An introduction to Machine Learning

Pierre Geurts
p.geurts@ulg.ac.be

Department of EE and CS &
GIGA-R, Bioinformatics and modelling
University of Liège

Outline

- Introduction
- Supervised Learning
- Other learning protocols/frameworks

Machine Learning: definition

- Machine Learning is concerned with the development, the analysis, and the application of algorithms that allow computers to learn
- Learning:
  - A computer learns if it improves its performance at some task with experience (i.e. by collecting data)
  - Extracting a model of a system from the sole observation (or the simulation) of this system in some situations.
  - A model = some relationships between the variables used to describe the system.
- Two main goals: make prediction and better understand the system

Machine learning: when ?

- Learning is useful when:
  - Human expertise does not exist (navigating on Mars),
  - Humans are unable to explain their expertise (speech recognition)
  - Solution changes in time (routing on a computer network)
  - Solution needs to be adapted to particular cases (user biometrics)
- Example: It is easier to write a program that learns to play checkers or backgammon well by self-play rather than converting the expertise of a master player to a program.

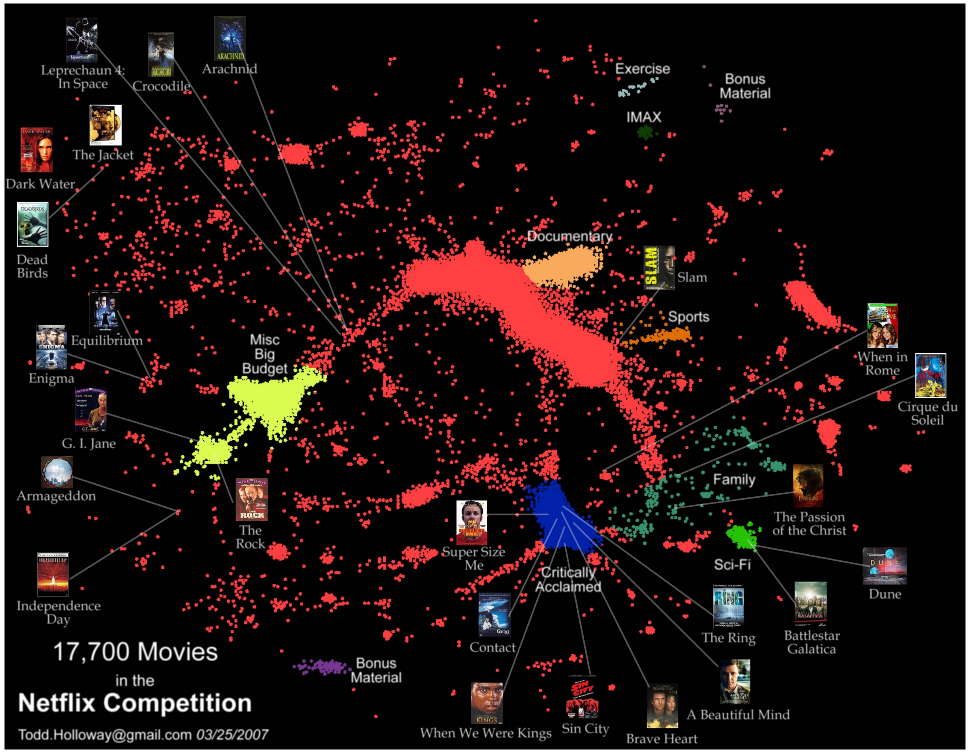
Applications: autonomous driving

- DARPA Grand challenge 2005: build a robot capable of navigating 175 miles through desert terrain in less than 10 hours, with no human intervention
- The actual wining time of Stanley [Thrun et al., 05] was 6 hours 54 minutes.

http://www.darpa.mil/grandchallenge/

Applications: recommendation system

- Netflix prize: predict how much someone is going to love a movie based on their movies preferences
- Data: over 100 million ratings that over 480,000 users gave to nearly 18,000 movies
- Reward: $1,000,000 dollars if 10% improvement with respect to Netflix's current system



http://www.netflixprize.com

Applications: credit risk analysis

- Data:

Customer103: (time=t0)	*Customer103:* (time=t1)	…	*Customer103:* (time=tn)
Years of credit: 9	Years of credit: 9		Years of credit: 9
Loan balance: $2,400	Loan balance: $3,250		Loan balance: $4,500
Income: $52k	Income: ?		Income: ?
Own House: Yes	Own House: Yes		Own House: Yes
Other delinquent accts: 2	Other delinquent accts: 2		Other delinquent accts: 3
Max billing cycles late: 3	Max billing cycles late: 4		Max billing cycles late: 6
Profitable customer?: ?	Profitable customer?: ?		**Profitable customer?: No**
...	...		...

- Logical rules automatically learned from data:

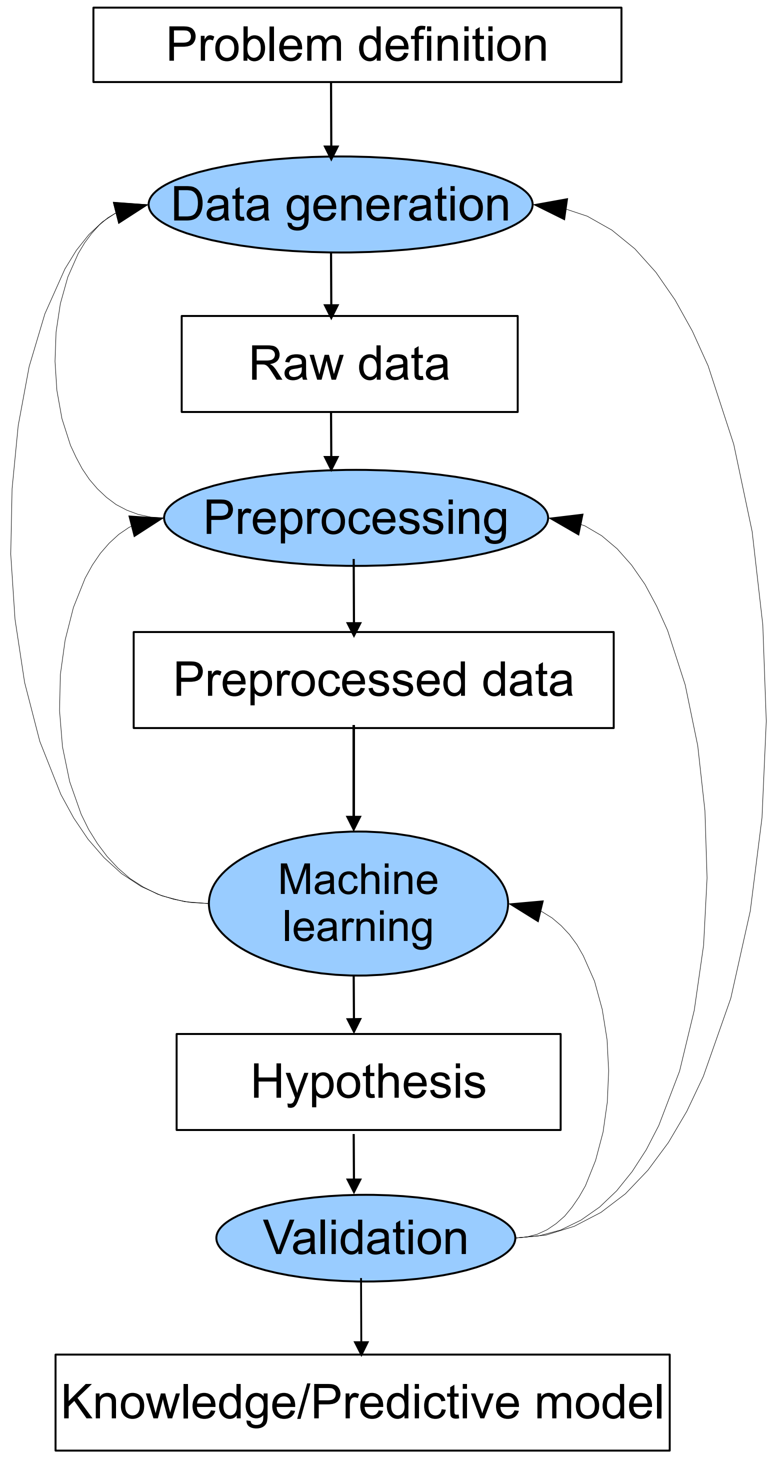
```
If   Other-Delinquent-Accounts > 2, and
     Number-Delinquent-Billing-Cycles > 1
Then Profitable-Customer? = No
     [Deny Credit Card application]
If   Other-Delinquent-Accounts = 0, and
     (Income > $30k)  OR  (Years-of-Credit > 3)
Then Profitable-Customer? = Yes
     [Accept Credit Card application]
```

Applications

- Machine learning has a wide spectrum of applications including:
  - Retail: Market basket analysis, Customer relationship management (CRM)
  - Finance: Credit scoring, fraud detection
  - Manufacturing: Optimization, troubleshooting
  - Medicine: Medical diagnosis
  - Telecommunications: Quality of service optimization, routing
  - Bioinformatics: Motifs, alignment
  - Web mining: Search engines
  - ...

Related fields

- Artificial Intelligence: smart algorithms
- Statistics: inference from a sample
- Computer Science: efficient algorithms and complex models
- Systems and control: analysis, modeling, and control of dynamical systems
- Data Mining: searching through large volumes of data



One part of the data mining process

- Each step generates many questions:
  - Data generation: data types, sample size, online/offline...
  - Preprocessing: normalization, missing values, feature selection/extraction...
  - Machine learning: hypothesis, choice of learning paradigm/algorithm...
  - Hypothesis validation: cross-validation, model deployment...

Glossary

- Data=a table (dataset, database, sample)

Variables (attributes, features) =
measurements made on objects

	VAR 1	VAR 2	VAR 3	VAR 4	VAR 5	VAR 6	VAR 7	VAR 8	VAR 9	VAR 10	VAR 11	...
Object 1	0	1	2	0	1	1	2	1	0	2	0	...
Object 2	2	1	2	0	1	1	0	2	1	0	2	...
Object 3	0	0	1	0	1	1	2	0	2	1	2	...
Object 4	1	1	2	2	0	0	0	1	2	1	1	...
Object 5	0	1	0	2	1	0	2	1	1	0	1	...
Object 6	0	1	2	1	1	1	1	1	1	1	1	...
Object 7	2	1	0	1	1	2	2	2	1	1	1	...
Object 8	2	2	1	0	0	0	1	1	1	1	2	...
Object 9	1	1	0	1	0	0	0	0	1	2	1	...
Object 10	1	2	2	0	1	0	1	2	1	0	1	...
...	...	...	...	...	...	...	...	...	...	...	...	...

Objects (samples, observations,
individuals, examples, patterns)

Dimension=number of variables
Size=number of objects

- Objects: samples, patients, documents, images...
- Variables: genes, proteins, words, pixels...

Outline

- Introduction
- Supervised Learning
  - Introduction
  - Model selection, cross-validation, overfitting
  - Some supervised learning algorithms
  - Beyond classification and regression
- Other learning protocols/frameworks

Supervised learning

Inputs | Output

A1	A2	A3	A4	Y
-0.69	-0.72	Y	0.47	Healthy
-2.3	-1.2	N	0.15	Disease
0.32	-0.9	N	-0.76	Healthy
0.37	-1	Y	-0.59	Disease
-0.67	-0.53	N	0.33	Healthy
0.51	-0.09	Y	-0.05	Disease

Learning sample

$$\xrightarrow{\text{Supervised learning}} \quad \hat{Y} = h(A_1, A_2, A_3, A_4)$$

model, hypothesis

- Goal: from the database (learning sample), find a function *h* of the inputs that approximates ***at best*** the output
- Symbolic output ⇒ *classification* problem,
- Numerical output ⇒ *regression* problem

Two main goals

- Predictive:

  Make predictions for a ***new*** sample described by its attributes

A1	A2	A3	A4	Y
0.83	-0.54	T	0.68	Healthy
-2.3	-1.2	F	-0.83	Disease
0.08	0.63	F	0.76	Healthy
0.06	-0.29	T	-0.57	Disease
-0.98	-0.18	F	-0.38	Healthy
-0.68	0.82	T	-0.95	Disease
0.92	-0.33	F	-0.48	?

- Informative:

  Help to understand the relationship between the inputs and the output

  Y=disease if $A3$=F and $A2$<0.3

  Find the most relevant inputs

Example of applications

- Biomedical domain: medical diagnosis, differentiation of diseases, prediction of the response to a treatment...

Gene expression, Metabolite concentrations...

Patients

A1	A2	...	A4	Y
-0.61	0.23	...	0.49	Healthy
-2.3	-1.2	...	-0.11	Disease
-0.82	-0.41	...	0.24	Healthy
-0.74	-0.1	...	-0.15	Disease
-0.14	0.98	...	-0.13	Healthy
-0.37	0.27	...	-0.67	Disease

Example of applications

- Perceptual tasks: handwritten character recognition, speech recognition...

Inputs:

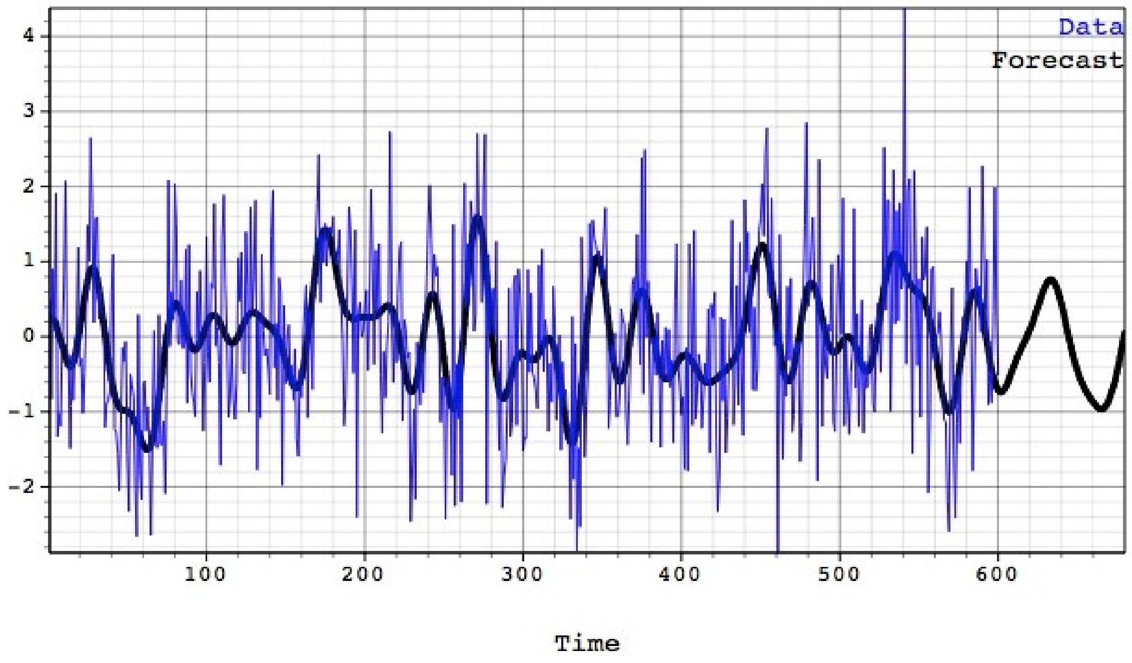
- a grey intensity [0,255] for each pixel
- each image is represented by a vector of pixel intensities
- eg.: 32x32=1024 dimensions

Output:

- 9 discrete values
- Y={0,1,2,...,9}

Example of applications

- Time series prediction: predicting electricity load, network usage, stock market prices...



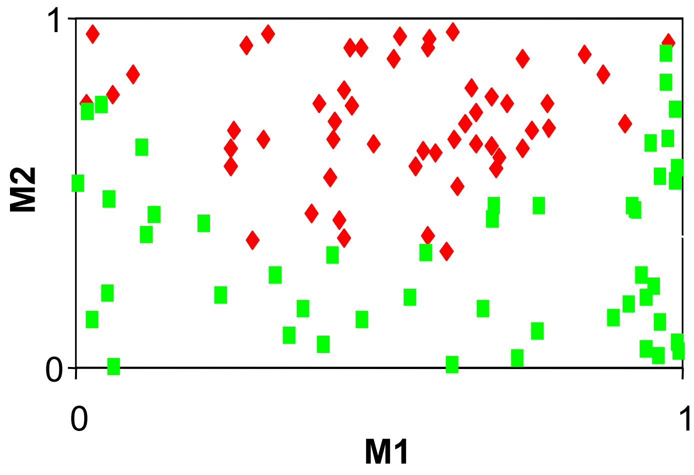
Outline

- Introduction
- Supervised Learning
  - Introduction
  - Model selection, cross-validation, overfitting
  - Some supervised learning algorithms
  - Beyond classification and regression
- Other learning protocols/frameworks

Illustrative problem

- Medical diagnosis from two measurements (eg., weights and temperature)

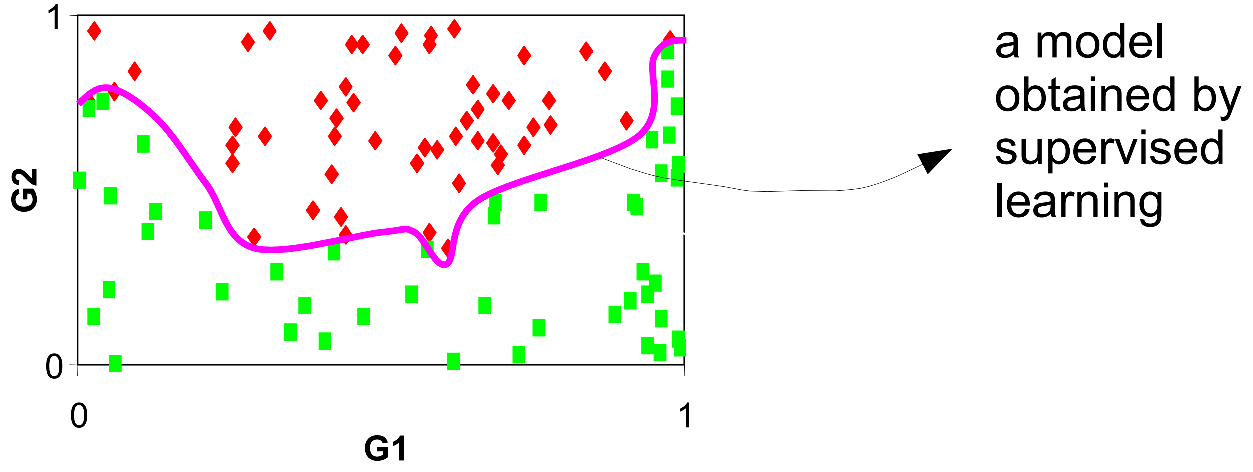
M1	M2	Y
0.52	0.18	Healthy
0.44	0.29	Disease
0.89	0.88	Healthy
0.99	0.37	Disease
…	…	…
0.95	0.47	Disease
0.29	0.09	Healthy



- Goal: find a model that classifies at best new cases for which M1 and M2 are known

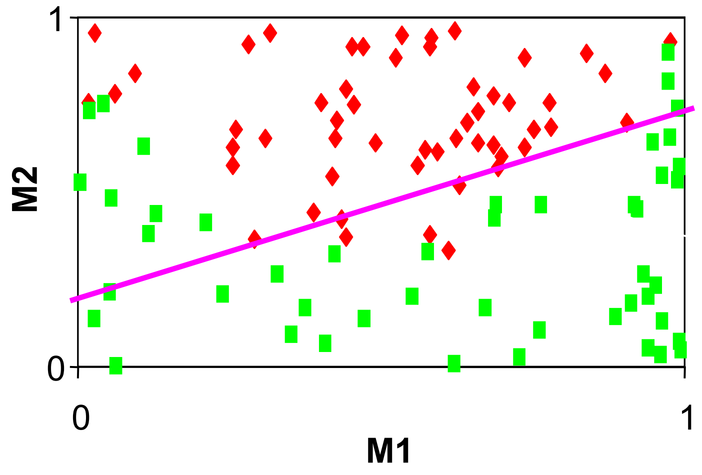
Learning algorithm

- A learning algorithm is defined by:
  - a family of candidate models (=hypothesis space *H*)
  - a quality measure for a model
  - an optimization strategy
- It takes as input a learning sample and outputs a function *h* in *H* of maximum quality



Linear model

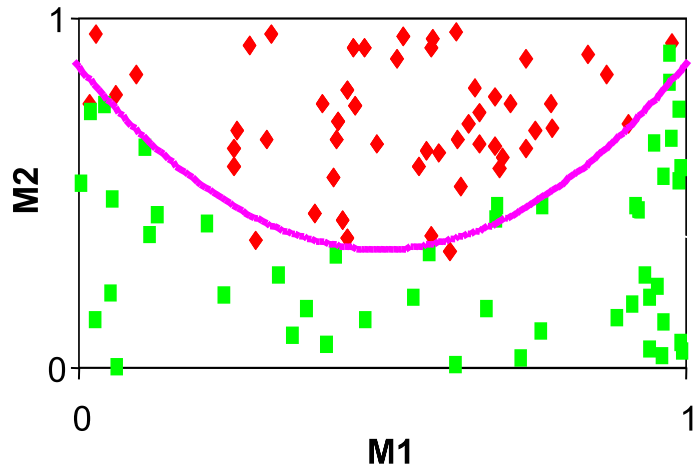
$$h(M1,M2)=\begin{cases}\text{Disease} & \text{if } w0+w1*M1+w2*M2>0\\ \text{Normal} & \text{otherwise}\end{cases}$$



- Learning phase: from the learning sample, find the best values for w0, w1 and w2
- Many alternatives even for this simple model (LDA, Perceptron, SVM...)

Quadratic model

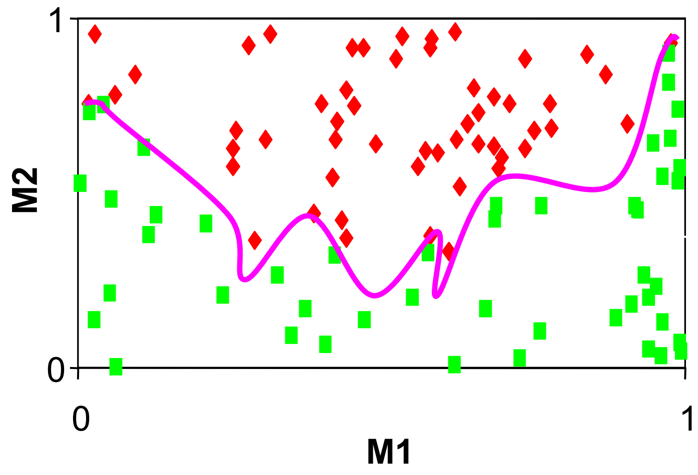
$$h(M1,M2)=\begin{cases}\text{Disease} & \text{if } w0+w1*M1+w2*M2+w3*M1^2+w4*M2^2>0\\ \text{Normal} & \text{otherwise}\end{cases}$$



- Learning phase: from the learning sample, find the best values for w0, w1,w2, w3 and w4
- Many alternatives even for this simple model (LDA, Perceptron, SVM...)

