance
- They have fast training time, but slow prediction time

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

The Notion of Similarity

- The key issue on kNN is the notion of similarity
- This notion is strongly related with the notion of distance between observations
- Distances among observations in a data set can be used to find the neighbors of a test case

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

How to Calculate the Distance between 2 Cases?

- The notion of distance is related to the differences between the values on the variables describing the cases

ID	Income	Sex	Position	Age
1	2500	f	manager	35
2	2750	f	manager	30
3	4550	m	director	50

Case 1 is “closer” to case 2 than to 3

The Euclidean Distance Function

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

where x_i is the value of case $\mathbf{x}$ on variable i

Example

Given two cases $\mathbf{x} =< 3, 5, 1 >$ and $\mathbf{y} =< 12, 5.4, -3 >$ their Euclidean distance is given by

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(3-12)^2 + (5-5.4)^2 + (1-(-3))^2} = 9.85697$$

A Generalization - the Minkowski distance

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{p} |x_i - y_i|^r \right)^{1/r}$$

where if

- $r = 1$ we have what is known as the Manhattan distance (or L_1-norm)
- $r = 2$ we have the Euclidean distance
- etc.

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Potential Problems with Distance Calculation

In domains where cases are described by many variables several problems may arise that may distort the notion of distance between any two cases.

- Different scales of variables
- Different importance of variables
- Different types of data (e.g. both numeric and nominal variables, et.c)
- etc.

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Heterogeneous Distance Functions

How to calculate the distance between two cases described by variables with different type (e.g. numeric and nominal variables)? A possible solution,

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{p} \delta_i(x_i, y_i)$$

emque,

$$\delta_i(v_1, v_2) = \begin{cases} 1 & \text{if } i \text{ is nominal e } v_1 \neq v_2 \\ 0 & \text{if } i \text{ is nominal e } v_1 = v_2 \\ \frac{|v_1 - v_2|}{range(i)} & \text{if } i \text{ is numeric} \end{cases}$$

The distance between $<$ 2500, f, director, 35 $>$ and $<$ 2750, f, director, 30 $>$ would be given by $\frac{|2500-2750|}{range(Salary)} + 0 + 0 + \frac{|35-30|}{range(Age)}$

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

1-Nearest Neighbor Classifier

Method

- Search for the training case most similar to the test case
- Predict for the test case the class of this nearest neighbor

- Very simple method
- May suffer with the presence of outliers
- Frequently achieves good results

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

k-Nearest Neighbor Classifier

- Use the *k* nearest neighbors to obtain the classification of the test case
- Namely, the majority class on the *k* neighbors is the prediction of the mehtod
- What should be the value of *k*?
  - Frequent values are 3, 5 and 7
  - Odd numbers to avoid draws!
  - It can be estimated experimentally
    - Global estimation searches for the ideal *k* for a given data set
    - Local estimation methods try to estimate the ideal *k* for each test case (computationally very demanding!)

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

k-nearest neighbors in R

- Package **class** contains function `knn()`

```
library(class)
set.seed(1234)
sp <- sample(1:150,100)
tr <- iris[sp,]
ts <- iris[-sp,]
nn3 <- knn(tr[,-5],ts[,-5],tr[,5],k=3)
(mtrx <- table(nn3,ts$Species))

##
## nn3           setosa versicolor virginica
##   setosa          12          0         0
##   versicolor       0         20         1
##   virginica        0          1        16

(err <- 1-sum(diag(mtrx))/sum(mtrx))

## [1] 0.04
```

k-nearest neighbors in R - 2

- Package **DMwR** has a wrapper function with a "standard" interface,

```
library(class)
library(DMwR)
set.seed(1234)
sp <- sample(1:150,100)
tr <- iris[sp,]
ts <- iris[-sp,]
nn3 <- kNN(Species ~ .,tr,ts,k=3,norm=TRUE)
(mtrx <- table(nn3,ts$Species))

##
## nn3           setosa versicolor virginica
##   setosa          12          0         0
##   versicolor       0         20         3
##   virginica        0          1        14

(err <- 1-sum(diag(mtrx))/sum(mtrx))

## [1] 0.08
```


Trying to find the "ideal" value of *k* in R

```
trials <- c(1,3,5,7,11,13,15)
nreps <- 10
res <- matrix(NA,nrow=length(trials),ncol=2)
for(k in seq_along(trials)) {
  errs <- rep(0,nreps)
  for(r in 1:nreps) {
    sp <- sample(1:150,100)
    tr <- iris[sp,]
    ts <- iris[-sp,]
    nn3 <- kNN(Species ~ .,tr,ts,k=trials[k],norm=TRUE)
    mtrx <- table(nn3,ts$Species)
    errs[r] <- 1-sum(diag(mtrx))/sum(mtrx)
  }
  res[k,] <- c(mean(errs),sd(errs))
}
dimnames(res) <- list(paste('k',trials,sep='='),c('avg','std'))
res

##        avg     std
## k=1  0.082 0.02741
## k=3  0.050 0.02708
## k=5  0.056 0.01265
## k=7  0.052 0.03676
## k=11 0.040 0.02108
## k=13 0.046 0.02503
## k=15 0.070 0.02539
```

k-Nearest Neighbor Regression

Method

- Search for the training case most similar to the test case
- Predict for the test case the **average** of the target variable values of the neighbors

k-nearest neighbors regression in R

- Package **caret** has a function that to obtain these models

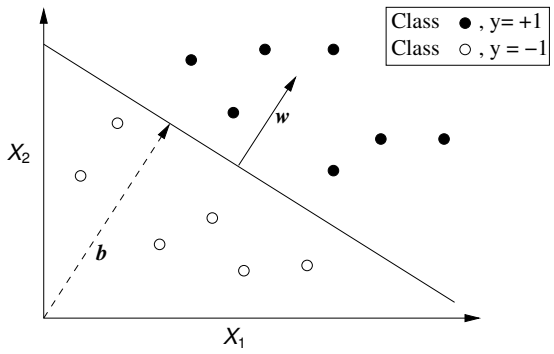
```
library(caret)
data(Boston,package="MASS")
set.seed(1234)
sp <- sample(1:506,354)
tr <- Boston[sp,]
ts <- Boston[-sp,]
tgt <- which(colnames(Boston) == "medv")
nn3 <- knnreg(tr[,-tgt],tr[,tgt],k=3)
pnn3 <- predict(nn3,ts[,-tgt])
(mse <- mean((pnn3-ts[,tgt])^2))

## [1] 44.93
```

Support Vector Machines

A Bit of History...

- SVM's were introduced in 1992 at the COLT-92 conference
- They gave origin to a new class of algorithms named *kernel machines*
- Since then there has been a growing interest on these methods
- More information may be obtained at `www.kernel-machines.org`
- A good reference on SVMs:
  N. Cristianini and J. Shawe-Taylor: An introduction to Support Vector Machines. Cambridge University Press, 2000.
- SVMs have been applied with success in a wide range of areas like: bio-informatics, text mining, hand-written character recognition, etc.

Two Linearly Separable Classes



- Obtain a linear separation of the cases (binary classification problems)
- Very simple and effective for linearly separable problems
- Most real-world problems are not linearly separable!

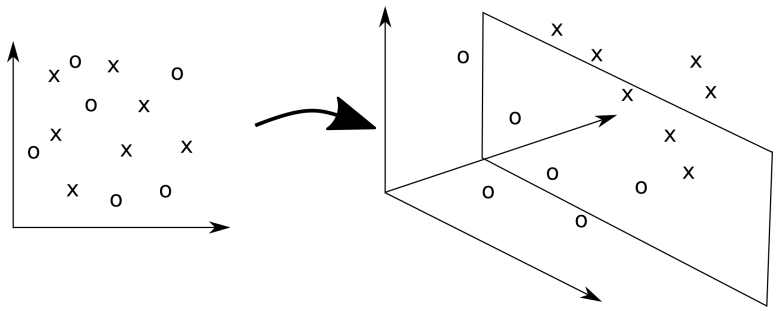
FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

The Basic Idea of SVMs

- Map the original data into a new space of variables with very high dimension.
- Use a linear approximation on this new input space.

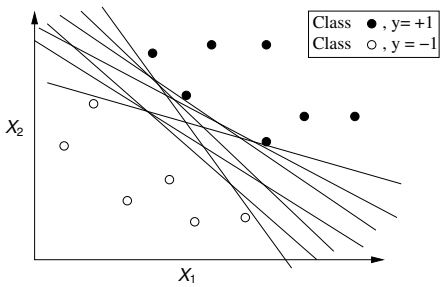
FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

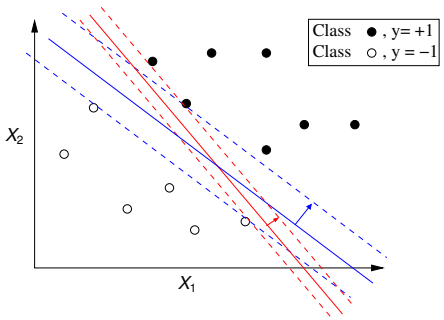
The Idea in a Figure



Map the original data into a new (higher dimension) coordinates system where the classes are linearly separable

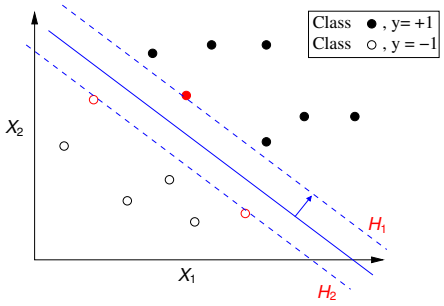
Maximum Margin Hyperplane





- There is an infinite number of hyperplanes separating the two classes!
- Which one should we choose?!
- We want the one that ensures a better classification accuracy on unseen data
- SVMs approach this problem by searching for the maximum margin hyperplane

The Support Vectors



- All cases that fall on the hyperplanes H_1 and H_2 are called the support vectors.
- Removing all other cases would not change the solution!

The Optimal Hyperplane

- SVMs use quadratic optimization algorithms to find the optimal hyperplane that maximizes the margin that separates the cases from the 2 classes
- Namely, these methods are used to find a solution to the following equation,

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\text{Subject to :}$$

$$\alpha_i \geq 0$$

$$\sum_i \alpha_i y_i = 0$$

- In the found solution, the α_i's > 0 correspond to the support vectors that represent the optimal solution

Recap

- Most real world problems are not linearly separable
- SVMs solve this by “moving” into a extended input space where classes are already linearly separable
- This means the maximum margin hyperplane needs to be found on this new very high dimension space

The Kernel trick

- The solution to the optimization equation involves dot products that are computationally heavy on high-dimensional spaces
- It was demonstrated that the result of these complex calculations is equivalent to the result of applying certain functions (the kernel functions) in the space of the original variables.

The Kernel Trick

Instead of calculating the dot products in a high dimensional space, take advantage of the proof that $K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{z})$ and simply replace the complex dot products by these simpler and efficient calculations

Summary of the SVMs Method

- As problems are usually non-linear on the original feature space, move into a high-dimension space where linear separability is possible
- Find the optimal separating hyperplane on this new space using quadratic optimization algorithms
- Avoid the heavy computational costs of the dot products using the kernel trick

How to handle more than 2 classes?

