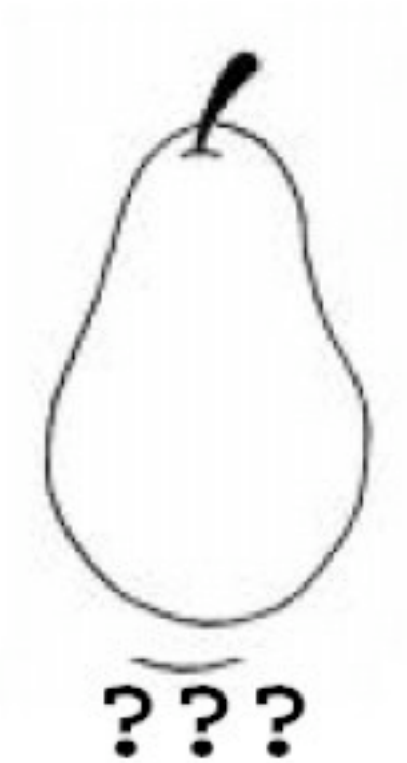
Machine Learning in Computational Biology

Jean-Philippe Vert

Mines ParisTech / Institut Curie / INSERM

Apple or pear?



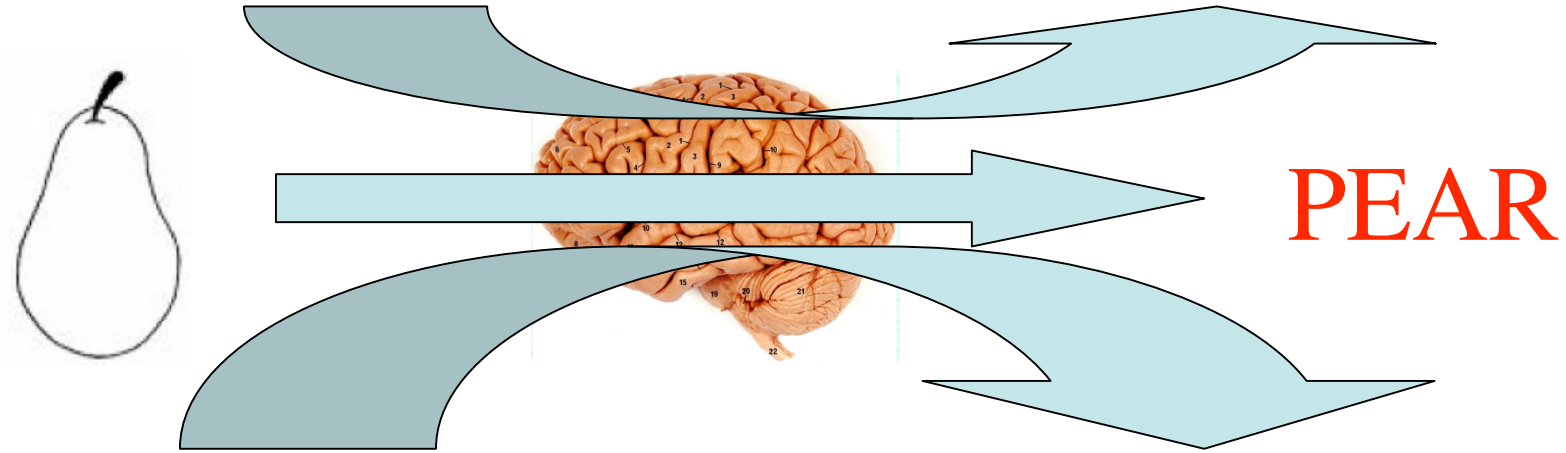
???

???

???

We need a predictor

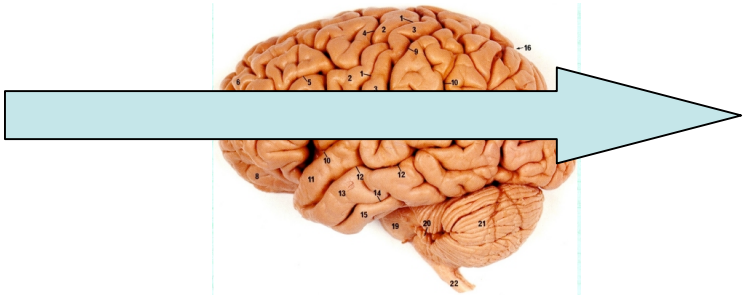
APPLE



PEAR

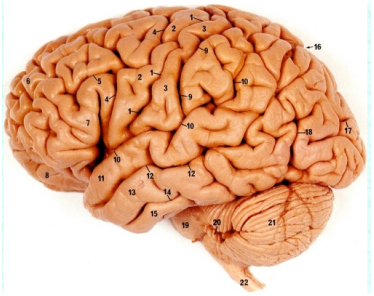
PEAR

How to make a predictor?

IMAGE  APPLE/PEAR

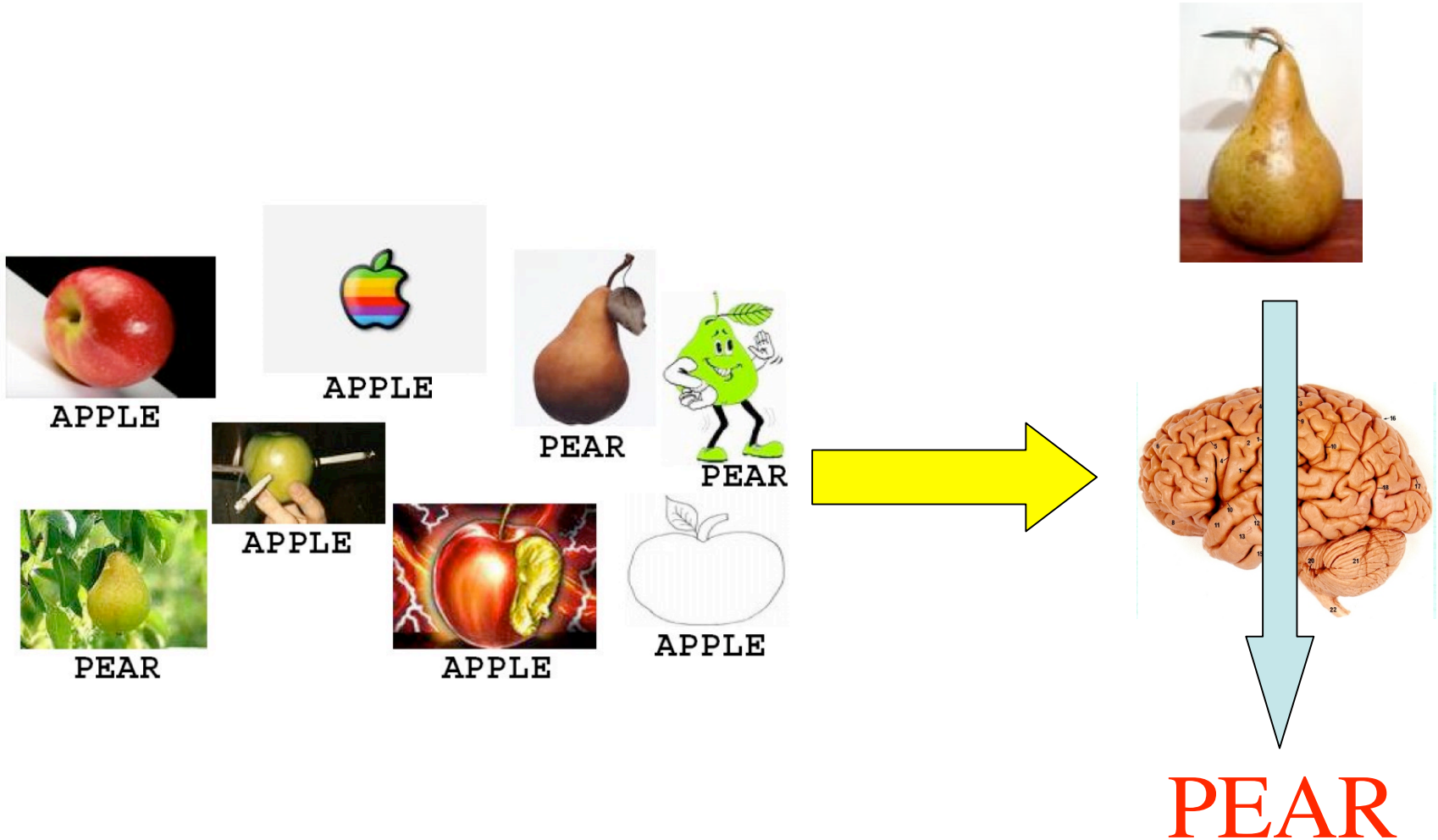
1. Knowledge-based, « intelligent design »
2. Data-based, « machine learning »

Knowledge-based predictor



- Based on shape, texture, color, …
- Usually difficult to engineer
- Can not be used for other problems (eg, discriminate strawberries vs grapes)