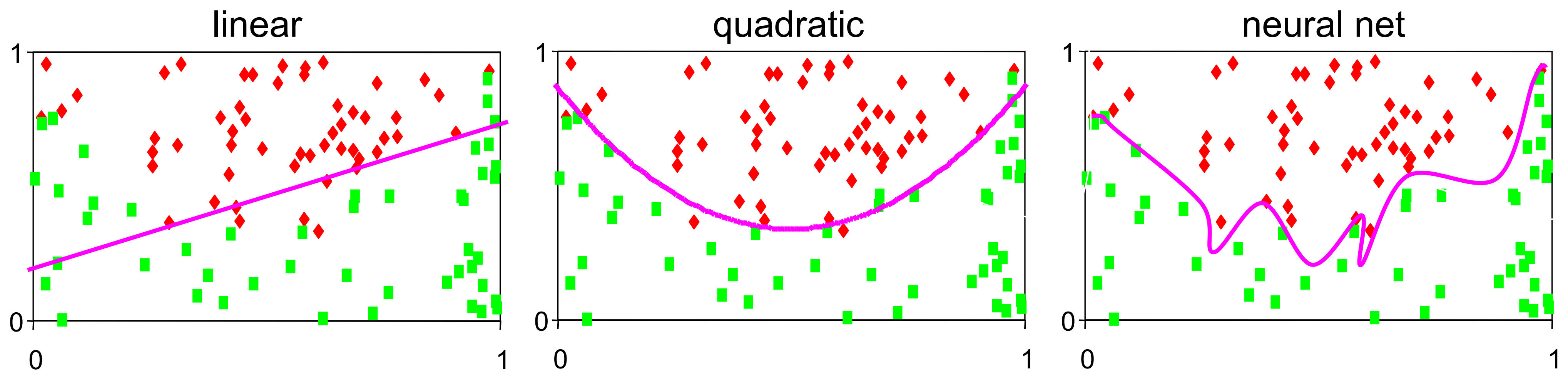
Artificial neural network

$$h(M1,M2)=\begin{cases}\text{Disease} & \text{if } \textit{some very complex function of M1,M2} > 0\\ \text{Normal} & \text{otherwise}\end{cases}$$



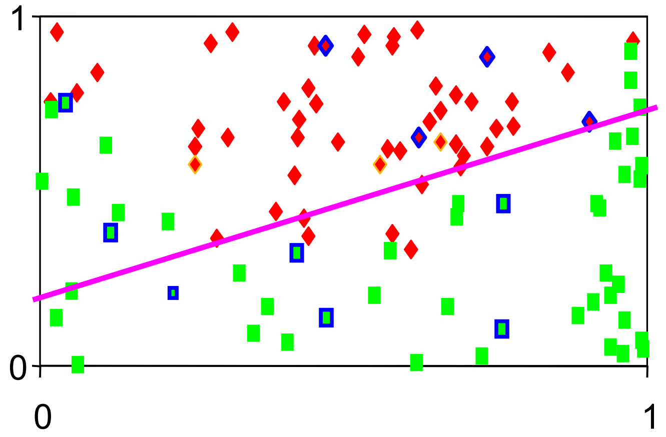
- Learning phase: from the learning sample, find the numerous parameters of the very complex function

Which model is the best?



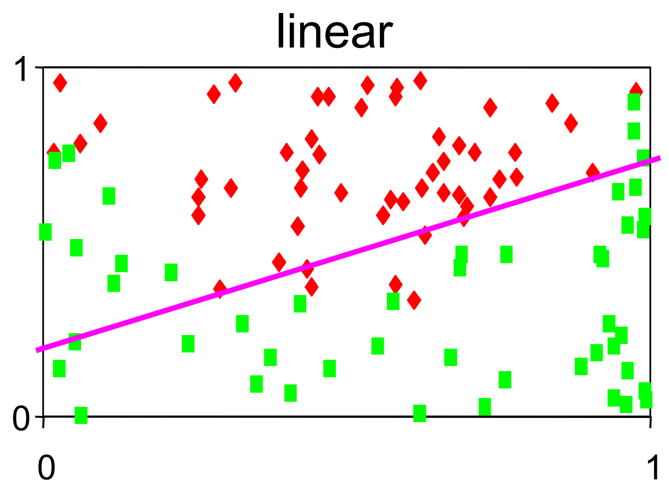
- Why not choose the model that minimises the error rate on the learning sample? (also called *re-substitution error*)
- How well are you going to predict future data drawn from the same distribution? (*generalisation error*)

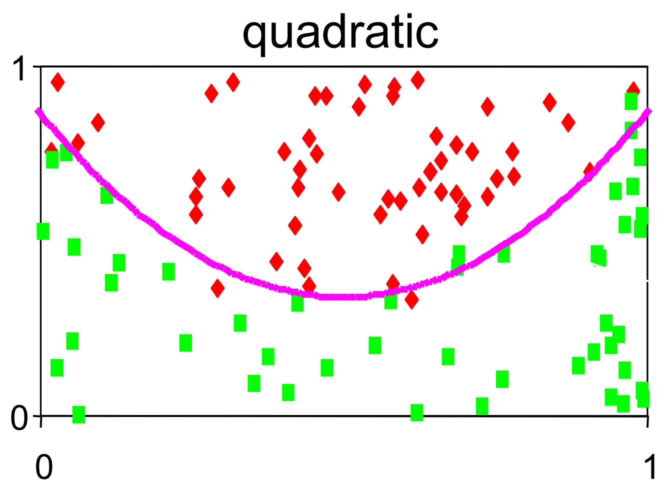
The test set method

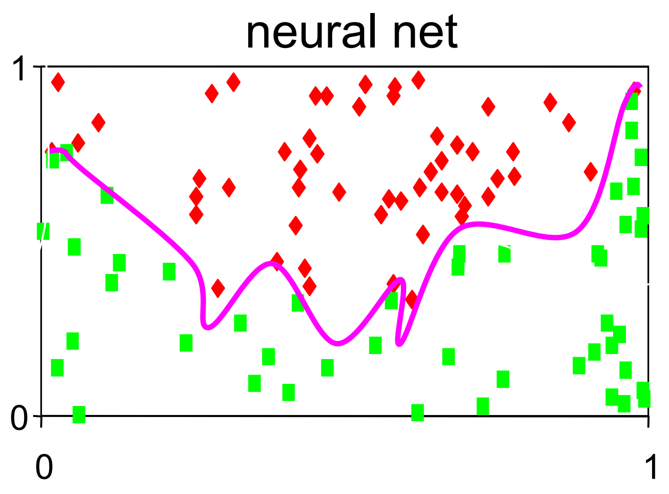


1. Randomly choose 30% of the data to be in a test sample
2. The remainder is a learning sample
3. Learn the model from the learning sample
4. Estimate its future performance on the test sample

Which model is the best?







LS error= 3.4%
TS error= 3.5%

LS error= 1.0%
TS error= 1.5%

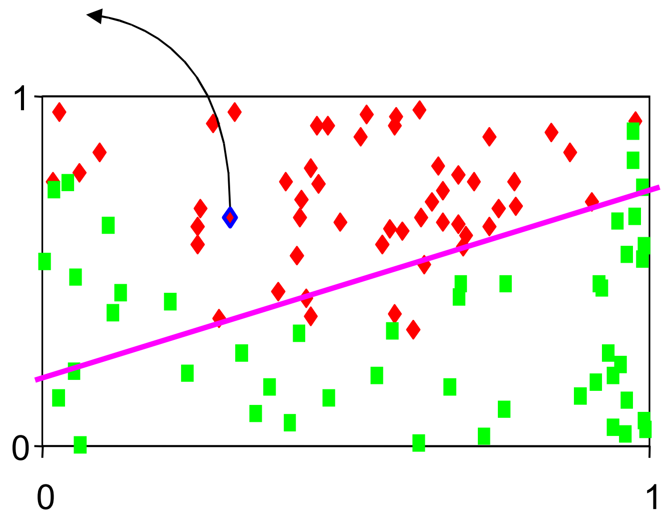
LS error= 0%
TS error= 3.5%

- We say that the neural network overfits the data
- Overfitting occurs when the learning algorithm starts fitting noise.
- (by opposition, the linear model underfits the data)

The test set method

- Upside:
  - very simple
  - Computationally efficient
- Downside:
  - Wastes data: we get an estimate of the best method to apply to 30% less data
  - Very unstable when the database is small (the test sample choice might just be lucky or unlucky)

Leave-one-out Cross Validation



For k=1 to N

- remove the kth object from the learning sample
- learn the model on the remaining objects
- apply the model to get a prediction for the kth object

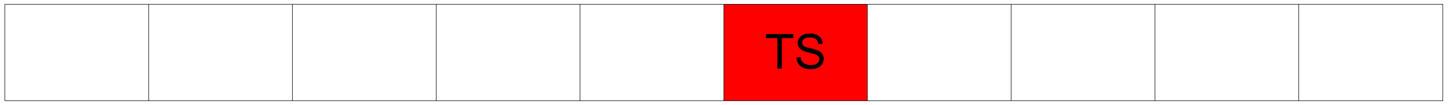
report the proportion of missclassified objects

Leave-one-out Cross Validation

- Upside:
  - Does not waste the data (you get an estimate of the best method to apply to N-1 data)
- Downside:
  - Expensive (need to train N models)
  - High variance

k-fold Cross Validation

- Randomly partition the dataset into *k* subsets (for example 10)

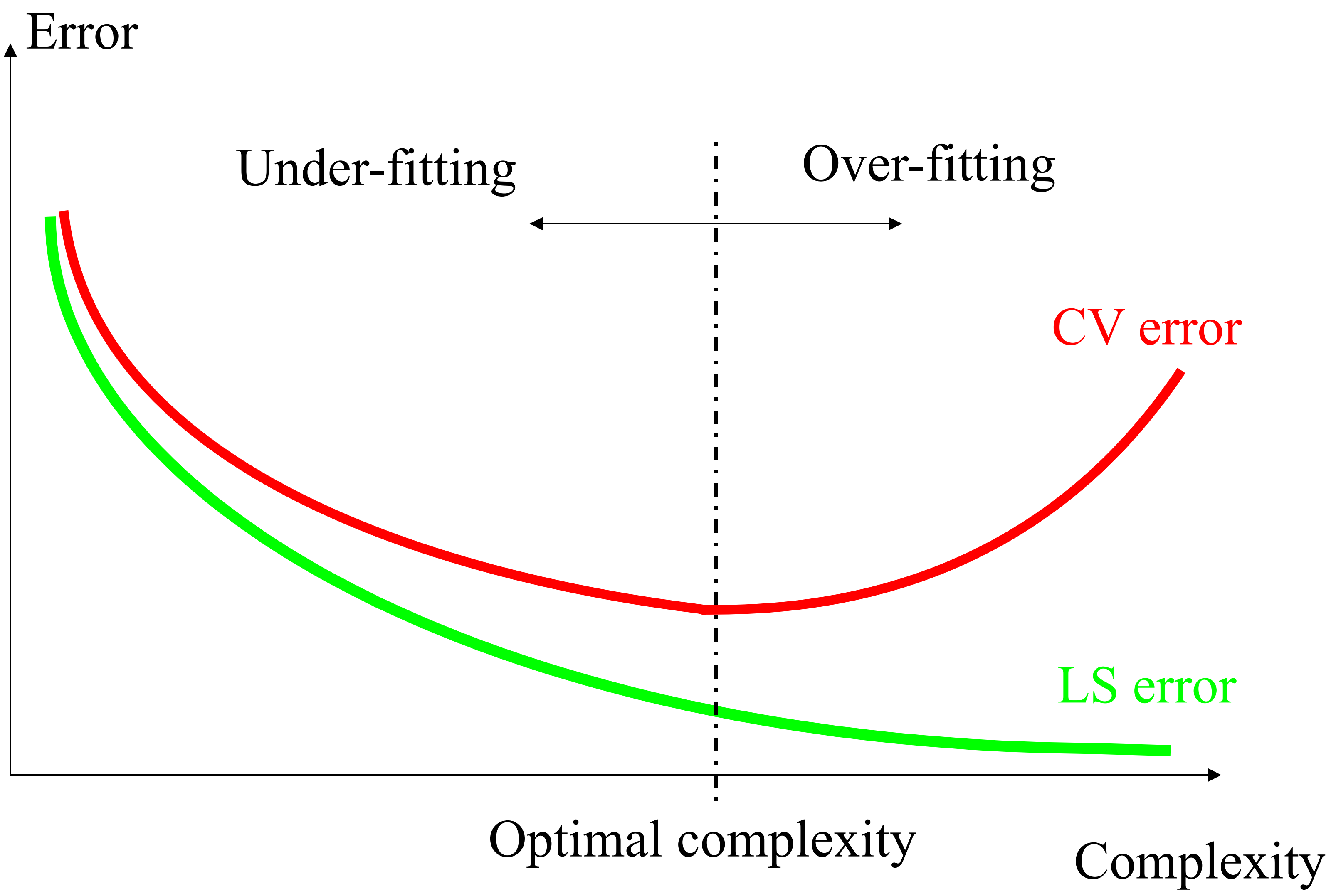


- For each subset:
  - learn the model on the objects that are not in the subset
  - compute the error rate on the points in the subset
- Report the mean error rate over the *k* subsets
- When *k*=the number of objects ⇒ leave-one-out cross validation

Which kind of Cross Validation?

- Test set:
  - Cheap but waste data and unreliable when few data
- Leave-one-out:
  - Doesn't waste data but expensive
- k-fold cross validation:
  - compromise between the two
- Rule of thumb:
  - a lot of data (>1000): test set validation
  - small data (100-1000): 10-fold CV
  - very small data(<100): leave-one-out CV

CV-based complexity control



Complexity

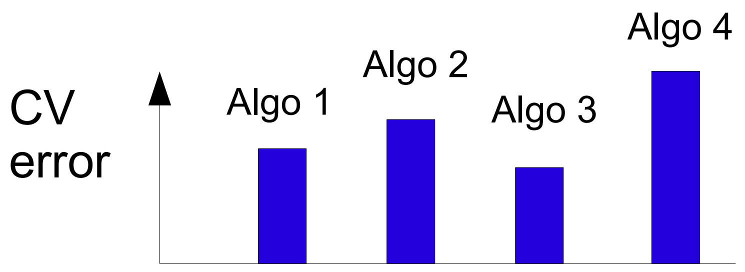
- Controlling complexity is called regularization or smooting
- Complexity can be controlled in several ways
  - The size of the hypothesis space: number of candidate models, range of the parameters...
  - The performance criterion: learning set performance versus parameter range, eg. minimizes

$$\mathrm{Err}(LS)+\lambda\, C(\mathrm{model})$$

  - The optimization algorithms: number of iterations, nature of the optimization problem (one global optimum versus several local optima)...

CV-based algorithm choice

- Step 1: compute 10-fold (or test set or LOO) CV error for different algorithms



- Step 2: whichever algorithm gave best CV score: learn a new model with all data, and that's the predictive model
- What is the expected error rate of this model?

Warning: Intensive use of CV can overfit

- If you compare many (complex) models, the probability that you will find a good one by chance on your data increases
- Solution:
  - Hold out an additional test set before starting the analysis (or, better, generate this data afterwards)
  - Use it to estimate the performance of your final model

(For small datasets: use two stages of 10-fold CV)

A note on performance measures

	True class	Model 1	Model 2
1	Negative	Positive	Negative
2	Negative	Negative	Negative
3	Negative	Positive	Positive
4	Negative	Positive	Negative
5	Negative	Negative	Negative
6	Negative	Negative	Negative
7	Negative	Negative	Positive
8	Negative	Negative	Negative
9	Negative	Negative	Negative
10	Positive	Positive	Positive
11	Positive	Positive	Negative
12	Positive	Positive	Positive
13	Positive	Positive	Positive
14	Positive	Negative	Negative
15	Positive	Positive	Negative

- Which of these two models is the best?
- The choice of an error or quality measure is highly application dependent.

A note on performance measures

- The error rate is not the only way to assess a predictive model
- In binary classification, results can be summarized in a contingency table (aka confusion matrix)

	Predicted class		
Actual class	p	n	Total
p	**T**rue **P**ositive	**F**alse **N**egative	P
n	**F**alse **P**ositive	**T**rue **N**egative	N

- Various criterion

Error rate = $(FP+FN)/(N+P)$

Accuracy = $(TP+TN)/(N+P)$
= 1-Error rate

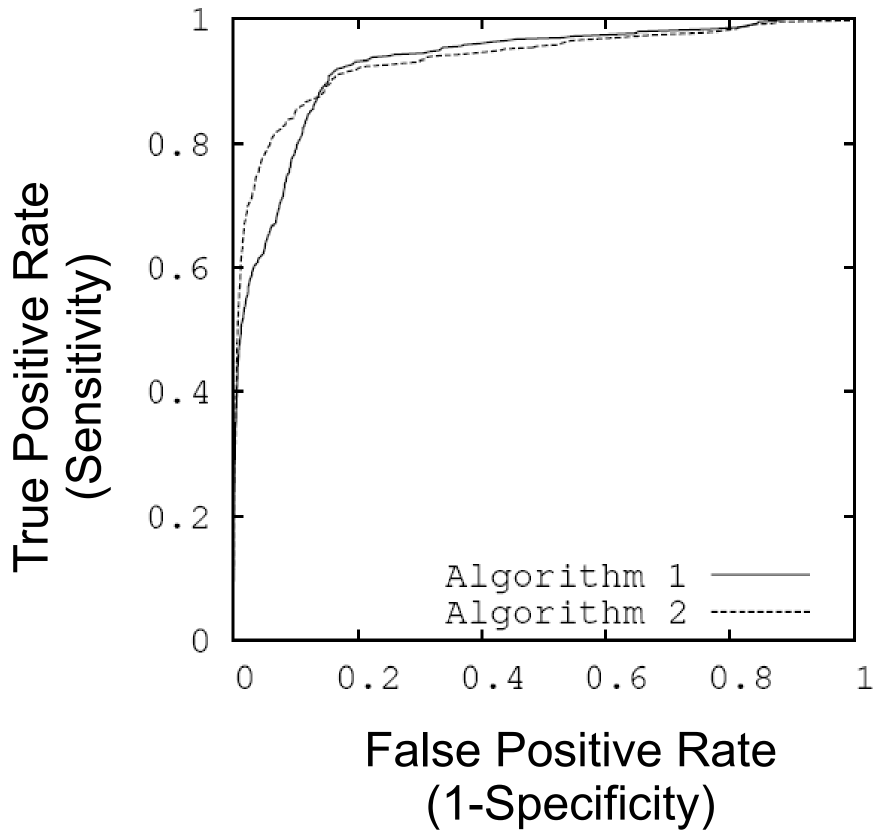
Sensitivity = TP/P (aka recall)

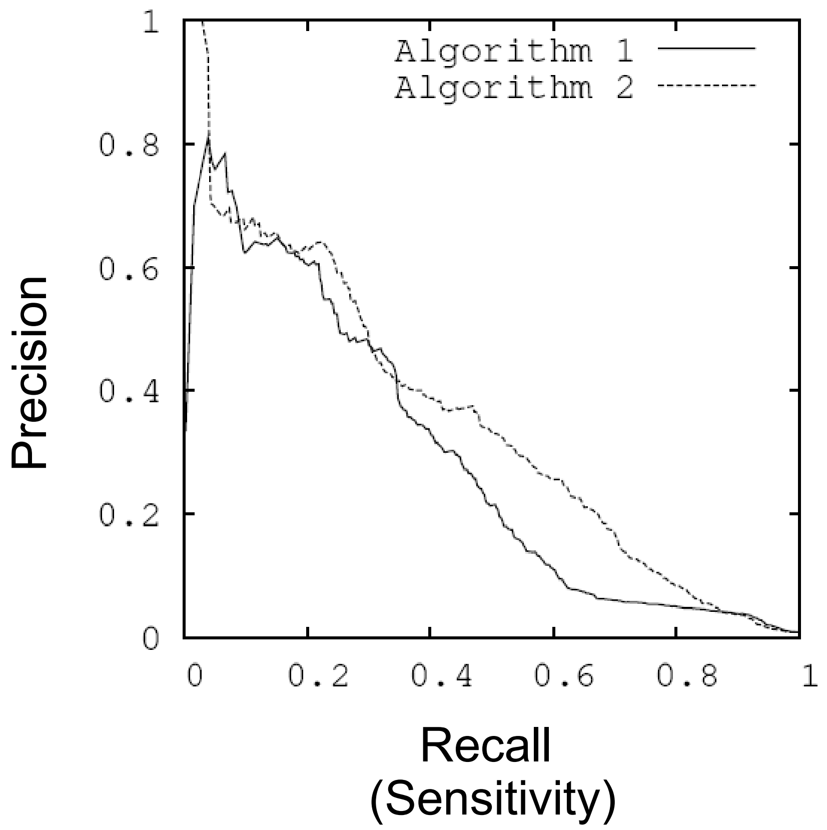
Specificity = $TN/(TN+FP)$

Precision = $TP/(TP+FP)$ (aka PPV)

ROC and Precision/recall curves

- Each point corresponds to a particular choice of the decision threshold





Outline

- Introduction
- Model selection, cross-validation, overfitting
- Some supervised learning algorithms
  - k-NN
  - Linear methods
  - Artificial neural networks
  - Support vector machines
  - Decision trees
  - Ensemble methods
- Beyond classification and regression

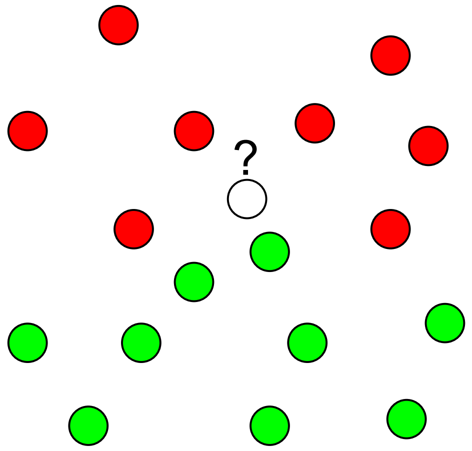
Comparison of learning algorithms

- Three main criteria:
  - Accuracy:
    - Measured by the generalization error (estimated by CV)
  - Efficiency:
    - Computing times and scalability for learning and testing
  - Interpretability:
    - Comprehension brought by the model about the input-output relationship
- Unfortunately, there is usually a tradeoff between these criteria

1-Nearest Neighbor (1-NN)

(prototype based method, instance based learning, non-parametric method)

- One of the simplest learning algorithm:
  - outputs as a prediction the output associated to the sample which is the closest to the test object



	M1	M2	Y
1	0.32	0.81	Healthy
2	0.15	0.38	Disease
3	0.39	0.34	Healthy
4	0.62	0.11	Disease
5	0.92	0.43	?

