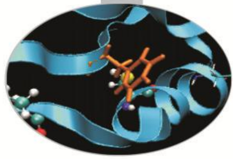
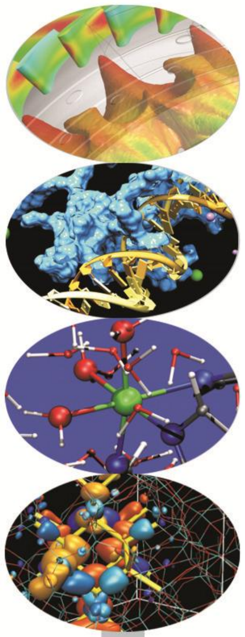


Predictive modelling / Machine Learning

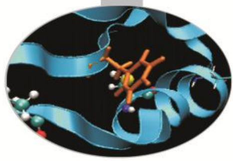
School on Scientific Data Analytics and Visualization



Roberta Turra, *Cineca*

21 June 2016

SuperComputing Applications and Innovation



Going back to the definition of data analytics …

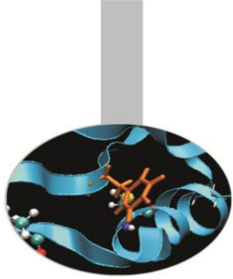
process of extracting valuable information from raw **data** using **algorithms** that discover hidden patterns

… **in order to correctly classify new observations**

It's always a data driven approach

Supervised

(use training samples with known classes to classify new data)



Going back to the techniques …

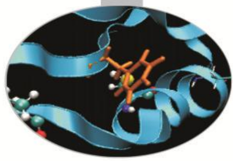
- predictive
  - classification (the learned attribute is categorical ,“nominal”)
    - Naive Bayes
    - Decision Trees
    - Neural Networks
    - KNN
    - Rocchio
    - Support Vectors Machine
    - …
  - regression (the learned attribute is numeric)

infer how to map input to output

Statisticians: model the process that gave rise to data
ML: make an accurate prediction, given the data

Pre–processing

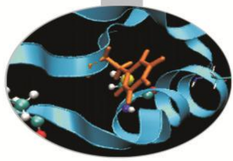
- data understanding and data quality assessment
  - Presence of missing values, outliers, inconsitencies
  - Level of noise
  - Redundance
- data preparation
  - Cleaning
  - Transformation (normalization, discretization, aggregation, new variables computation…)
  - Feature extraction
  - Selection / filtering
  - Train / Test set splitting



Data representation

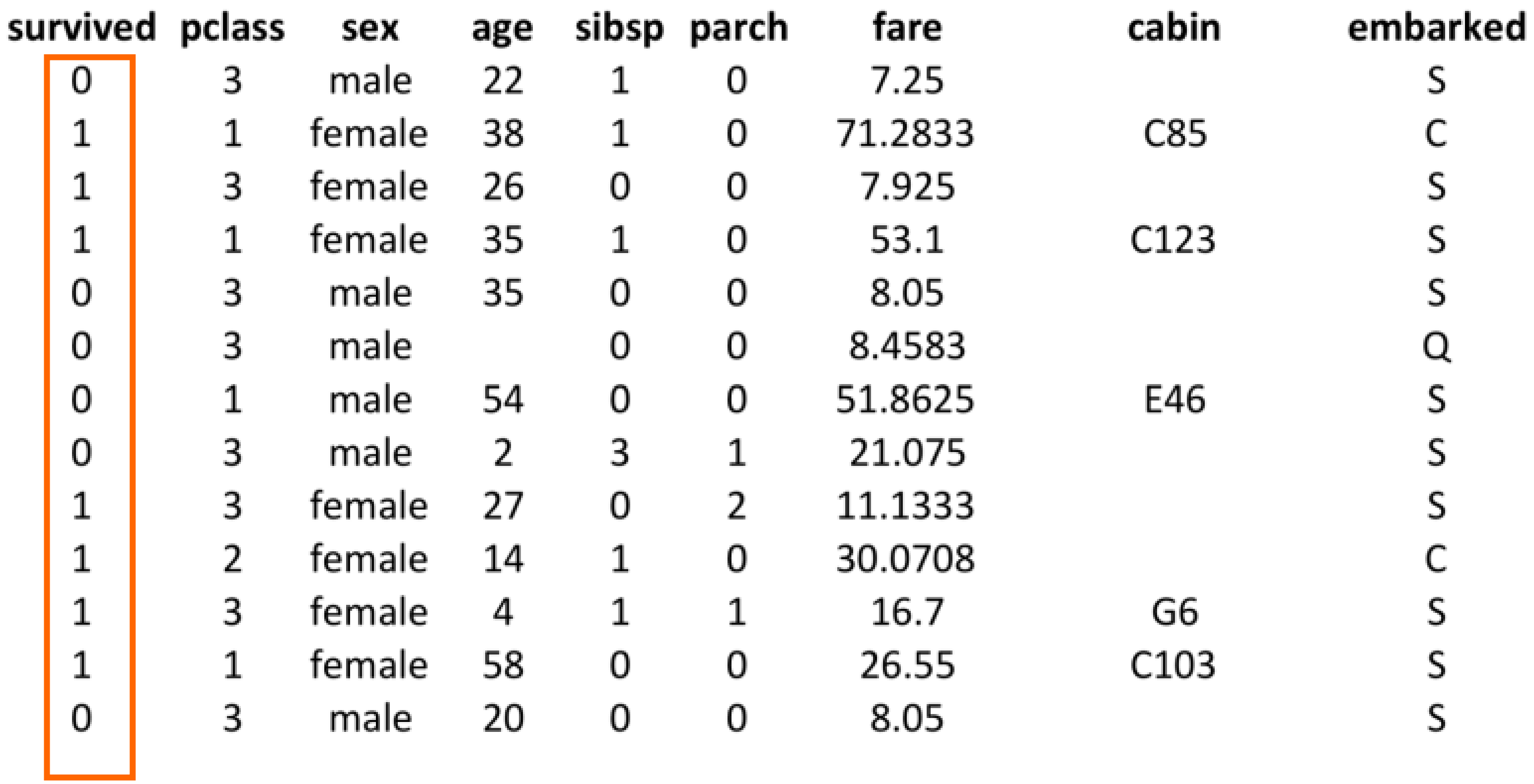
Analysis matrix

$$
\begin{array}{cccccc|l}
 & & \text{variable} & & & \text{target} & \\
X_{11} & X_{12} & X_{13} & \dots & X_{1d} & C_1 & \\
X_{21} & X_{22} & X_{23} & \dots & X_{2d} & C_2 & \text{observation} \\
\dots & & & & & & \\
X_{n1} & X_{n2} & X_{n3} & \dots & x_{nd} & C_n &
\end{array}
$$