Data-based predictor



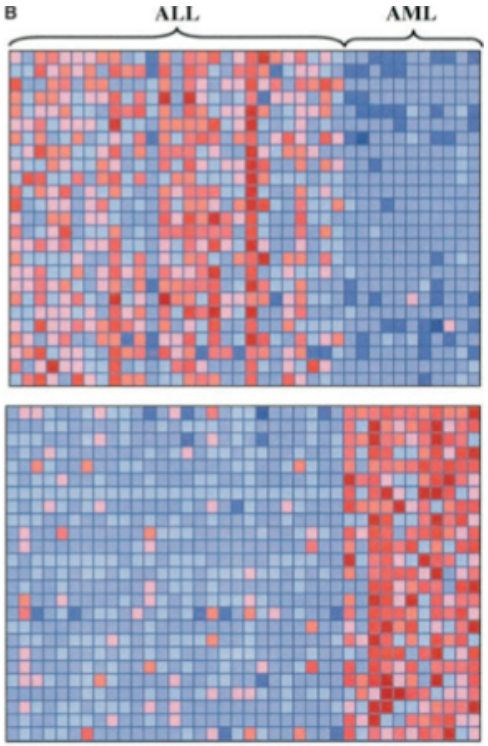
Data-based predictor

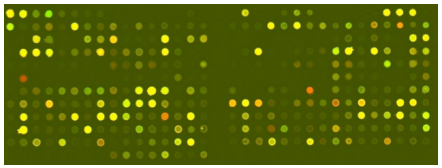
- Needs a database of labeled examples
- Does not always provide a simple rule (« black-box »)
- The more data the better!
- The algorithm can be quite generic

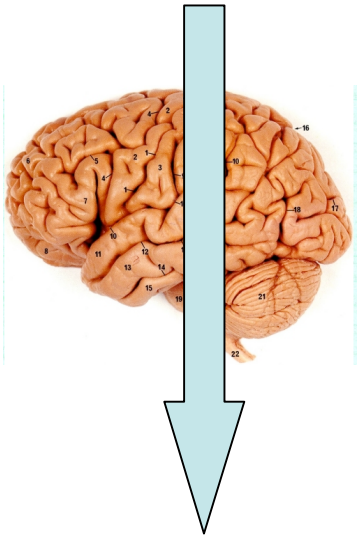
Ok, but there is no apple is bioinformatics!

- Sure, but:
  - There are many data
  - Many problems can be formulated as that of « learning a predictor from data »
  - It is often difficult to design knowledge-based predictors (no clear biological theory, noise, large number of features…)

Example: diagnosis/prognosis from microarray data

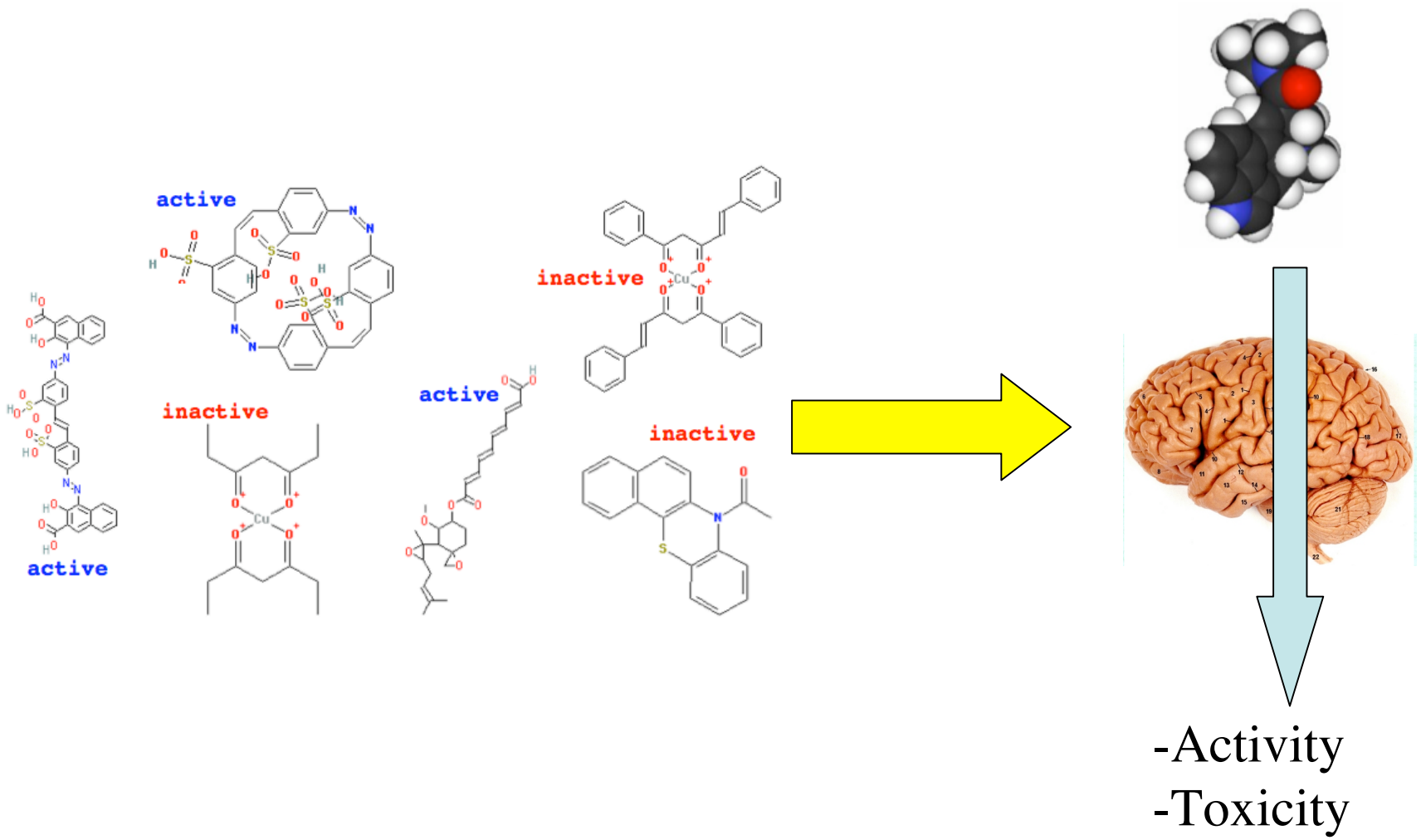






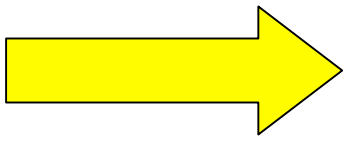
- Cancer type
- Future evolution

Example: virtual screening

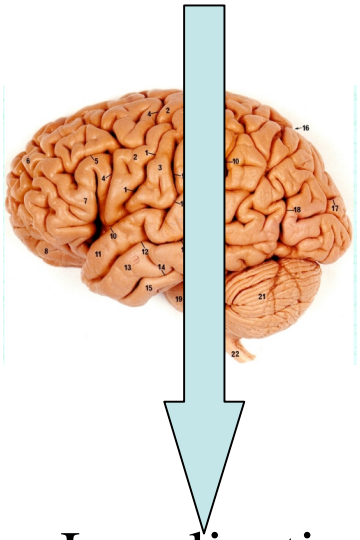


Example: gene annotation

- Secreted proteins:
  MASKATLLLAFTLLFATCIARHQQRQQQQNQCQLQNIEA...
  MARSSLFTFLCLAVFINGCLSQIEQQSPWEFQGSEVW...
  MALHTVLIMLSLLPMLEAQNPEHANITIGEPITNETLGWL...
  ...
- Non-secreted proteins:
  MAPPSVFAEVPQAQPVLVFKLIADFREDPDPRKVNLGVG...
  MAHTLGLTQPNSTEPHKISFTAKEIDVIEWKGDILVVG...
  MSISESYAKEIKTAFRQFTDFPIEGEQFEDFLPIIGNP..
  ...



MAHSKMQN…



-Localization
-Function
-Structure

Other examples

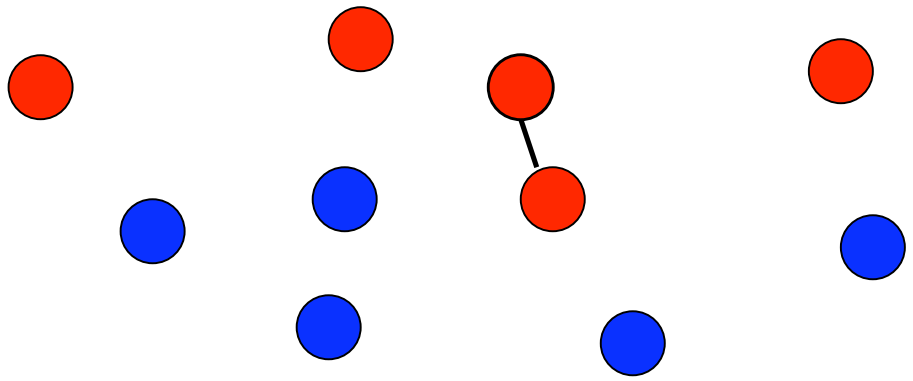
- Predict function from structure
- Predict splicing sites
- Predict binding sites
- Predict regulated genes
- …

Summary

- Patterns X (image/sequence/structure/…)
- Label Y (binary here, but can be more general)
- We want to build a predictor Y=f(X)
- For this we need a training set of (X,Y) pairs
- We need an algorithm that estimates the predictor f from the training set
- We can then use the predictor to make predictions on new patterns X by f(Y)

My first machine learning algorithm: nearest neighbour

- Define a similarity measure s(X,X’) between patterns
- For a new pattern X, predict as label f(X) the label of the most similar pattern in the training set