$$d(5,1)=\sqrt{(0.32-0.92)^2+(0.81-0.43)^2}=0.71$$

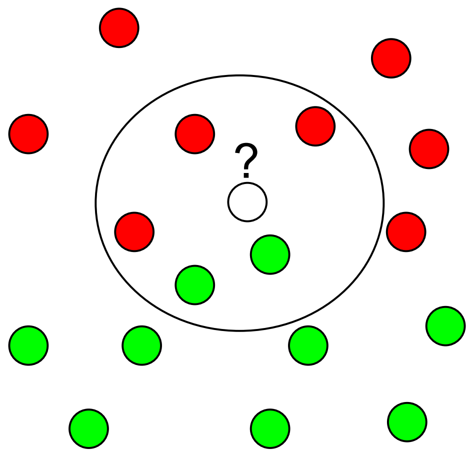
$$d(5,2)=\sqrt{(0.15-0.92)^2+(0.38-0.43)^2}=0.77$$

$$d(5,3)=\sqrt{(0.39-0.92)^2+(0.34-0.43)^2}=0.71$$

$$d(5,4)=\sqrt{(0.62-0.92)^2+(0.43-0.43)^2}=0.44$$

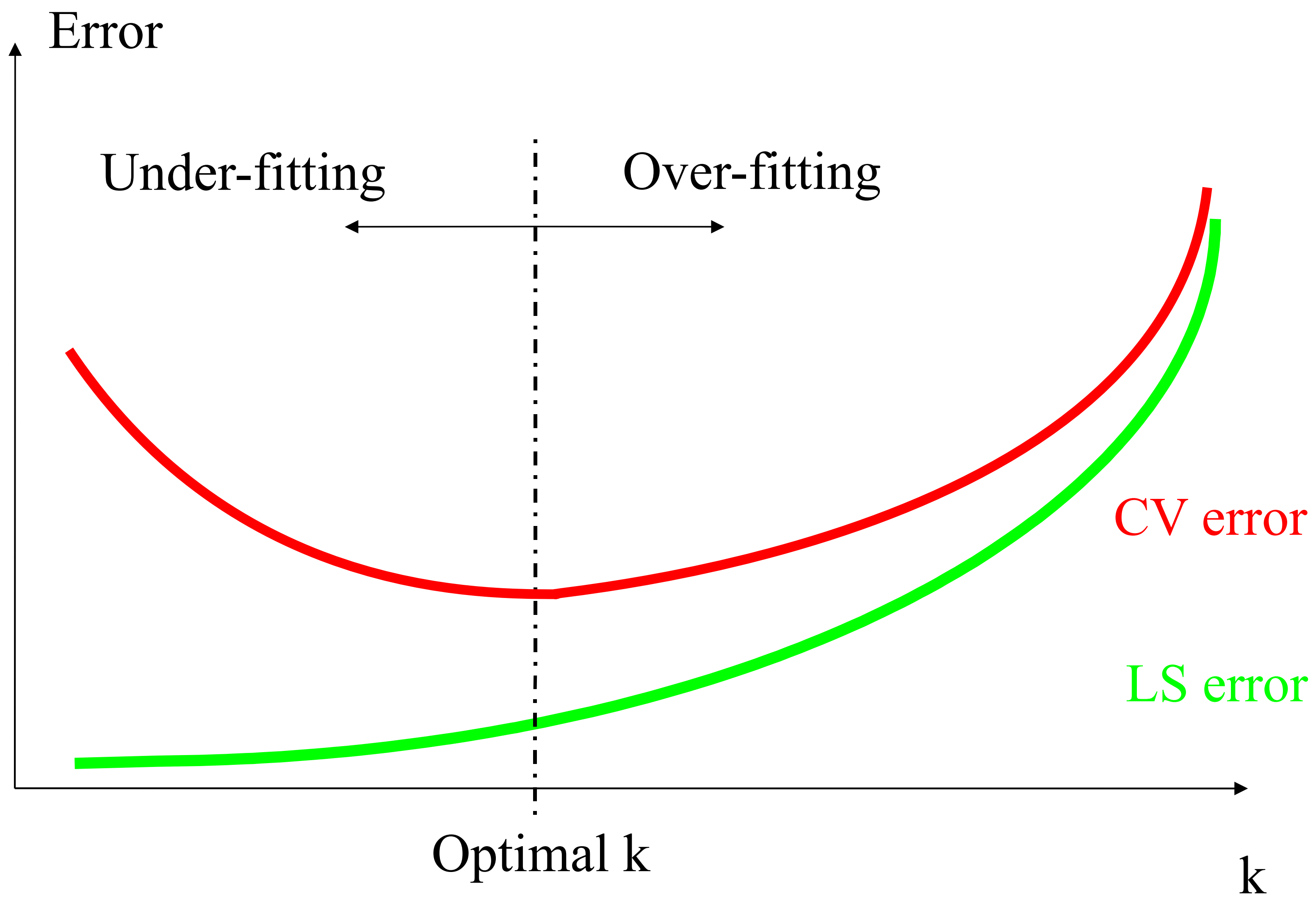
  - closest=usually of minimal Euclidian distance

Obvious extension: k-NN

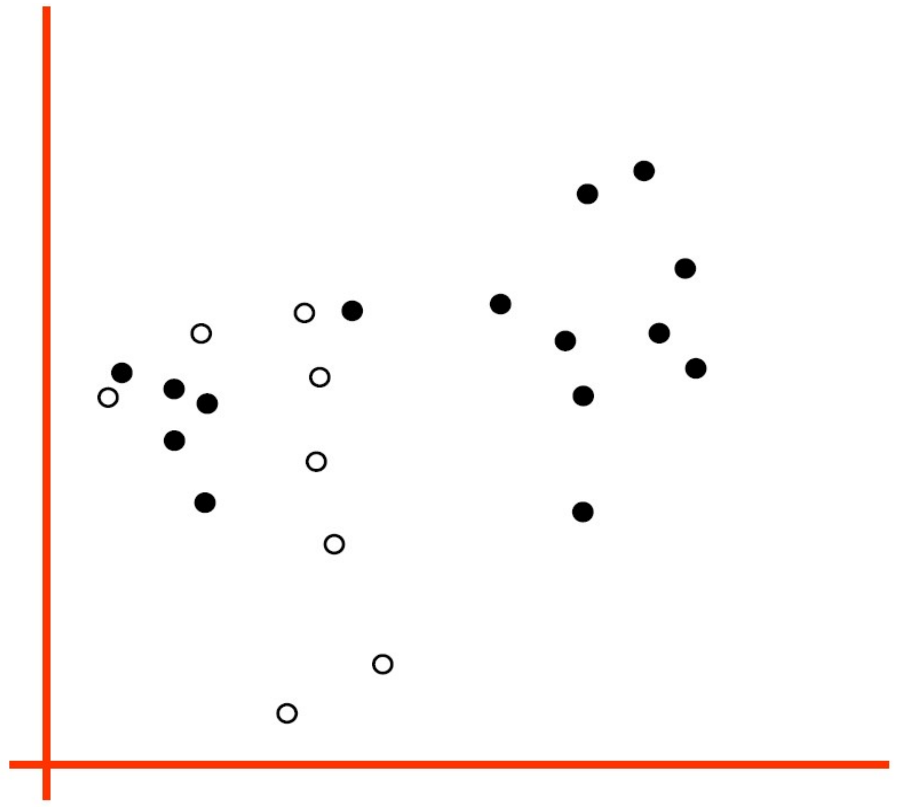


- Find the k nearest neighbors (instead of only the first one) with respect to Euclidian distance
- Output the most frequent class (classification) or the average outputs (regression) among the k neighbors.

Effect of k on the error



Small exercise



- In this classification problem with two inputs:
  - What it the resubstitution error (LS error) of 1-NN?
  - What is the LOO error of 1-NN?
  - What is the LOO error of 3-NN?
  - What is the LOO error of 22-NN?

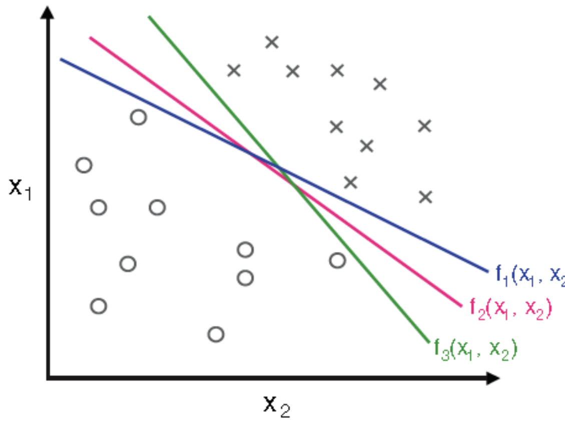
Andrew Moore

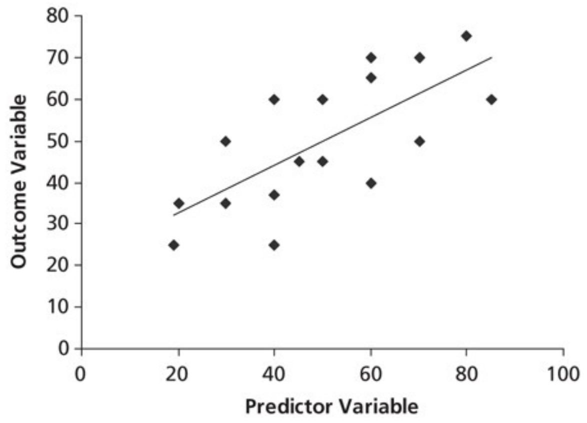
k-NN

- Advantages:
  - very simple
  - can be adapted to any data type by changing the distance measure
- Drawbacks:
  - choosing a good distance measure is a hard problem
  - very sensitive to the presence of noisy variables
  - slow for testing

Linear methods

- Find a model which is a linear combinations of the inputs
  - Regression: $y = w_0 + w_1 x_1 + w_2 x_2 + ... + w_n w_n$
  - Classification: $y = c_1$ if $w_0 + w_1 x_1 + ... + w_n x_n > 0$, c_2 otherwise





- Several methods exist to find coefficients w_0, w_1... corresponding to different objective functions, optimization algorithms, eg.:
  - Regression: least-square regression, ridge regression, partial least square, support vector regression, LASSO...
  - Classification: linear discriminant analysis, PLS-discriminant analysis, support vector machines...

Example: ridge regression

- Find ***w*** that minimizes ($\lambda > 0$):

$$\sum_i (y_i - w x_i)^2 + \lambda \|w\|^2$$

- From simple algebra, the solution is given by:

$$w^r = (X^T X + \lambda I)^{-1} X^T y$$

where X is the input matrix and y is the output vector

- λ regulates complexity (and avoids problems related to the singularity of $X^T X$)

Example: perceptron

- Find ***w*** that minimizes:

$$\sum_i (y_i - w x_i)^2$$

using gradient descent: given a training example (x, y)

$$\delta \leftarrow y - w^T x$$
$$\forall j \, w_j \leftarrow w_j + \eta \delta x_j$$

- **Online** algorithm, ie. that treats every example in turn (vs **Batch** algorithm that treats all examples at once)
- Complexity is regulated by the learning rate η and the number of iterations
- Can be adapted to classification

Linear methods

- Advantages:
  - simple
  - there exist fast and scalable variants
  - provide interpretable models through variable weights (magnitude and sign)
- Drawbacks:
  - often not as accurate as other (non-linear) methods

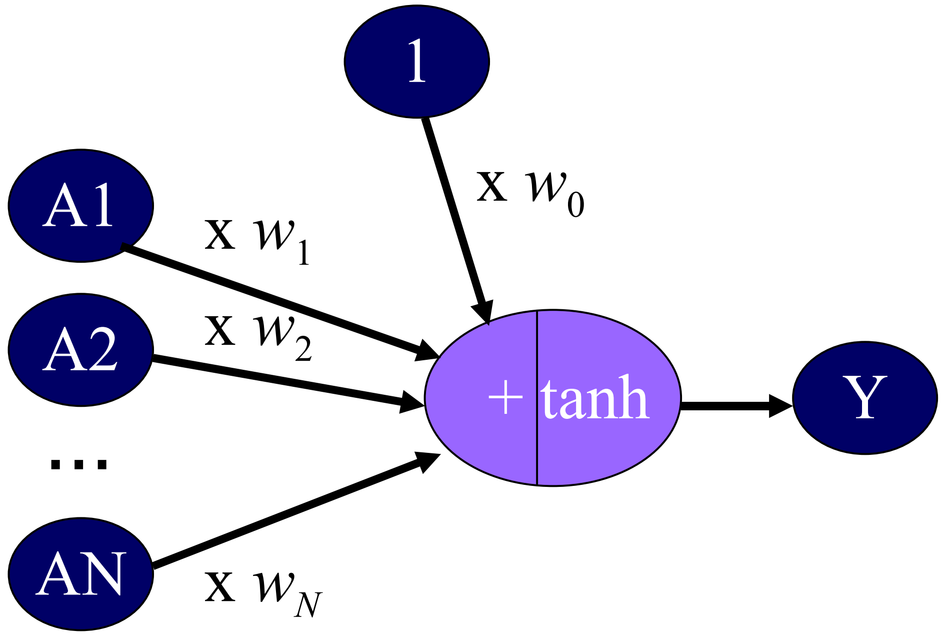
Non-linear extensions

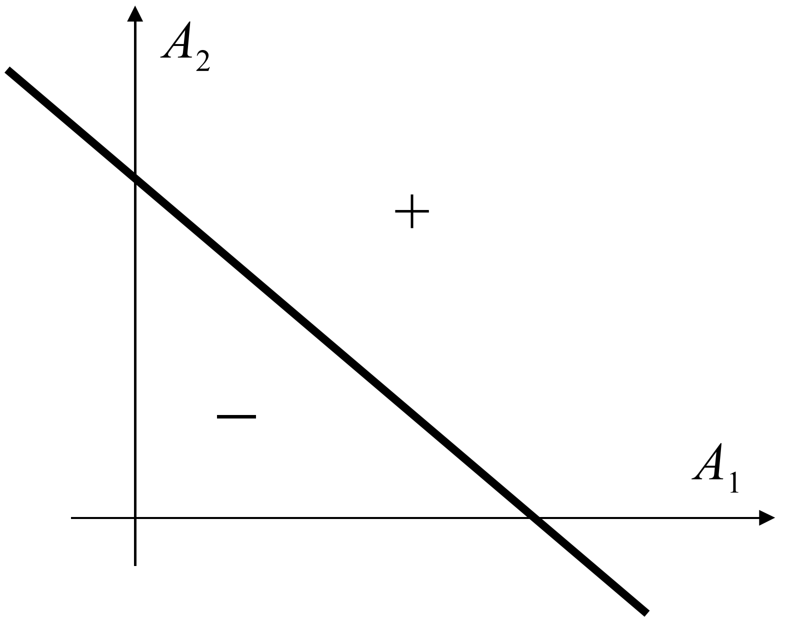
- Generalization of linear methods:
  - $y=w_0+w_1\phi_1(x)+w_2\phi_2(x_2)+...+w_n\phi_n(x)$
  - Any linear methods can be applied (but regularization becomes more important)
- Artificial neural networks (with a single hidden layer):
  - $y=g(\sum_j W_j g(\sum_i w_{i,j} x_i))$
    where g is a non linear function (eg. sigmoid)
  - (a non linear function of a linear combination of non linear functions of linear combinations of inputs)
- Kernel methods:
  - $y=\sum_i w_i \phi_i(x) \quad \Leftrightarrow \quad y=\sum_j \alpha_j k(x_j, x)$
    where $k(x,x')=\langle\phi(x),\phi(x')\rangle$ is the dot-product in the feature space and *j* indexes training examples

Artificial neural networks

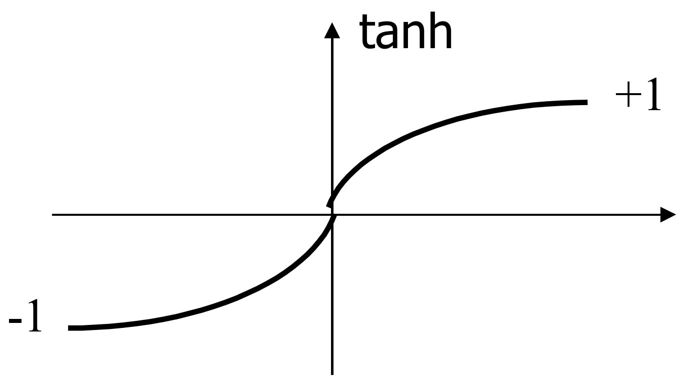
- Supervised learning method initially inspired by the behaviour of the human brain
- Consists of the inter-connection of several small units
- Essentially numerical but can handle classification and discrete inputs with appropriate coding
- Introduced in the late 50s, very popular in the 90s

Hypothesis space: a single neuron



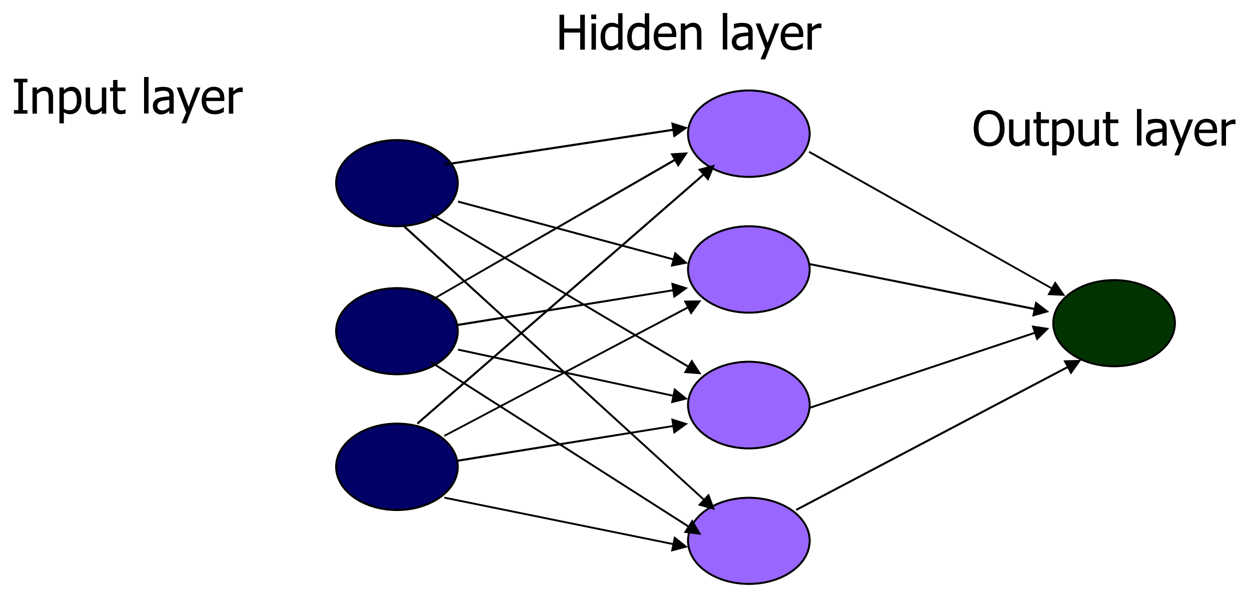


$Y=\tanh(w_1*A_1+w_2*A_2+\ldots+w_N*A_N+w_0)$



Hypothesis space: Multi-layers Perceptron

- Inter-connection of several neurons (just like in the human brain)



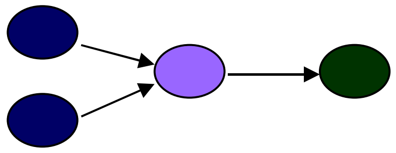
- With a sufficient number of neurons and a sufficient number of layers, a neural network can model any function of the inputs.

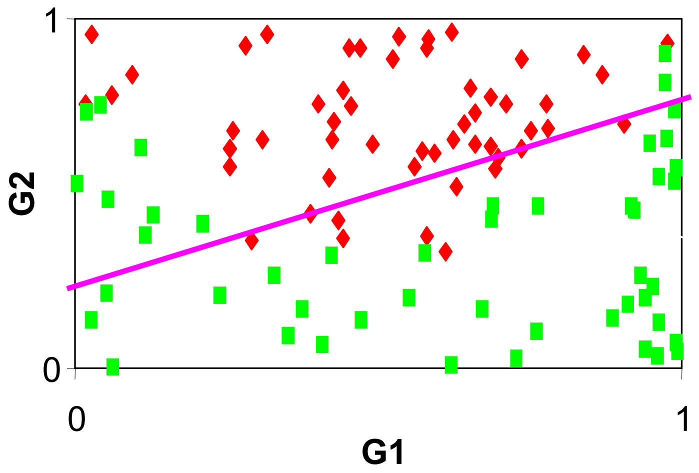
Learning

- Choose a structure
- Tune the value of the parameters (connections between neurons) so as to minimize the learning sample error.
  - Non-linear optimization by the back-propagation algorithm. In practice, quite slow.
- Repeat for different structures
- Select the structure that minimizes CV error

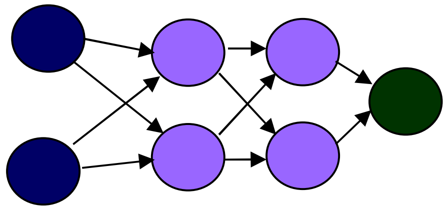
Illustrative example

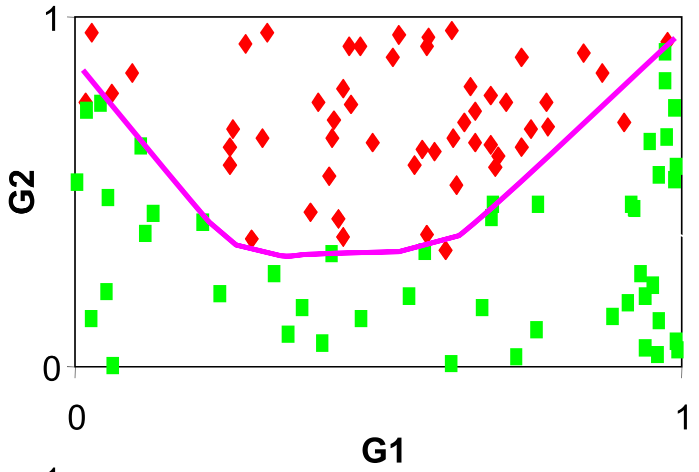
1 neuron



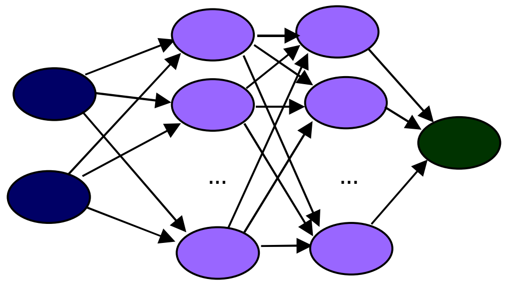


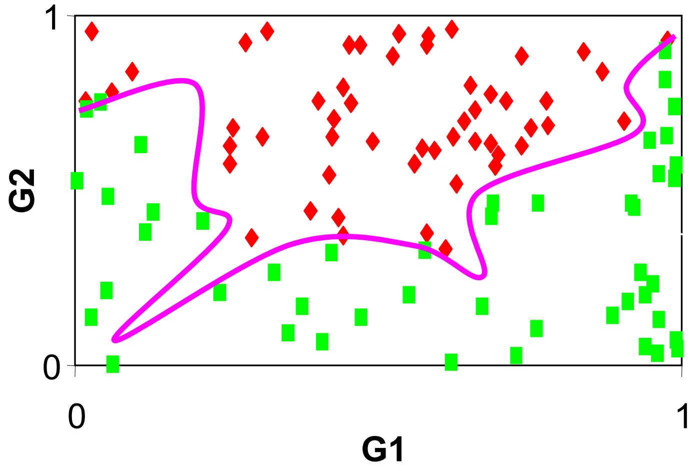
2 +2 neurons





10 +10 neurons





Artificial neural networks

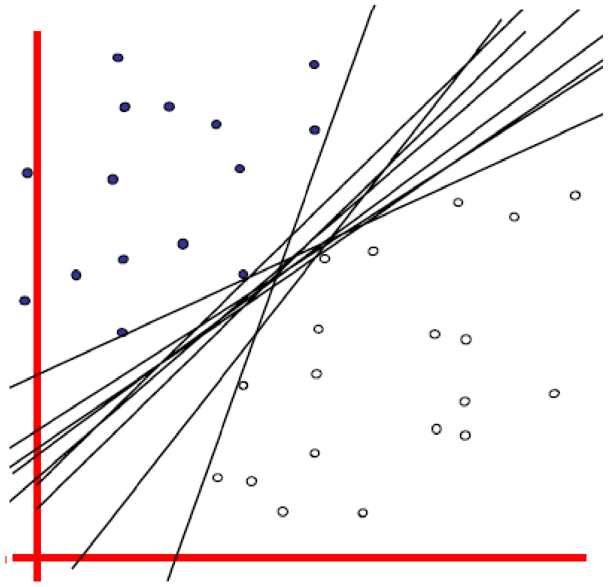
- Advantages:
  - Universal approximators
  - May be very accurate (if the method is well used)
- Drawbacks:
  - The learning phase may be very slow
  - Black-box models, very difficult to interprete
  - Scalability