- Solve several binary classification tasks
- Essentially find the support vectors that separate each class from all others

The Algorithm

- Given a *m* classes task
- Obtain *m* SVM classifiers, one for each class
- Given a test case assign it to the class whose separating hyperplane is more distant from the test case

Obtaining an SVM in R

The package **e1071**

```
library(e1071)
data(Glass,package='mlbench')
tr <- Glass[1:200,]
ts <- Glass[201:214,]
s <- svm(Type ~ .,Glass)
predict(s,ts)
```

```
## 201 202 203 204 205 206 207 208 209 210 211 212 213 214
##   7   7   7   7   7   7   7   7   7   7   7   7   7   7
## Levels: 1 2 3 5 6 7
```


Obtaining an SVM in R (2)

The package **e1071**

```
ps <- predict(s,ts)
table(ps,ts$Type)
```

```
##
## ps   1  2  3  5  6  7
##   1  0  0  0  0  0  0
##   2  0  0  0  0  0  0
##   3  0  0  0  0  0  0
##   5  0  0  0  0  0  0
##   6  0  0  0  0  0  0
##   7  0  0  0  0  0 14
```

```
mc <- table(ps,ts$Type)
error <- 100*(1-sum(diag(mc))/sum(mc))
error
```

```
## [1] 0
```

ε-SV Regression

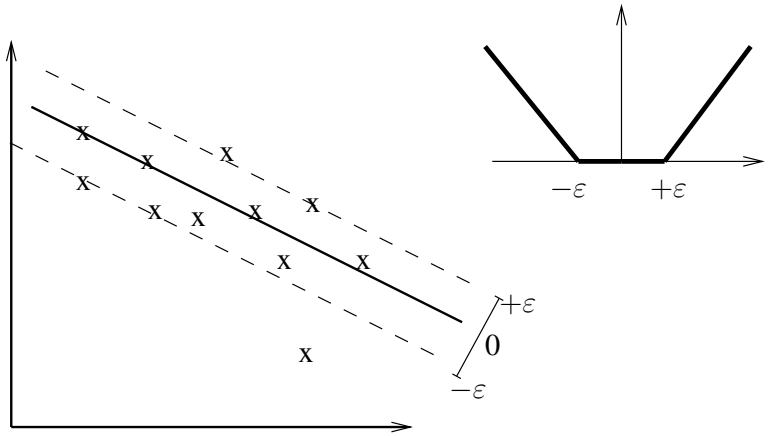
- Vapnik (1995) proposed the notion of ε support vector regression
- The goal in ε-SV Regression is to find a function $f(x)$ that has at most ε deviation from the given training cases
- In other words we do not care about errors smaller than ε

V. Vapnik (1995). The Nature of Statistical Learning Theory. Springer.

ε-SV Regression (cont.)

- ε-SV Regression uses the following error metric,

$$|\xi|_\varepsilon = \begin{cases} 0 & \text{if } |\xi| \leq \varepsilon \\ |\xi| - \varepsilon & \text{otherwise} \end{cases}$$



ε-SV Regression (cont.)

- The theoretical development of this idea leads to the following optimization problem,

$$\text{Minimize} : \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*)$$

$$\text{Subject to} : \begin{cases} y_i - \mathbf{w}\cdot\mathbf{x} - b & \leq \varepsilon + \xi_i \\ \mathbf{w}\cdot\mathbf{x} + b - y_i & \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq 0 \end{cases}$$

where C corresponds to the cost to pay for each violation of the error limit ε

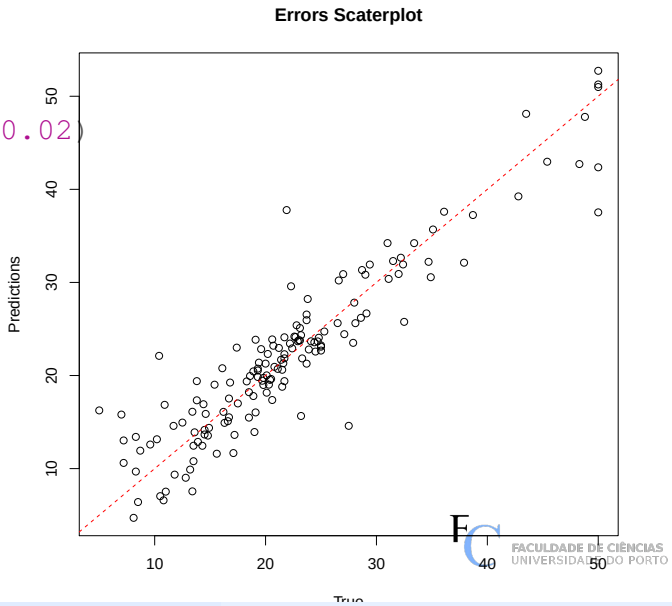
ε-SV Regression (cont.)

- As within classification we use the kernel trick to map a non-linear problem into a high dimensional space where we solve the same quadratic optimization problem as in the linear case
- In summary, by the use of the $|\xi|_\varepsilon$ loss function we reach a very similar optimization problem to find the support vectors of any non-linear regression problem.

SVMs for regression in R

```
library(e1071)
library(DMwR)
data(Boston,package='MASS')
set.seed(1234)
sp <- sample(1:nrow(Boston),354)
tr <- Boston[sp,]
ts <- Boston[-sp,]
s <- svm(medv ~ .,tr,cost=10,epsilon=0.02)
preds <- predict(s,ts)
regr.eval(ts$medv,preds)

##      mae      mse     rmse     mape
##   2.6678 13.8211   3.7177   0.1634
```

```
plot(ts$medv,preds,main='Errors Scaterplot',
     ylab='Predictions',xlab='True')
abline(0,1,col='red',lty=2)
```



Hands on SVMs

The file `Wine.Rdata` contains 2 data frames with data about the quality of "green" wines: i) `redWine` and ii) `whiteWine`. Each of these data sets has information on a series of wine tasting sessions to "green" wines (both red and white). For each wine sample several physico-chemical properties of the wine sample together with a quality score assigned by a committee of wine experts (variable `quality`).

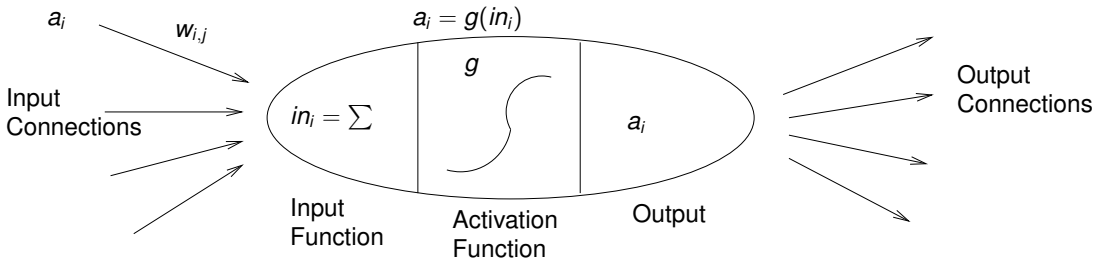
1. Obtain and SVM for forecasting the quality of the red variant of "green" wines
2. Split the data set in two parts: one with 70% of the samples and the other with the remaining 30%. Obtain an SVM with the first part and apply it to the second. What was the resulting mean absolute error?
3. Using the `round()` function, round the predictions obtained in the previous question to the nearest integer. Calculate the error rate of the resulting integers when compared to the true values

Artificial Neural Networks

Artificial Neural Networks (ANNs)

- Models with a strong biological inspiration.
- McCulloch e Pitts (1943) proposed the first artificial model of a neuron.
- An artificial neural network is composed by a set of units (neurons) that are connected. These connections have an associated weight.
- Each unit has an activation level as well as means to update this level.
- Some units are connected to the outside world. We have input and output neurons.
- Learning within ANNs consists of updating the weights of the network connections.

The Problem



- Each unit has a very simple function:
  - Receive the input impulses and calculate its ouput as a function of these impulses.
- This calculation is divided in two parts:
  - a linear computation (of the inputs):

$$in_i = \sum_j w_{j,i} \cdot a_j$$

  - A non-linear computation (the activation function)

Activation Functions

Different activation functions provide different behaviours.

- Some common functions
  - The Step Function

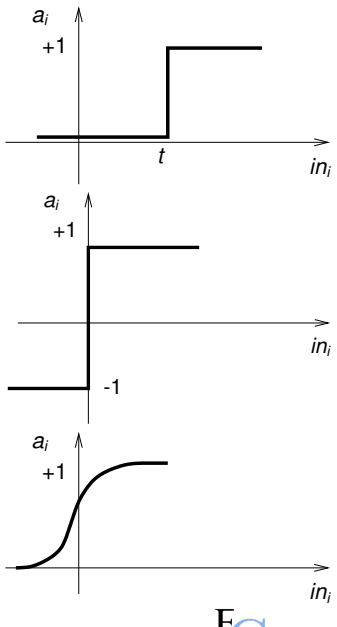
$$step(x) = \begin{cases} 1 & \text{if } x \geq t \\ 0 & \text{if } x < t \end{cases}$$

  - The Sign Function

$$sign(x) = \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

  - The Sigmoid Functions

$$sigmoid(x) = \frac{1}{1 + \exp^{-x}}$$



Activation Functions - 2

- The units can have thresholds that represent the minimum value of the weighted sum of the inputs (in_i) that activates the neuron.
  - There are also thresholded versions of the sign and sigmoid functions
- In the majority of cases we treat the thresholds as an extra input unit
  - The extra unit has a fixed activation value $a_0 = -1$
  - The weight of this unit is equal to $-t$
  - With this approximation we have units with a threshold of zero (as the real threshold is treated as an extra weight).
  - Mathematically we get:

$$a_i = step_t\left(\sum_{j=1}^{n} w_{j,i} \cdot a_j\right) = step_0\left(\sum_{j=0}^{n} w_{j,i} \cdot a_j\right)$$

where $w_{0,i} = t$ and $a_0 = -1$

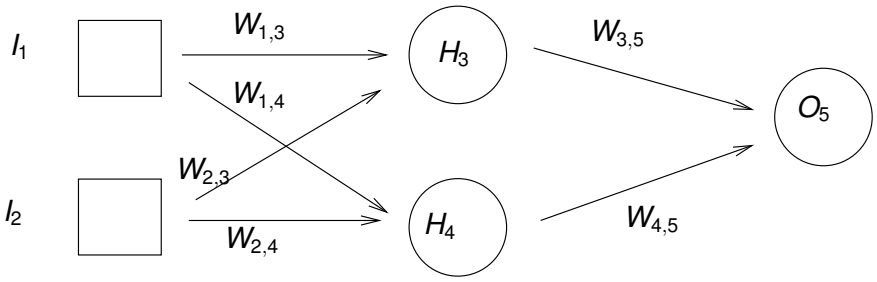
FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Networks of Computation Units (ANNs)

- There are two main types of ANNs:
  - Feed-forward networks
    - Networks with uni-directional connections (from input to output), and without cycles.
  - Recurrent networks
    - Networks with arbitrary connections
- Usually the networks are structured in layers
  - On a feed-forward network each unit is connected only to units in the following layer. There are not connections from units on a certain layer and units on previous layers.
  - Due to the possible feedback effects, recurrent networks are potentially more instable, possibly exhibiting caotic behaviors. Usually they take longer to converge.