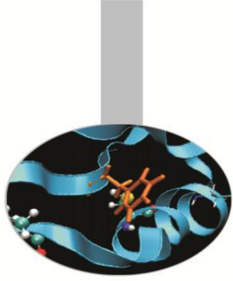
Titanic dataset

In 2012 Kaggle published this dataset to let researchers test the efficacy of their algorithms in **predicting survival on the Titanic**



survived	pclass	sex	age	sibsp	parch	fare	cabin	embarked
0	3	male	22	1	0	7.25		S
1	1	female	38	1	0	71.2833	C85	C
1	3	female	26	0	0	7.925		S
1	1	female	35	1	0	53.1	C123	S
0	3	male	35	0	0	8.05		S
0	3	male		0	0	8.4583		Q
0	1	male	54	0	0	51.8625	E46	S
0	3	male	2	3	1	21.075		S
1	3	female	27	0	2	11.1333		S
1	2	female	14	1	0	30.0708		C
1	3	female	4	1	1	16.7	G6	S
1	1	female	58	0	0	26.55	C103	S
0	3	male	20	0	0	8.05		S

target



Titanic dataset

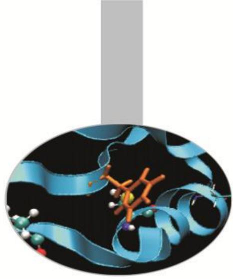
IF sex=‘female’ THEN survive=yes

ELSE IF sex=‘male’ THEN survive = no

confusion matrix

	no	yes	classified as
no	468	81	
yes	109	233	

(468 + 233) / (468+109+81+233) = 79% correct



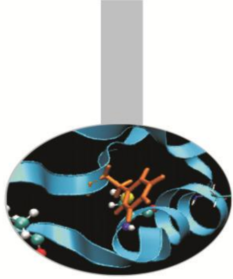
Titanic dataset

```
IF pclass='1' THEN survive=yes
ELSE IF pclass='2' THEN survive=yes
ELSE IF pclass='3' THEN survive=no
```

confusion matrix

	no	yes	classified as
no	372	177	
yes	119	223	

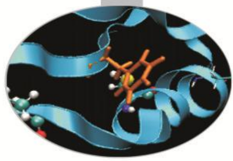
(372 + 223) / (372+119+223+177) = 67% correct



Titanic dataset

Strategy

- For each attribute A:
  - For each value V of that attribute, create a rule:
    1. count how often each class appears
    2. find the most frequent class, c
    3. make a rule "if A=V then Class=c"
  - Calculate the error rate of this rule
- Pick the attribute whose rules produce the lowest error rate



Titanic dataset

IF pclass=‘1’ AND sex=‘female’ THEN survive=yes
IF pclass=‘2’ AND sex=‘female’ THEN survive=yes
IF pclass=‘3’ AND sex=‘female’ AND age < 4 THEN survive=yes
IF pclass=‘3’ AND sex=‘female’ AND age >= 4 THEN survive=no
IF pclass=‘2’ AND sex=‘male’ THEN survive=no
IF pclass=‘3’ AND sex=‘male’ THEN survive=no
IF pclass=‘1’ AND sex=‘male’ AND age < 5 THEN survive=yes
…

Titanic dataset

IF pclass='1' AND sex='female' THEN survive=yes
IF pclass='2' AND sex='female' THEN survive=yes
IF pclass='3' AND sex='female' AND age < 4 THEN survive=yes
IF pclass='3' AND sex='female' AND age >= 4 THEN survive=no
IF pclass='2' AND sex='male' THEN survive=no
IF pclass='3' AND sex='male' THEN survive=no
IF pclass='1' AND sex='male' AND age < 5 THEN survive=yes
…

We might consider grouping redundant conditions

```
IF pclass='1' THEN
    IF sex='female' THEN survive=yes
    IF sex='male' AND age < 5 THEN survive=yes
IF pclass='2' …
```

$\Rightarrow$ Decision Tree

Learning

Three core components

- Representation

  how the data is classified (a hyperplane that separates the two classes? a decision tree? a neural network?)

  Usually a conditional probability distribution P(y|x) or a decision function f (x). The set of classifiers(or decision functions) is called the hypothesis space of the model.

- Evaluation

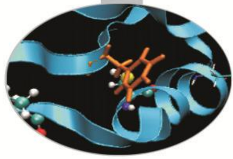
  how to determine if the classifier is a good representation (# of errors on some test set? precision and recall? residual sum of squares? likelihood?)

  In order to measure how well a function fits the training data, a loss function is defined (e.g. quadratic loss function $L(Y,f(X))=(Y-f(X))^2$). The risk function is the expected loss of f: $E[L(Y,f(X))]$ and can be estimated from the training data.

- Optimization

  how to make the model more efficient by reducing the search space (greedy search? gradient descent?)

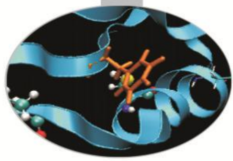
  The training (or learning) algorithm searches among the classifiers in the hypothesis space for the highest-scoring one. The choice of optimization technique is key to the efficiency of the model.