Support vector machines

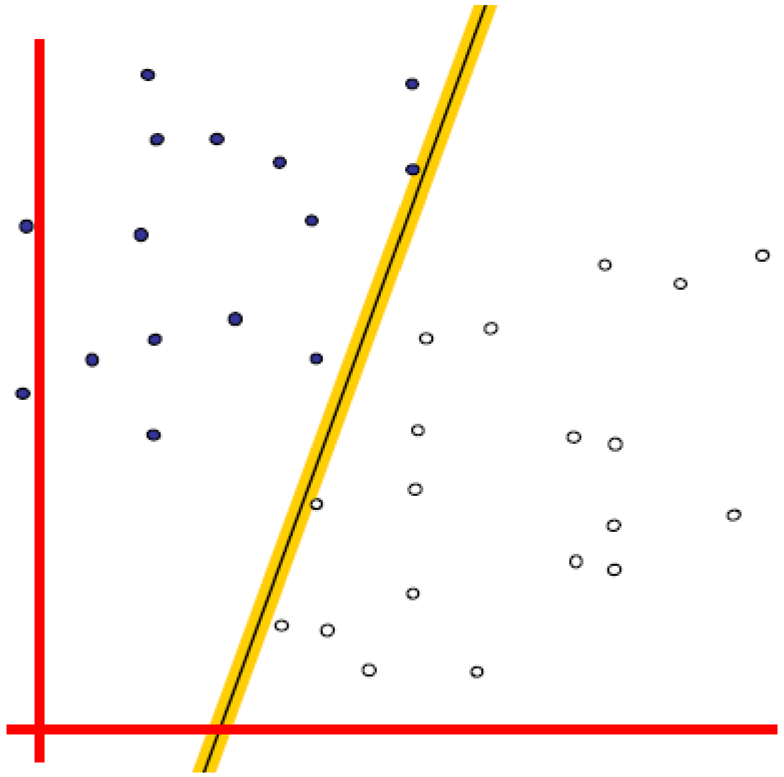
- Recent (mid-90's) and very successful method
- Based on two smart ideas:
  - large margin classifier
  - kernelized input space

Linear classifier

- Where would you place a linear classifier?

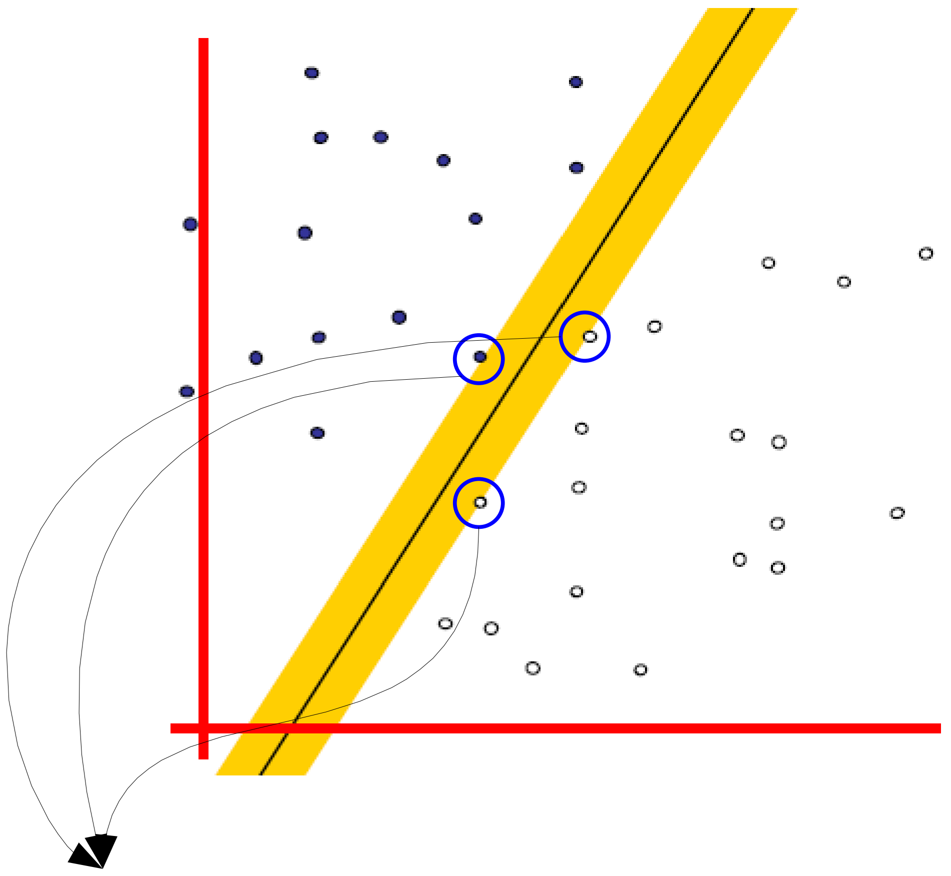


Margin of a linear classifier



- The margin = the width that the boundary could be increased by before hitting a datapoint.

Maximum-margin linear classifier



Support vectors: the samples the closest to the hyperplane

- The linear classifier with the maximum margin (= Linear SVM)
- Intuitively, this feels safest
- Works very well in practice

Mathematically

- Linearly separable case: amount at solving the following quadratic programming optimization problem:

  minimizes $\frac{1}{2}\|w\|^2$

  subject to $y_i(w^T x_i - b) \geqslant 1, \forall i = 1, \ldots, N$

- Decision function:
  - $y = +1$ if $w^T x - b > 0$, $y = -1$ otherwise

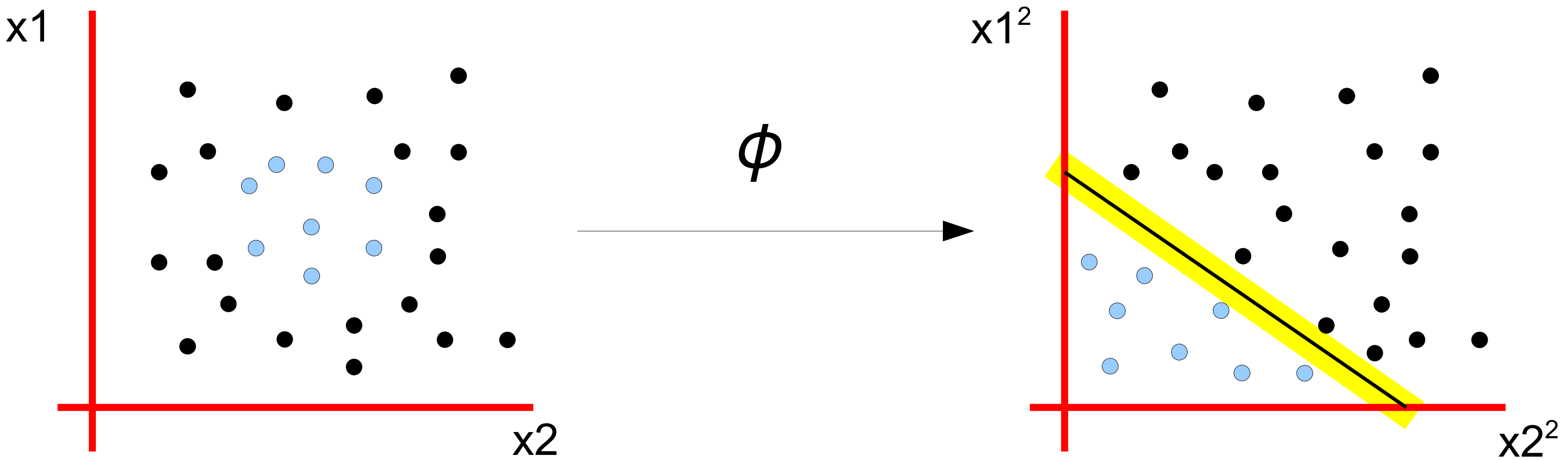
- Non linearly separable case:

  minimizes $\frac{1}{2}\|w\|^2 + C \sum_i \xi_i$

  subject to $y_i(w^T x_i - b) \geqslant 1 - \xi_i, \forall i = 1, \ldots, N$

Non-linear boundary

– What about this problem?



- Solution:
  - map the data into a new feature space where the boundary is linear
  - Find the maximum margin model in this new space

The kernel trick

- Intuitively:
  - You don't need to compute explicitly the mapping φ
  - All you need is a (special) similarity measure between objects (like for the kNN)
  - This similarity measure is called a kernel
- Mathematically:
  - The maximum-margin classifier in some feature space can be written only in terms of dot-products in that feature space:

$$k(x,x') = <\phi(x), \phi(x')>$$

Mathematically

- Primal form of the optimization problem:

  minimizes $\frac{1}{2}\|w\|^2$

  subject to $y_i(\langle w, \phi(x_i)\rangle - b) \geqslant 1, \forall i = 1, \ldots, N$

- Dual form:

  minimizes $\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \phi(x_i), \phi(x_j)\rangle$

  subject to $\alpha_i \geqslant 0$ and $\sum_i \alpha_i y_i = 0$

$(w = \sum_i \alpha_i y_i \phi(x_i))$

- Decision function:
  - $y = +1$ if $\langle w, x\rangle = \sum_i \alpha_i y_i \langle \phi(x_i), \phi(x)\rangle = \sum_i \alpha_i y_i k(x_i, x) > 0$
  - $y = -1$ otherwise

Support vector machines

	G1	G2	Y
1	0.21	0.64	C1
2	0.57	0.26	C2
3	0.21	0.68	C2
4	0.69	0.52	C1
5	0.83	0.96	C1
6	0.48	-0.52	C2

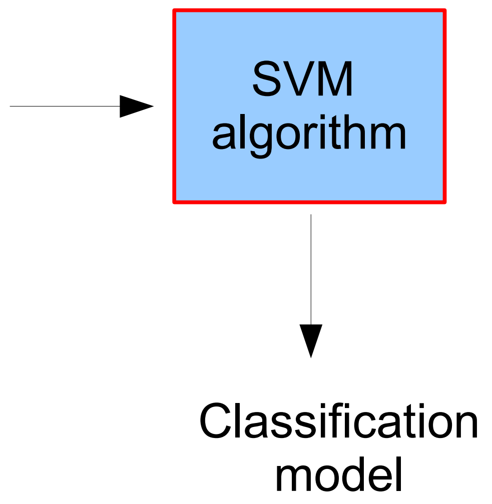
	G1	Y
1	ACGCTCTATAG	C1
2	ACTCGCTTAGA	C2
3	GTCTCTGAGAG	C2
4	CGCTAGCGTCG	C1
5	CGATCAGCAGC	C1
6	GCTCGCGCTCG	C2

kernel matrix

	1	2	3	4	5	6
1	1	0.14	0.96	0.17	0.01	0.24
2	0.14	1	0.02	0.17	0.22	0.67
3	0.96	0.02	1	0.15	0.27	0.07
4	0.17	0.7	0.15	1	0.37	0.55
5	0.01	0.22	0.27	0.37	1	-0.25
6	0.24	0.67	0.07	0.55	-0.25	1

Class labels

	Y
1	C1
2	C2
3	C2
4	C1
5	C1
6	C2



Examples of kernels

- Linear kernel:

  $k(x,x') = <x,x'>$

- Polynomial kernel

  $k(x,x')=(<x,x'>+1)^d$

(main parameter: d, the maximum degree)

- Radial basis function kernel:

  $k(x,x')=\exp(-\|x-x'\|^2/(2\sigma^2))$

(main parameter: σ, the spread of the distribution)

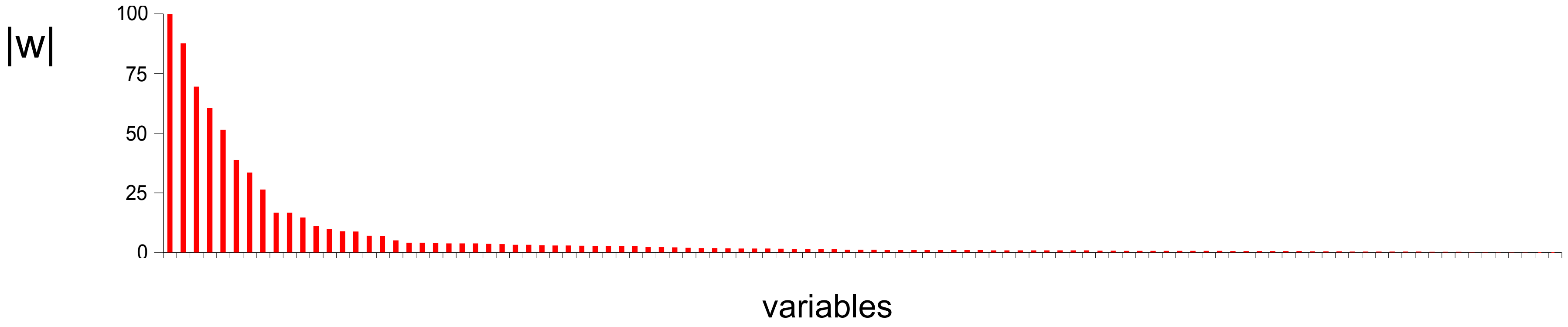
- + many kernels that have been defined for structured data types (eg. texts, graphs, trees, images)

Feature ranking with linear kernel

- With a linear kernel, the model looks like:

$$h(x1,x2,\ldots,xK)=\begin{cases} C1 & \text{if } w0+w1*x1+w2*x2+\ldots+wK*xK>0 \\ C2 & \text{otherwise} \end{cases}$$

- Most important variables are those corresponding to large $|wi|$



SVM parameters

- Mainly two sets of parameters in SVM:
  - Optimization algorithm's parameters:
    - Control the number of training errors versus the margin (when the learning sample is not linearly separable)
  - Kernel's parameters:
    - choice of particular kernel
    - given this choice, usually one complexity parameter
    - eg, the degree of the polynomial kernel
- Again, these parameters can be determined by cross-validation

Support vector machines

- Advantages:
  - State-of-the-art accuracy on many problems
  - Can handle any data types by changing the kernel (many applications on sequences, texts, graphs...)
- Drawbacks:
  - Tuning the parameter is very crucial to get good results and somewhat tricky
  - Black-box models, not easy to interprete

A note on kernel methods

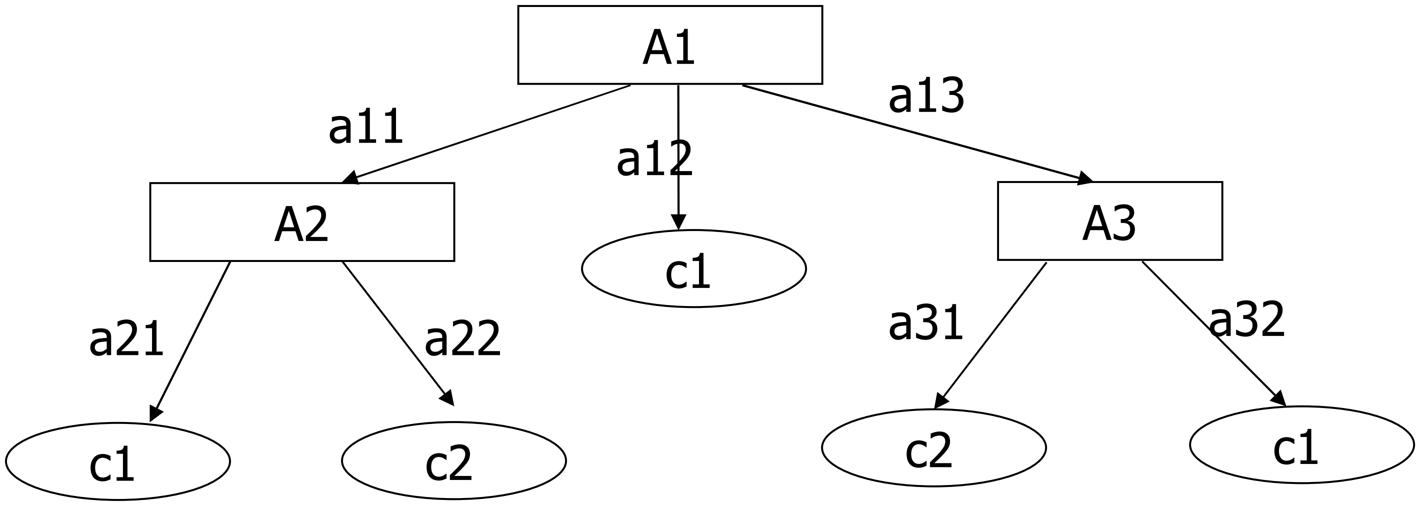
- The kernel trick can be applied to any (learning) algorithm whose solution can be expressed in terms of dot-products in the original input space
  - It makes a non-linear algorithm from a linear one
  - Can work in a very highly dimensional space (even infinite) without requiring to explicitly compute the features
  - Decouple the representation stage from the learning stage. The same learning machine can be applied to a large range of problems
- Examples: ridge regression, perceptron, PCA, k-means...

Decision (classification) trees

- A learning algorithm that can handle:
  - Classification problems (binary or multi-valued)
  - Attributes may be discrete (binary or multi-valued) or continuous.
- Classification trees were invented at least twice:
  - By statisticians: CART (Breiman et al.)
  - By the AI community: ID3, C4.5 (Quinlan et al.)

Decision trees

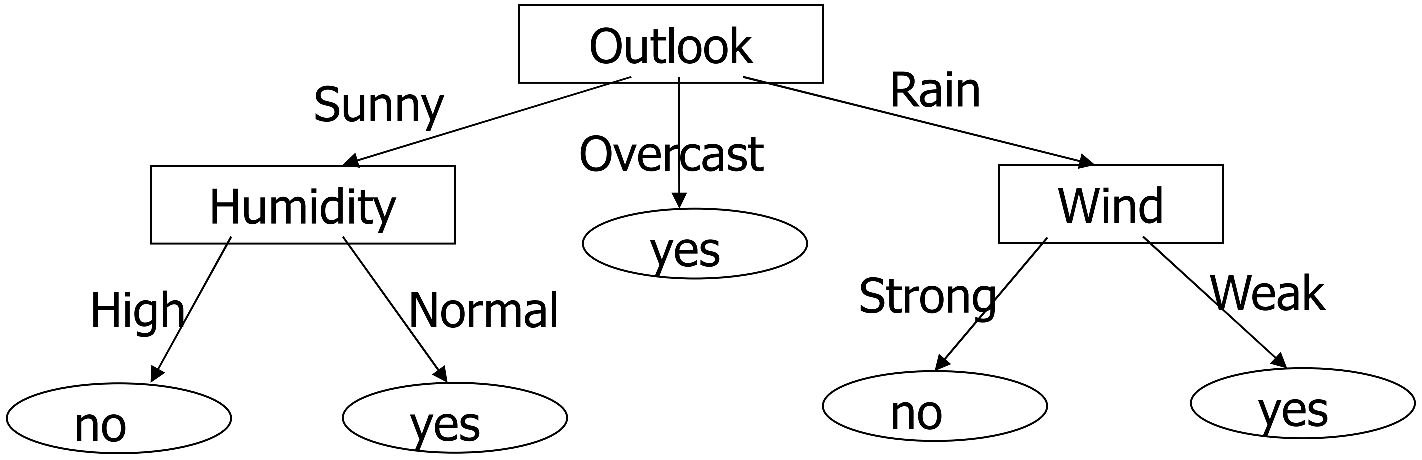
- A decision tree is a tree where:
  - Each interior node tests an attribute
  - Each branch corresponds to an attribute value
  - Each leaf node is labeled with a class



A simple database: playtennis

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	Normal	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	High	Strong	Yes
D8	Sunny	Mild	Normal	Weak	No
D9	Sunny	Hot	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Cool	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

A decision tree for playtennis

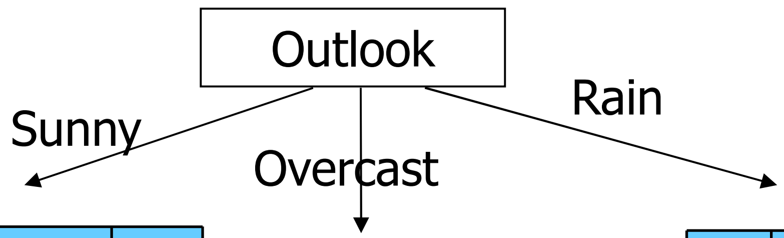


Should we play tennis on D15?

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D15	Sunny	Hot	High	Weak	?

Top-down induction of DTs

- Choose « best » attribute
- Split the learning sample
- Proceed recursively until each object is correctly classified



Day	Outlook	Temp.	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Hot	Normal	Wea	
D11	Sunny	Cool	Normal	Stro	

Day	Outlook	Temp.	Humidity	Wind	Play
D3	Overcast	Hot	High	Weak	Yes
D7	Overcast	Cool	High	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes

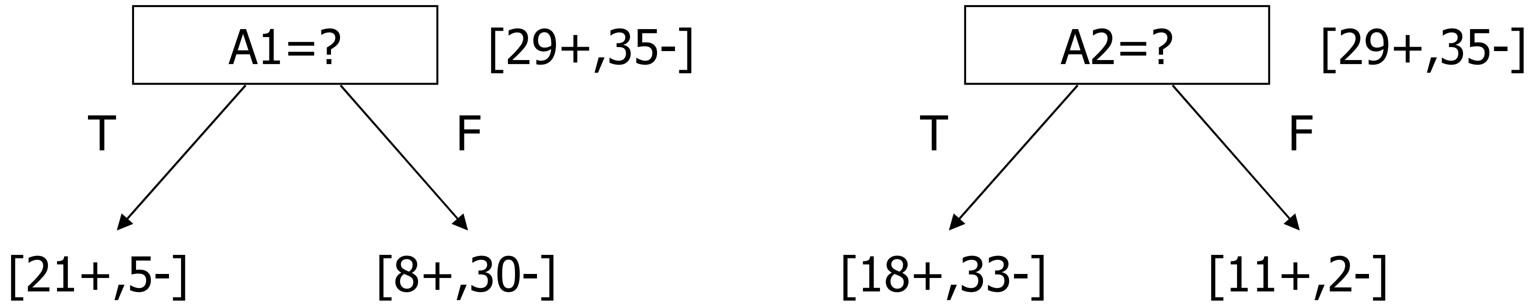
Day	Outlook	Temp.	Humidity	Wind	Play
D4	Rain	Mild	Normal	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
	Rain	Mild	Normal	Strong	Yes
	Rain	Mild	High	Strong	No

Top-down induction of DTs

Procedure learn_dt(learning sample, LS)

- If all objects from LS have the same class
  - Create a leaf with that class
- Else
  - Find the « best » splitting attribute A
  - Create a test node for this attribute
  - For each value a of A
    - Build $LS_a = \{o \in LS \mid A(o) \text{ is } a\}$
    - Use Learn_dt(LS_a) to grow a subtree from LS_a.