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Feed-forward Networks



- There are networks without "hidden" layers. These are usually known as perceptron networks.
  - These networks have a very simple learning process but strong limitations in the functions they may learn
- Networks with hidden layers are called multilayer networks
- If we fix the structure of the network we may represent the output of a network as follows:

$$a_5 = g(W_{3,5}a_3 + W_{4,5}a_4) = g\left(W_{3,5}g(W_{1,3}a_1 + W_{2,3}a_2) + W_{4,5}g(W_{1,4}a_1 + W_{2,4}a_2)\right)$$

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Perceptron Networks

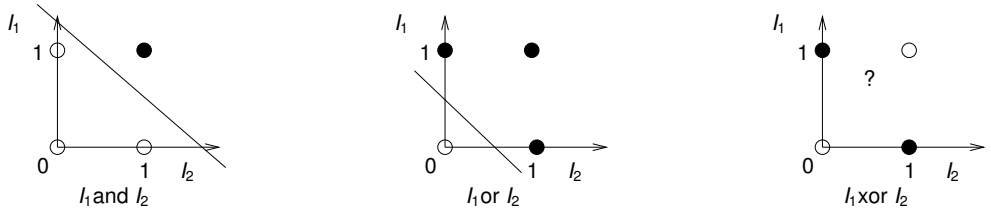
- Rosenblatt (1958) introduced the notion of perceptron networks. This work was then further extended by Minsky and Papert (1969).
- Each output unit, O, gets its activation value from,

$$O = step_0\left(\sum_{j=0}^{n} W_j I_j\right) = step_0(\mathbf{W} \cdot \mathbf{I})$$

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

The Limitations of Perceptron Networks

- This class of networks may only represent certain classes of boolean functions.

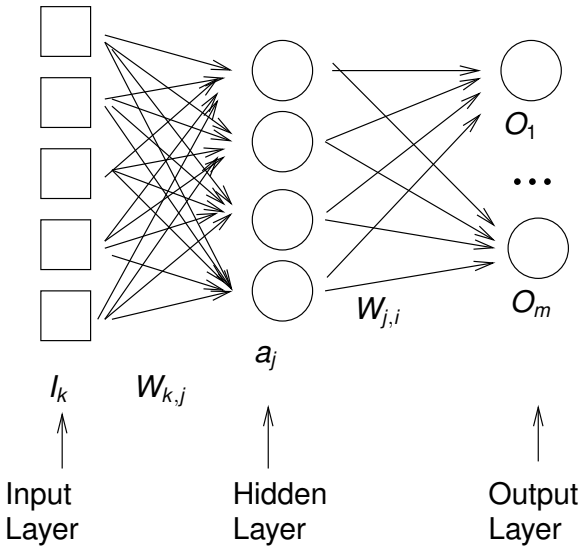


- Perceptrons are limited to linearly separable functions
  - A perceptron produces a 1 iif $\mathbf{W} \cdot \mathbf{I} > 0$
    - In the case I_1 and I_2 the plane separating 1 from 0 is $I_1 + I_2 = 1.5$
- There is a learning algorithm for perceptrons that ensures the ability to learn any linearly separable functions, provided enough examples are given.

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Feed-forward Multi-Layer Architectures

- The following is a frequently used multilayer feed-forward network architecture:

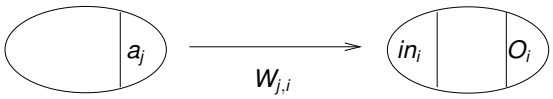


FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

The Backpropagation Algorithm

- This is the most popular algorithm for learning ANNs
- It has similarities with the learning algorithm used in perceptron networks
  - Each example is presented to the network
  - If the correct output is produced nothing is done
  - If there is an error we need to re-adjust the network weights
  - This adjustment is simple in perceptrons as there is a single connection between the input and output nodes
  - In multilayer networks things are not that simple as we need to divide the adjustments across the nodes and layers of the network

Weight Adjustment through Backpropagation



- Let $Err_i = (T_i - O_i)$ be the error in output unit O_i
- The updating rule from the hidden units to the output unit O_i is

$$W_{j,i} = W_{j,i} + \eta \cdot a_j \cdot \Delta_i \qquad \text{where } \Delta_i = g'(in_i) \cdot Err_i$$

  where $g'(.)$ is the derivative of the activation function $g(.)$ and η is known as the learning rate.
- If using the sigmoid activation function this results in:

$$\Delta_i = O_i \cdot (1 - O_i) \cdot Err_i = O_i \cdot (1 - O_i) \cdot (T_i - O_i)$$

Weight Adjustment through Backpropagation - 2

- To update the weights of the connections between the input units and the hidden units we need a quantity similar to Err_i. This is where the backpropagation idea enters:
  - Each hidden unit j is responsible for a certain fraction of the error Δ_i in the output nodes to which it is connected. Thus each Δ_i value is going to be divided according to the weight of the connection between the respective hidden and output units, thus propagating the errors backwards:

$$\Delta_j = g'(in_j) \cdot \sum_i W_{j,i} \cdot \Delta_i \qquad W_{k,j} = W_{k,j} + \eta \cdot I_k \cdot \Delta_j$$

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

The Backpropagation Algorithm

```
1: function BACKPROPAGATION(ANN, Data, η)
2:     repeat
3:         for all ex ∈ Data do
4:             O ← OutputANN(ANN, ex)
5:             for all i ∈ OutputUnits(ANN) do
6:                 Err_i ← T_i − O_i
7:                 Δ_i ← Err_i · g'(in_i)
8:             end for
9:             for all j ∈ HiddenUnits(ANN) do
10:                Err_j ← 0
11:                for all i ∈ OutputUnits(ANN) do
12:                    Err_j ← w_{j,i} · Δ_i
13:                    w_{j,i} ← W_{j,i} + η · a_j · Δ_i
14:                end for
15:                Delta_j ← Err_j · g'(in_j)
16:            end for
17:            for all k ∈ InputUnits(ANN) do
18:                for all j ∈ HiddenUnits(ANN) do
19:                    W_{k,j} = W_{k,j} + η · I_k · Δ_j
20:                end for
21:            end for
22:        end for
23:    until Converge(ANN)
24:    return ANN
25: end function
```

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Obtaining a feed-forward multilayer ANN in R

```
library(nnet)
data(iris)
set.seed(1234)
sp <- sample(1:150,100)
tr <- iris[sp,]
ts <- iris[-sp,]
nn <- nnet(Species ~ .,tr,size=5, decay =0.1, maxit=1000)

## # weights:  43
## initial  value 119.514232
## iter  10 value 54.207924
## iter  20 value 23.457619
## iter  30 value 18.693044
## iter  40 value 17.815623
## iter  50 value 17.705735
## iter  60 value 17.407647
## iter  70 value 17.377002
## iter  80 value 17.361747
## iter  90 value 17.257140
## iter 100 value 17.221449
## iter 110 value 17.218631
## final  value 17.218628
## converged
```


Obtaining a feed-forward multilayer ANN in R

```
(mtrx <- table(predict(nn,ts,type='class'),ts$Species))

##
##              setosa versicolor virginica
##   setosa         12          0         0
##   versicolor      0         20         0
##   virginica       0          1        17
```

Obtaining a feed-forward multilayer ANN in R

```
nn


## a 4-5-3 network with 43 weights
## inputs: Sepal.Length Sepal.Width Petal.Length Petal.Width
## output(s): Species
## options were - softmax modelling  decay=0.1


summary(nn)


## a 4-5-3 network with 43 weights
## options were - softmax modelling  decay=0.1
##  b->h1 i1->h1 i2->h1 i3->h1 i4->h1
##   1.47   0.97   1.29  -1.65  -2.02
##  b->h2 i1->h2 i2->h2 i3->h2 i4->h2
##  -0.16  -0.30  -0.85   1.44   0.63
##  b->h3 i1->h3 i2->h3 i3->h3 i4->h3
##   0.19   0.34   0.97  -1.66  -0.73
##  b->h4 i1->h4 i2->h4 i3->h4 i4->h4
##  -1.95  -1.19  -1.66   2.04   2.53
##  b->h5 i1->h5 i2->h5 i3->h5 i4->h5
##  -0.16  -0.30  -0.85   1.44   0.63
##  b->o1 h1->o1 h2->o1 h3->o1 h4->o1 h5->o1
##   0.57   1.33  -1.77   2.45  -0.65  -1.77
##  b->o2 h1->o2 h2->o2 h3->o2 h4->o2 h5->o2
##  -0.55   1.73   1.35  -2.10  -2.98   1.35
##  b->o3 h1->o3 h2->o3 h3->o3 h4->o3 h5->o3
##  -0.02  -3.06   0.42  -0.35   3.64   0.42
```


Obtaining a feed-forward multilayer ANN in R

```
nn


## a 4-5-3 network with 43 weights
## inputs: Sepal.Length Sepal.Width Petal.Length Petal.Width
## output(s): Species
## options were - softmax modelling  decay=0.1


head(predict(nn,ts))


##       setosa  versicolor    virginica
## 1  0.9889885 0.010335105 0.0006764155
## 3  0.9869150 0.012323611 0.0007613524
## 9  0.9783846 0.020543622 0.0010718246
## 11 0.9897715 0.009585496 0.0006430209
## 12 0.9837816 0.015337819 0.0008805736
## 14 0.9875598 0.011704266 0.0007359794
```

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Tunning the ANN

```
set.seed(1234)
trials <- expand.grid(sz=c(3,5,7),dc=c(0.1,0.01,0.05))
nreps <- 10
res <- matrix(NA,nrow=nrow(trials),ncol=2)
for(k in 1:nrow(trials)) {
  errs <- rep(0,nreps)
  for(r in 1:nreps) {
    sp <- sample(1:150,100)
    tr <- iris[sp,]
    ts <- iris[-sp,]
    nn <- nnet(Species ~ .,tr,
               size=trials[k,"sz"],decay=trials[k,"dc"],
               maxit=1000,trace=F)
    mtrx <- table(predict(nn,ts,type='class'),ts$Species)
    errs[r] <- 1-sum(diag(mtrx))/sum(mtrx)
  }
  res[k,] <- c(mean(errs),sd(errs))
}
```


Tunning the ANN - 2

```
dimnames(res) <- list(apply(trials,1,
    function(x) paste(colnames(trials)[1],x[1],colnames(trials)[2],x[2],sep='_')),
                      c('avg','std'))
res
```

```
##                avg        std
## sz_3_dc_0.1  0.020 0.01333333
## sz_5_dc_0.1  0.018 0.01475730
## sz_7_dc_0.1  0.028 0.02699794
## sz_3_dc_0.01 0.026 0.02118700
## sz_5_dc_0.01 0.022 0.01751190
## sz_7_dc_0.01 0.022 0.01988858
## sz_3_dc_0.05 0.026 0.02319004
## sz_5_dc_0.05 0.034 0.01897367
## sz_7_dc_0.05 0.026 0.01349897
```

```
res[which.min(res[,1]),,drop=FALSE]
```

```
##               avg       std
## sz_5_dc_0.1 0.018 0.0147573
```

Pros and Cons of ANNs

- Pros
  - Tolerance of noisy data
  - Ability to classify patterns on which they have not been trained
  - Successful on a wide range of real-world problems
  - Algorithms are inherently parallel
- Cons
  - Long training times
  - Resulting models are essentially black boxes

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Multivariate Adaptive Regression Splines

Additive Models

- Main idea:
  - A complex function may be decomposed in an additive way such that each term has a simpler form.
  - Main advantage/motivation: additive models are very interpretable
- A Generalized Additive Model (GAM) (Hastie and Tibshirani, 1990) can be defined as,

$$r(\mathbf{x}) = \alpha + \sum_{i=1}^{a} f_i(X_i)$$

where the f_i's are univariate functions.