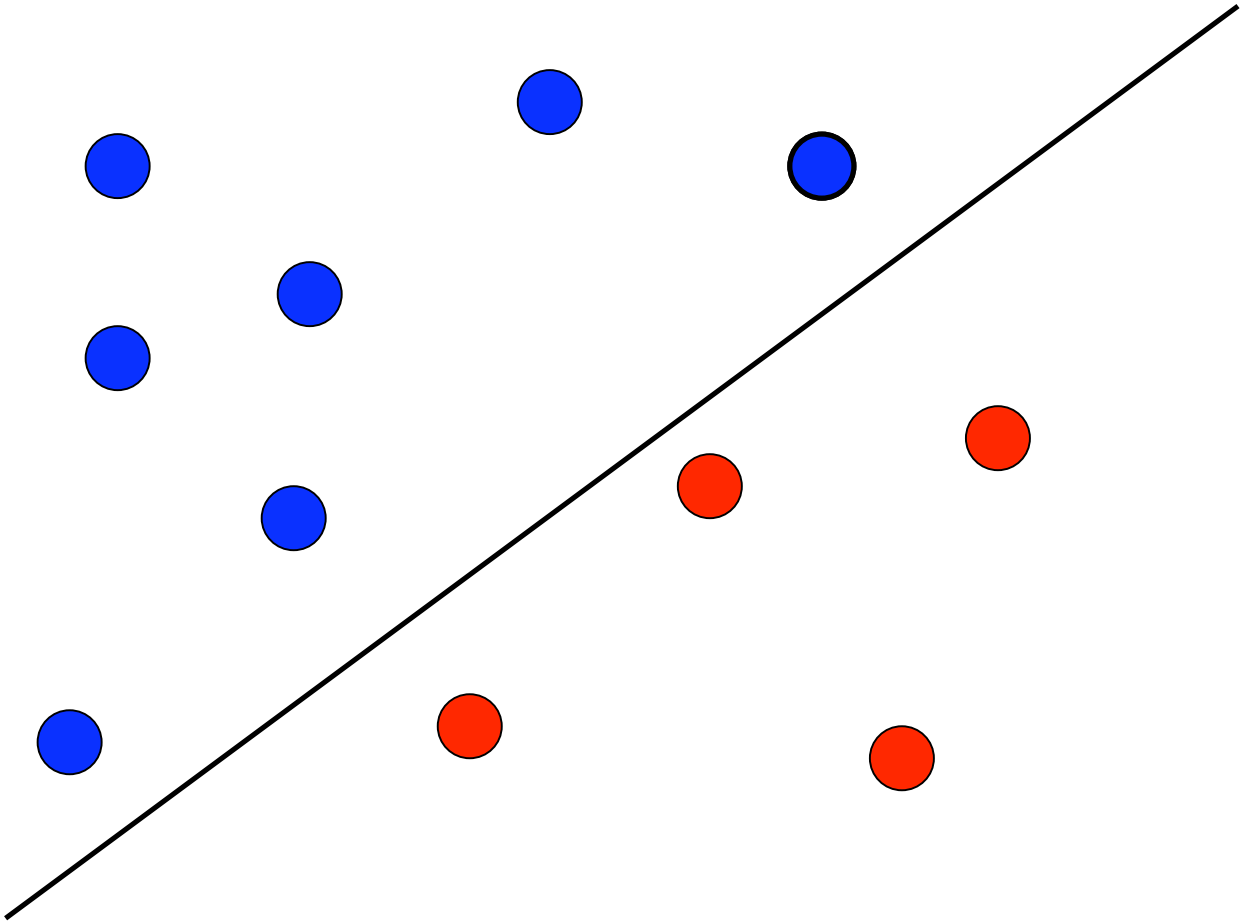
Nearest neighbours

- Very simple to implement
- Good baseline method
- Simple extension: make a majority vote of the k nearest neighbors (k-NN)

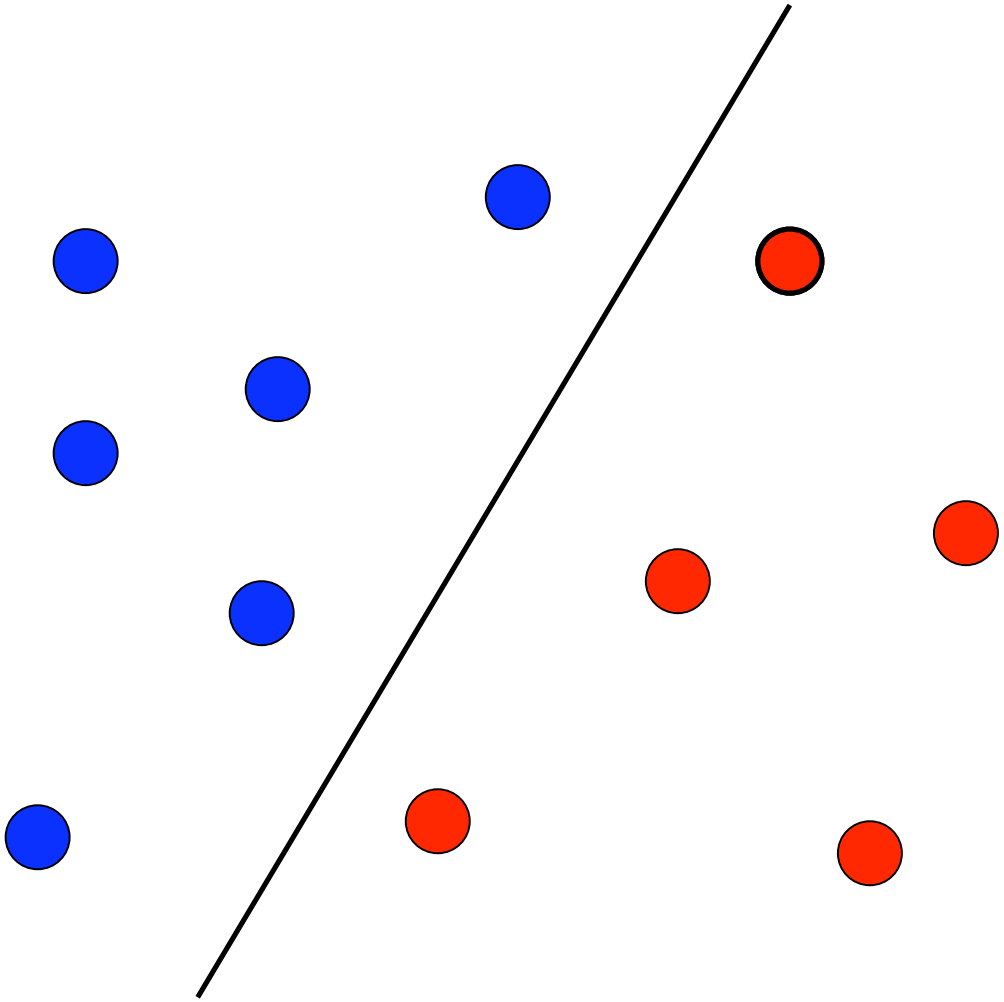
Other popular algorithms

- Decision trees, random forests
- Fisher linear discriminant
- Artificial neural networks (ANN)
- Logistic regression
- Boosting
- Support vector machines (SVM)

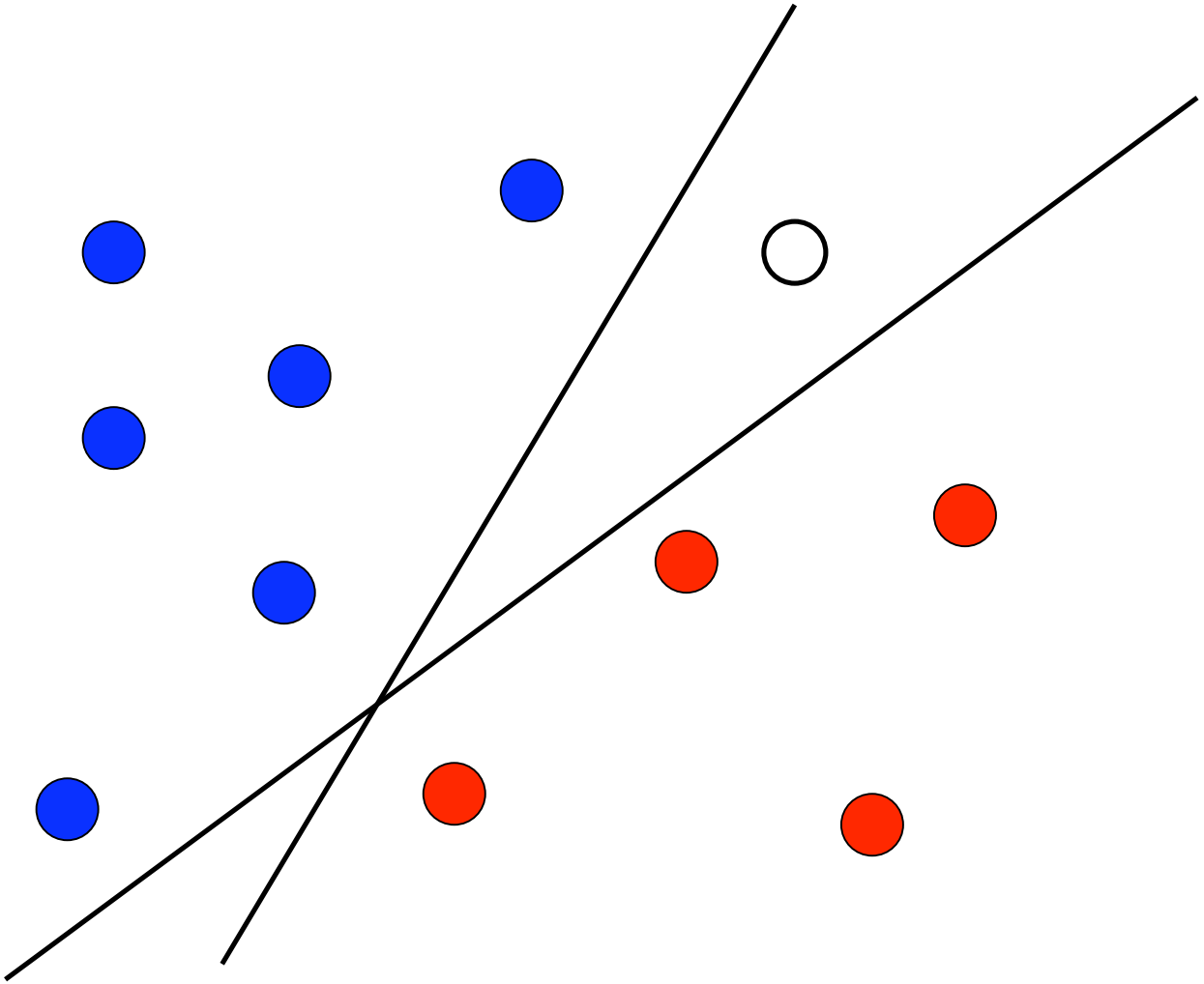
Linear classifier (simple case)