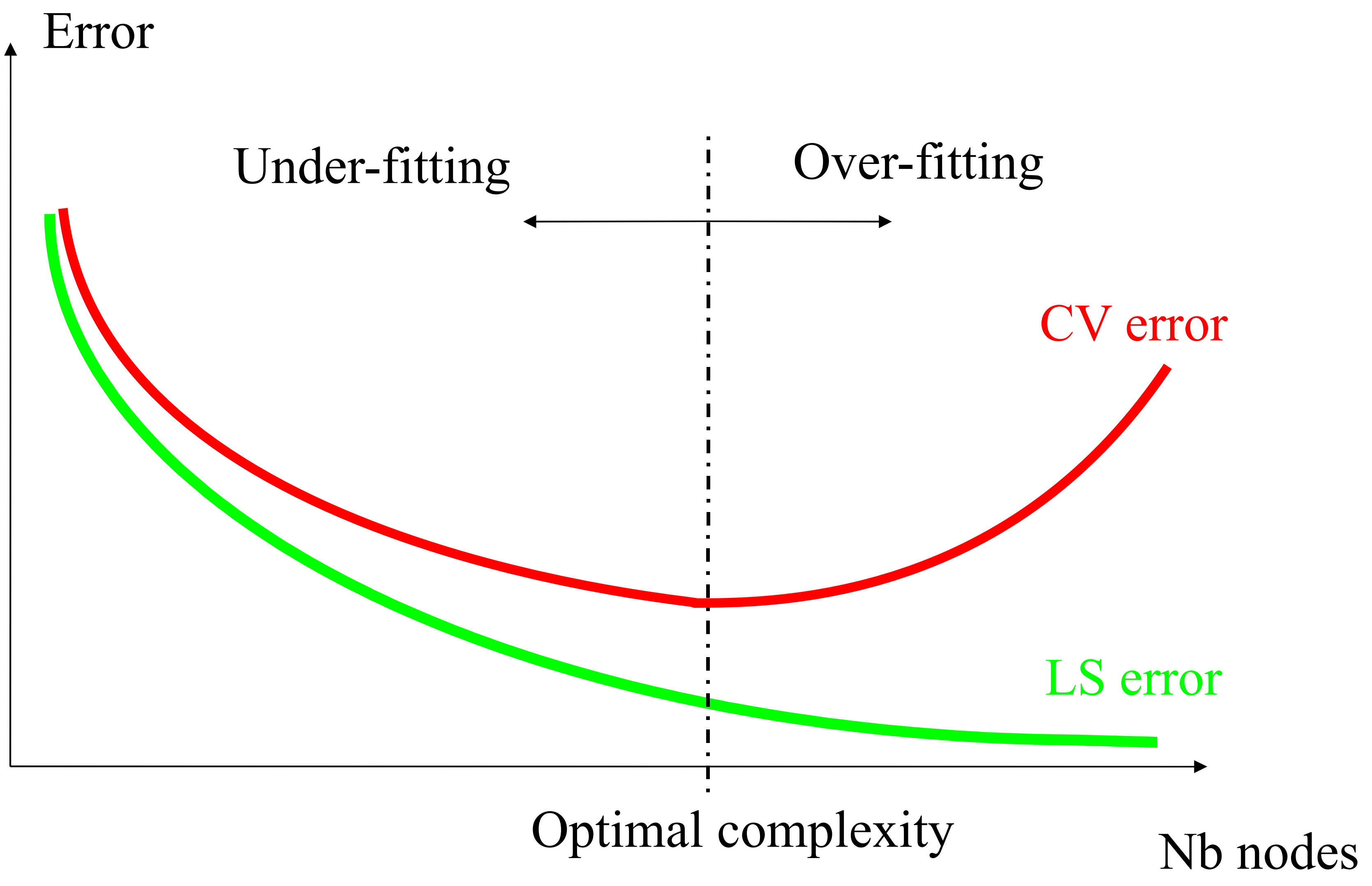
Which attribute is best ?



- A “score” measure is defined to evaluate splits
- This score should favor class separation at each step (to shorten the tree depth)
- Common score measures are based on information theory

$$I(LS,A)=H(LS)-\frac{|LS_{left}|}{|LS|}H(LS_{left})-\frac{|LS_{right}|}{|LS|}H(LS_{right})$$

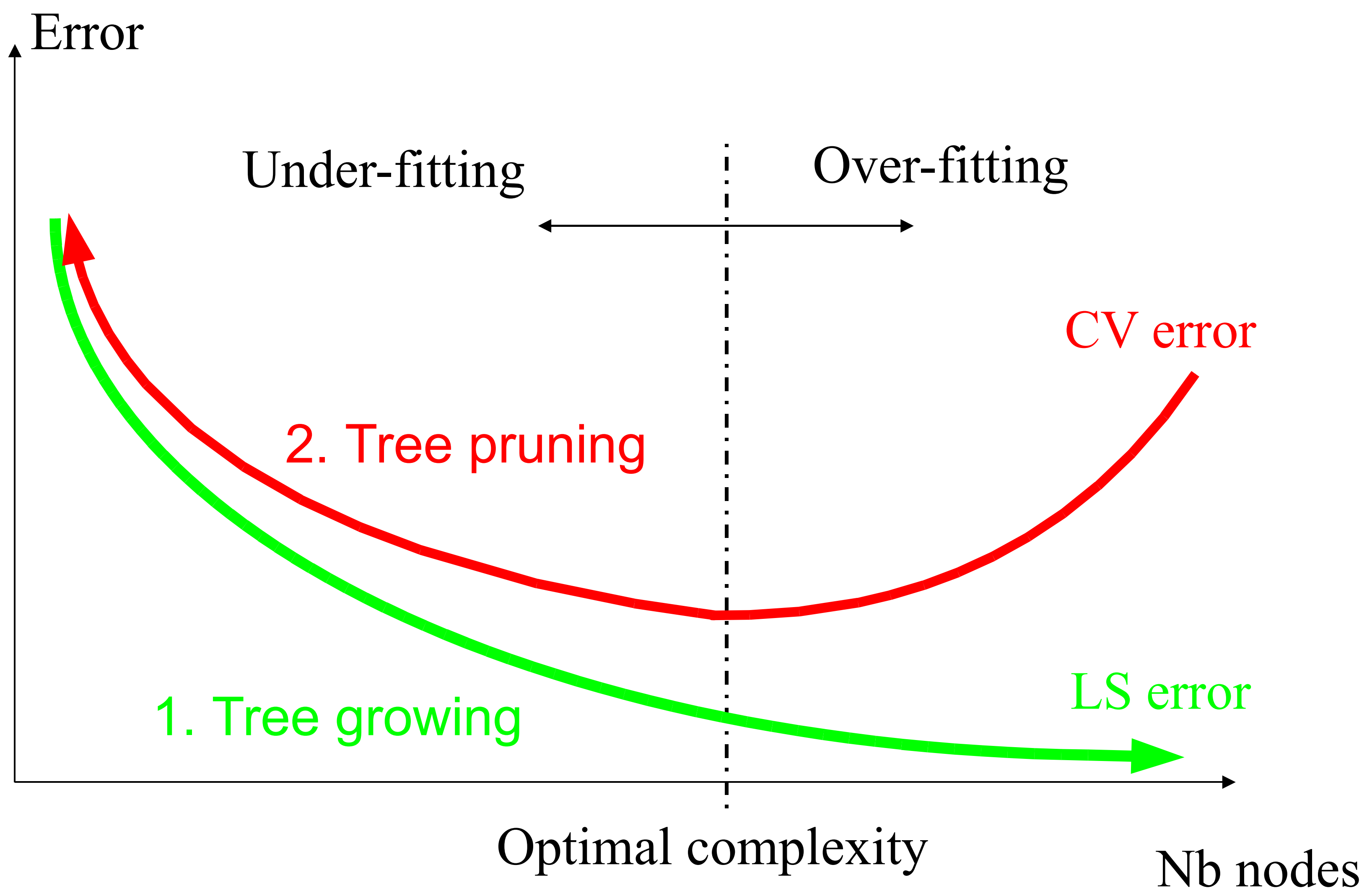
Effect of number of nodes on error



How can we avoid overfitting?

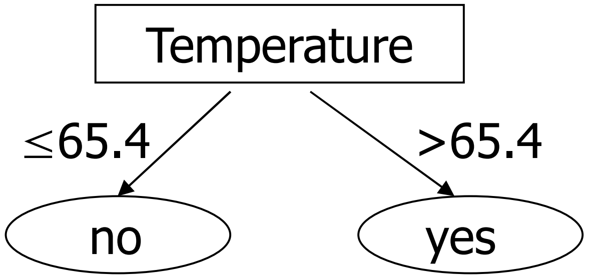
- Pre-pruning: stop growing the tree earlier, before it reaches the point where it perfectly classifies the learning sample
- Post-pruning: allow the tree to overfit and then post-prune the tree
- Ensemble methods (later)

Post-pruning



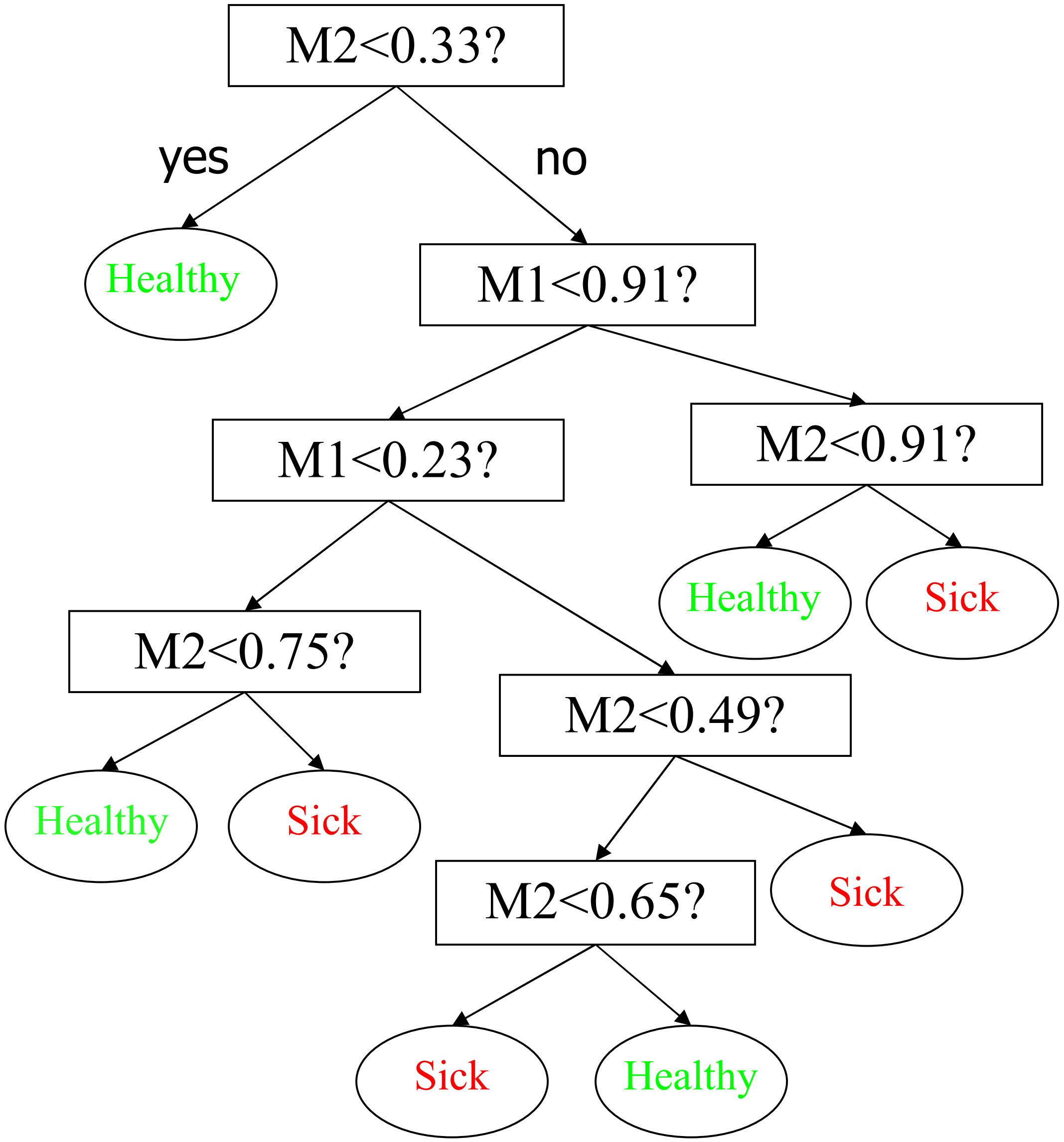
Numerical variables

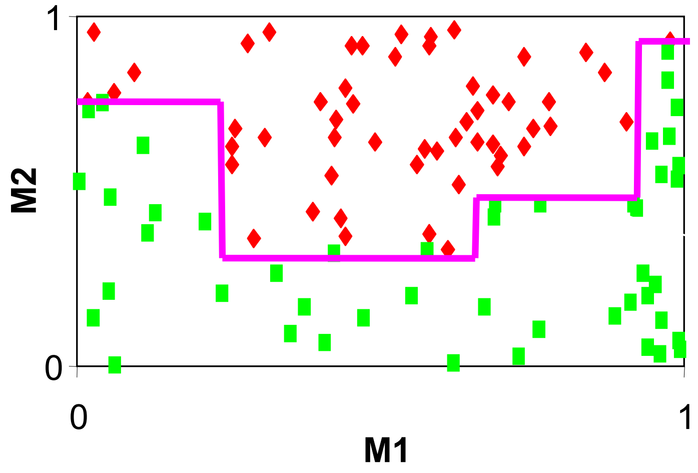
- Example: temperature as a number instead of a discrete value
- Two solutions:
  - Pre-discretize: Cold if Temperature<70, Mild between 70 and 75, Hot if Temperature>75
  - Discretize during tree growing:



optimization of the threshold to maximize the score

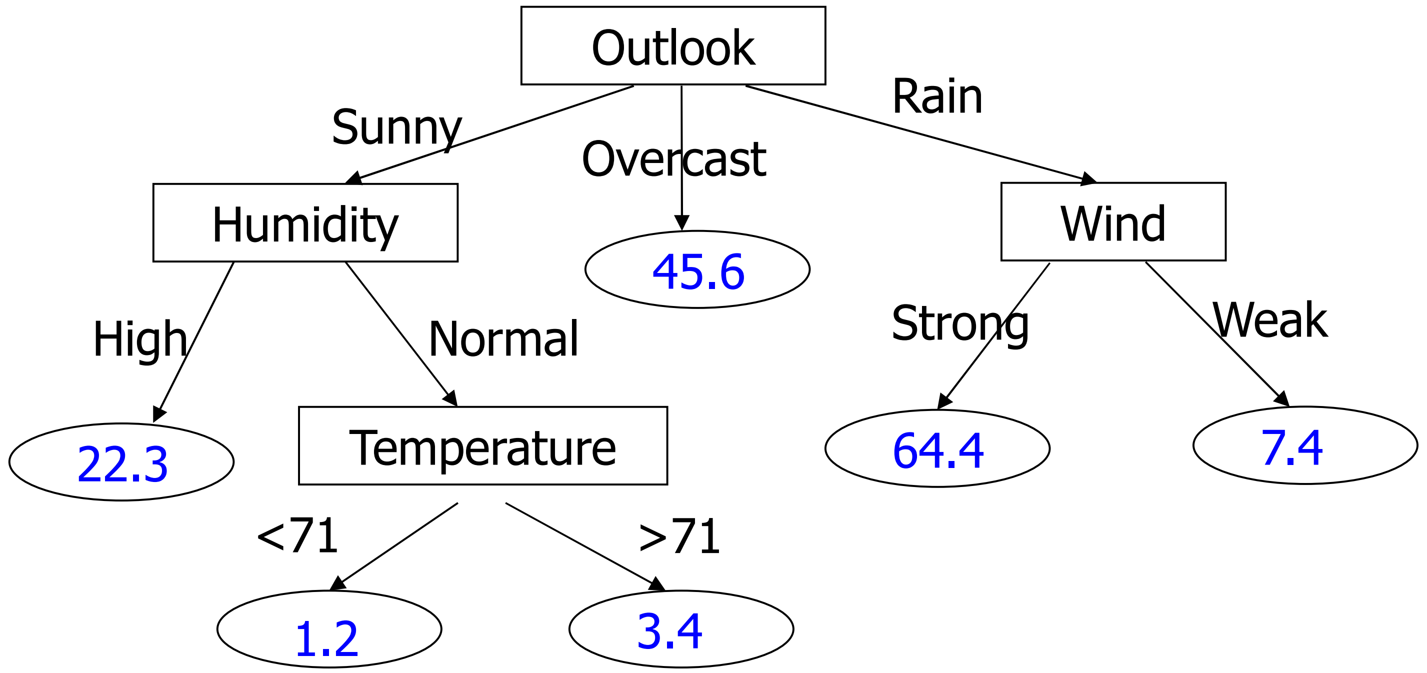
Illustrative example





Regression trees

Trees for regression problems: exactly the same model but with a number in each leaf instead of a class

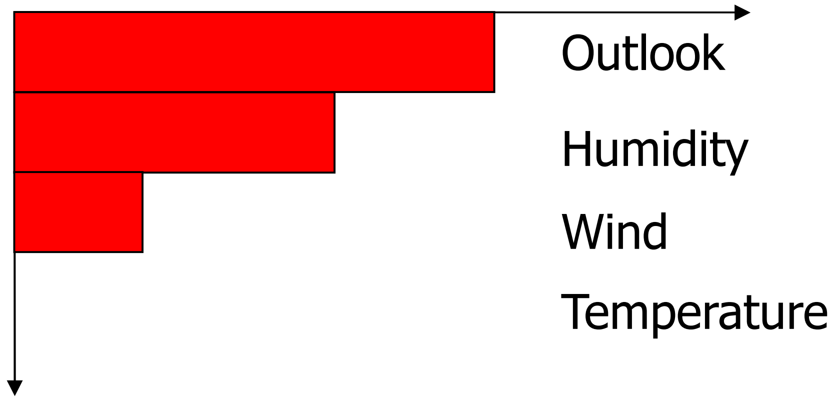


Interpretability and attribute selection

- Interpretability
  - Intrinsically, a decision tree is highly interpretable
  - A tree may be converted into a set of “if…then” rules.
- Attribute selection
  - If some attributes are not useful for classification, they will not be selected in the (pruned) tree
  - Of practical importance, if measuring the value of a variable is costly (e.g. medical diagnosis)
  - Decision trees are often used as a pre-processing for other learning algorithms that suffer more when there are irrelevant variables

Attribute importance

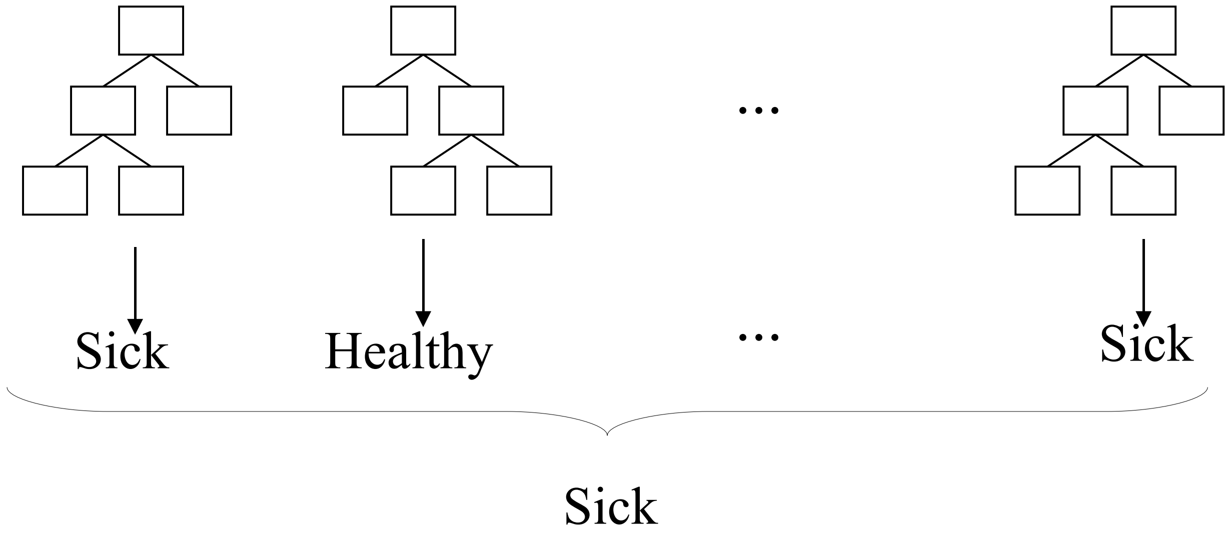
- In many applications, all variables do not contribute equally in predicting the output.
- We can evaluate variable importances with trees



Decision and regression trees

- Advantages:
  - very fast and scalable method (able to handle a very large number of inputs and objects)
  - provide directly interpretable models and give an idea of the relevance of attributes
- Drawbacks:
  - high variance (more on this later)
  - often not as accurate as other methods

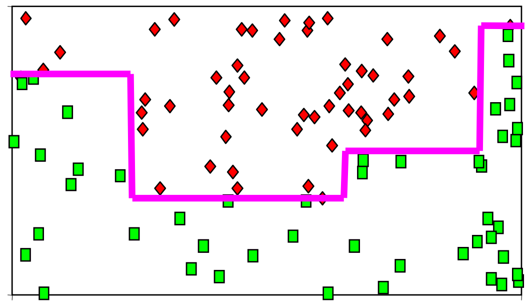
Ensemble methods

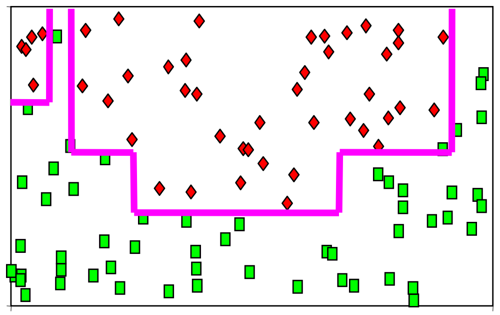


- Combine the predictions of several models built with a learning algorithm. Often improve very much accuracy.
- Often used in combination with decision trees for efficiency reasons
- Examples of algorithms: Bagging, Random Forests, Boosting...