Hastie,T., Tibshirani,R. (1990) : Generalized Additive Models. Chapman & Hall.

Additive Models (cont.)

$$r(\mathbf{x}) = \alpha + \sum_{i=1}^{a} f_i(X_i)$$

- These models can be further generalized over functions with more than one variable.
- The model parameters are usually obtained through the *backfitting* algorithm (Friedman and Stuetzle, 1981).

Friedman,J., Stuetzle,W. (1981) : Projection pursuit regression. Journal of the American Statistical Association, 76 (376), 817-823

Multivariate Adaptive Regression Splines (MARS)

- These are another example of additive models, this time with the form,

$$r(\mathbf{x}) = c_0 + \sum_{i=1}^{p} c_i \prod_{k=1}^{K_i} \left[s_{k,i} \left(X_{v(k,i)} - t_{k,i} \right) \right]_+$$

  where $\left[s_{k,i} \left(X_{v(k,i)} - t_{k,i} \right) \right]_+$ are two-sided trucanted base functions.
- These models can be re-written in an easier to understand format as follows,

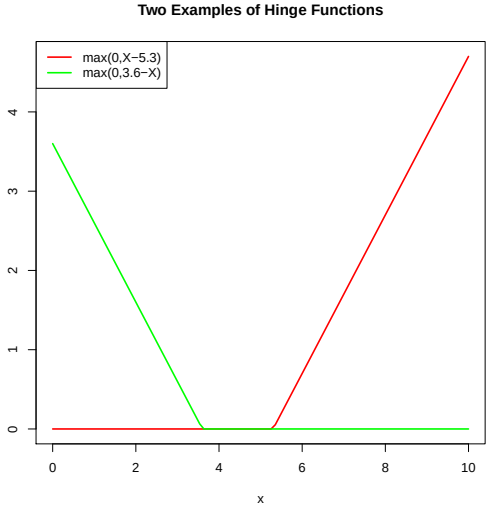
$$r(\mathbf{x}) = c_0 + \sum c_i \cdot B_i(\mathbf{x})$$

  where the B_i's are basis functions and the c_i's are constants.

Friedman,J. (1991) : Multivariate Adaptive Regression Splines. Annals of Statistics, 19:1, 1-141.

Multivariate Adaptive Regression Splines (MARS) - 2

- The basis functions usually take one of the following forms:
  1. the constant 1 (for the intercept)
  2. a hinge function with the form $\max(0, X - k)$ or $\max(0, k - X)$, where k are constants
  3. a product of two or more hinge functions, which try to capture the interactions between two or more variables



MARS - the algorithm

- MARS builds models in two phases: the forward and backward passes
  1. Forward pass
     - start with an intercept (mean of the target)
     - iteratively keep adding new basis function terms
     - this is carried out until a certain termination criterion is met
  2. Backward pass
     - iteratively tries to remove each term in turn
     - use a cross validation criterion to compare and select alternatives

Obtaining MARS Models in R

```
library(DMwR)
library(earth)  # extra package to install
data(Boston, package="MASS")
sp <- sample(1:nrow(Boston),as.integer(0.7*nrow(Boston)))
tr <- Boston[sp,]
ts <- Boston[-sp,]
mars <- earth(medv ~ .,tr)
preds <- predict(mars,ts)
regr.eval(ts$medv,preds,train.y=tr$medv)

##      mae      mse     rmse     mape     nmse     nmae
##   2.7063 12.7442   3.5699   0.1264   0.1722   0.4122
```

Obtaining MARS Models in R (cont.)

```
summary(mars)

## Call: earth(formula=medv~., data=tr)
##
##                 coefficients
## (Intercept)          30.3372
## h(crim-4.22239)      -0.4881
## h(crim-22.0511)       0.4375
## h(nox-0.488)        -23.4996
## h(rm-6.405)           6.6240
## h(rm-7.454)          10.5300
## h(rm-7.923)         -22.6268
## h(dis-2.4298)        -0.7862
## h(2.4298-dis)         6.6821
## h(rad-7)              0.4358
## h(tax-300)           -0.0141
## h(ptratio-14.7)      -0.7006
## h(black-395.5)       -1.3363
## h(395.5-black)       -0.0074
## h(lstat-6.07)        -0.6514
## h(6.07-lstat)         2.5575
## h(lstat-24.56)        0.7851
##
## Selected 17 of 23 terms, and 9 of 13 predictors
## Importance: rm, lstat, ptratio, nox, dis, crim, rad, tax, black, ...
## Number of terms at each degree of interaction: 1 16 (additive model)
## GCV 15.53    RSS 4520    GRSq 0.8263    RSq 0.8564
```


Model Ensembles

Model Ensembles

What?

- Ensembles are collections of models that are used together to address a certain prediction problem

Why? (Diettrich, 2002)

- For complex problems it is hard to find a model that “explains” all observed data.
- Averaging over a set of models typically leads to significantly better results.

Dietterich, T. G. (2002). Ensemble Learning. In The Handbook of Brain Theory and Neural Networks, Second edition, (M.A. Arbib, Ed.), Cambridge, MA: The MIT Press, 2002. 405-408.

The Bias-Variance Decomposition of Prediction Error

- The prediction error of a model can be split in two main components: the bias and the variance components
- The bias component is the part of the error that is due to the poor ability of the model to fit the seen data
- The variance component has to do with the sensibility of the model to the given training data

The Bias-Variance Decomposition of Prediction Error

- Decreasing the bias by adjusting more to the training sample will most probably lead to a higher variance - the over-fitting phenomenon
- Decreasing the variance by being less sensitive to the given training data will most probably have as consequence a higher bias
- In summary: there is a well-known bias-variance trade-off in learning a prediction model

Ensembles are able to reduce both components of the error

Their approach consist on applying the same algorithm to different samples of the data and use the resulting models in a voting schema to obtain predictions for new cases

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Independent or Parallel Models

- One of the ways of obtaining an ensemble is to construct them independently in a way that ensures some diversity among them
- There are several ways we can reach this diversity among which we may refer:
  - Applying the models on somewhat different training sets
  - Applying the models on data sets using different predictors

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Coordinated or Sequential Models

- Another way of obtaining an ensemble is to construct a “larger” model by composing it from smaller models integrated somehow where each simpler model has some weighted participation in the ensemble predictions
- The task of this type of ensembles is then to choose the right component models and their respective weight, so that the weighted sum of these components has a good predictive performance

Bagging (Breiman, 1996)

- Bagging (Bootstrap Aggregating) is a method that obtains a set of k models using different bootstrap samples of the given training data
  - For each model a sample with replacement of the same size as the available data is obtained
  - This means that for each model there is a small proportion of the examples that will be different
- If the base learner has a high variance (i.e. very sensitive to variations on the training sample), this will ensure diversity among the k models
- In this context, bagging should be applied to base learners with high variance

Breiman, L. (1996): Bagging predictors. In *Machine Learning*, 24: 123–140.

A Simple Implementation of Bagging in R

```
simpleBagging <- function(form,data,model='rpartXse',nModels=100,...) {
    ms <- list()
    n <- nrow(data)
    for(i in 1:nModels) {
        tr <- sample(n,n,replace=T)
        ms[[i]] <- do.call(model,c(list(form,data[tr,]),...))
    }
    ms
}

predict.simpleBagging <- function(models,test) {
   ps <- sapply(models,function(m) predict(m,test))
   apply(ps,1,mean)
}
```

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Using it...

```
data(Boston,package='MASS')
library(DMwR)

## Loading required package:  methods
## Loading required package:  lattice
## Loading required package:  grid

set.seed(123)
trPerc <- 0.7
sp <- sample(1:nrow(Boston),as.integer(trPerc*nrow(Boston)))
tr <- Boston[sp,]
ts <- Boston[-sp,]

m <- simpleBagging(medv ~ .,tr,nModels=300,se=0.5)
ps <- predict.simpleBagging(m,ts)
mean(abs(ps-ts$medv))

## [1] 2.316893
```

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

A More "Sophisticated" Implementation

Package ipred

```
library(ipred)
data(Boston,package='MASS')
set.seed(123)
trPerc <- 0.7
sp <- sample(1:nrow(Boston),as.integer(trPerc*nrow(Boston)))
tr <- Boston[sp,]
ts <- Boston[-sp,]

m <- bagging(medv ~ .,tr,nbagg=100)
ps <- predict(m,ts)
mean(abs(ps-ts$medv))

## [1] 2.839413
```


Varying the Predictors

- Another way of generating a diverse set of models is by using different randomly chosen predictors
- The idea is similar to bagging but instead of generating samples of the cases we generate samples of the variables

A Simple Implementation in R

```
randPreds <- function(tgtName,data,model='rpartXse',
                      nVars=(ncol(data)-1)%/%2,nModels=20,...) {

   np <- ncol(data)-1
   if (np <= nVars)
     stop(paste("Nro de colunas nos dados insuficiente para escolher",
                nVar,"variáveis"))

   tgtCol <- which(colnames(data) == tgtName)
   preds <- (1:ncol(data))[-tgtCol]

   ms <- list()
   for(i in 1:nModels) {
     cols <- sample(preds,nVars)
     form <- as.formula(paste(paste(names(data)[tgtCol],'~'),
                              paste(names(data)[cols],collapse='+')))
     ms[[i]] <- do.call(model,c(list(form,data),...))
   }
   ms
}

predict.randPreds <- function(models,test) {
   ps <- sapply(models,function(m) predict(m,test))
   apply(ps,1,mean)
}
```


Using it...

```
data(Boston,package='MASS')
library(DMwR)
set.seed(123)
trPerc <- 0.7
sp <- sample(1:nrow(Boston),as.integer(trPerc*nrow(Boston)))
tr <- Boston[sp,]
ts <- Boston[-sp,]

m <- randPreds("medv",tr,nModels=300,se=0.5)
ps <- predict.randPreds(m,ts)
mean(abs(ps-ts$medv))


## [1] 2.940712
```

Random Forests (Breiman, 2001)

- Random Forests put the ideas of sampling the cases and sampling the predictors, together in a single method
  - Random Forests combine the ideas of bagging together with the idea of random selection of predictors
- Random Forests consist of sets of tree-based models where each tree is obtained from a bootstrap sample of the original data and uses some form of random selection of variables during tree growth

Breiman, L. (2001): "Random Forests". Machine Learning 45 (1): 5—32.