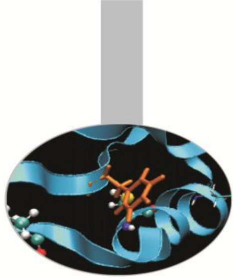
Learning

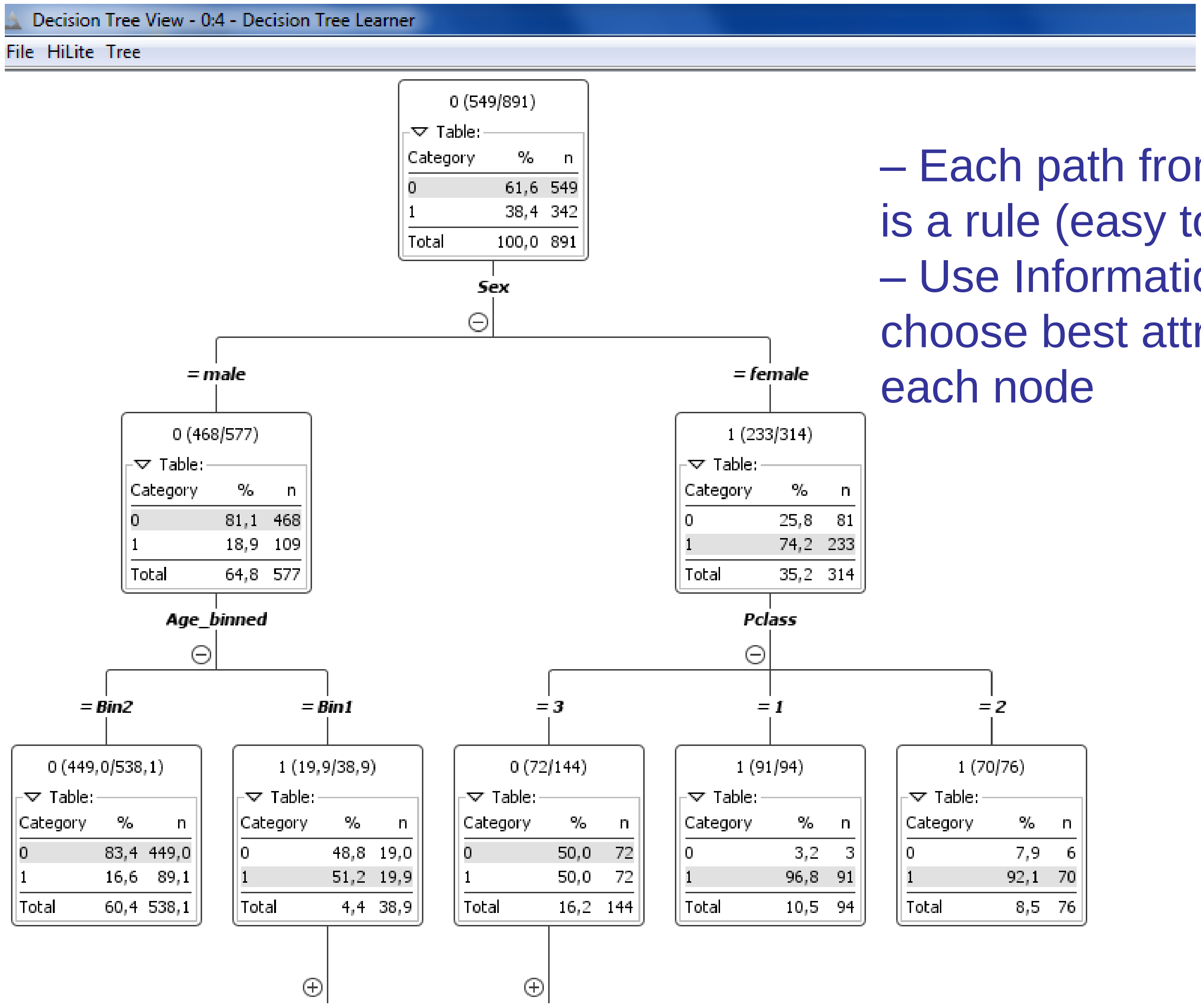
Three core components

- Representation
  - A set of rules: IF…THEN conditions
- Evaluation
  - coverage: # of data points that satisfy conditions
  - accuracy = # of correct predictions / coverage
- Optimization
  - Build rules by finding conditions that maximize accuracy



Decision Trees





0 (549/891)

Table:

Category	%	n
0	61,6	549
1	38,4	342
Total	100,0	891

Sex

= male

0 (468/577)

Table:

Category	%	n
0	81,1	468
1	18,9	109
Total	64,8	577

Age_binned

= Bin2

0 (449,0/538,1)

Table:

Category	%	n
0	83,4	449,0
1	16,6	89,1
Total	60,4	538,1

= Bin1

1 (19,9/38,9)

Table:

Category	%	n
0	48,8	19,0
1	51,2	19,9
Total	4,4	38,9

= female

1 (233/314)

Table:

Category	%	n
0	25,8	81
1	74,2	233
Total	35,2	314

Pclass

= 3

0 (72/144)

Table:

Category	%	n
0	50,0	72
1	50,0	72
Total	16,2	144

= 1

1 (91/94)

Table:

Category	%	n
0	3,2	3
1	96,8	91
Total	10,5	94

= 2

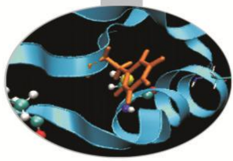
1 (70/76)

Table:

Category	%	n
0	7,9	6
1	92,1	70
Total	8,5	76

– Each path from the root is a rule (easy to interpret)

– Use Information Gain to choose best attribute at each node



Information gain

The expected information gain is the change in information entropy H from a prior state to a state that takes some information as given:

IG(T,a) = H(T) - H(T|a)

T = training set; a = an attribute value

$$\text{Entropy} = -\sum_i p_x \log_2 p_x$$

Higher entropy means the events being measured are less predictable (e.g. in a coin toss entropy is 1).

Which attribute do we choose at each level?

The one with the highest information gain

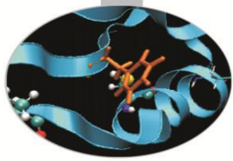
i.e. the one that reduces the unpredictability the most

Information gain

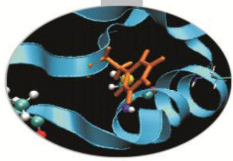
How unpredictable is your data?

342/891 survivors in titanic training set

$$-\left(\frac{342}{891}\log_2\frac{342}{891}+\frac{549}{891}\log_2\frac{549}{891}\right)=0.96$$



SuperComputing Applications and Innovation



Information gain

outlook	temperature	humidity	windy	play
overcast	cool	normal	TRUE	yes
overcast	hot	high	FALSE	yes
overcast	hot	normal	FALSE	yes
overcast	mild	high	TRUE	yes
rainy	cool	normal	TRUE	no
rainy	mild	high	TRUE	no
rainy	cool	normal	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
sunny	mild	normal	TRUE	yes

Before: 14 records, 9 are "yes"

$$-\left(\frac{9}{14}\log_2\frac{9}{14}+\frac{5}{14}\log_2\frac{5}{14}\right)=0.94$$

If we choose **outlook**:

overcast : 4 records, 4 are "yes"

$$-\left(\frac{4}{4}\log_2\frac{4}{4}\right)=0$$

rainy : 5 records, 3 are "yes"

$$-\left(\frac{3}{5}\log_2\frac{3}{5}+\frac{2}{5}\log_2\frac{2}{5}\right)=0.97$$

sunny : 5 records, 2 are "yes"

$$-\left(\frac{2}{5}\log_2\frac{2}{5}+\frac{3}{5}\log_2\frac{3}{5}\right)=0.97$$

Expected new entropy:

$$\frac{4}{14}\times 0.0+\frac{5}{14}\times 0.97+\frac{5}{14}\times 0.97$$

= <u>0.69</u>

Information gain

outlook	temperature	humidity	windy	play
overcast	cool	normal	TRUE	yes
overcast	hot	high	FALSE	yes
overcast	hot	normal	FALSE	yes
overcast	mild	high	TRUE	yes
rainy	cool	normal	TRUE	no
rainy	mild	high	TRUE	no
rainy	cool	normal	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
sunny	mild	normal	TRUE	yes

Before: 14 records, 9 are "yes"

$$-\left(\frac{9}{14}\log_2\frac{9}{14}+\frac{5}{14}\log_2\frac{5}{14}\right)=0.94$$

outlook

0.94 – 0.69 = $\boxed{0.25}$

temperature

0.94 – 0.91 = 0.03

humidity

0.94 – 0.725 = 0.215

windy

0.94 – 0.87 = 0.07

SuperComputing Applications and Innovation

Information gain - Continuous Attributes

Consider every possible binary partition; choose the partition with the highest gain

outlook	temperature	humidity	windy	play
rainy	mild	54	FALSE	yes
overcast	hot	58	FALSE	yes
overcast	cool	59	TRUE	yes
rainy	cool	60	FALSE	yes
overcast	mild	60	TRUE	yes
overcast	hot	62	FALSE	yes
rainy	mild	63	TRUE	no
sunny	cool	80	FALSE	yes
rainy	mild	81	FALSE	yes
sunny	mild	89	TRUE	yes
sunny	hot	90	FALSE	no
rainy	cool	90	TRUE	no
sunny	hot	90	TRUE	no
sunny	mild	92	FALSE	no

E(6/6)