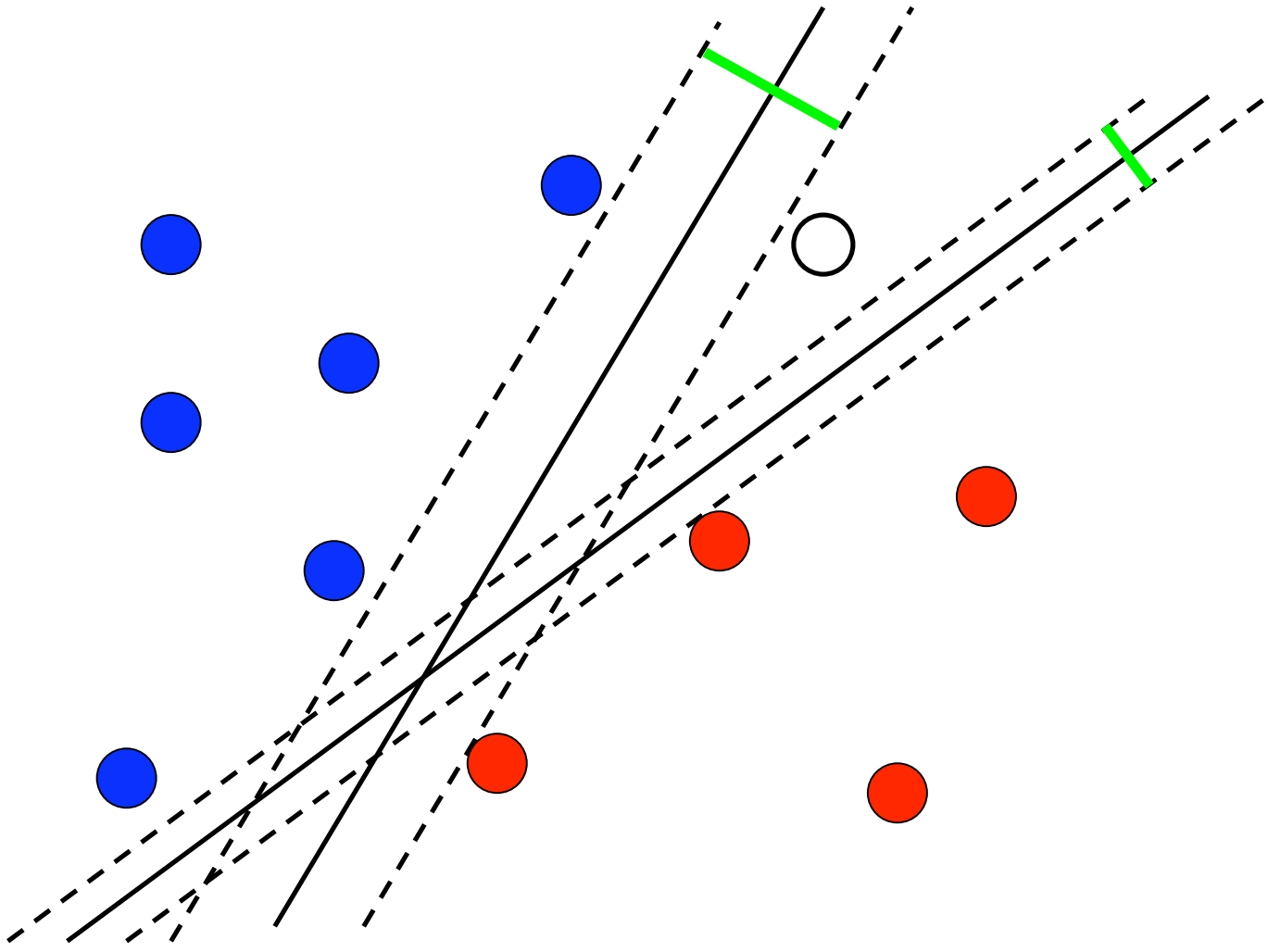
Linear classifier (simple case)



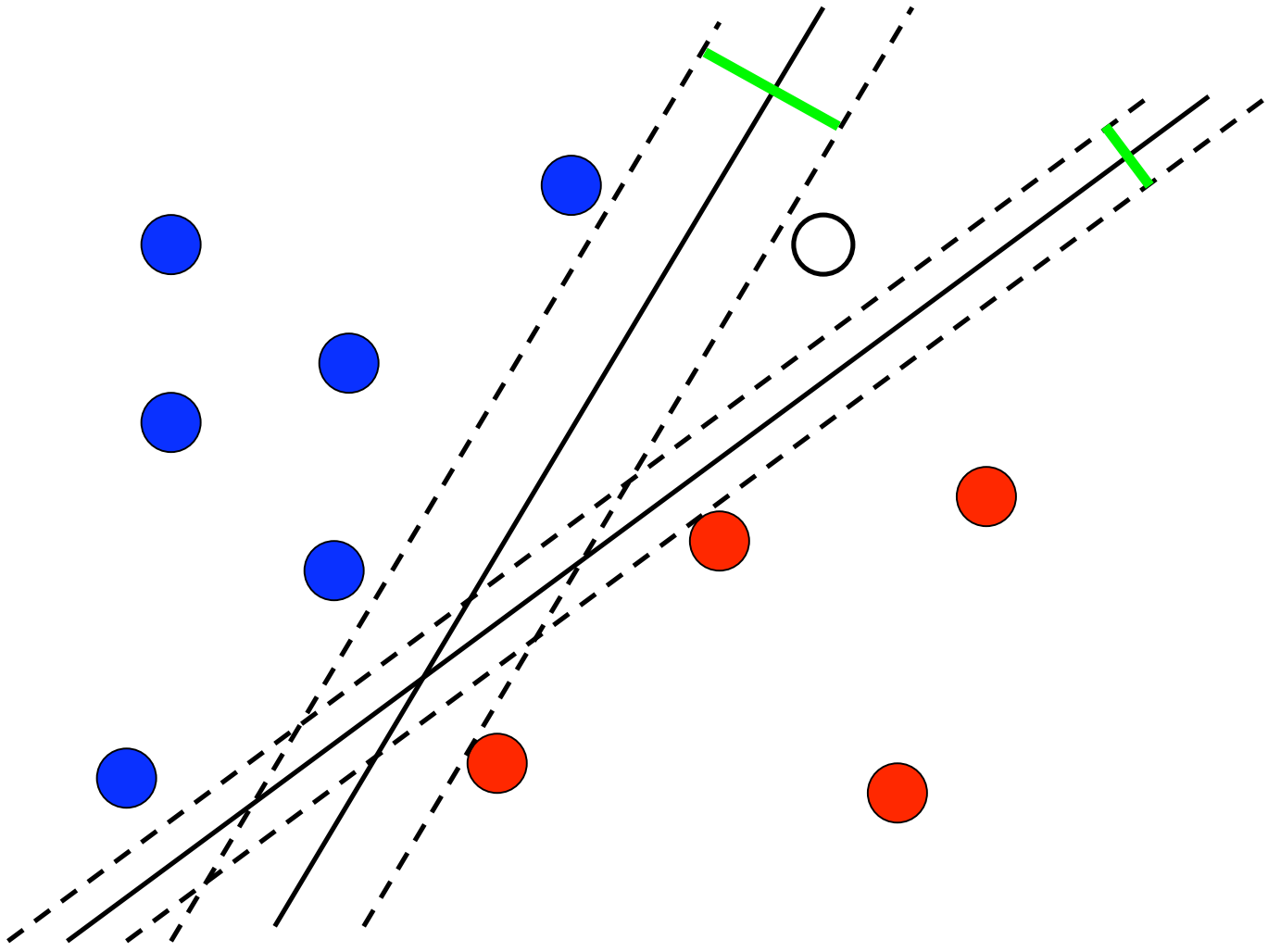
Which is better?