Random Forests - the algorithm

- For each of the *k* models
  - Draw a random sample with replacement to obtain the training set
  - Grow a classification or regression tree
    - On each node of the tree choose the best split from a randomly selected subset *m* of the predictors
- The trees are fully grown, i.e. no pruning is carried out

Random Forests in R

The package `randomForest`

```
library(randomForest)
data(Boston,package="MASS")
samp <- sample(1:nrow(Boston),354)
tr <- Boston[samp,]
ts <- Boston[-samp,]
m <- randomForest(medv ~ ., tr)
ps <- predict(m,ts)
mean(abs(ts$medv-ps))

## [1] 2.378855
```


A classification example

```
data(Glass,package='mlbench')
set.seed(1234)
sp <- sample(1:nrow(Glass),150)
tr <- Glass[sp,]
ts <- Glass[-sp,]
m <- randomForest(Type ~ ., tr,ntree=3000)
ps <- predict(m,ts)
table(ps,ts$Type)

##
## ps   1  2  3  5  6  7
##   1 13  5  3  0  0  1
##   2  2 18  0  3  0  2
##   3  0  0  1  0  0  0
##   5  0  0  0  4  0  0
##   6  0  1  0  0  3  0
##   7  0  0  0  0  0  8

mc <- table(ps,ts$Type)
err <- 100*(1-sum(diag(mc))/sum(mc))
err

## [1] 26.5625
```

Other Uses of Random Forests

Variable Importance

```
data(Boston,package='MASS')
library(randomForest)
m <- randomForest(medv ~ ., Boston,
                  importance=T)
importance(m)
```

```
##           %IncMSE IncNodePurity
## crim    16.001604     2511.3914
## zn       2.719681      184.4274
## indus   11.992644     2501.0269
## chas     4.496731      208.3667
## nox     18.440180     2702.4705
## rm      37.873226    13288.7533
## age     11.793865     1198.7370
## dis     17.957678     2423.8487
## rad      7.259293      320.4829
## tax     14.721102     1157.0856
## ptratio 15.715445     2716.8744
## black   11.498495      826.2531
## lstat   29.172401    11871.6578
```


```
varImpPlot(m,main="Feature Relevance Scores")
```


Hands on Linear Regression and Random Forests

the Algae data set

Load in the data set `algae` and answer the following questions:

1. How would you obtain a random forest to forecast the value of alga *a4*
2. Repeat the previous exercise but now using a linear regression model. Try to simplify the model using the `step()` function.
3. Obtain the predictions of the two previous models for the data used to obtain them. Draw a scatterplot comparing these predictions
4. Load the data set `testAlgae`. It contains a data frame named `test.algae` with some extra 140 water samples for which we want predictions. Use the previous two models to obtain predictions for `a4` on these new samples. Check what happened to the test cases with NA's. Fill-in the NA's on the test set and repeat the experiment.

Boosting

- Boosting was developed with the goal of answering the question: Can a set of weak learners create a single strong learner?
- In the above question a “weak” learner is a model that alone is unable to correctly approximate the unknown predictive function, while a “strong” learner has that ability
- Boosting algorithms work by iteratively creating a strong learner by adding at each iteration a new weak learner to make the ensemble
- Weak learners are added with weights that reflect the learner’s accuracy
- After each addition the data is re-weighted such that cases that are still poorly predicted gain more weight
- This means that each new weak learner will focus on the errors of the previous ones

Rob Schapire (1990). Strength of Weak Learnability. Machine Learning Vol. 5, pages 197–227.

The AdaBoost Algorithm (Freund & Shapire, 1996)

- AdaBoost (Adaptive Boosting) is an ensemble algorithm that can be used to improve the performance of a base algorithm
- It consists of an iterative process where new models are added to form an ensemble
- It is adaptive in the sense that at each new iteration of the algorithm the new models are built to try to overcome the errors made in the previous iterations
- At each iteration the weights of the training cases are adjusted so that cases that were wrongly predicted get their weight increased to make new models focus on accurately predicting them
- AdaBoost was created for classification although variants for regression exist

Y. Freund and R. Schapire (1996). Experiments with a new boosting algorithm, in Proc. of 13th International Conference on Machine Learning

The AdaBoost Algorithm

- The goal of the algorithm is to reach a form of additive model composed of k weak models

$$H(\mathbf{x}_i) = \sum_k w_k h_k(\mathbf{x}_i)$$

  where w_k is the weight of the weak model $h_k(\mathbf{x}_i)$
- All training cases start with a weight equal to $d_1(\mathbf{x}_i) = 1/n$, where n is the training set size

The AdaBoost Algorithm

- At iteration r the algorithm builds the weak model $h_r(\mathbf{x}_i)$ such that this model minimizes the weighted training error. This error is $e = \sum_i d_r(\mathbf{x}_i) I(y_i \neq h_r(\mathbf{x}_i))$, where $d_r(\mathbf{x}_i)$ is the weight of case $\langle \mathbf{x}_i, y_i \rangle$ at iteration r
- The weight of this weak model is calculated by

$$w_r = \frac{1}{2} \ln \left(\frac{1-e}{e} \right)$$

- Finally, the case weights for iteration $r+1$ are updated by

$$d_{r+1}(\mathbf{x}_i) = d_r(\mathbf{x}_i) \frac{\exp(-w_r I(y_i \neq h_r(\mathbf{x}_i)))}{Z_r}$$

  where Z_r is chosen to make all d_{r+1} sum up to 1

AdaBoost for Classification in R

Package adabag

This package uses classification trees as the weak learners

```
library(adabag)
data(iris)
set.seed(1234)
trPerc <- 0.7
sp <- sample(1:nrow(iris),as.integer(trPerc*nrow(iris)))

tr <- iris[sp,]
ts <- iris[-sp,]

m <- boosting(Species ~ ., tr)
ps <- predict(m,ts)

ps$confusion
```

```
##                Observed Class
## Predicted Class setosa versicolor virginica
##      setosa         11          0         0
##      versicolor      0         20         1
##      virginica       0          1        12
```

```
ps$error
```

```
## [1] 0.04444
```


Bagging in package adabag

This package also includes a function implementing bagging,

```
library(adabag)
data(iris)
set.seed(1234)
trPerc <- 0.7
sp <- sample(1:nrow(iris),as.integer(trPerc*nrow(iris)))

tr <- iris[sp,]
ts <- iris[-sp,]

m <- bagging(Species ~ ., tr,mfinal=50)
ps <- predict(m,ts)

ps$confusion
```

```
##                Observed Class
## Predicted Class setosa versicolor virginica
##      setosa         11          0         0
##      versicolor      0         20         1
##      virginica       0          1        12
```

```
ps$error
```

```
## [1] 0.04444
```

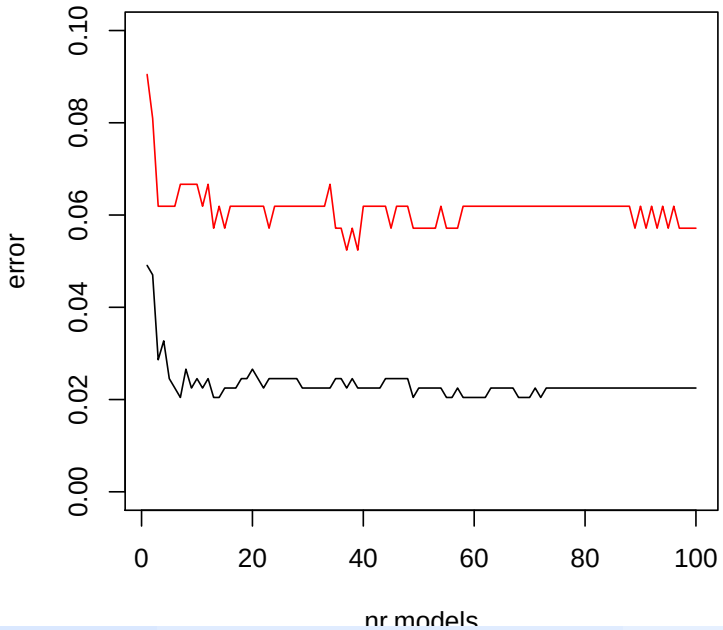
Error curves

This package also includes a function that allows you to check the evolution of the error as you increase the number of weak learners,

```
library(adabag)
data(BreastCancer,package="mlbench")
set.seed(1234)
trPerc <- 0.7
sp <- sample(1:nrow(BreastCancer),as.integer(trPerc*nrow(BreastCancer)))

tr <- BreastCancer[sp,-1]
ts <- BreastCancer[-sp,-1]

m <- bagging(Class ~ ., tr,mfinal=100)
ps <- predict(m,ts)

ptr <- errorevol(m,tr)
pts <- errorevol(m,ts)
```


Error curves (cont.)

```
plot(ptr$error,type="l",xlab="nr.models",ylab="error",ylim=c(0,0.1))
lines(pts$error,col="red")
```



AdaBoost for Regression in R

Package gbm

Package gbm implements the Gradient Boosting Machine (Friedman, 2001)

```
library(gbm)
data(Boston,package='MASS')

set.seed(1234)
trPerc <- 0.7
sp <- sample(1:nrow(Boston),as.integer(trPerc*nrow(Boston)))
tr <- Boston[sp,]
ts <- Boston[-sp,]

m <- gbm(medv ~ .,distribution='gaussian',data=tr,
         n.trees=20000,verbose=F)
ps <- predict(m,ts,type='response',n.trees=20000)
mean(abs(ps-ts$medv))


## [1] 2.636
```

J.H. Friedman (2001). Greedy Function Approximation: A Gradient Boosting Machine, Annals of Statistics, 29(5):1189-1232.

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Hands on Mars and Gradient Boosting

Load in the data set Boston Housing from package **MASS** and answer the following questions:

1. Split the data set into training and testing sets and obtain a MARS model and a GBM model with the training set.
2. Check the predictions of the models on the test set. Obtain their mean squared error.
3. Read the help pages of the two functions that obtain MARS and GBM and explore a few variants of these models and check the results in terms of the performance on the test set
4. Produce a barplot with the errors of the different variants you have considered.

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Evaluation Methodologies and Comparison of Models

Performance Estimation

The setting

- Predictive task: unknown function $Y = f(\mathbf{x})$ that maps the values of a set of predictors into a target variable value (can be a classification or a regression problem)
- A (training) data set $\{< \mathbf{x}_i, y_i >\}_{i=1}^{N}$, with known values of this mapping
- Performance evaluation criterion(a) - metric(s) of predictive performance (e.g. error rate or mean squared error)
- How to obtain a **reliable estimates** of the predictive performance of any solutions we consider to solve the task using the available data set?

Reliability of Estimates

Resubstitution estimates

- Given that we have a data set one possible way to obtain an estimate of the performance of a model is to evaluate it on this data set
- This leads to what is known as a **resubstitution estimate** of the prediction error
- These estimates are unreliable and should not be used as they tend to be over-optimistic!