= 0.0

E(3/8) + E(5/8)

= 0.95

E(9/10)+ E(1/10)

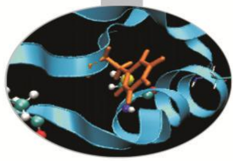
= 0.47

E(4/4)

= 0.0

Expect = 8/14*0.95 + 6/14*0
= 0.54

Expect = 10/14*0.47 + 4/14*0
= 0.33



Building a Decision Tree

- Assume attributes are discrete
  - Discretize continuous attributes
- Choose the attribute with the highest Information Gain
- Create branches for each value of attribute
- Partition examples on the basis of selected attributes
- Repeat with remaining attributes
- Stopping conditions
  - All examples assigned the same label
  - No examples left

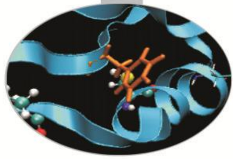
Problems

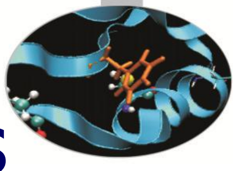
- Expensive to train
- Prone to **overfitting**
  - perform well on training data, bad on test data
  - pruning can help: remove or aggregate subtrees that provide little discriminatory power [are overspecialized …]

Test

Is the model able to generalize? Can it deal with unseen data, or does it overfit the data? Test on hold-out data:

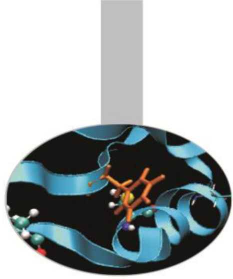
- **split** data to be modeled in training and test set
- **train** the model on training set
- evaluate the model on the training set
- **evaluate** the model on the test set
- difference between the fit on training data and test data measures the model’s ability to *generalize*





Division into training and test sets

- Fixed split
  - Leave out random N% of the data
- K-fold Cross-Validation
  - Select K folds without replace
- Leave-One-Out Cross Validation
  - Special case
- Bootstrap
  - Generate new training sets by sampling with replacement



Evaluation

Confusion matrix

The known class of test samples is matched against the class predicted by the model

		Predicted labels (model)		
		False	**True**	
True labels (target)	**False**	TN	FP	Specificity TN / (FP+TN)
	True	FN	TP	Sensitivity TP / (TP+FN)
		Negative Predictive Value TN / (TN + FN)	Positive Predictive Value TP / (TP + FP)	Accuracy (TP+TN) / (TP+FP+TN+FN)

⇨ Recall

⇩ Precision

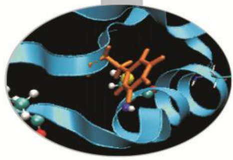
F-score = 2*Precision*Recall / (Precision + Recall)
Error rate = 1 – Precision
FP rate = 1 – Specificity

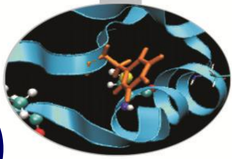
Evaluation

Accuracy

Need a baseline

- Base Rate
    - Accuracy of trivially predicting the most-frequent class
- Random Rate
    - Accuracy of making a random class assignment
- Naive Rate
    - Accuracy of some simple default or pre-existing model (e.g. “All females survived”)



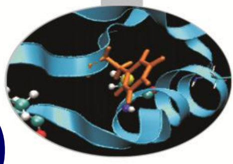


Kappa coefficient (Cohen's Kappa)

Measure of agreement between two raters

Kappa measures the percentage of data values in the main diagonal of the table and then adjusts these values for the amount of agreement that could be expected due to chance alone.

$$\kappa = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)}$$



Kappa coefficient (Cohen's Kappa)

Calculation (example)

		B	
		Yes	No
A	Yes	20	5
	No	10	15

Evaluation of grant proposals.

The observed agreement is Pr(a) = (20 + 15) / 50 = 0.70

To calculate Pr(e) (the probability of random agreement) we note that:

Rater A said "Yes" to 25 applicants and "No" to 25 applicants. Thus rater A said "Yes" 50% of the time.

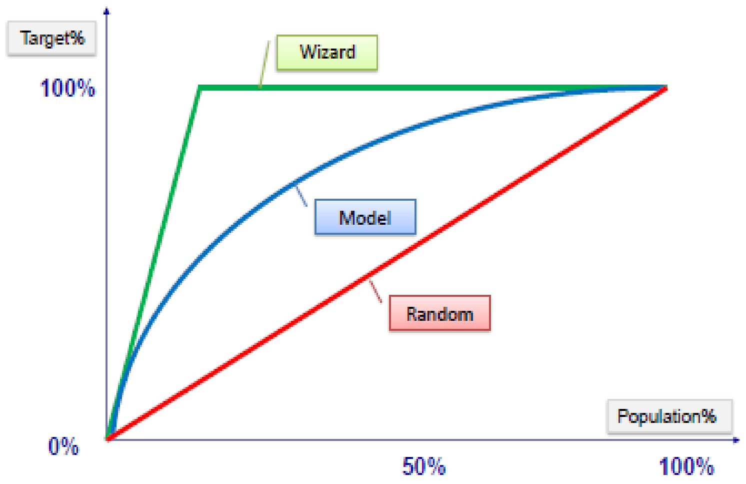
Rater B said "Yes" to 30 applicants and "No" to 20 applicants. Thus rater B said "Yes" 60% of the time.

Therefore the probability that both of them would say "Yes" randomly is 0.50 · 0.60 = 0.30 and the probability that both of them would say "No" is 0.50 · 0.40 = 0.20. Thus the overall probability of random agreement is Pr(e) = 0.3 + 0.2 = 0.5.

$$\kappa = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)} = \frac{0.70 - 0.50}{1 - 0.50} = 0.40$$

CINECA SCAI SuperComputing Applications and Innovation

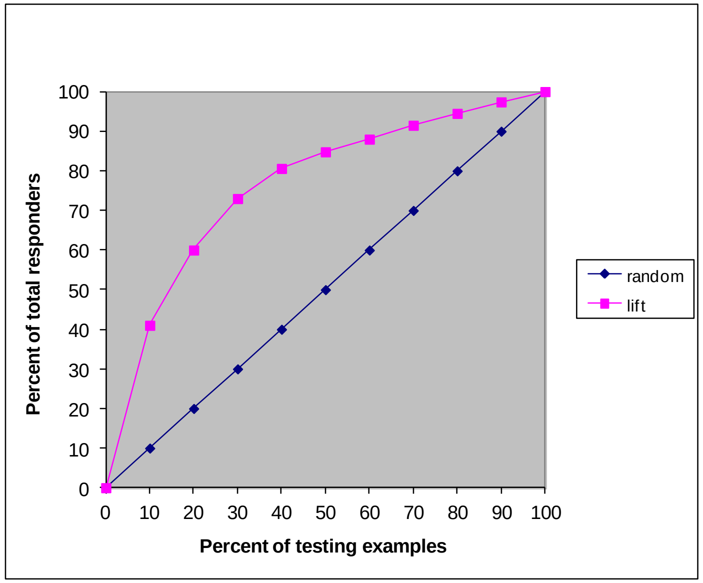
Gain and Lift curves



Visual aids for evaluating the performance of classification models in a portion of the population.