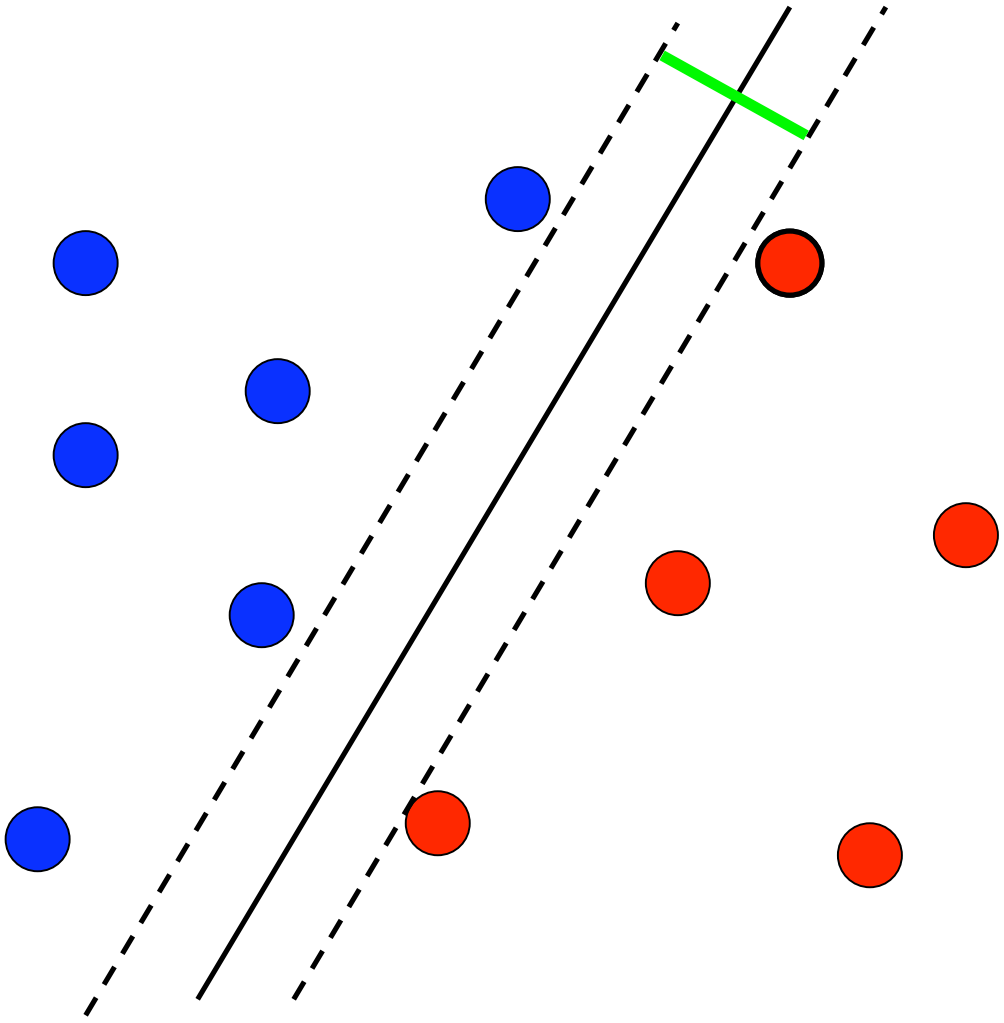
Vapnik's answer: margin



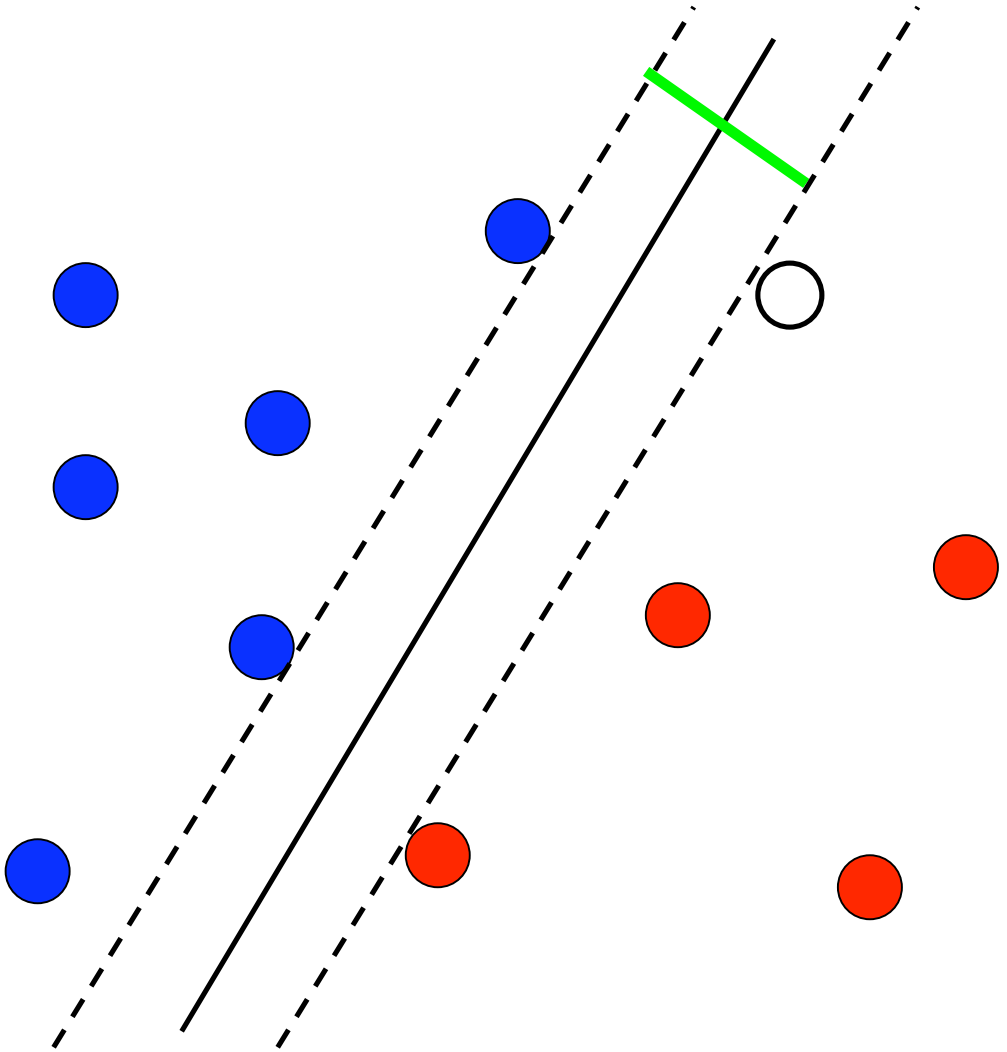
Vapnik's answer: margin



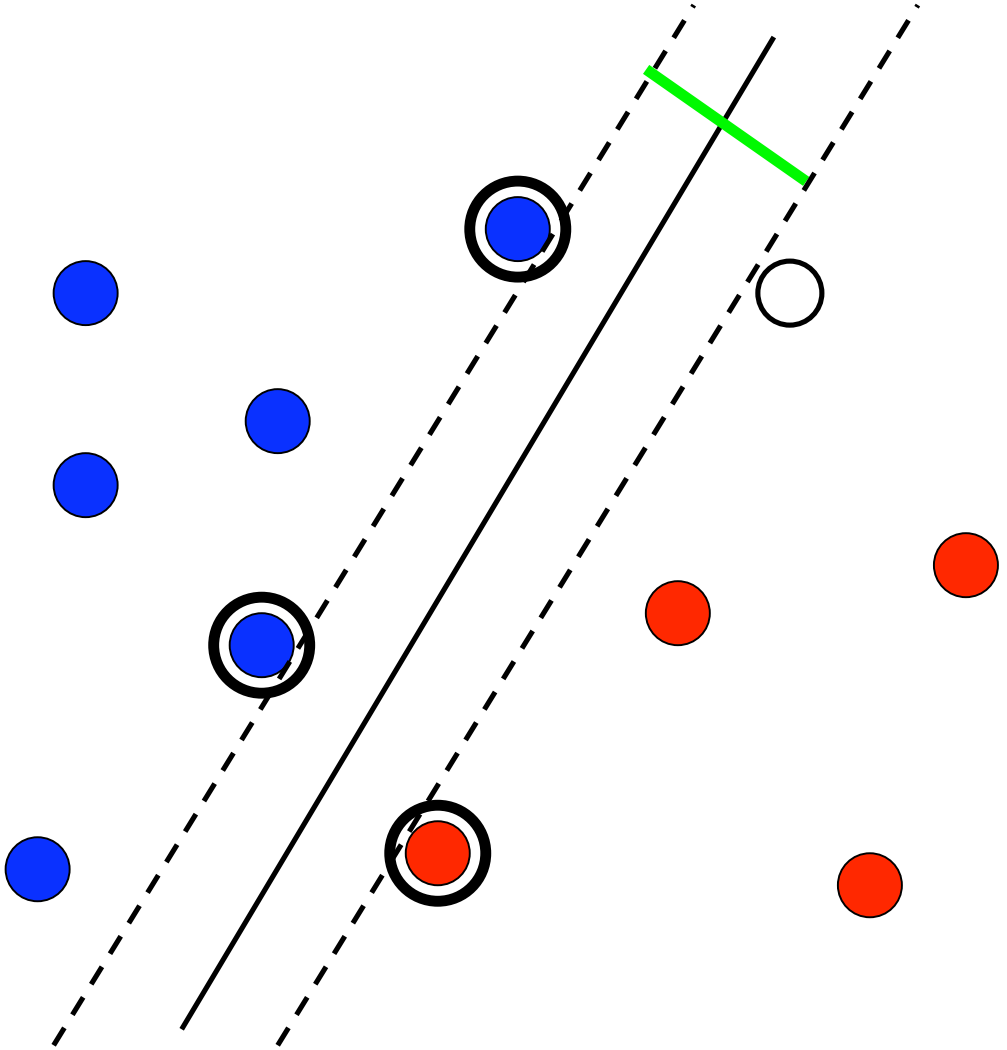
Vapnik's answer: margin



The best: largest margin



Support vectors

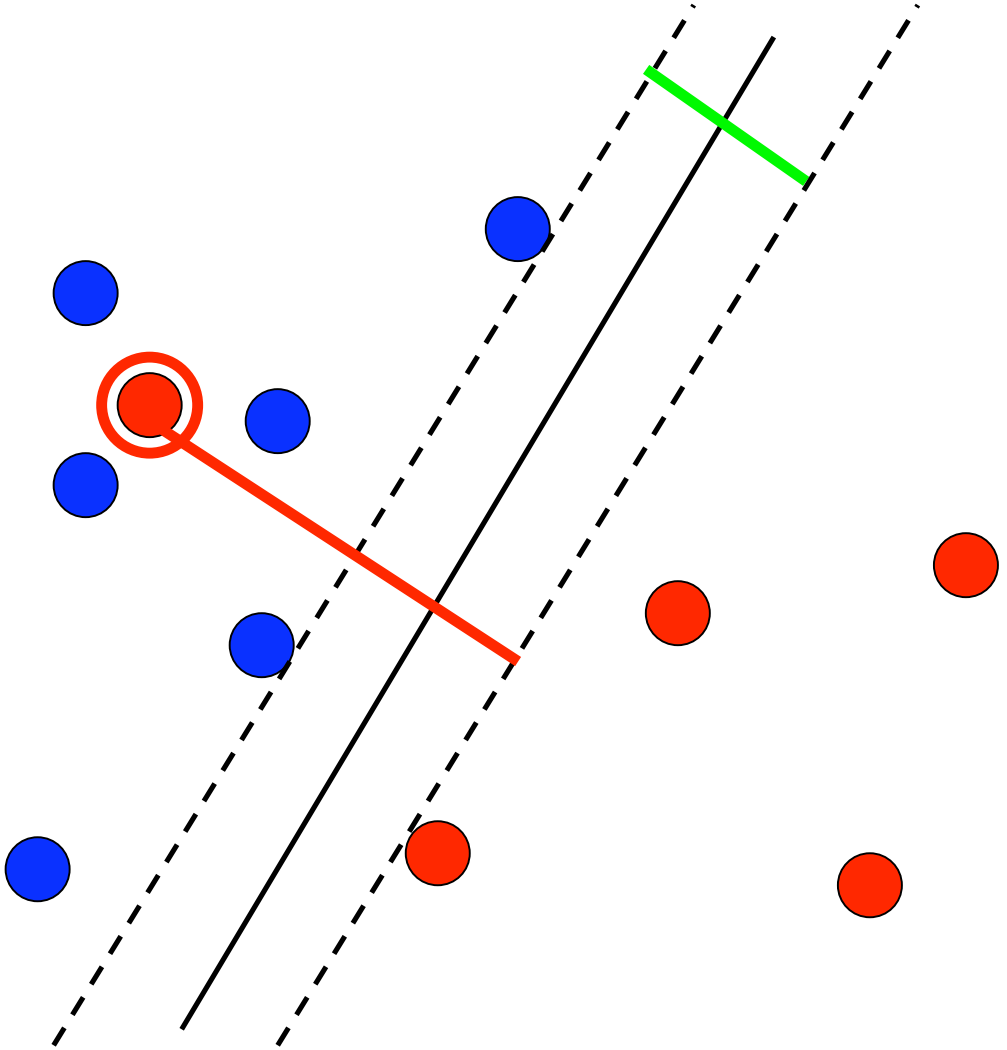


Implementation

$$\max_f \{margin(f)\} \Longleftrightarrow \min_f \left\{ \frac{1}{margin(f)} \right\}$$

- The problème of finding the largest margin hyperplane is easy to solve (but not by yourself!)
- Unique solution, no local optimum (convex optimization problem)
- Only depends on the support vectors

New problem



Soft-margin SVM

- Find a trade-off between:
  - Large margin
  - Few misclassification
- Mathematically:

$$\min_f \left\{ \frac{1}{margin(f)} + C \times error(f) \right\}$$

- Still easy to solve (for a good choice of « error »). C is a parameter.

Some limitations

- What if the data are not vectors?
- What if instead I have a way to measure a distance between patterns (e.g., alignment of sequences, stuctures, …)