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Reliability of Estimates

Resubstitution estimates (2)

- Why are they unreliable?
  - Models are obtained with the goal of optimizing the selected prediction error statistic on the given data set
  - In this context it is expected that they get good scores!
  - The given data set is just a sample of the unknown distribution of the problem being tackled
  - What we would like is to have the performance of the model on this distribution
  - As this is usually impossible the best we can do is to evaluate the model on **new samples** of this distribution

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Goal of Performance Estimation

Main Goal of Performance Estimation

Obtain a **reliable estimate** of the expected prediction error of a model on the unknown data distribution

- In order to be reliable it should be based on evaluation on unseen cases - *a test set*

Goal of Performance Estimation (2)

- Ideally we want to repeat the testing several times
- This way we can collect a series of scores and provide as our estimate the average of these scores, together with the standard error of this estimate
- In summary:
  - calculate the sample mean prediction error on the repetitions as an estimate of the true population mean prediction error
  - complement this sample mean with the standard error of this estimate

Goal of Performance Estimation (3)

- The golden rule of Performance Estimation:

  The data used for evaluating (or comparing) any models cannot be seen during model development.

Goal of Performance Estimation (4)

- An experimental methodology should:
  - Allow obtaining several prediction error scores of a model, $E_1, E_2, \cdots, E_k$
  - Such that we can calculate a sample mean prediction error

$$\overline{E} = \frac{1}{k} \sum_{i=1}^{k} E_i$$

  - And also the respective standard error of this estimate

$$SE(\overline{E}) = \frac{s_E}{\sqrt{k}}$$

  where s_E is the sample standard deviation of E measured as $\sqrt{\frac{1}{k-1} \sum_{i=1}^{k} (E_i - \overline{E})^2}$

The Holdout Method and Random Subsampling

- The holdout method consists on **randomly dividing the available data sample in two sub-sets** - one used for training the model; and the other for testing/evaluating it
  - A frequently used proportion is 70% for training and 30% for testing

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

The Holdout Method (2)

- If we have a small data sample there is the danger of either having a too small test set (unreliable estimates as a consequence), or removing too much data from the training set (worse model than what could be obtained with the available data)
- We only get one prediction error score - no average score nor standard error
- If we have a very large data sample this is actually the preferred evaluation method

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Random Subsampling

- The Random Subsampling method is a variation of holdout method and it simply consists of repeating the holdout process several times by randomly selecting the train and test partitions
- Has the same problems as the holdout with the exception that we already get several scores and thus can calculate means and standard errors
- If the available data sample is too large the repetitions may be too demanding in computation terms

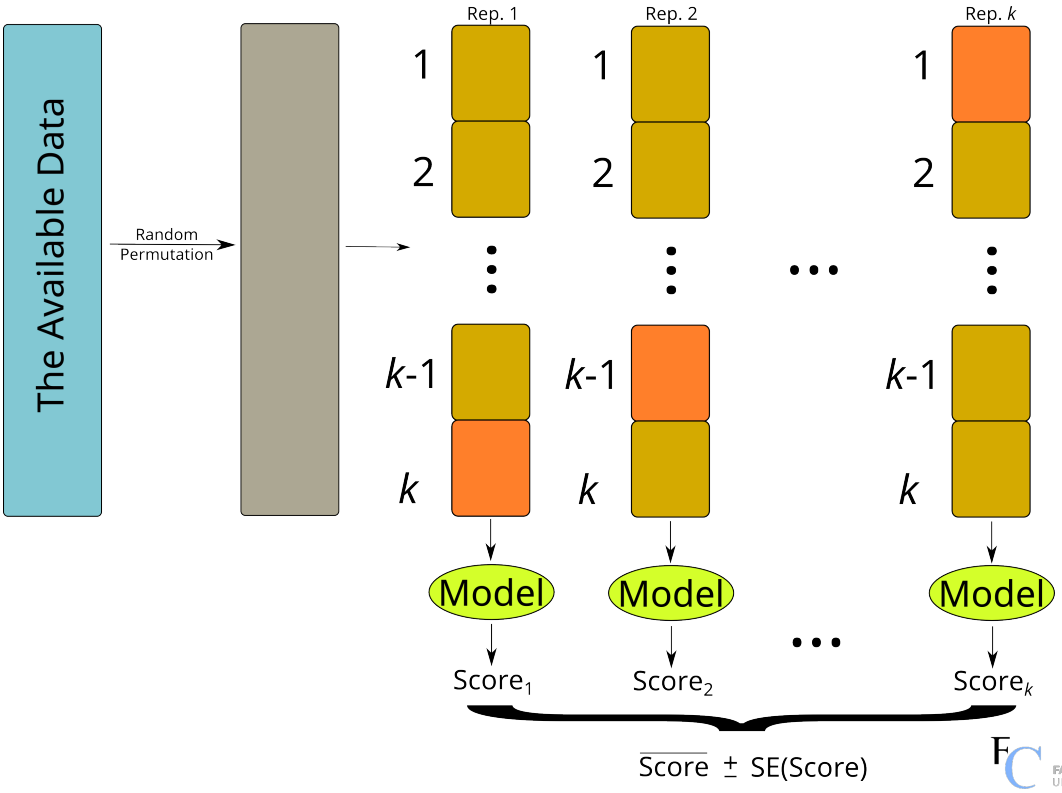
The Holdout method in R

```
library(DMwR)
set.seed(1234)
data(Boston,package='MASS')
## random selection of the holdout
trPerc <- 0.7
sp <- sample(1:nrow(Boston),as.integer(trPerc*nrow(Boston)))
## division in two samples
tr <- Boston[sp,]
ts <- Boston[-sp,]
## obtaining the model and respective predictions on the test set
m <- rpartXse(medv ~.,tr)
p <- predict(m,ts)
## evaluation
regr.eval(ts$medv,p,train.y=tr$medv)

##         mae         mse        rmse        mape        nmse        nmae
##   3.2320067 22.1312980  4.7043913  0.1731599  0.2369631  0.4529675
```

The k-fold Cross Validation Method

- The idea of k-fold Cross Validation (CV) is similar to random subsampling
- It essentially consists of *k* repetitions of training on part of the data and then test on the remaining
- The diference lies on the way the partitions are obtained

The k-fold Cross Validation Method (cont.)



Leave One Out Cross Validation Method (LOOCV)

- Similar idea to k-fold Cross Validation (CV) but in this case on each iteration a single case is left out of the training set
- This means it is essentially equivalent to *n*-fold CV, where *n* is the size of the available data set

The Bootstrap Method

- Train a model on a random sample of size *n* with replacement from the original data set (of size *n*)
  - Sampling with replacement means that after a case is randomly drawn from the data set, it is "put back on the sampling bag"
  - This means that several cases will appear more than once on the training data
  - On average only 63.2% of all cases will be on the training set
- Test the model on the cases that were not used on the training set
- Repeat this process many times (typically around 200)
- The average of the scores on these repetitions is known as the ϵ_0 bootstrap estimate
- The .632 bootstrap estimate is obtained by $.368 \times \epsilon_r + .632 \times \epsilon_0$, where ϵ_r is the resubstitution estimate

Bootstrap in R

```
data(Boston,package='MASS')
nreps <- 200
scores <- vector("numeric",length=nreps)
n <- nrow(Boston)
set.seed(1234)
for(i in 1:nreps) {
   # random sample with replacement
   sp <- sample(n,n,replace=TRUE)
   # data splitting
   tr <- Boston[sp,]
   ts <- Boston[-sp,]
   # model learning and prediction
   m <- lm(medv ~.,tr)
   p <- predict(m,ts)
   # evaluation
   scores[i] <- mean((ts$medv-p)^2)
}
# calculating means and standard errors
summary(scores)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   16.37   21.70   24.20   24.56   26.47   48.82
```


The Infra-Structure of package **performanceEstimation**

- The package **performanceEstimation** provides a set of functions that can be used to carry out comparative experiments of different models on different predictive tasks
- This infra-structure can be applied to any model/task/evaluation metric
- Installation:
  - Official release (from CRAN repositories):

```
install.packages("performanceEstimation")
```

  - Development release (from Github):

```
library(devtools)  # You need to install this package before!
install_github("ltorgo/performanceEstimation",ref="develop")
```

  - **Install from github!**

The Infra-Structure of package **performanceEstimation**

- The main function of the package is `performanceEstimation()`
  - It has 3 arguments:
    1. The predictive tasks to use in the comparison
    2. The models to be compared
    3. The estimation task to be carried out
- The function implements a wide range of experimental methodologies including all we have discussed

A Simple Example

- Suppose we want to estimate the mean squared error of regression trees in a certain regression task using cross validation

```
library(performanceEstimation)
library(DMwR)
data(Boston,package='MASS')
res <- performanceEstimation(
    PredTask(medv ~ .,Boston),
    Workflow("standardWF",learner="rpartXse"),
    EstimationTask(metrics="mse",method=CV(nReps=1,nFolds=10)))
```

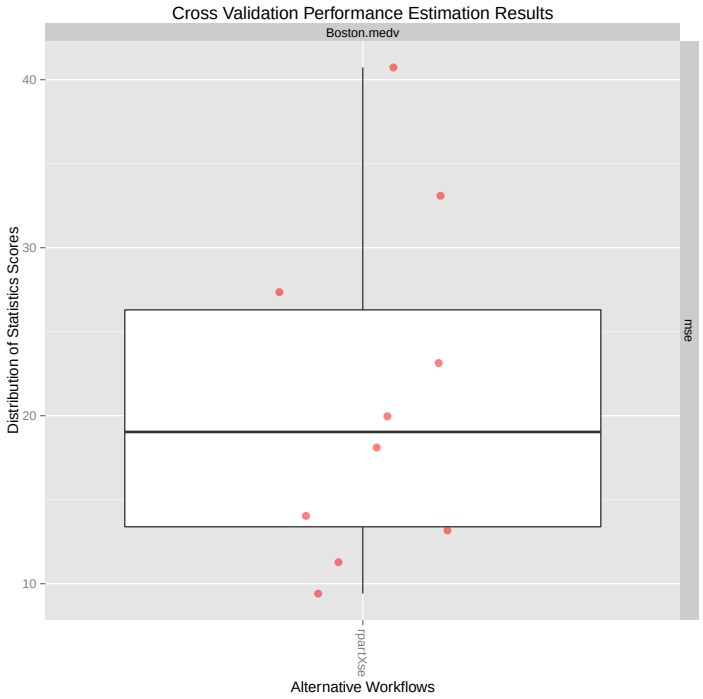
A Simple Example (2)

```
summary(res)
```

```
##
## == Summary of a  Cross Validation Performance Estimation Experiment ==
##
## Task for estimating  mse  using
##  1 x 10 - Fold Cross Validation
##   Run with seed =  1234
##
## * Predictive Tasks ::  Boston.medv
## * Workflows  ::  rpartXse
##
## -> Task:  Boston.medv
##   *Workflow: rpartXse
##               mse
## avg     21.02393
## std     10.15683
## med     19.02955
## iqr     12.91203
## min      9.40574
## max     40.72403
## invalid  0.00000
```


A Simple Example (3)

```
plot(res)
```



Predictive Tasks

- Objects of class **PredTask** describing a predictive task
  - Classification
  - Regression
  - Time series forecasting
- Created with the constructor with the same name

```
data(iris)
PredTask(Species ~ ., iris)

## Prediction Task Object:
##   Task Name         :: iris.Species
##   Task Type         :: classification
##   Target Feature    :: Species
##   Formula           :: Species ~ .
##   Task Data Source  :: iris

PredTask(Species  ~ ., iris,"IrisDS",copy=TRUE)

## Prediction Task Object:
##   Task Name         :: IrisDS
##   Task Type         :: classification
##   Target Feature    :: Species
##   Formula           :: Species ~ .
##   Task Data Source  :: internal  150x5 data frame.
```

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Workflows

- Objects of class **Workflow** describing an approach to a predictive task
  - Standard Workflows
    - Function `standardWF` for classification and regression
    - Function `timeseriesWF` for time series forecasting
  - User-defined Workflows

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Standard Workflows for Classification and Regression Tasks

```
library(e1071)
Workflow("standardWF",learner="svm",learner.pars=list(cost=10,gamma=0.1))

## Workflow Object:
##  Workflow ID       ::  svm
##  Workflow Function ::  standardWF
##       Parameter values:
##   learner  ->  svm
##   learner.pars  ->  cost=10 gamma=0.1
```

"standardWF" can be omitted ...

```
Workflow(learner="svm",learner.pars=list(cost=5))

## Workflow Object:
##  Workflow ID       ::  svm
##  Workflow Function ::  standardWF
##       Parameter values:
##   learner  ->  svm
##   learner.pars  ->  cost=5
```


Standard Workflows for Classification and Regression Tasks (cont.)