Test output is usually a score. Compute percentiles on the score distribution and identify the TP in each percentile.

$$lift = \frac{\%\ positives > threshold}{\%\ dataset > threshold}$$

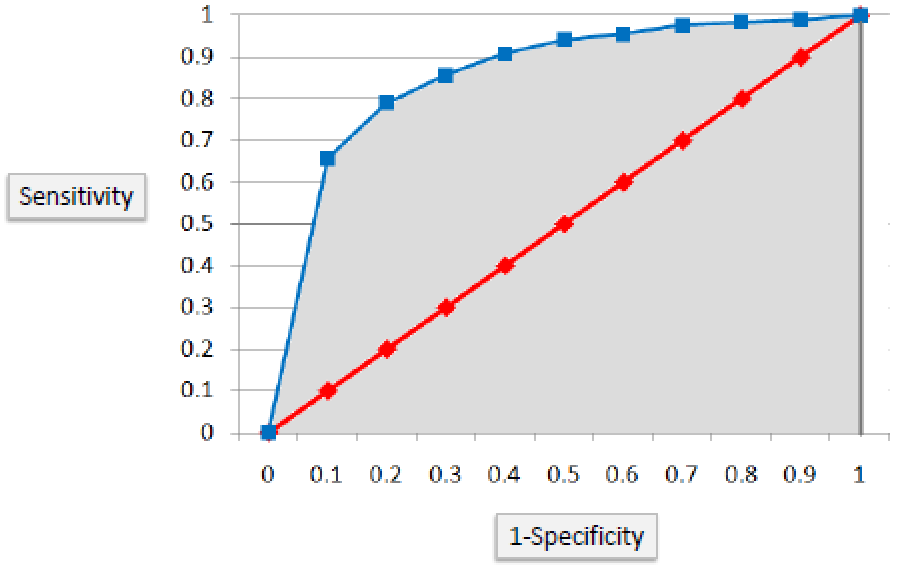


Lift curve for L'Equité: shows how much more likely we are to receive positive responses than if we contacted a random sample of customers. For example, by contacting only 10% of customers based on the predictive model we will reach 4 times as many respondents, as if we use no model.

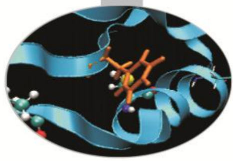
ROC curves

The ROC provides a means of comparison between classification models. The ROC chart shows false positive rate (1-specificity) on X-axis, the probability of target=1 when its true value is 0, against true positive rate (sensitivity) on Y-axis, the probability of target=1 when its true value is 1.

Area under ROC curve is often used as a measure of quality of the classification models. A random classifier has an area under the curve of 0.5, while AUC for a perfect classifier is equal to 1.



Receiver Operator Characteristic (used to measure accuracy of radar operators)



Bootstrap

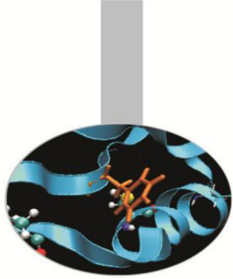
Given a dataset of size N

- Draw N samples with replacement to create a new dataset
- Repeat ~1000 times
- You now have ~1000 sample datasets
  - All drawn from the same population
  - You can compute ~1000 sample statistics
  - You can interpret these as repeated experiments

Very elegant use of computational resources

The bootstrap allows you to simulate repeated statistical experiments

Statistics computed from bootstrap samples are typically unbiased estimators



Ensembles

Combining classifiers

The output of a set of classifiers can be combined to derive a stronger classifier

(e.g. average results from different models)

- Better classification performance than individual classifiers
- More resilience to noise
- Time consuming
- Models become difficult to explain

Bagging

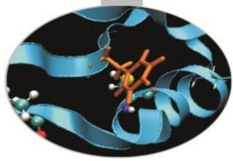
- Draw N bootstrap samples
- Retrain the model on each sample
- Average the results
    - Regression: Averaging
    - Classification: Majority vote

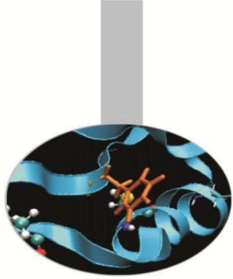
Works great for overfit models

Boosting: instead of selecting data points randomly, favor the misclassified points

- Initialize the weights
- Repeat:
    - Resample with respect to weights
    - Retrain the model
    - Recompute weights

The disadvantage of boosting, relative to big data, is that it’s inherently sequential (weights in time t depend from weights in time t-1; while in bagging everything can go parallel)





Random forest

Ensemble method based on decision trees

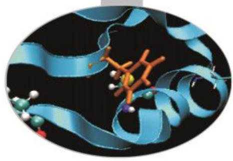
Repeat k times:

- Draw a bootstrap sample from the dataset
- Train a decision tree

  Until the tree is maximum size

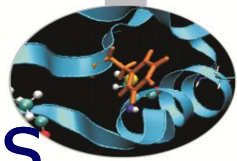
  - Choose next leaf node
  - Select <u>m attributes at random </u>from the p available
  - Pick the best attribute/split as usual
- Measure out-of-bag error
  - Evaluate against the samples that were not selected in the bootstrap
  - Provides measures of strength (inverse error rate), correlation between trees, and variable importance

Make a prediction by majority vote among the k trees



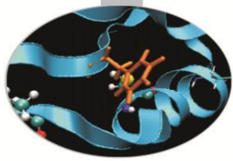
Random Forests

- General and powerful technique
- Easy to parallelize
  - Trees are built independently
- Work on categorical attributes
- Handles “small n big p” problems naturally
  - A subset of attributes are selected by importance
- Avoids overfitting (ensemble of models)



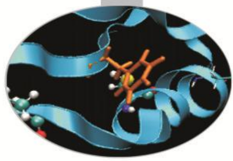
Decision Trees and Random Forests

- Representation
  - Decision Trees
  - Sets of decision trees with majority vote
- Evaluation
  - Accuracy
  - Random forests: out-of-bag error
- Optimization
  - Information Gain or Gini Index to measure impurity and select best attributes



Agenda

- Introduction to Supervised Learning: Decision Trees
  - Information gain and Entropy
  - Overfitting
- Evaluation
  - Train and test splitting
  - Accuracy, Precision and Recall
  - Lift and ROC curves
- Ensembles
  - Bootstrap
  - Bagging and Boosting
- Random Forests
- Other approaches – overview
- Model selection



Classification algorithms

- probabilistic classifiers
- symbolic algorithms
- example-based classifiers
- regression methods
- linear classifiers
- neural networks
- support vector machines
- …

Naïve Bayes

Decision trees, decision rules

KNN

LLSF

Rocchio

Discriminant analysis, profile based, incremental

SVM

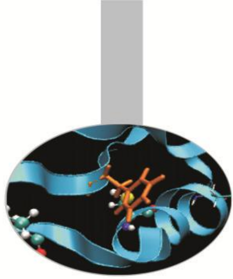
Maximum Entropy

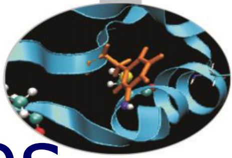
Bayesian classification

The classification problem may be formalized using **a-posteriori probabilities**:

P(C|X) = probability that the sample tuple $X=<x_1,\ldots,x_k>$ is of class C

Idea: assign to sample X the class label C such that **P(C|X) is maximal**





Estimating a-posteriori probabilities

Bayes theorem:

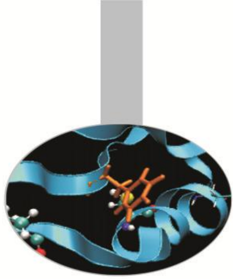
$$P(C|X) = P(X|C) \cdot P(C) / P(X)$$

P(X) is constant for all classes

P(C) = relative freq of class C samples

C such that **P(C|X)** is maximum = C such that **P(X|C)·P(C)** is maximum

Problem: computing P(X|C) is unfeasible!