An interesting property

- To train a SVM we just need the matrix of pairwise distances:

$$D_{i,j} = ||X_i - X_j||^2$$

- The predictor has the form:

$$f(X) = \sum_{i \in SV} w_i ||X_i - X||^2$$

An interesting generalization

- Take a distance d(X,X')
- Train a SVM from the matrix of pairwise distances:

$$D_{i,j} = d(X_i, X_j)^2$$

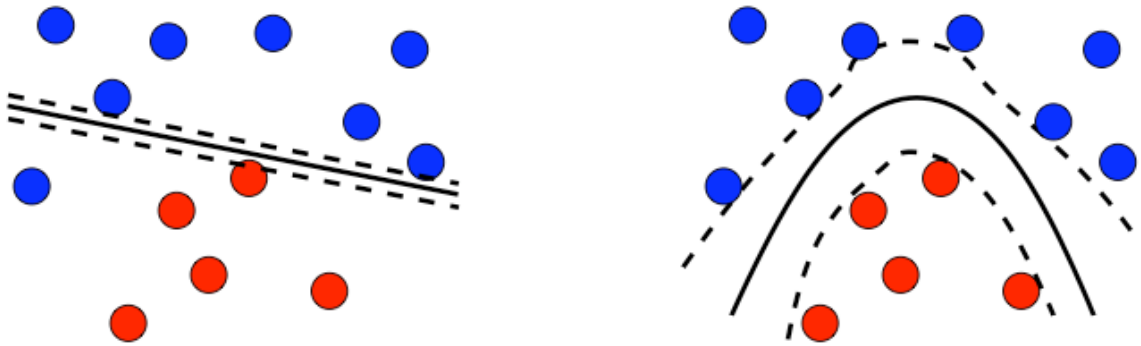
- The predictor now is:

$$f(X) = \sum_{i \in SV} w_i d(X_i, X)^2$$

Technical details

- This will work very well if the distance d(X,X') satisfies some mathematical conditions (« conditionally positive definiteness »)
- If not there still exist tricks to make it work

Example: nonlinear SVM



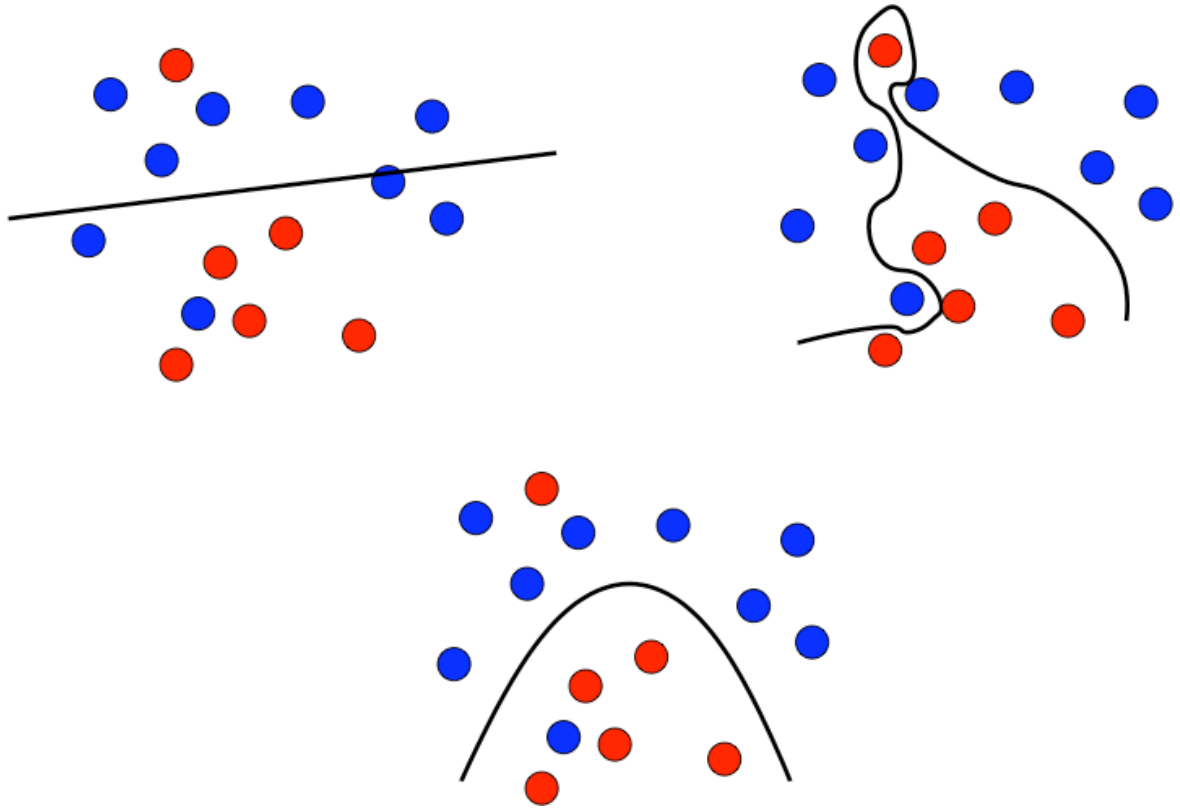
- Take a Gaussian distance:

$$d(X, X')^2 = 1 - \exp\left(-\frac{||X - X'||^2}{2\sigma^2}\right)$$

- We can then learn nonlinear predictors:

$$f(X) = \sum_{i \in SV} w_i \exp\left(-\frac{||X - X_i||^2}{2\sigma^2}\right) + cte$$