- Main parameters of the constructor:
  - Learning stage
    - `learner` - which function is used to obtain the model for the training data
    - `learner.pars` - list with the parameter settings to pass to the learner
  - Prediction stage
    - `predictor` - function used to obtain the predictions (defaults to `predict()`)
    - `predictor.pars` - list with the parameter settings to pass to the predictor

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Standard Workflows for Classification and Regression Tasks (cont.)

- Main parameters of the constructor (cont.):
  - Data pre-processing
    - `pre` - vector with function names to be applied to the training and test sets before learning
    - `pre.pars` - list with the parameter settings to pass to the functions
  - Predictions post-processing
    - `post` - vector with function names to be applied to the predictions
    - `post.pars` - list with the parameter settings to pass to the functions

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Standard Workflows for Classification and Regression Tasks (cont.)

```
data(algae,package="DMwR")
res <- performanceEstimation(
    PredTask(a1 ~ .,algae[,1:12],"A1"),
    Workflow(learner="lm",pre="centralImp",post="onlyPos"),
    EstimationTask("mse",method=CV())       # defaults to 1x10-fold CV
                                )

##
##
## ##### PERFORMANCE ESTIMATION USING  CROSS VALIDATION  #####
##
## ** PREDICTIVE TASK :: A1
##
## ++ MODEL/WORKFLOW :: lm
## Task for estimating  mse  using
##  1 x 10 - Fold Cross Validation
##   Run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
```

Evaluating Variants of Workflows

Function `workflowVariants()`

```
library(e1071)
data(Boston,package="MASS")
res2 <- performanceEstimation(
    PredTask(medv ~ .,Boston),
    workflowVariants(learner="svm",
                     learner.pars=list(cost=1:5,gamma=c(0.1,0.01))),
    EstimationTask(metrics="mse",method=CV()))
```


Evaluating Variants of Workflows (cont.)

```
summary(res2)
```

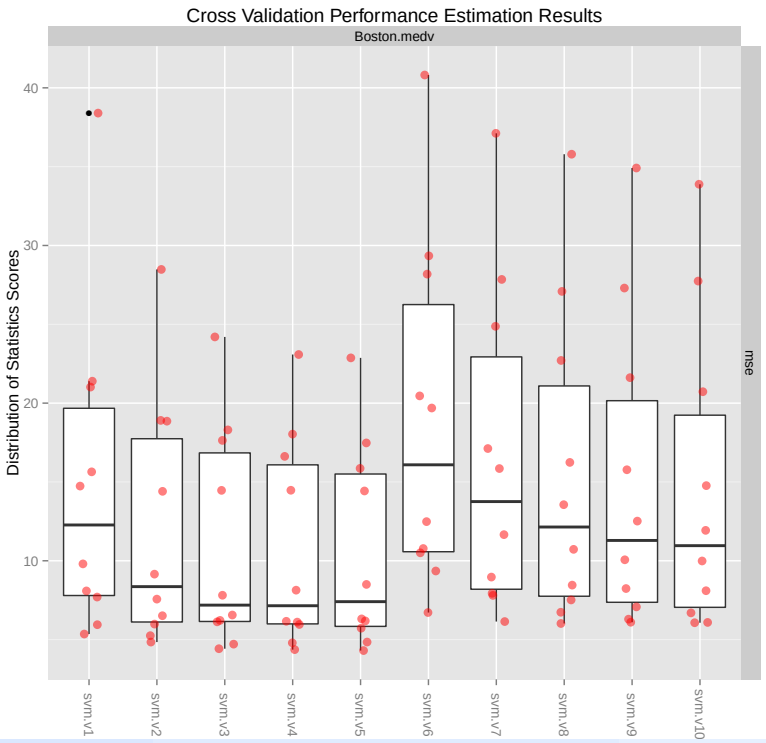
```
##
## == Summary of a  Cross Validation Performance Estimation Experiment ==
##
## Task for estimating  mse  using
##  1 x 10 - Fold Cross Validation
##   Run with seed =  1234
##
## * Predictive Tasks ::  Boston.medv
## * Workflows  ::  svm.v1, svm.v2, svm.v3, svm.v4, svm.v5, svm.v6, svm.v7, svm.v8, svm.v9, svm.v10
##
## -> Task:  Boston.medv
##   *Workflow: svm.v1
##              mse
## avg     14.80685
## std     10.15295
## med     12.27015
## iqr     11.87737
## min      5.35198
## max     38.39681
## invalid  0.00000
##
##   *Workflow: svm.v2
##               mse
## avg     11.995178
## std      7.908371
## med      8.359433
## iqr     11.626306
## min      4.842848
## invalid  0.000000
##
```

Exploring the Results

```
getWorkflow("svm.v1",res2)

## Workflow Object:
##  Workflow ID       ::  svm.v1
##  Workflow Function ::  standardWF
##       Parameter values:
##   learner.pars  ->  cost=1 gamma=0.1
##   learner  ->  svm

topPerformers(res2)

## $Boston.medv
##     Workflow Estimate
## mse   svm.v5    10.65
```


Visualizing the Results

```
plot(res2)
```



Estimation Tasks

- Objects of class **EstimationTask** describing the estimation task
  - Main parameters of the constructor
    - **metrics** - vector with names of performance metrics
    - **method** - object of class **EstimationMethod** describing the method used to obtain the estimates

```
EstimationTask(metrics=c("F","rec","prec"),method=Bootstrap(nReps=100))

## Task for estimating  F,rec,prec  using
## 100  repetitions of  e0  Bootstrap experiment
##   Run with seed =  1234
```


Performance Metrics

- Many classification and regression metrics are available
  - Check the help page of functions `classificationMetrics` and `regressionMetrics`
- User can provide a function that implements any other metric she/he wishes to use
  - Parameters **evaluator** and **evaluator.pars** of the `EstimationTask` constructor

Comparing Different Algorithms on the Same Task

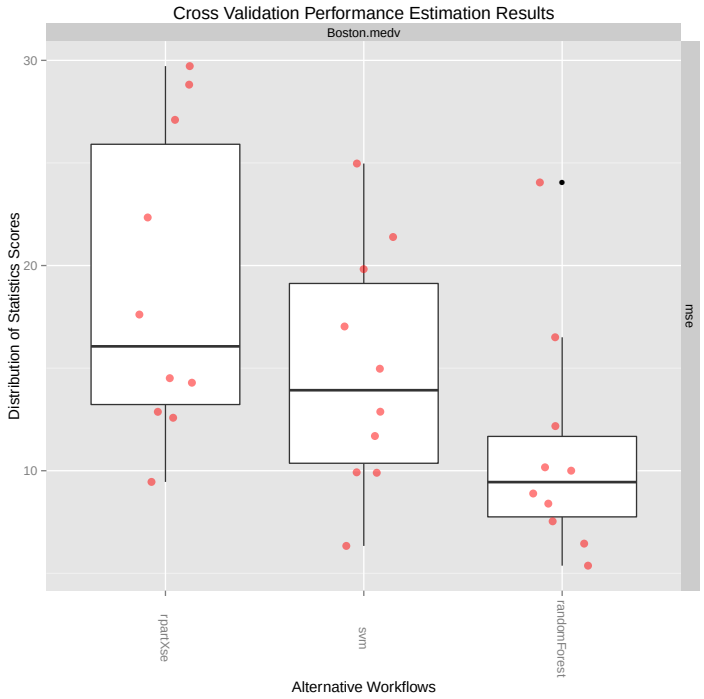
```
library(randomForest)
library(e1071)
res3 <- performanceEstimation(
    PredTask(medv ~ ., Boston),
    workflowVariants("standardWF",
             learner=c("rpartXse","svm","randomForest")),
    EstimationTask(metrics="mse",method=CV(nReps=2,nFolds=5)))
```


Some auxiliary functions

```
rankWorkflows(res3,3)

## $Boston.medv
## $Boston.medv$mse
##       Workflow Estimate
## 1 randomForest 10.95412
## 2          svm 14.89183
## 3     rpartXse 18.92990
```

The Results

```
plot(res3)
```

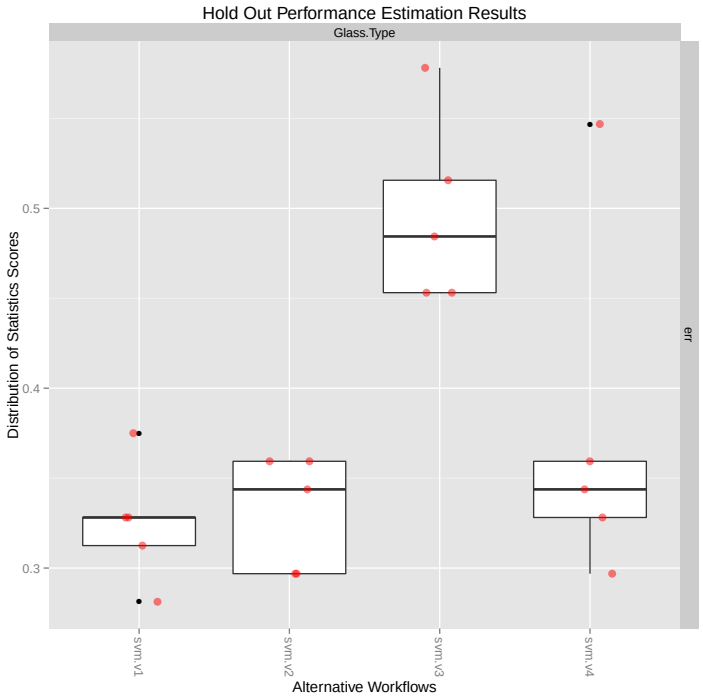


An example using Holdout and a classification task

```
data(Glass,package='mlbench')
res4 <- performanceEstimation(
    PredTask(Type ~ ., Glass),
    workflowVariants(learner="svm",   # You may omit "standardWF" !
                     learner.pars=list(cost=c(1,10),
                                       gamma=c(0.1,0.01))),
    EstimationTask(metrics="err",method=Holdout(nReps=5,hldSz=0.3)))
```