Naïve Bayesian Classification

Naïve assumption: **attribute independence**

$$P(x_1,\ldots,x_k|C) = P(x_1|C)\cdot\ldots\cdot P(x_k|C)$$

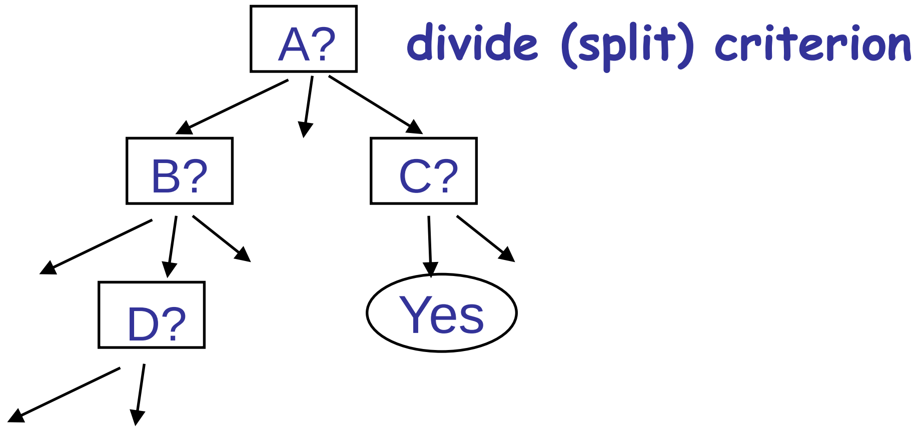
- If i-th attribute is categorical:
  $P(x_i|C)$ is estimated as the relative freq of samples having value x_i as i-th attribute in class C
- If i-th attribute is continuous:
  $P(x_i|C)$ is estimated thru a Gaussian density function

Computationally easy in both cases

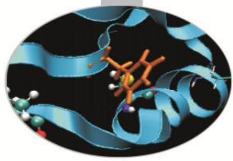
Decision trees

A tree where

internal node	= test on a single attribute
branch	= an outcome of the test
leaf node	= class or class distribution



One rule is generated for each path in the tree from the root to a leaf



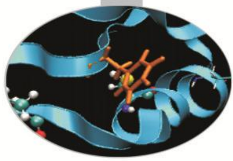
Symbolic classifiers

The key step is the choice of the condition on which to operate the partition, a choice which is generally made according to an **information gain** or **entropy** criterion.

“Fully grown” tree may be prone to **overfitting**, as some branches may be too specific to the training data. Most DT learning methods thus include a method for growing the tree and one for pruning it, that is, for removing the overly specific branches.

The decision tree method characterizes an observation in terms of a logical combination of features, which is simply a statement on the observation’s attributes, and does not involve any numeric computation.

A decision tree can be considered as a **set of rules**, since each path between the root and a leaf node specifies a set of conjoined conditions upon the outcome at the leaf.



Linear classifiers

A linear classifier is a **vector** belonging to the same d-dimensional space in which observations are represented (it is a separator in a metric space).

Classifying a new observation means measuring how close this observation is to the weight vector .

Weights are learned using training data.

Rocchio's classifier consists of an explicit profile (or prototypical observation) of the category.

It rewards the closeness of a test observation to the centroid of the positive training examples, and its distance from the centroid of the negative training examples.

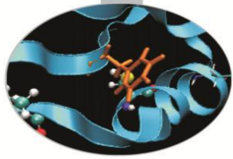
Linear classifiers

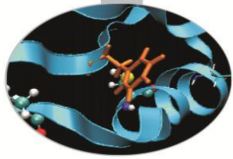
Batch methods build a classifier by analyzing the training set all at once.

One example of a batch method is linear discriminant analysis, however, the foremost example of a batch method is the Rocchio method.

On-line (incremental) methods build a classifier soon after examining the first training observation, and incrementally refine it as they examine new ones. This may be an advantage in the applications in which the training set is not available in its entirety from the start, or in which the "meaning" of the category may change in time.

A simple on-line method is the Perceptron algorithm.





k-nearest neighbors (KNN)

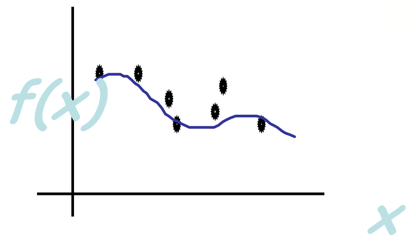
Rely on the category labels attached to the **k training observations that are most similar to the test observation**.

Don’t build an explicit representation of each category.

Need to define a **distance metric** and criteria for assigning a category, given the categories assigned to its k nearest neighbors:

- **majority class** among the k nearest neighbors
- use a **distance-weighted criteria**, so that the further a neighbor is from the observation, the less it contributes in the decision to assign that neighbor’s category

Regression



Model of continuous attributes as functions of other attributes.

The constructed model can be used for prediction (e.g., a model to predict the sales of a product given its price)

Many problems solvable by **linear regression**, where attribute Y (*response variable*) is modeled as a linear function of other attribute(s) X (*predictor variable(s)*):

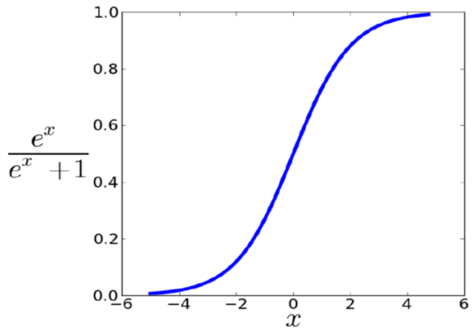
$$Y = a + b \cdot X$$

Coefficients a and b are computed from the samples using the **least square method**, that minimizes the error on the training set.

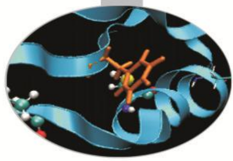
Logistic regression

You want to produce a categorical output (survived / not survived) and still use this numerical technique

Predict survival (y-axis) from (normalized) age (x-axis)



- Maps any number to the range (0,1)
- Interpret the result as a probability (what is the probability a passenger survived?)
- Interpret categorical classes numerically
- The optimization maximizes the probability of correct classification



Backpropagation

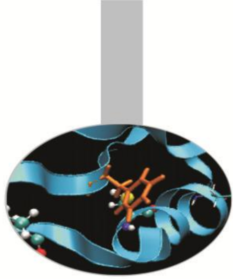
It’s a **neural network** algorithm, performing on multilayer feed-forward networks.