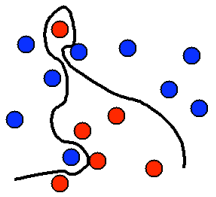
The fundamental trade-off: regularity (margin) vs error



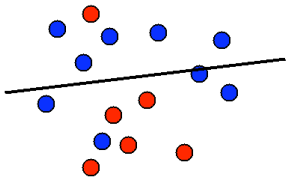
C controls the trade-off

$$\min_f \left\{ \frac{1}{margin(f)} + C \times error(f) \right\}$$

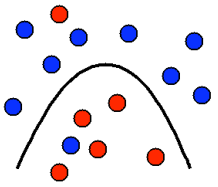
- Large C :
  - makes few errors



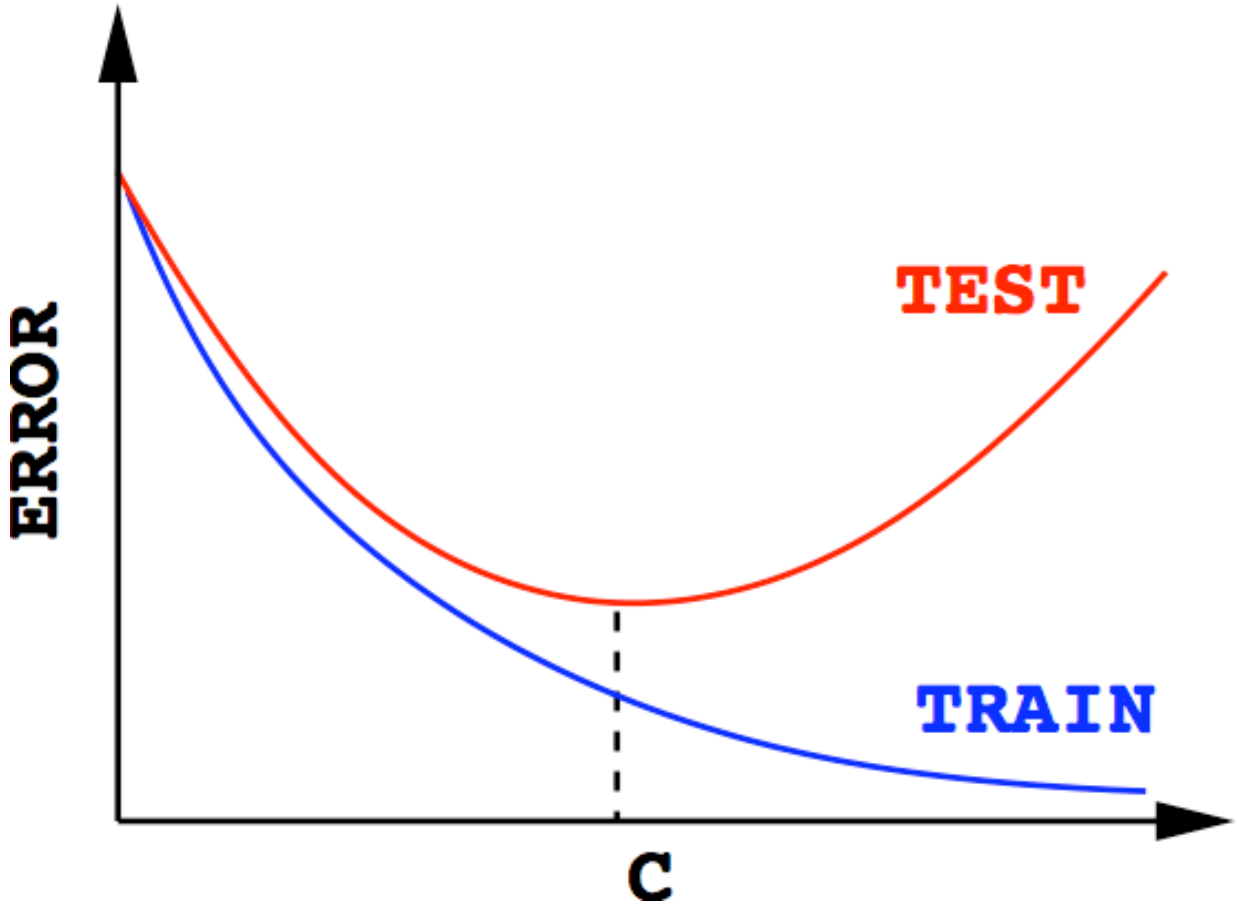
- Small C :
  - ensure a large margin



- Intermediate C:
  - finds a trade-off



Why it is important to care about the trade-off



Choosing C

- Split the annotated data in 2: training / validation
- Train a predictor on the training set
- Evaluate the performance on the validation set
- Choose C to minimize the validation error
- (you may repeat all this several times -> cross-validation)

SVM: summary

- You need a training set of labeled patterns, i.e., of (X,Y) pairs
- You need a distance d(X,X') between patterns
- You need to choose the parameter C (e.g., cross-validation)
- You plug this into any SVM implementation to train a predictor

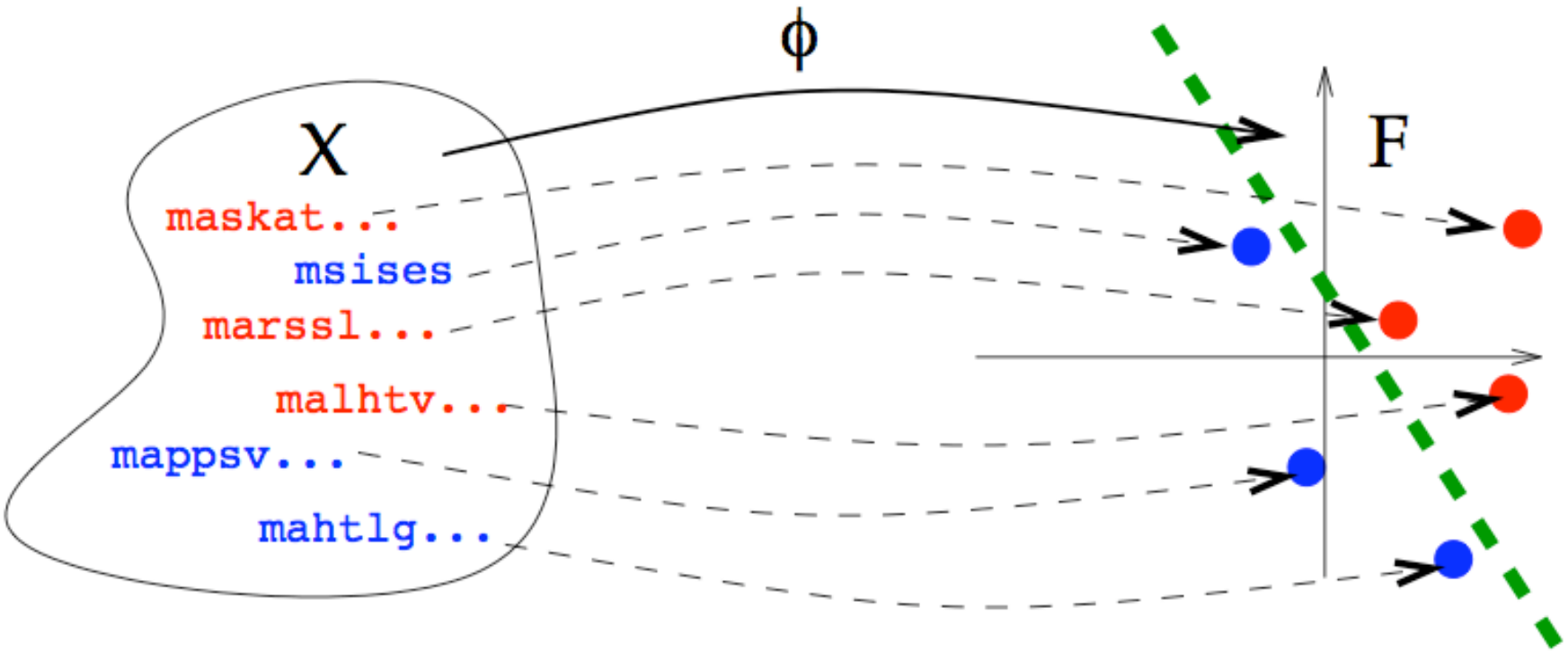
SVM in practice
(eg: libsvm with Python)

```
1> from svm import *
2> param = svm_parameter(kernel_type=LINEAR,C=10)
3> prob = svm_problem([1,-1],[[1,0,1],[-1,0,-1]])
4> m = svm_model(prob, param)
5> r = m.predict([1, 1, 1])
```

How to choose the distance

- The distance is where you can put prior knowledge
- « Similar points have similar labels »
- Strategy 1: define features to represent a pattern as a vector, then use a distance for vectors (Euclidean, Gaussian…)
- Strategy 2: use directly your own distance (but be careful if it is not conditionally positive definite)

Example: sequence classification



The spectrum kernel

- Features are the number of occurrences of each k-mer
- Example: X=AATCGCAA
- For k=2: [AA:2, AC:0, AG:0, AT:1, CA:1, CC:0, CG:1, CT:0, GA:0, GC:1, GG:0, GT:0, TA:0, TC:1, TG:0, TT:0]
- Fast (tricks…) and good baseline

Motif kernel

General idea

- Conserved motif in sequences indicate structural and functional characteristics
- Model sequence as feature vector representing motifs
- i-th vector component is 1 $\Leftrightarrow$ $\mathbf{x}$ contains i-th motif

Motif databases

- Protein: Pfam, PROSITE, ...
- DNA: Transfac, Jaspar, ...
- RNA: Rfam, Structures, Regulatory sequences, ...