Bagging: motivation

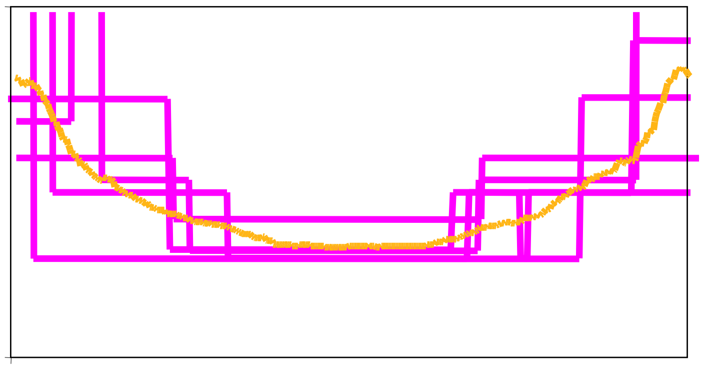
- Different learning samples yield different models, especially when the learning algorithm overfits the data





As there is only one optimal model, this *variance* is source of error

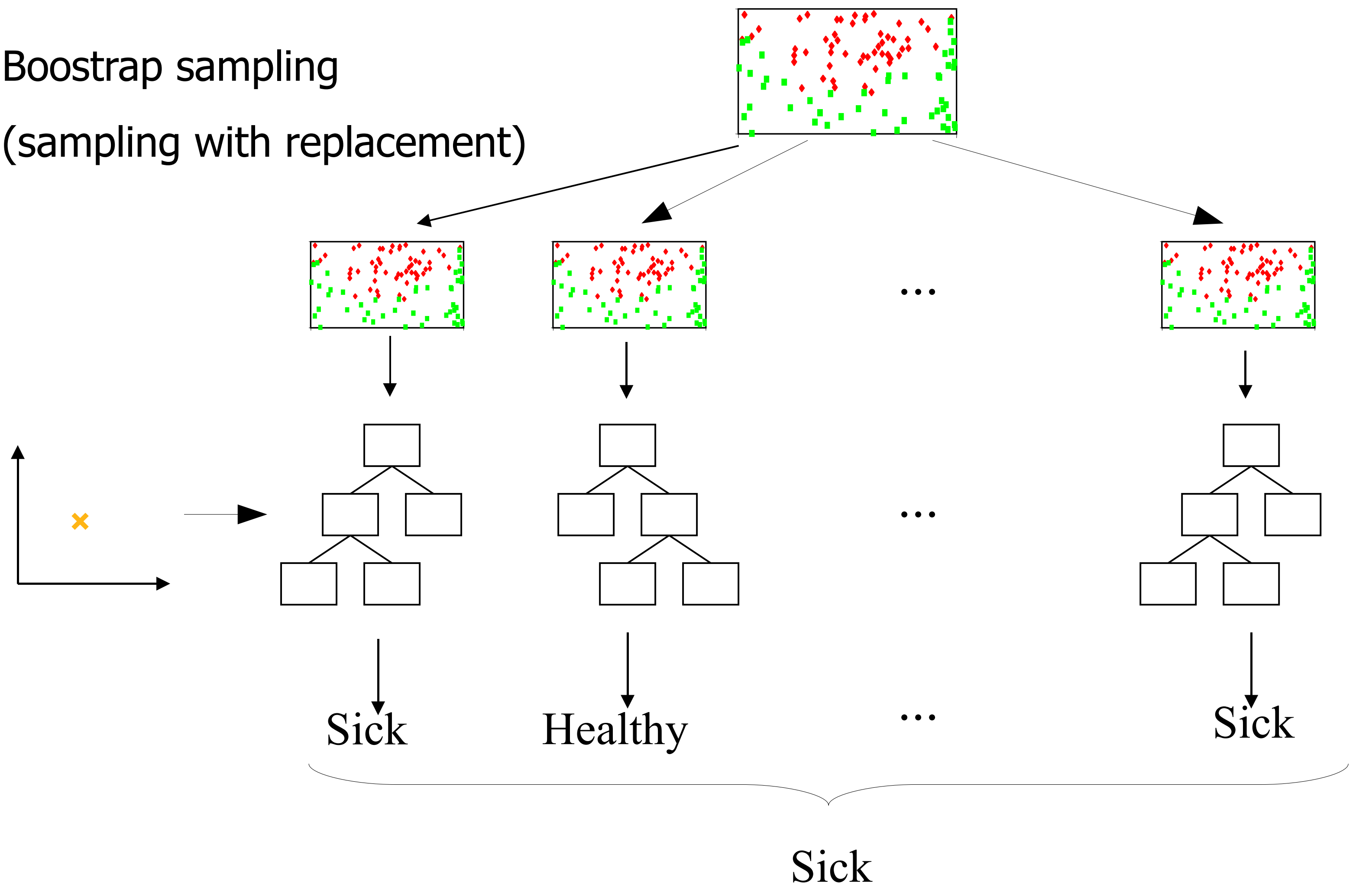
- Solution: aggregate several models to obtain a more stable one



Bagging: bootstrap aggregating

Boostrap sampling

(sampling with replacement)



Note: the more models, the better.

Bootstrap sampling

- Sampling with replacement

	G1	G2	Y
1	0.74	0.68	Healthy
2	0.78	0.45	Disease
3	0.86	0.09	Healthy
4	0.2	0.61	Disease
5	0.2	-5.6	Healthy
6	0.32	0.6	Disease
7	-0.34	-0.45	Healthy
8	0.89	-0.34	Disease
9	0.1	0.3	Healthy
10	-0.34	-0.65	Healthy

$\longrightarrow$

	G1	G2	Y
3	0.86	0.09	Healthy
7	-0.34	-0.45	Healthy
2	0.78	0.45	Disease
9	0.1	0.3	Healthy
3	0.86	0.09	Healthy
10	-0.34	-0.65	Healthy
1	0.74	0.68	Healthy
8	0.89	-0.34	Disease
6	0.32	0.6	Disease
10	-0.34	-0.65	Healthy

- Some objects do not appear, some objects appear several times

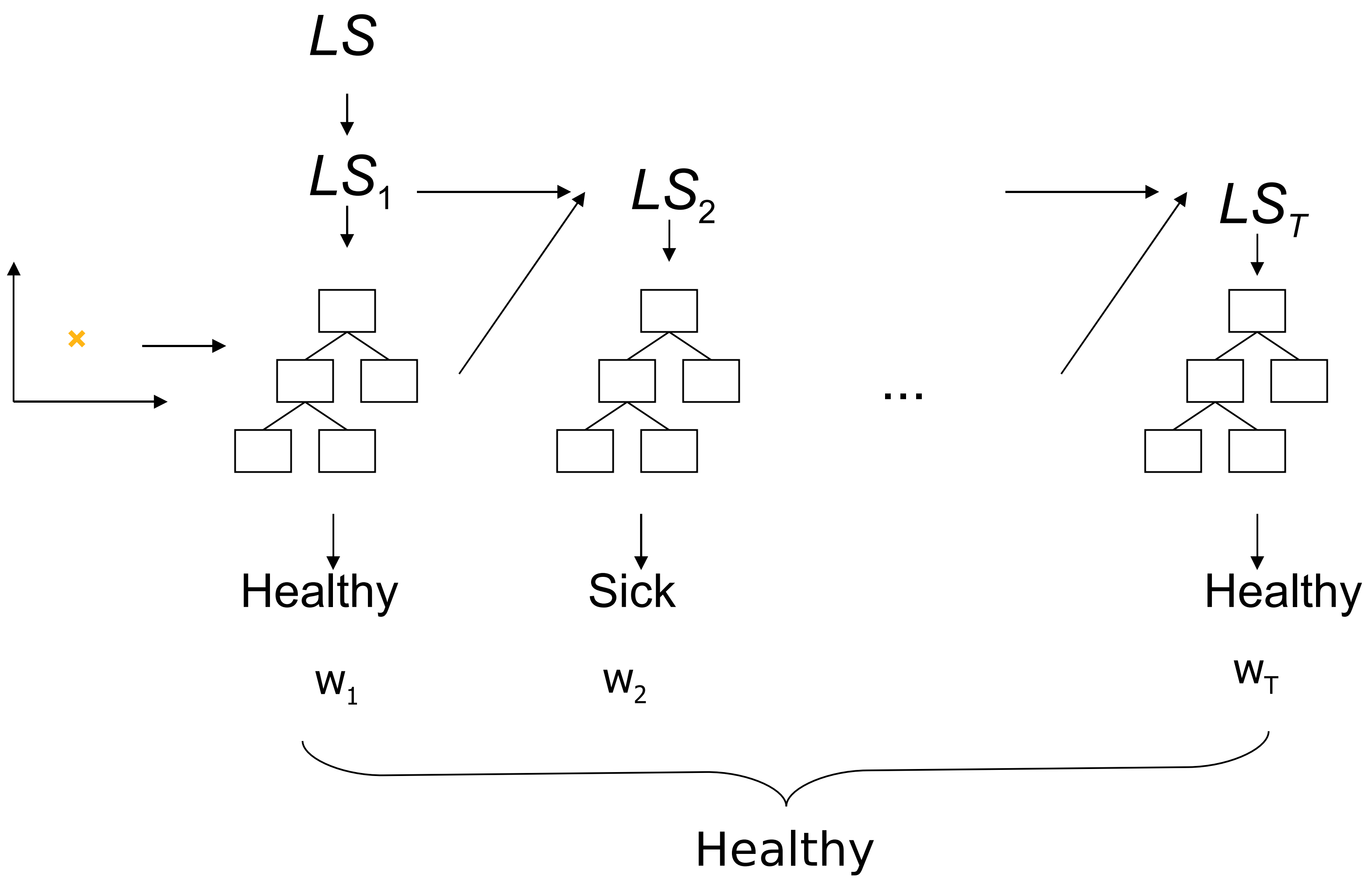
Boosting

- Idea of boosting: combine many « weak » models to produce a more powerful one.
- Weak model = a model that underfits the data (strictly, in classification, a model slightly better than random guessing)

- Adaboost:
  - At each step, adaboost forces the learning algorithm to focus on the cases from the learning sample misclassified by the last model
  - The predictions of the models are combined through a weighted vote. More accurate models have more weights in the vote.

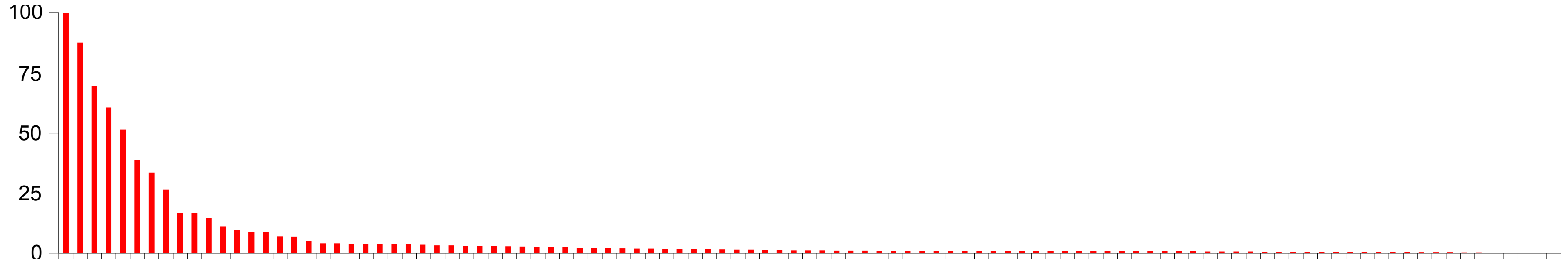
Eg., by duplicating the missclassified examples in the learning sample

Boosting



Interpretability and efficiency

- When combined with decision trees, ensemble methods loose interpretability and efficiency
- However,
    - We still can use the ensemble to compute the importance of variables (by averaging it over all trees)



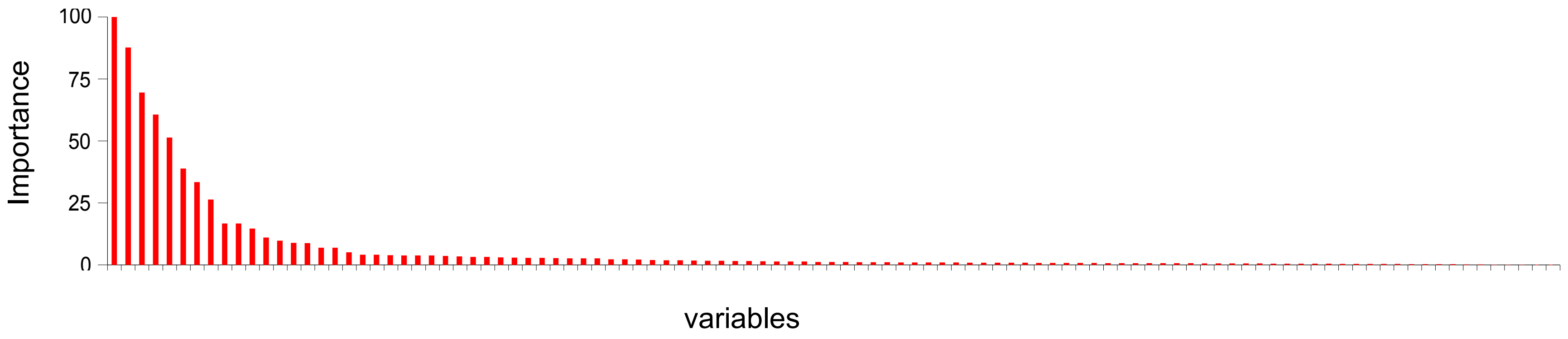
    - Ensemble methods can be parallelized and boosting type algorithm uses smaller trees. So, the increase of computing times is not so detrimental.

Example on microarray data

- 72 patients, 7129 gene expressions, 2 classes of Leukemia (ALL and AML) (Golub et al., Science, 1999)
- Leave-one-out error with several variants

Method	Error
1 decision tree	22.2% (16/72)
Random forests (k=85,T=500)	9.7% (7/72)
Extra-trees (s_{th}=0.5, T=500)	5.5% (4/72)
Adaboost (1 test node, T=500)	1.4% (1/72)

- Variable importance with boosting



Method comparison

Method	Accuracy	Efficiency	Interpretability	Ease of use
kNN	++	+	+	++
DT	+	+++	+++	+++
Linear	++	+++	++	+++
Ensemble	+++	+++	++	+++
ANN	+++	+	+	++
SVM	++++	+	+	+

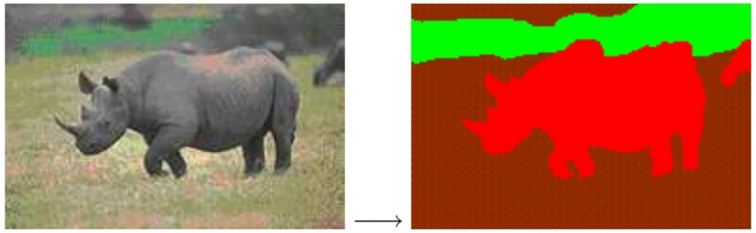
- Note:
  - The relative importance of the criteria depends on the specific application
  - These are only general trends. Eg., in terms of accuracy, no algorithm is always better than all others.

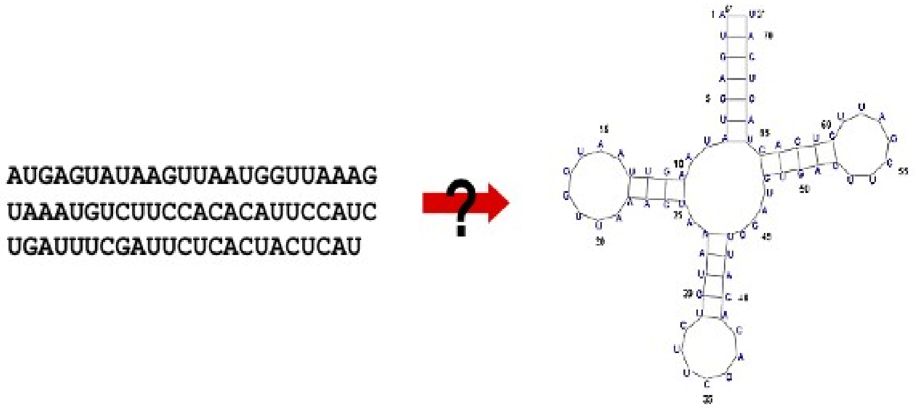
Outline

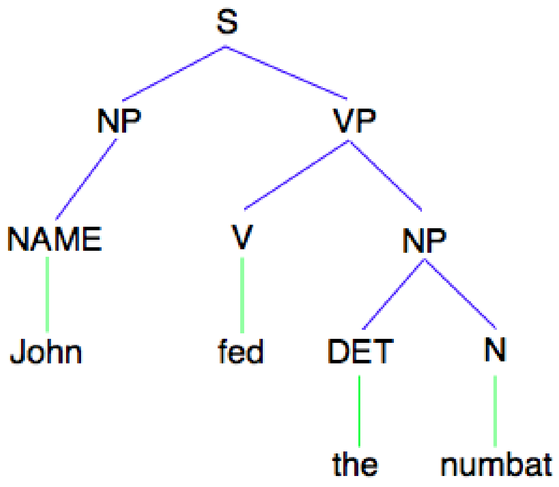
- Introduction
- Supervised Learning
  - Introduction
  - Model selection, cross-validation, overfitting
  - Some supervised learning algorithms
  - Beyond classification and regression
- Other learning protocols/frameworks

Beyond classification and regression

- All supervised learning problems can not be turned into standard classification or regression problems
- Examples:
  - Graph predictions
  - Sequence labeling
  - image segmentation







Structured output approaches

- Decomposition:
  - Reduce the problem to several simpler classification or regression problems by decomposing the output
  - Not always possible and does not take into account interactions between suboutputs
- Kernel output methods
  - Extend regression methods to handle an output space endowed with a kernel
  - This can be done with regression trees or ridge regression for example
- Large margin methods
  - Use SVM-based approaches to learn a model that scores directly input-output pairs:

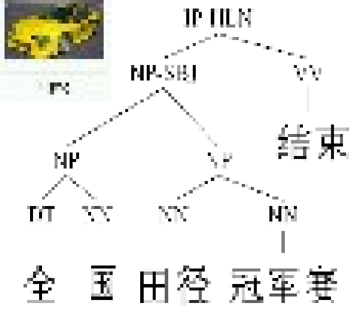
$$y = arg\,max_{y'} \sum_i w_i \phi_i(x, y')$$

Outline

- Introduction
- Supervised learning
- Other learning protocols/frameworks
  - Semi-supervised learning
  - Transductive learning
  - Active learning
  - Reinforcement learning
  - Unsupervised learning

Labeled versus unlabeled data

- Unlabeled data=input-output pairs without output value
- In many settings, unlabeled data is cheap but labeled data can be hard to get
  - labels may require human experts
  - human annotation is expensive, slow, unreliable
  - labels may require special devices
- Examples:
  - Biomedical domain
  - Speech analysis
  - Natural language parsing
  - Image categorization/segmentation
  - Network measurement

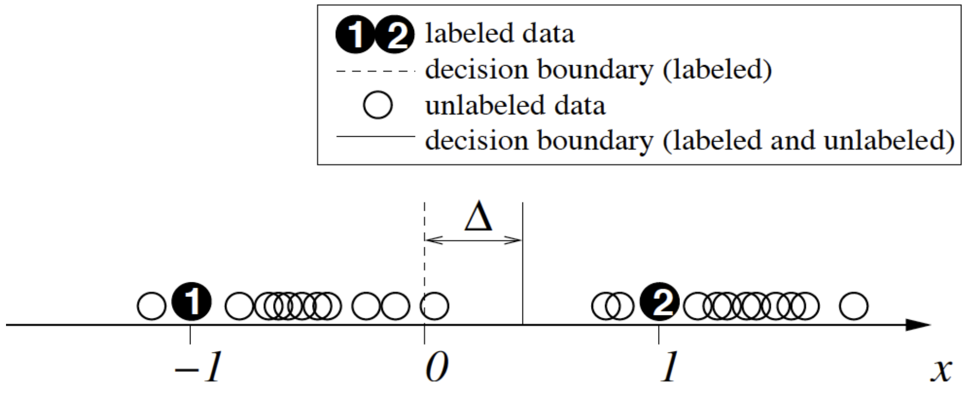


Semi-supervised learning

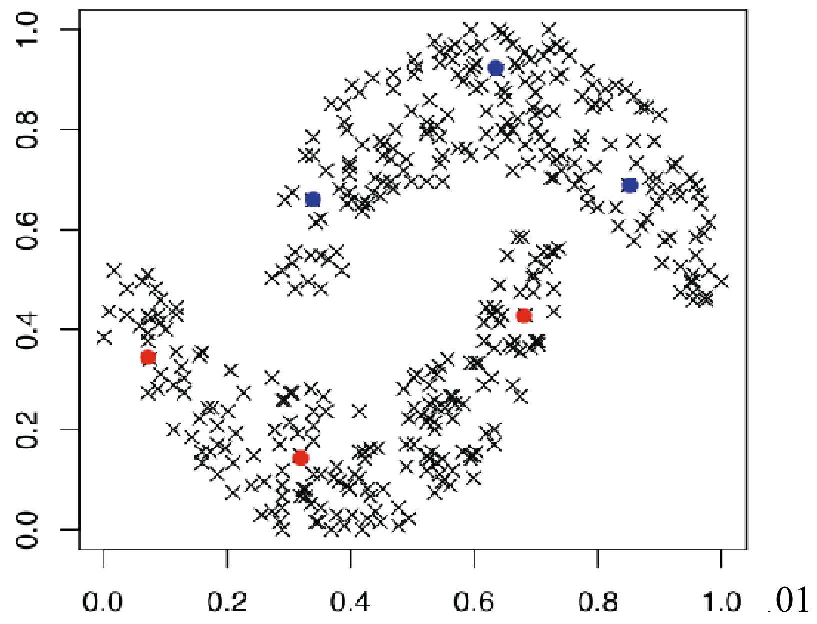
- Goal: exploit both labeled and unlabeled data to build better models than using each one alone

	A1	A2	A3	A4	Y
labeled data	0.01	0.37	T	0.54	Healthy
	-2.3	-1.2	F	0.37	Disease
	0.69	-0.78	F	0.63	Healthy
unlabeled data	-0.56	-0.89	T	-0.42	
	-0.85	0.62	F	-0.05	
	-0.17	0.09	T	0.29	
test data	-0.09	0.3	F	0.17	?

- Why would it improve?