A network is a set of connected **input/output units** where each connection has an associated **weight**.

The weights are adjusted during the training phase, in order to correctly predict the class label for samples.

The simplest type of NN classifier is the *perceptron*, which is a linear classifier.

A nonlinear NN is a network with one ore more additional “layers” of units, that represent higher-order interactions between features



Radial Basis Function

It’s a neural **network algorithm**, performing on feed-forward networks with a single hidden layer of units whose activation function is a **basis function** (Gaussian or other) rather than the sigmoidal activation function.

The hidden layer units are called **centers**. Centers weights can be computed using Kohonen feature maps, k-means clustering, ...

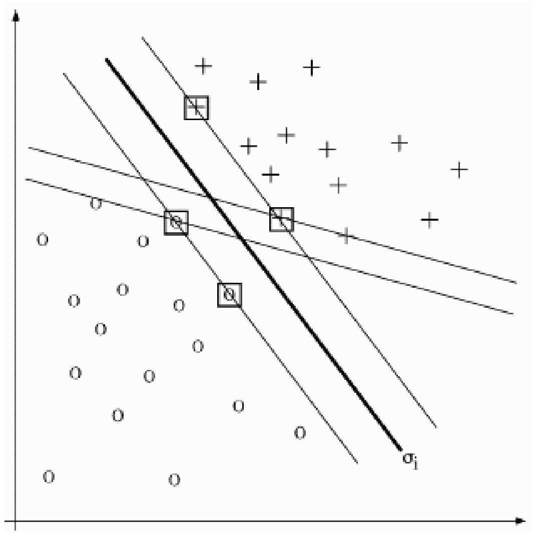
The basis function is a non linear function of distance from the center (radial).

Once the hidden layer weights are set, output weights are adjusted using the standard back propagation rule.

The prediction given for a particular set of attributes (a *point*) is a weighted sum of the basis functions at that point.

Train much faster than back propagation networks.

Support Vector Machines (SVM)



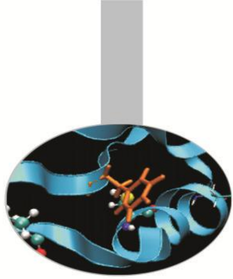
Find, among all the surfaces σ1, σ2, … in *d*-dimensional space that separate the positive from the negative training examples (**decision surfaces**), the σi that separates the positives from the negatives by the **widest possible margin** (such that the separation property is invariant with respect to the widest possible traslation of σi).

If the positives and the negatives are linearly separable, the decision surfaces are (d-1)-hyperplanes.

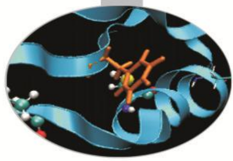
The “best” decision surface is determined by only a small set of training examples, called the *support vectors.*

We can soften the definition of margin to allow for missclassified points. Simple models tend to generalize better. **Regularization** strikes a balance between simplicity of the model and classification error and helps prevent models from overfitting the training data.

Overview of Classifiers



- probabilistic classifiers
  look at the distribution of features and compute probability of each class
- symbolic algorithms
  characterize an observation in terms of a logical combination of features, which is a statement on the observation’s attributes, and does not involve any numeric computation
- linear classifiers
  assume the existence of a multidimensional feature space and membership in a class is determined by the observation’s position in that space
- example-based classifiers
  don’t learn through induction (no explicit model of the class is built), only memorize the observations in the training set and their features
- regression methods
  approximation of a *real-valued* function by means of a function that fits the training data
- neural networks
  network of input/output units where each connection has an associated weight
- support vector machines
  identify the surface that separates the positives from the negatives by the widest possible margin
- …



Model selection

There is not a general rule, main criteria are based on:

- the **target variable**
  - quantitative (Regression, Regression Trees, KNN)
  - categorical (Logistic Regression, SVM, DT, Naive Bayes, NN, KNN, Rocchio, ...)
- the **main objective**
  - prediction accuracy (Random Forests, classifiers' comitee, Neural Networks)
  - interpretability (Linear Regression, Decision Trees)

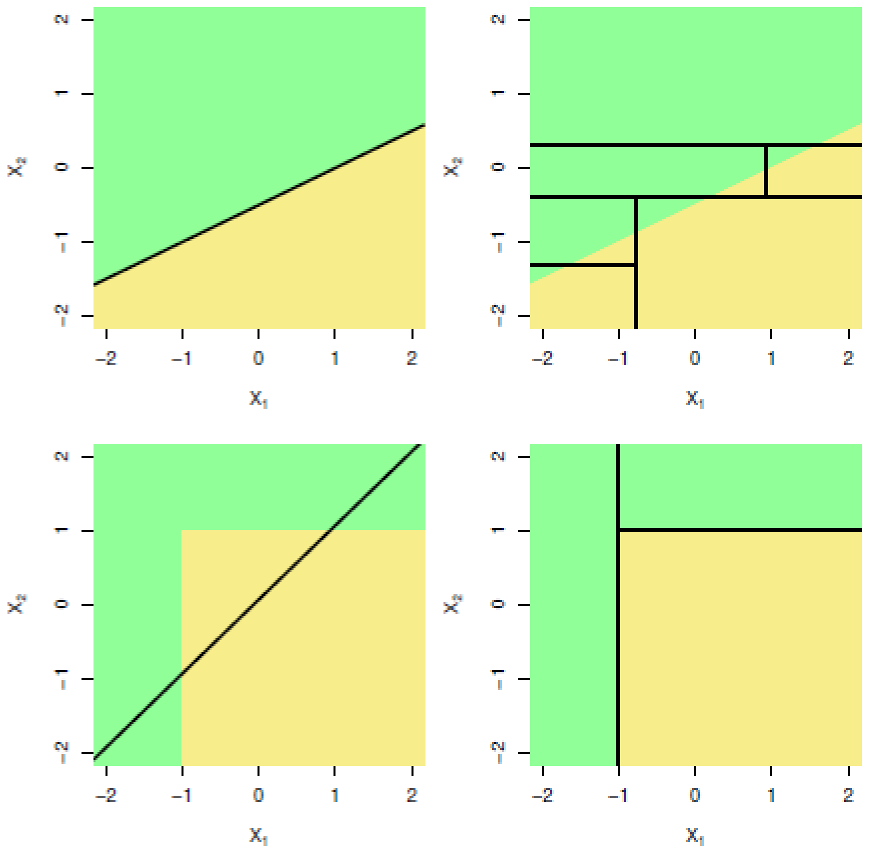
but should also be taken into account: the data quality (noise and missing values), the dimensionality, the computational effort, ...