The Results

```
plot(res4)
```

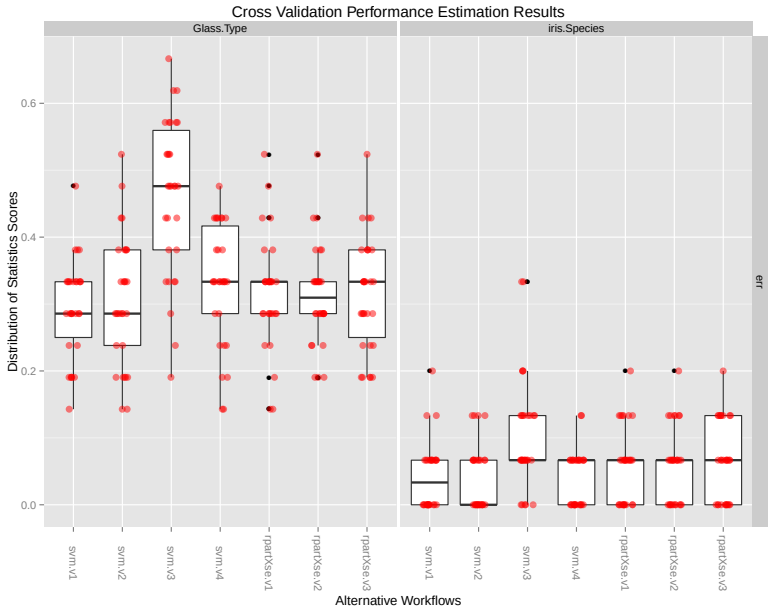


An example involving more than one task

```
data(Glass,package='mlbench')
data(iris)
res5 <- performanceEstimation(
    c(PredTask(Type ~ ., Glass),PredTask(Species ~.,iris)),
    c(workflowVariants(learner="svm",
                       learner.pars=list(cost=c(1,10),
                                         gamma=c(0.1,0.01))),
      workflowVariants(learner="rpartXse",
                       learner.pars=list(se=c(0,0.5,1)),
                       predictor.pars=list(type="class"))),
    EstimationTask(metrics="err",method=CV(nReps=3)))
```

The Results

```
plot(res5)
```



The Results (2)

```
topPerformers(res5)

## $Glass.Type
##     Workflow Estimate
## err   svm.v1    0.294
##
## $iris.Species
##     Workflow Estimate
## err   svm.v2     0.04

topPerformer(res5,"err","Glass.Type")

## Workflow Object:
##  Workflow ID       ::  svm.v1
##  Workflow Function ::  standardWF
##       Parameter values:
##   learner.pars  ->  cost=1 gamma=0.1
##   learner  ->  svm
```

Hands on Performance Estimation

the Algae data set

Load in the data set `algae` and answer the following questions:

1. Estimate the MSE of a regression tree for forecasting alga *a1* using 10-fold Cross validation.
2. Repeat the previous exercise this time trying some variants of random forests. Check what are the characteristics of the best performing variant.
3. Compare the results in terms of mean absolute error of the default variants of a regression tree, a linear regression model and a random forest, in the task of predicting alga a3. Use 2 repetitions of a 5-fold Cross Validation experiment.
4. Carry out an experiment designed to select what are the best models for each of the seven harmful algae. Use 10-fold Cross Validation. For illustrative purposes consider only the default variants of regression trees, linear regression and random forests.

Are the Observed Differences Statistically Significant?

Statistical Hypothesis Testing

- Test if some result is unlikely to have occurred by chance
- The null hypothesis: there is no difference among a set of alternative workflows
- This hypothesis is rejected if the result of the test has a p-value less than a certain selected threshold (typically 0.01 or 0.05, i.e. 99% or 95% confidence)

- There are many statistical tests that could be used
- The work by Demsar (2006) includes what are the current recommendations for different experimental setups

J. Demsar. Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research, 7:1–30, 2006.

Paired Comparisons on a Task

Wilcoxon Signed Rank Test

- The null hypothesis: the difference between the two workflows is zero
- This hypothesis is rejected if the result of the test has a p-value less than a certain selected threshold (typically 0.01 or 0.05, i.e. 99% or 95% confidence)

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

A Simple Example

```
library(performanceEstimation)
library(DMwR) # because of rpartXse
data(Boston,package="MASS")
res <- performanceEstimation(
    PredTask(medv ~ .,Boston),
    workflowVariants(learner="rpartXse",learner.pars=list(se=c(0,0.5,1))),
    EstimationTask(metrics="mse",method=CV(nReps=3,nFolds=10)))
```

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

A Simple Example

```
pres <- pairedComparisons(res)

## Warning in pairedComparisons(res): With less 2 tasks the Friedman,
Nemenyi and Bonferroni-Dunn tests are not calculated.

pres$mse$WilcoxonSignedRank.test

## , , Boston.medv
##
##              MedScore DiffMedScores   p.value
## rpartXse.v1 18.18101            NA        NA
## rpartXse.v2 19.54956   -1.36855309 0.5837571
## rpartXse.v3 18.21299   -0.03198033 0.5027610
```


Which ones are significant at some level?

```
signifDiffs(pres,p.limit=0.05)

## $mse
## $mse$WilcoxonSignedRank.test
## $mse$WilcoxonSignedRank.test$Boston.medv
##      MedScore DiffMedScores       p.value
##      18.18101            NA            NA
##
##
## $mse$t.test
## $mse$t.test$Boston.medv
##      AvgScore DiffAvgScores       p.value
##      19.65952            NA            NA
```

Paired Comparisons on a Multiple Tasks

Demsar (2006) recommended procedure

- Step 1: Friedman test
  - Null hypothesis: all workflows are equivalent and so their rankings across the tasks are equal
- If this hypothesis is rejected then we can move to the second step
  - Paired comparisons among all pairs of workflows
    - Nemenyi post-hoc test
    - Null hypothesis: there is no significant difference among the ranks of a certain pair of workflows
  - Paired comparisons against a baseline
    - Bonferroni-Dunn post-hoc test
    - Null hypothesis: there is no significant difference among the ranks of a certain workflow and the baseline

An Example with Several Tasks

```
library(performanceEstimation)
library(e1071)
library(randomForest)
tgts <- 12:18
tasks <- c()
for(t in tgts)
    tasks <- c(tasks,
               PredTask(as.formula(paste(colnames(algae)[t],'~ .')),
                        algae[,c(1:11,t)],
                        paste0("algaA",t-11),
                        copy=TRUE))
res.algae <- performanceEstimation(
    tasks,
    workflowVariants(learner=c("svm","lm","randomForest"),
                     pre="knnImp"),
    EstimationTask("mae",method=CV())
    )
```

An Example with Several Tasks (cont.)

Can we reject the hypothesis that the workflows have the same ranking across all tasks?

```
pres <- pairedComparisons(res.algae)
pres$mae$F.test

## $chi
## [1] 12.28571
##
## $FF
## [1] 43
##
## $critVal
## [1] 0.3574087
##
## $rejNull
## [1] TRUE
```


An Example with Several Tasks (cont.)

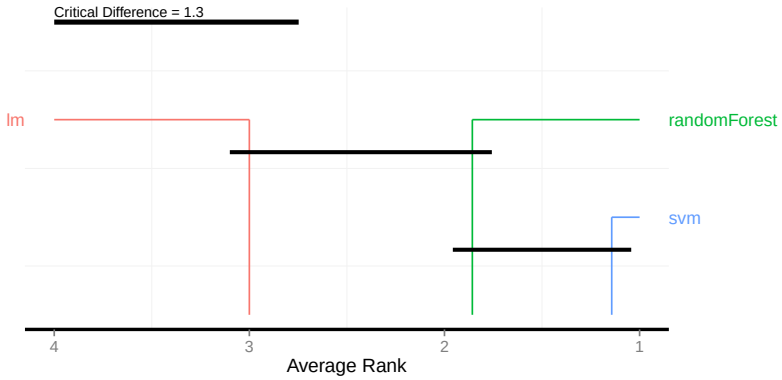
Are there any significant differences among the workflows?

```
pres$mae$Nemenyi.test

## $critDif
## [1] 1.252761
##
## $rkDifs
##                     svm       lm randomForest
## svm          0.0000000 1.857143    0.7142857
## lm           1.8571429 0.000000    1.1428571
## randomForest 0.7142857 1.142857    0.0000000
##
## $signifDifs
##                   svm    lm randomForest
## svm           FALSE  TRUE        FALSE
## lm             TRUE FALSE        FALSE
## randomForest FALSE FALSE        FALSE
```

CD diagrams for the Nemenyi test

Average rank differences that are not statistically significant are connected

```
CDdiagram.Nemenyi(pres)
```



An Example with Several Tasks (cont.)

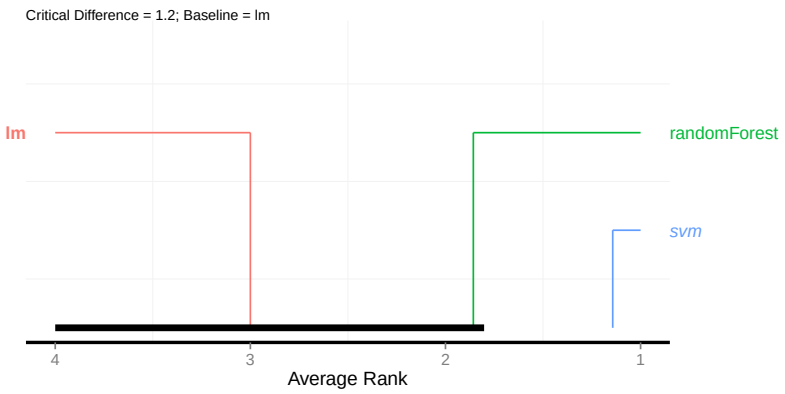
Suppose “lm” was our baseline system and we wanted to check if the other alternatives were able to improve over it on these tasks

```
pres <- pairedComparisons(res.algae,baseline="lm")
pres$mae$BonferroniDunn.test

## $critDif
## [1] 1.19808
##
## $baseline
## [1] "lm"
##
## $rkDifs
##          svm randomForest
##     1.857143     1.142857
##
## $signifDifs
##          svm randomForest
##         TRUE        FALSE
```

CD diagrams for the Bonferroni Dunn test

Are the average ranks of the other systems significantly better than the one of "lm"?

```
CDdiagram.BD(pres)
```



Hands on Statistical Significance

Using the *algae* data set from package **DMwR** answer the following questions

1. For the 7 different algae, choose a reasonable set of SVM variants and estimate their MSE error.
2. Check if these alternatives are significantly better than the SVM with the default parameter settings
3. Present the results of the previous question visually