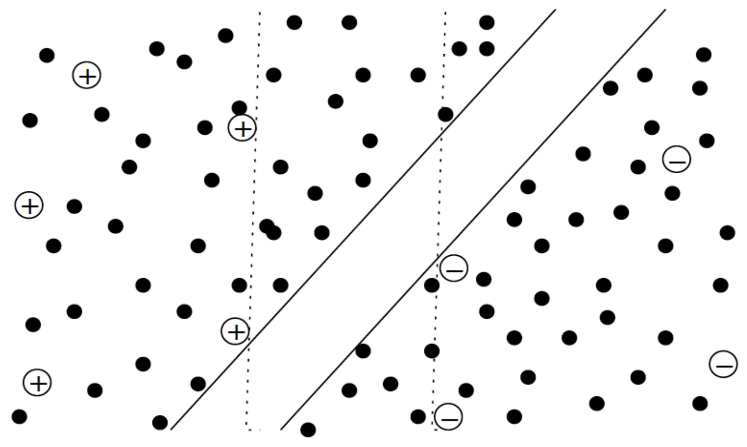


labeled data
decision boundary (labeled)
unlabeled data
decision boundary (labeled and unlabeled)
Δ
−1
0
1
x



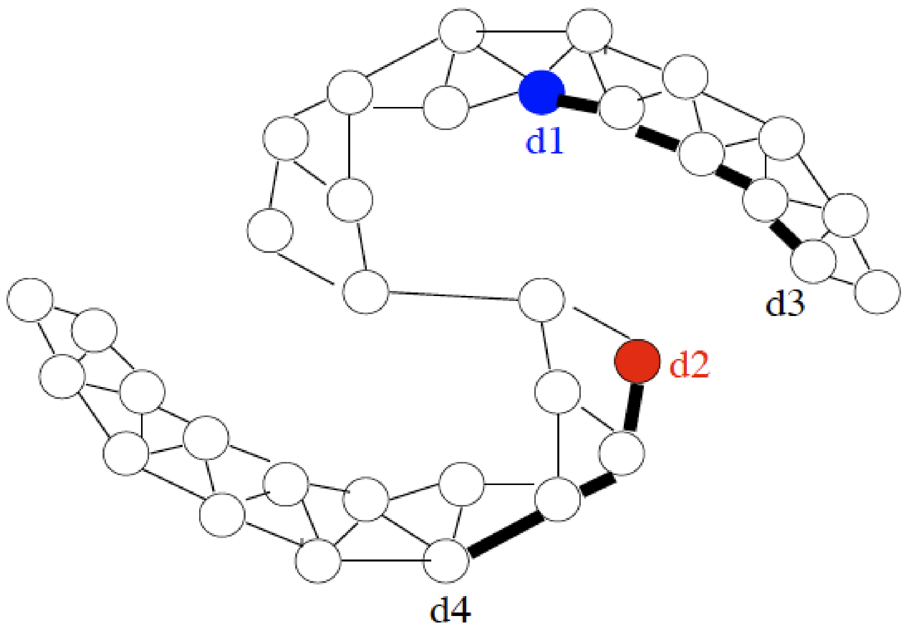
Some approaches

- Self-training
  - Iteratively label some unlabeled examples with a model learned from the previously labeled examples
- Semi-supervised SVM (S3VM)
  - Enumerate all possible labeling of the unlabeled examples
  - Learn an SVM for each labeling
  - Pick the one with the largest margin



Some approaches

- Graph-based algorithms
  - Build a graph over the (labeled and unlabeled) examples (from the inputs)
  - Learn a model that predicts well labeled examples and is smooth over the graph



Transductive learning

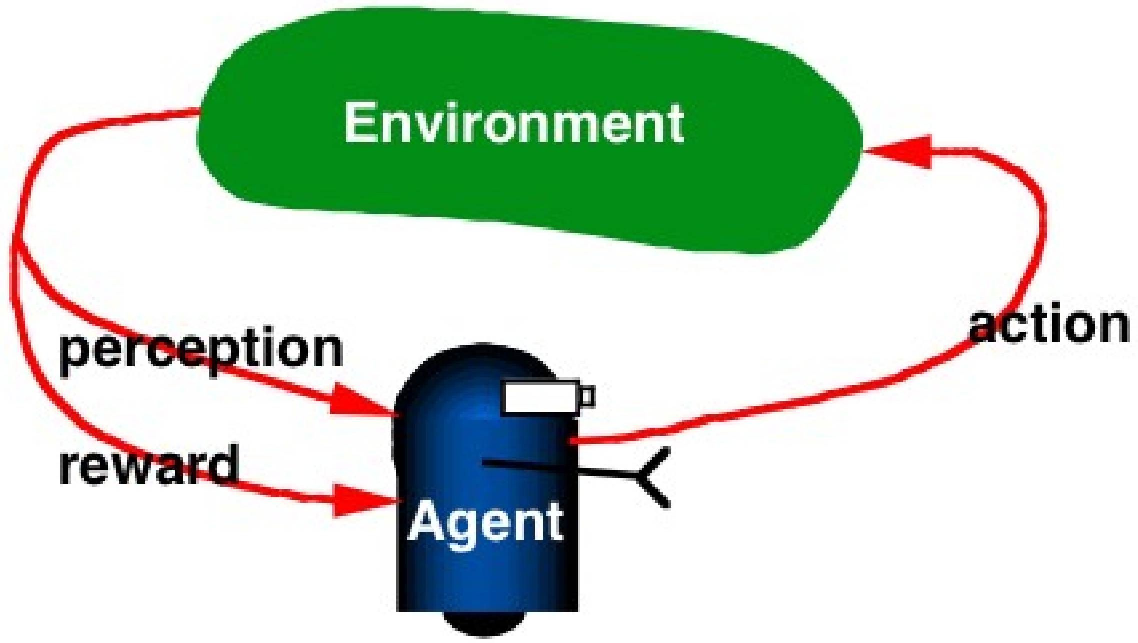
- Like supervised learning but we have access to the test data from the beginning and we want to exploit it
- We don't want a model, only compute predictions for the unlabeled data
- Simple solution:
    - Apply semi-supervised learning techniques using the test data as unlabeled data to get a model
    - Use the resulting model to make predictions on the test data
- There exist also specific algorithms that avoid building a model

Active learning

- Goal:
  - given unlabeled data, find (adaptively) the examples to label in order to learn an accurate model
  - The hope is to reduce the number of labeled instances with respect to the standard batch SL
- Usually, in an online setting:
  - choose the k “best” unlabeled examples
  - determine their labels
  - update the model and iterate
- Algorithms differ in the way the unlabeled examples are selected
  - Example: choose the k examples for which the model predictions are the most uncertain

Reinforcement learning

Learning form interactions



$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} \ldots$$

Goal: learn to choose sequence of actions (= policy) that maximizes $r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots,$ where $0 \leqslant \gamma < 1$

RL approaches

- System is usually modeled by
  - state transition probabilities $P(s_{t+1}|s_t, a_t)$
  - reward probabilities $P(r_{t+1}|s_t, a_t)$

  (= Markov Decision Process)
- Model of the dynamics and reward is known $\Rightarrow$ try to compute optimal policy by dynamic programming
- Model is unknown
  - Model-based approaches $\Rightarrow$ first learn a model of the dynamics and then derive an optimal policy from it (DP)
  - Model-free approaches $\Rightarrow$ learn directly a policy from the observed system trajectories

Reinforcement versus supervised learning

- Batch-mode SL: learn a mapping from input to output from observed input-output pairs
- Batch-mode RL: learn a mapping from state to action from observed (state,action,reward) triplets
- Online active learning: combine SL and (online) selection of instances to label
- Online RL: combine policy learning with control of the system and generation of the training trajectories
- Note:
  - RL would reduce to SL if the optimal action was known in each state
  - SL is used inside RL to model system dynamics and/or value functions

Examples of applications

- Robocup Soccer Teams (Stone & Veloso, Riedmiller et al.)
- Inventory Management (Van Roy, Bertsekas, Lee &Tsitsiklis)
- Dynamic Channel Assignment, Routing (Singh & Bertsekas, Nie & Haykin, Boyan & Littman)
- Elevator Control (Crites & Barto)
- Many Robots: navigation, bi-pedal walking, grasping, switching between skills...
- Games: TD-Gammon and Jellyfish (Tesauro, Dahl)

Unsupervised learning

- Unsupervised learning tries to find any regularities in the data without guidance about inputs and outputs

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
-0.27	-0.15	-0.14	0.91	-0.17	0.26	-0.48	-0.1	-0.53	-0.65	0.23	0.22	0.98	0.57	0.02	-0.55	-0.32	0.28	-0.33
-2.3	-1.2	-4.5	-0.01	-0.83	0.66	0.55	0.27	-0.65	0.39	-1.3	-0.2	-3.5	0.4	0.21	-0.87	0.64	0.6	-0.29
0.41	0.77	-0.44	0	0.03	-0.82	0.17	0.54	-0.04	0.6	0.41	0.66	-0.27	-0.86	-0.92	0	0.48	0.74	0.49
0.28	-0.71	-0.82	0.27	-0.21	-0.9	0.61	-0.57	0.44	0.21	0.97	-0.27	0.74	0.2	-0.16	0.7	0.79	0.59	-0.33
-0.28	0.48	0.79	-0.14	0.8	0.28	0.75	0.26	0.3	-0.78	-0.72	0.94	-0.78	0.48	0.26	0.83	-0.88	-0.59	0.71
0.01	0.36	0.03	0.03	0.59	-0.5	0.4	-0.88	-0.53	0.95	0.15	0.31	0.06	0.37	0.66	-0.34	0.79	-0.12	0.49
-0.53	-0.8	-0.64	-0.93	-0.51	0.28	0.25	0.01	-0.94	0.96	0.25	-0.12	0.27	-0.72	-0.77	-0.31	0.44	0.58	-0.86
0.04	0.94	-0.92	-0.38	-0.07	0.98	0.1	0.19	-0.57	-0.69	-0.23	0.05	0.13	-0.28	0.98	-0.08	-0.3	-0.84	0.47
-0.88	-0.73	-0.4	0.58	0.24	0.08	-0.2	0.42	-0.61	-0.13	-0.47	-0.36	-0.37	0.95	-0.31	0.25	0.55	0.52	-0.66
-0.56	0.97	-0.93	0.91	0.36	-0.14	-0.9	0.65	0.41	-0.12	0.35	0.21	0.22	0.73	0.68	-0.65	-0.4	0.91	-0.64

- Are there interesting groups of variables or samples? outliers? What are the dependencies between variables?

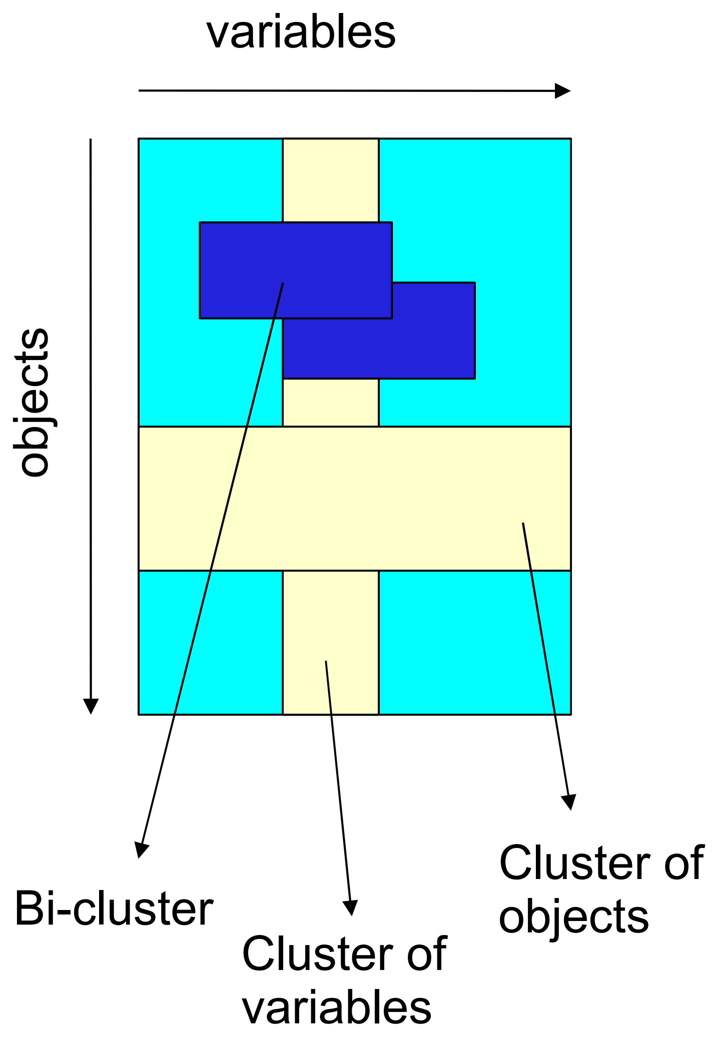
Unsupervised learning methods

- Many families of problems exist, among which:
  - Clustering: try to find natural groups of samples/variables
    - eg: k-means, hierarchical clustering
  - Dimensionality reduction: project the data from a high-dimensional space down to a small number of dimensions
    - eg: principal/independent component analysis, MDS
  - Density estimation: determine the distribution of data within the input space
    - eg: bayesian networks, mixture models.

Clustering

- Goal: grouping a collection of objects into subsets or “clusters”, such that those within each cluster are more closely related to one another than objects assigned to different clusters

Clustering



- Clustering rows

  grouping similar objects
- Clustering columns

  grouping similar variables across samples
- Bi-Clustering/Two-way clustering

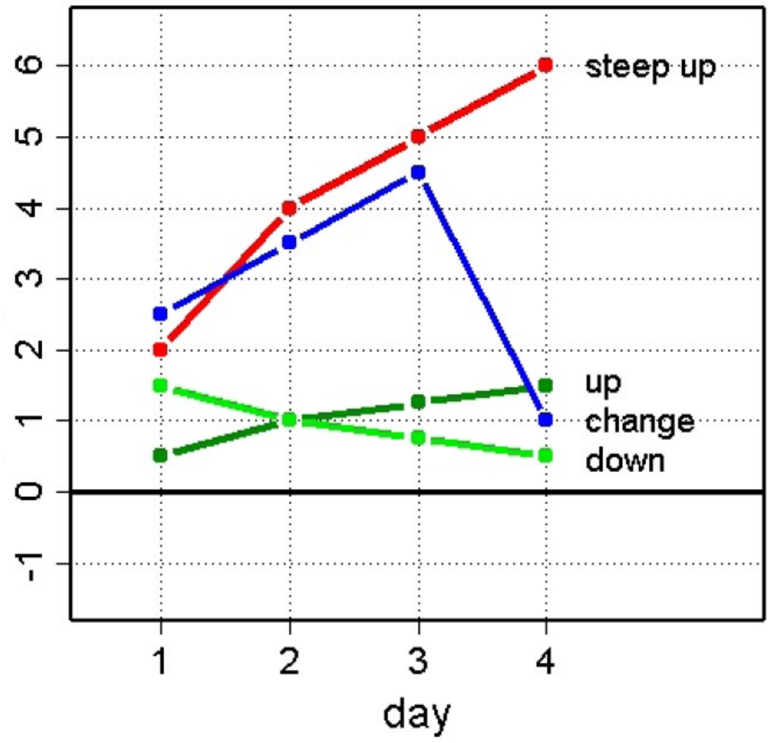
  grouping objects that are similar across a subset of variables

Clustering

- Two essential components of cluster analysis:
  - **Distance measure:** A notion of distance or similarity of two objects: When are two objects close to each other?
  - **Cluster algorithm:** A procedure to minimize distances of objects within groups and/or maximize distances between groups

Examples of distance measures

- **Euclidean** distance measures average difference across coordinates
- **Manhattan** distance measures average difference across coordinates, in a robust way
- **Correlation** distance measures difference with respect to trends



Clustering algorithms

- Popular algorithms for clustering
  - hierarchical clustering
  - K-means
  - SOMs (Self-Organizing Maps)
  - autoclass, mixture models...
- Hierarchical clustering allows the choice of the dissimilarity matrix.
- k-Means and SOMs take original data directly as input. Attributes are assumed to live in Euclidean space.

Hierarchical clustering

Agglomerative clustering:

1. Each object is assigned to its own cluster
2. Iteratively:
    - the two most similar clusters are joined and replaced by a new one
    - the distance matrix is updated with this new cluster replacing the two joined clusters

(divisive clustering would start from a big cluster)

Distance between two clusters

- Single linkage uses the smallest distance

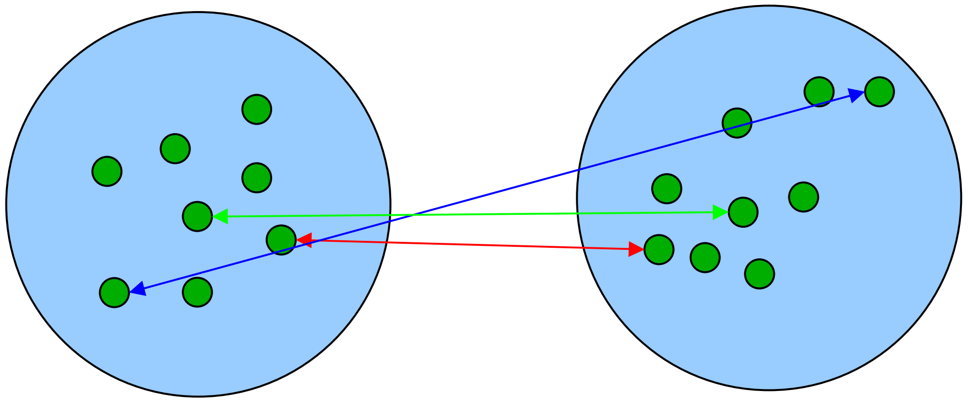
$$d_S(G,H) = \min_{i \in G, j \in H} d_{ij}$$

- Complete linkage uses the largest distance

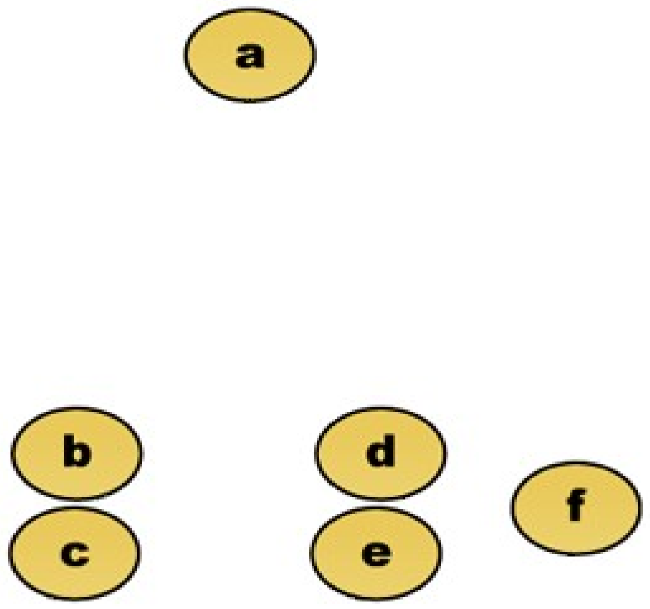
$$d_C(G,H) = \max_{i \in G, j \in H} d_{ij}$$

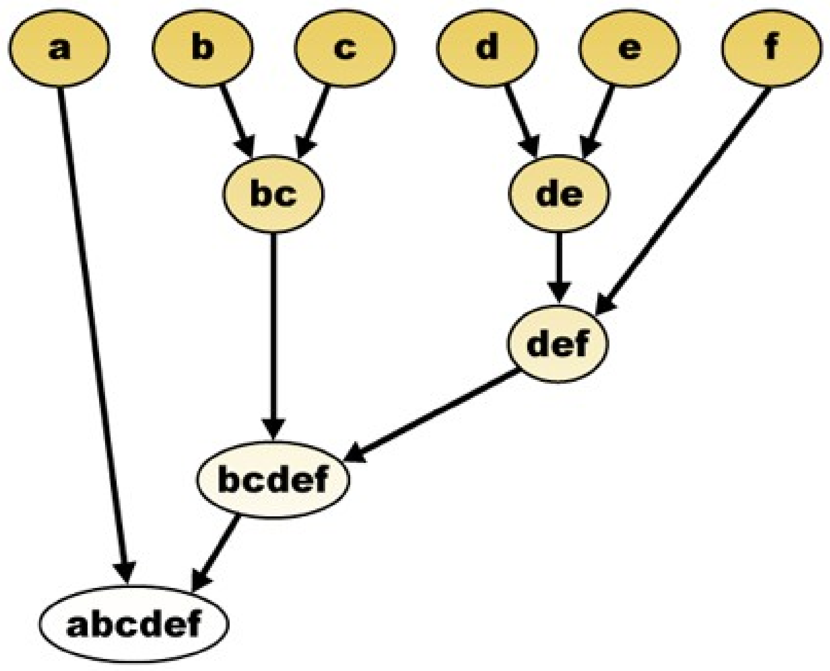
- Average linkage uses the average distance

$$d_A(G,H) = \frac{1}{N_G N_H} \sum_{i \in G} \sum_{j \in H} d_{ij}$$



Hierarchical clustering

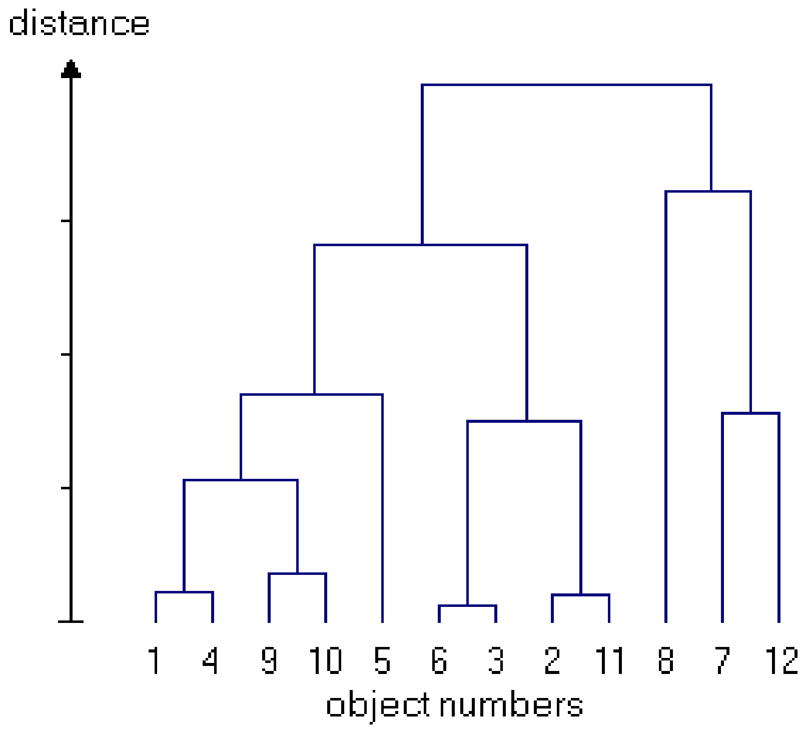




(wikipedia)

Dendrogram

- Hierarchical clustering are visualized through dendrograms
  - Clusters that are joined are combined by a line
  - Height of line is distance between clusters
  - Can be used to determine visually the number of clusters



Hierarchical clustering

- Strengths
  - No need to assume any particular number of clusters
  - Can use any distance matrix
  - Find sometimes a meaningful taxonomy
- Limitations
  - Find a taxonomy even if it does not exist
  - Once a decision is made to combine two clusters it cannot be undone
  - Not well theoretically motivated