Generated by

- manual construction/prior knowledge
- multiple sequence alignment (do not use test set!)

Local alignment kernel

$\mathbf{x}_1$ = CGGSLIAMMWFGV

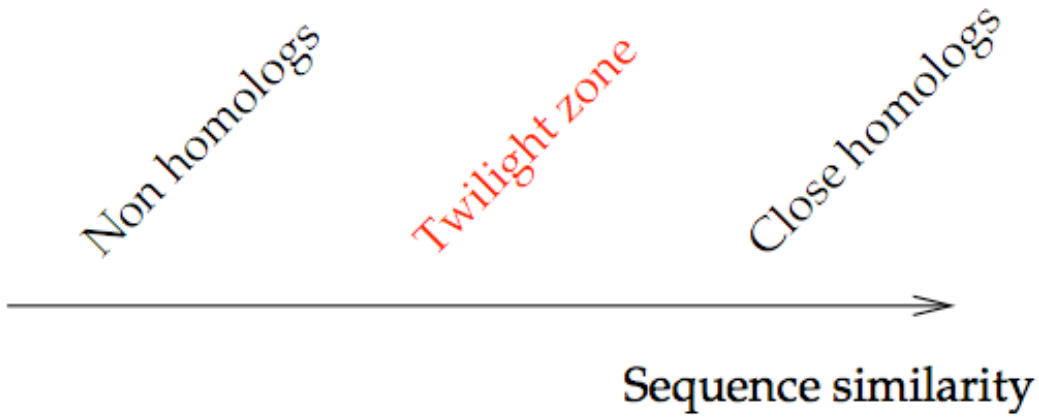
$\mathbf{x}_2$ = CLIVMMNRLMWFGV

```
CGGSLIAMM----WFGV
|...|||||....||||
C---LIVMMNRLMWFGV
```

$$s_{S,g}(\pi) = S(C,C) + S(L,L) + S(I,I) + S(A,V) + 2S(M,M) + S(W,W) + S(F,F) + S(G,G) + S(V,V) - g(3) - g(4)$$

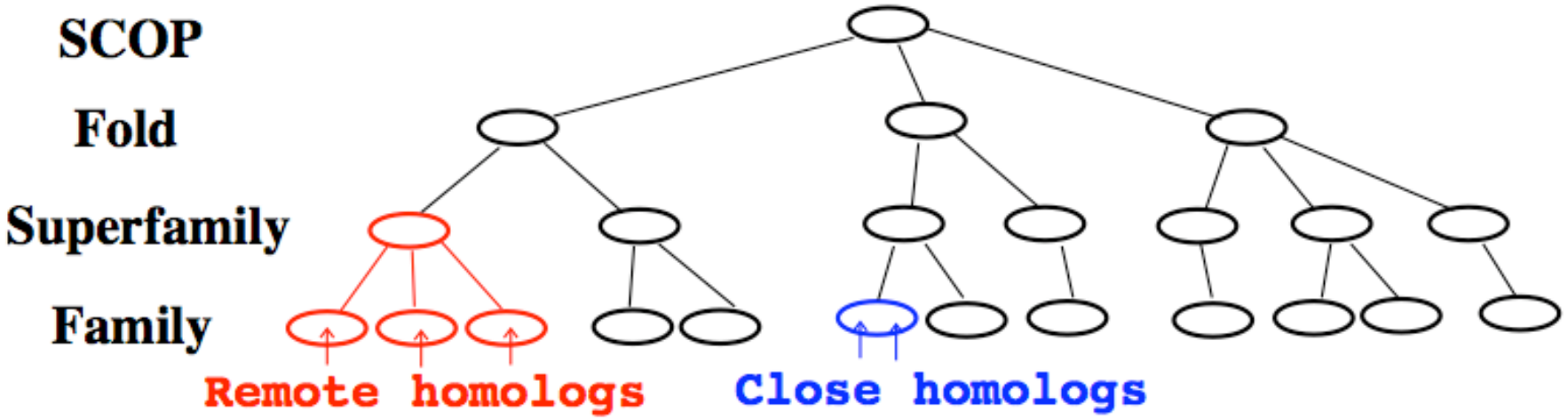
$$K_{LA}^{(\beta)}(\mathbf{x},\mathbf{y}) = \sum_{\pi \in \Pi(\mathbf{x},\mathbf{y})} \exp\left(\beta s_{S,g}(\mathbf{x},\mathbf{y},\pi)\right)$$

Application: remote homology detection



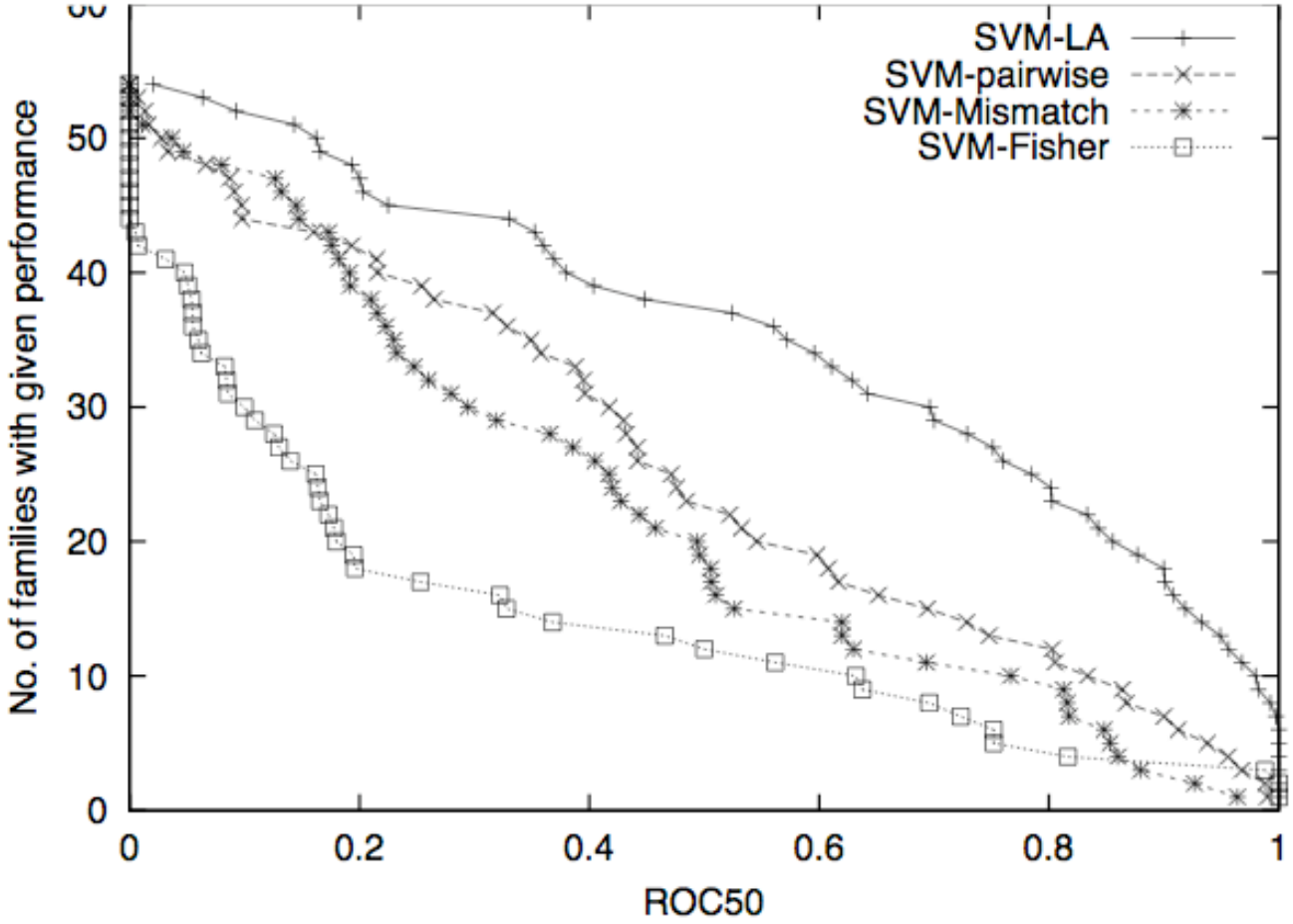
- Homologs have common ancestors
- Structures and functions are more conserved than sequences
- Remote homologs can not be detected by direct sequence comparison

SCOP hierarchy



- Goal: recognize directly the superfamily
- Training: for a sequence of interest, positive examples come from the same superfamily, but different families. Negative from other superfamilies.
- Test: predict the superfamily.

Performance of different distances



Take-home messages

- Machine learning relevant in bioinformatics
- Must find a trade-off between fitting the training set and controlling the capacity
- SVM is easy to use
- Prior knowledge can be put in the kernel/distance