Trees versus Linear models

Linear model | Tree-based model

True linear boudary

True non-linear boudary



Bias-Variance Trade-off

Suppose we have fit a model $\hat{f}(x)$ to some training data Tr, and let (x_0, y_0) be a test observation drawn from the population. If the true model is $Y = f(X) + \epsilon$ (with $f(x) = E(Y|X = x)$), then

expected prediction error

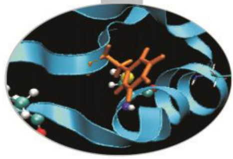
$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2 + \mathrm{Var}(\epsilon).$$

The expectation averages over the variability of y_0 as well as the variability in Tr. Note that $\mathrm{Bias}(\hat{f}(x_0))] = E[\hat{f}(x_0)] - f(x_0)$.

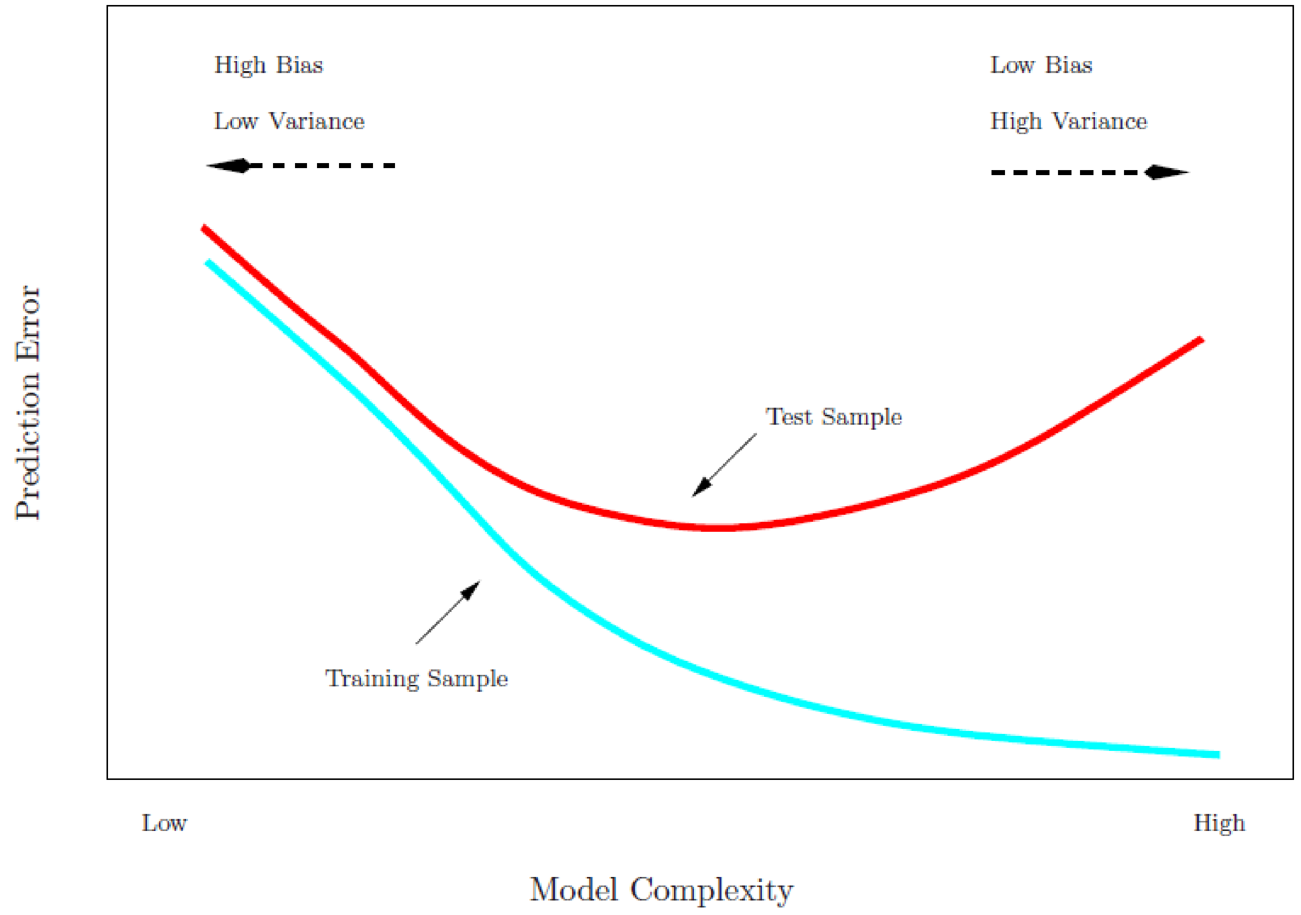
The **bias** is how far off on the average the model is from the truth.

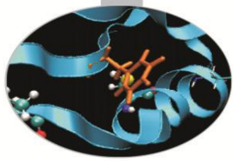
The **variance** is how much the estimate varies around its average.

Typically as the flexibility of the model increases, its variance increases, and its bias decreases. So choosing the flexibility based on average test error amounts to a bias-variance trade-off.



Training versus Test set performance





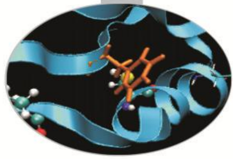
Model selection – a workflow

Identify the subset of models that suit your task.

Start with the simplest.

For each model set the parameters and the degree of complexity by means of a cross-validation process on the training set.

Use the test set to compare the performances.



Tools for classification / predictive modelling

(among many others)

KNIME

R

mahout

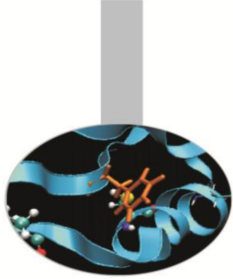
Spark MLlib

sas THE POWER TO KNOW.

stata®

CINECA SCAI SuperComputing Applications and Innovation

Model selection



Algorithm	Pros	Cons	Good at
Linear regression	- Very fast (runs in constant time) - Easy to understand the model - Less prone to overfitting	- Unable to model complex relationships -Unable to capture nonlinear relationships without first transforming the inputs	- The first look at a dataset - Numerical data with lots of features
Decision trees	- Fast - Robust to noise and missing values - Accurate	- Complex trees are hard to interpret - Duplication within the same sub-tree is possible	- Star classification - Medical diagnosis - Credit risk analysis
Neural networks	- Extremely powerful - Can model even very complex relationships - No need to understand the underlying data - Almost works by "magic"	- Prone to overfitting - Long training time - Requires significant computing power for large datasets - Model is essentially unreadable	- Images - Video - "Human-intelligence" type tasks like driving or flying - Robotics
Support Vector Machines	- Can model complex, nonlinear relationships - Robust to noise (because they maximize margins)	- Need to select a good kernel function - Model parameters are difficult to interpret - Sometimes numerical stability problems - Requires significant memory and processing power	- Classifying proteins - Text classification - Image classification - Handwriting recognition
K-Nearest Neighbors	- Simple - Powerful - No training involved ("lazy") - Naturally handles multiclass classification and regression	- Expensive and slow to predict new instances - Must define a meaningful distance function - Performs poorly on high-dimensionality datasets	- Low-dimensional datasets - Computer security: intrusion detection - Fault detection in semiconducter manufacturing - Video content retrieval - Gene expression - Protein-protein interaction

CINECA