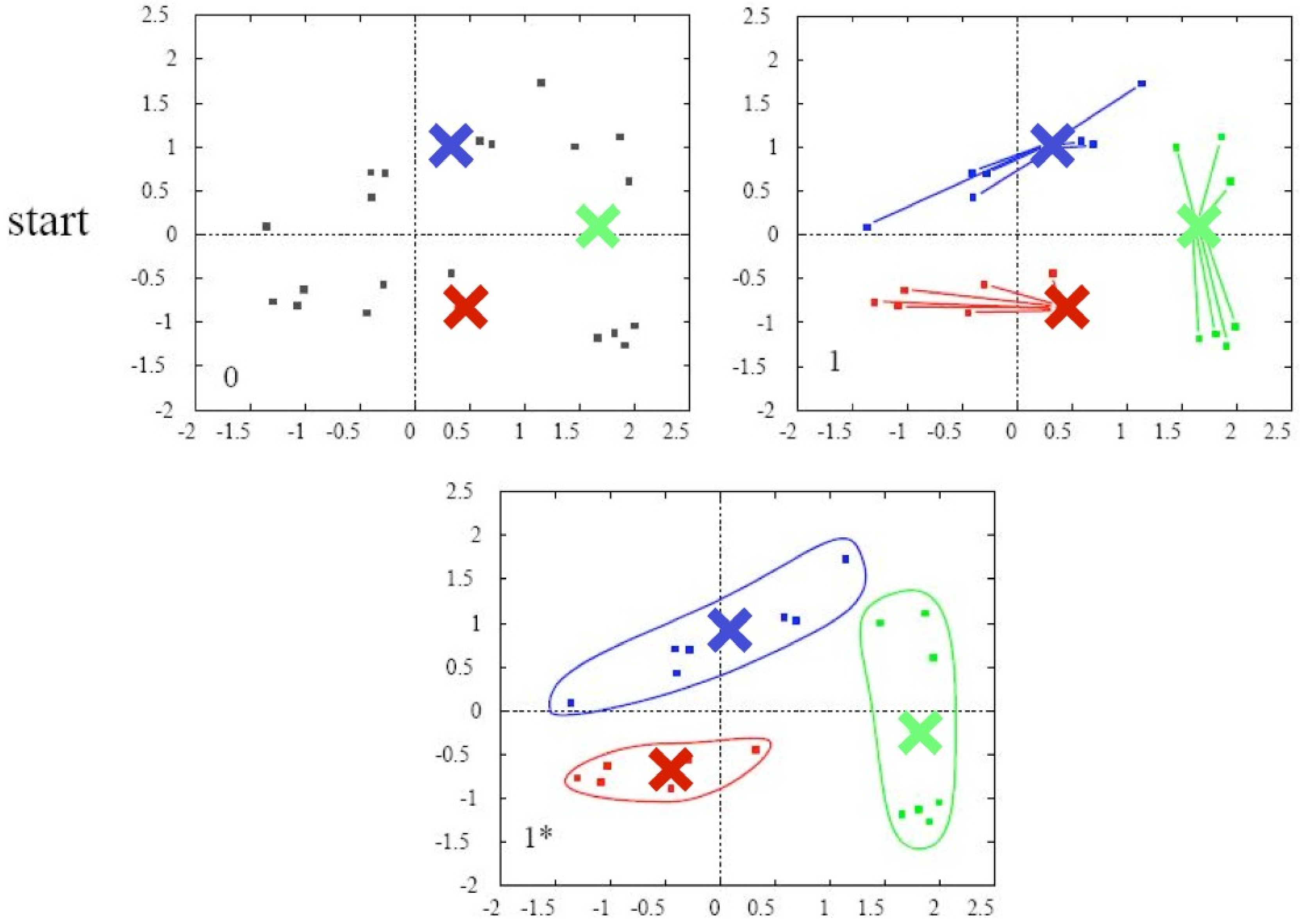
k-Means clustering

- Partitioning algorithm with a prefixed number k of clusters
- Use Euclidean distance between objects
- Try to minimize the sum of intra-cluster variances

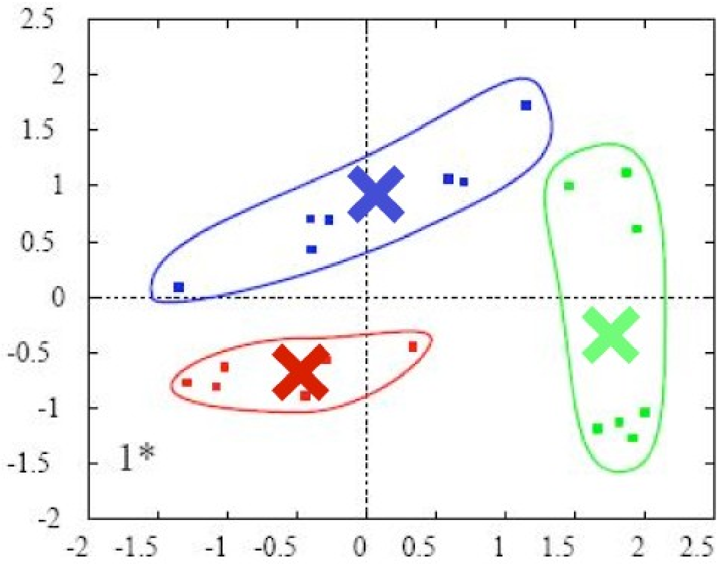
$$\sum_{j=1}^{k} \sum_{o \in Cluster\, j} d^2\left(o, c_j\right)$$

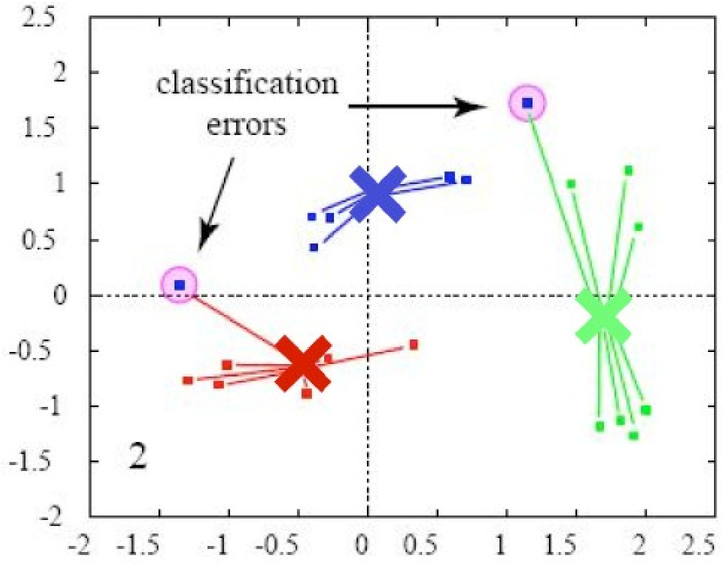
where c_j is the center of cluster j and d^2 is the Euclidean distance

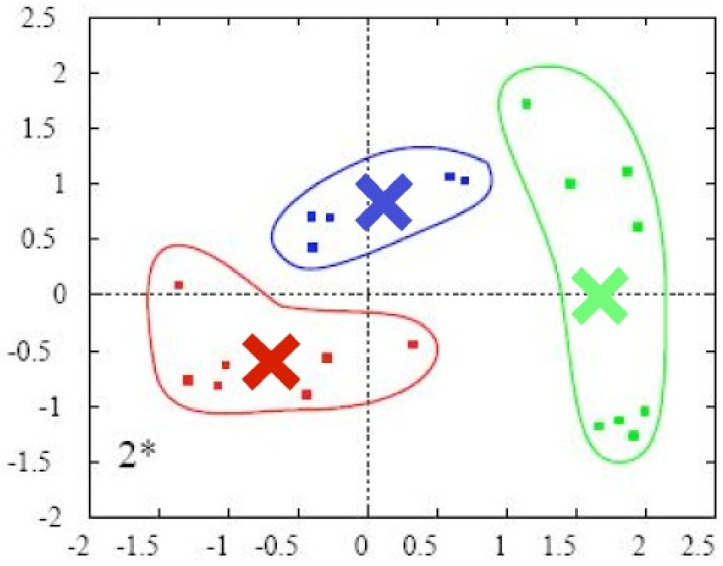
k-Means clustering



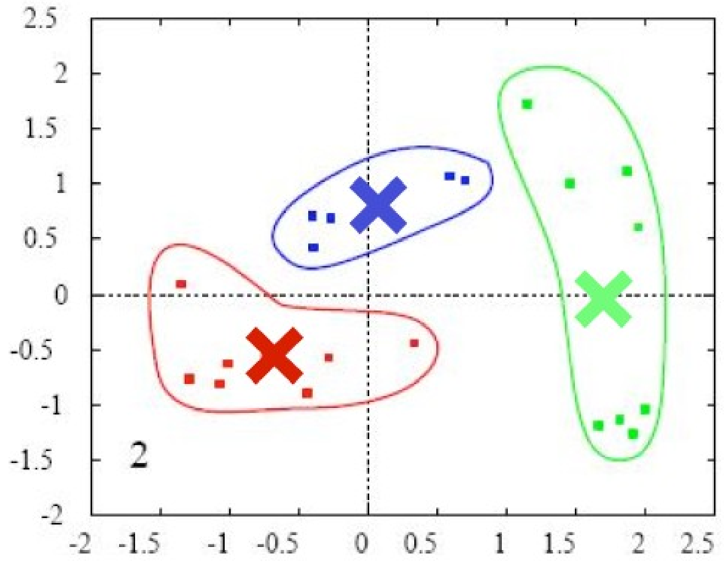
k-Means clustering

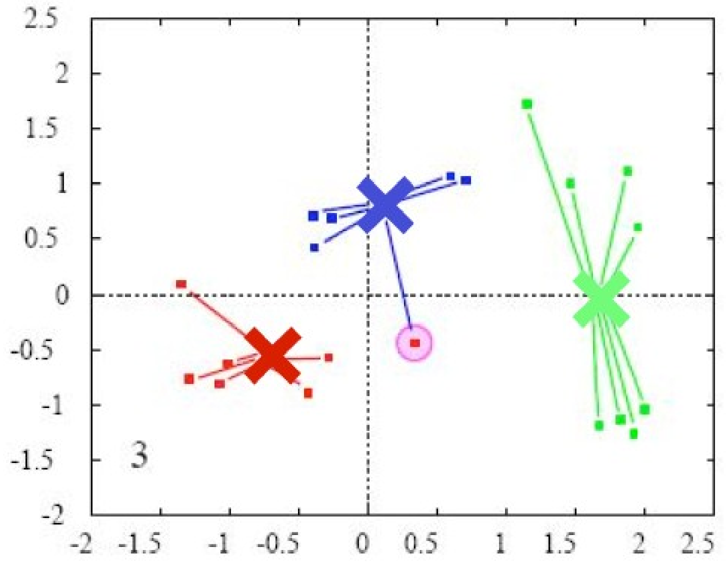


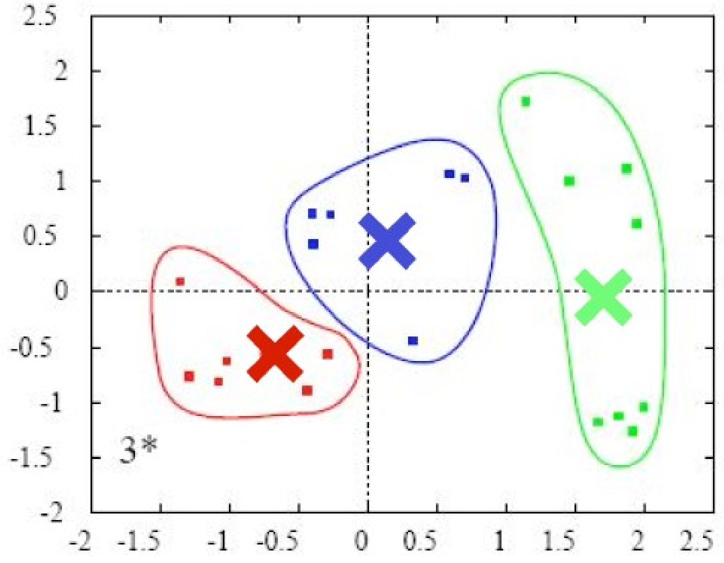




k-Means clustering







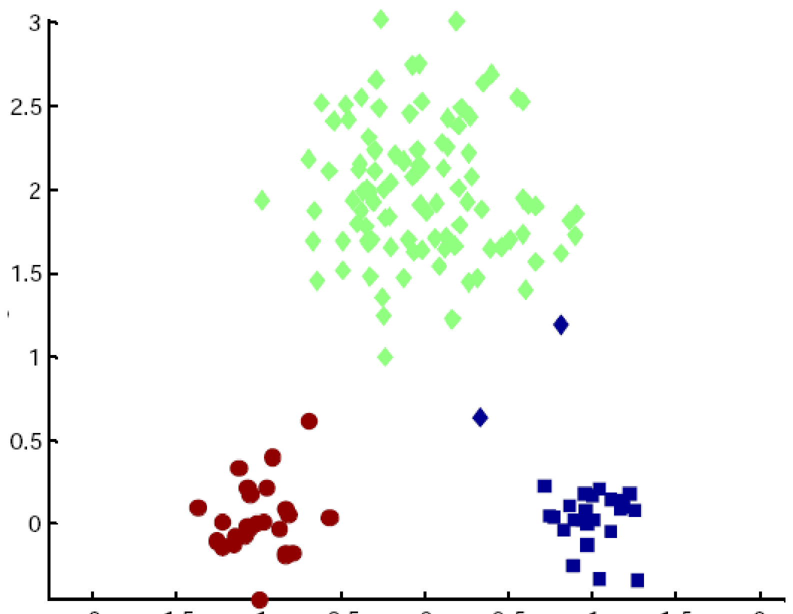
end

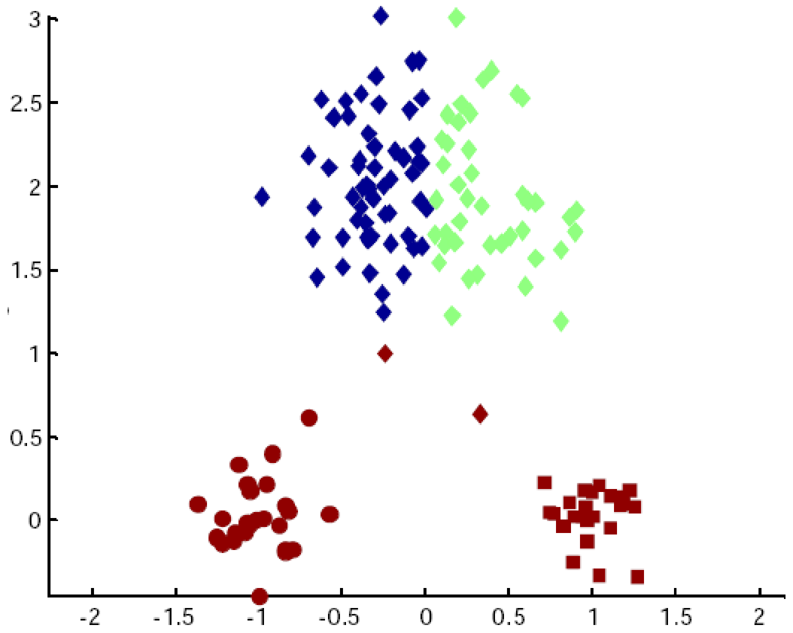
k-Means clustering

- Strengths
  - Simple, understandable
  - Can cluster any new point (unlike hierarchical clustering)
  - Well motivated theoretically
- Limitations
  - Must fix the number of clusters beforehand
  - Sensitive to the initial choice of cluster centers
  - Sensitive to outliers

Suboptimal clustering

- You could obtain any of these from a random start of k-means





- Solution: restart the algorithm several times

Principal Component Analysis

- An exploratory technique used to reduce the dimensionality of the data set to a smaller space (2D, 3D)

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
0.25	0.93	0.04	-0.78	-0.53	0.57	0.19	0.29	0.37	-0.22
-2.3	-1.2	-4.5	-0.51	-0.76	0.07	0.81	0.95	0.99	0.26
-0.29	-1	0.73	-0.33	0.52	0.13	0.13	0.53	-0.5	-0.48
-0.16	-0.17	-0.26	0.32	-0.08	-0.38	-0.48	0.99	-0.95	0.34
0.07	-0.87	0.39	0.5	-0.63	-0.53	0.79	0.88	0.74	-0.14
0.61	0.15	0.68	-0.94	0.5	0.06	-0.56	0.49	0	-0.77

$\longrightarrow$

PC1	PC2
0.36	0.1
-2.3	-1.2
0.27	-0.89
-0.19	0.7
-0.77	-0.7
-0.65	-0.99

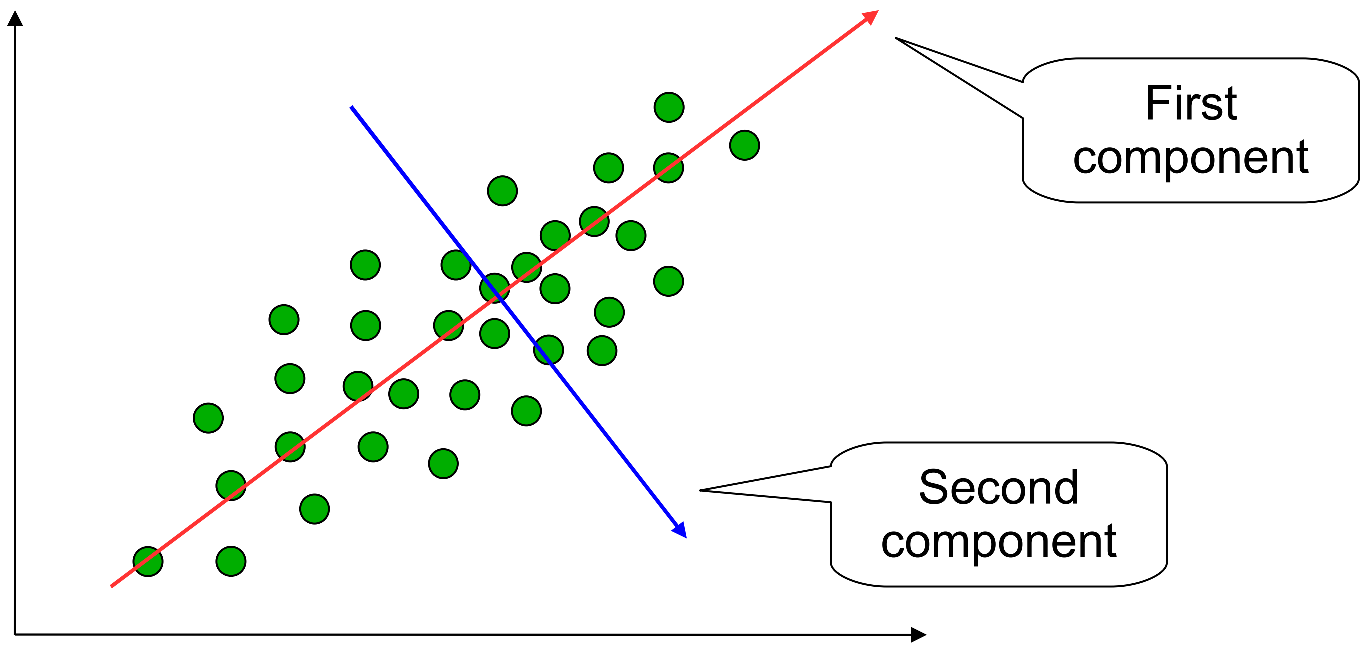
- Transform some large number of variables into a smaller number of uncorrelated variables called principal components (PCs)

Objectives of PCA

- Reduce dimensionality (pre-processing for other methods)
- Choose the most useful (informative) variables
- Compress the data
- Visualize multidimensional data
  - to identify groups of objects
  - to identify outliers

Basic idea

- Goal: map data points into a few dimensions while trying to preserve the variance of the data as much as possible



Each component is a linear combination of the original variables

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
-0.39	-0.38	0.29	0.65	0.15	0.73	-0.57	0.91	-0.89	-0.17
-2.3	-1.2	-4.5	-0.15	0.86	-0.85	0.43	-0.19	-0.83	-0.4
0.9	0.4	-0.11	0.62	0.94	0.97	0.1	-0.41	0.01	0.1
-0.82	-0.31	0.14	0.22	-0.49	-0.76	0.27	0	-0.43	-0.81
0.71	0.39	-0.09	0.26	-0.46	-0.05	0.46	0.39	-0.01	0.64
-0.25	0.27	-0.81	-0.42	0.62	0.54	-0.67	-0.15	-0.46	0.69

$\longrightarrow$

PC1	PC2
0.62	-0.33
-2.3	-1.2
0.88	0.31
-0.18	-0.05
-0.39	-0.01
-0.61	0.53

Scores for each sample and PC

$PC1=0.2*A1+3.4*A2-4.5*A3$

$PC2=0.4*A4+5.6*A5+2.3*A7$

…

$\mathrm{VAR}(PC1)=4.5 \rightarrow 45\%$

$\mathrm{VAR}(PC2)=3.3 \rightarrow 33\%$

…

Loading of a variable
- Gives an idea of its importance in the component
- Can be use for selecting biomarkers

For each component, we have a measure of the percentage of the **variance** of the initial data that it contains

Books

- Reference textbooks for the course
  - *The elements of statistical learning: data mining, inference, and prediction*. T. Hastie et al, Springer, 2001
  - *Pattern Recognition and Machine Learning (Information Science and Statistics)*. C.M.Bishop, Springer, 2004
- Other textbooks
  - *Pattern classification* (2nd edition). R.Duda, P.Hart, D.Stork, Wiley Interscience, 2000
  - *Introduction to Machine Learning*. Ethan Alpaydin, MIT Press, 2004.
  - *Machine Learning*. Tom Mitchell, McGraw Hill, 1997.

Books

- More advanced topics
  - *kernel methods for pattern analysis.* J. Shawe-Taylor and N. Cristianini. Cambridge University Press, 2004
  - *Reinforcement Learning: An Introduction.* R.S. Sutton and A.G. Barto. MIT Press, 1998
  - *Neuro-Dynamic Programming.* D.P Bertsekas and J.N. Tsitsiklis. Athena Scientific, 1996
  - *Semi-supervised learning.* Chapelle et al., MIT Press, 2006
  - *Predicting structured data.* G. Bakir et al., MIT Press, 2007

Softwares

- Pepito
  - www.pepite.be
  - Free for academic research and education
- WEKA
  - http://www.cs.waikato.ac.nz/ml/weka/
- Many R and Matlab packages
  - http://www.kyb.mpg.de/bs/people/spider/
  - http://www.cs.ubc.ca/~murphyk/Software/BNT/bnt.html
  -

Journals

- Journal of Machine Learning Research
- Machine Learning
- IEEE Transactions on Pattern Analysis and Machine Intelligence
- Journal of Artificial Intelligence Research
- Neural computation
- Annals of Statistics
- IEEE Transactions on Neural Networks
- Data Mining and Knowledge Discovery
- ...

Conferences

- International Conference on Machine Learning (ICML)
- European Conference on Machine Learning (ECML)
- Neural Information Processing Systems (NIPS)
- Uncertainty in Artificial Intelligence (UAI)
- International Joint Conference on Artificial Intelligence (IJCAI)
- International Conference on Artificial Neural Networks (ICANN)
- Computational Learning Theory (COLT)
- Knowledge Discovery and Data mining (KDD)
- ...