Statistical learning theory, Support vector machines, and Bioinformatics

Jean-Philippe.Vert@mines.org

Ecole des Mines de Paris
Computational Biology group

ENS Paris, november 25, 2003.

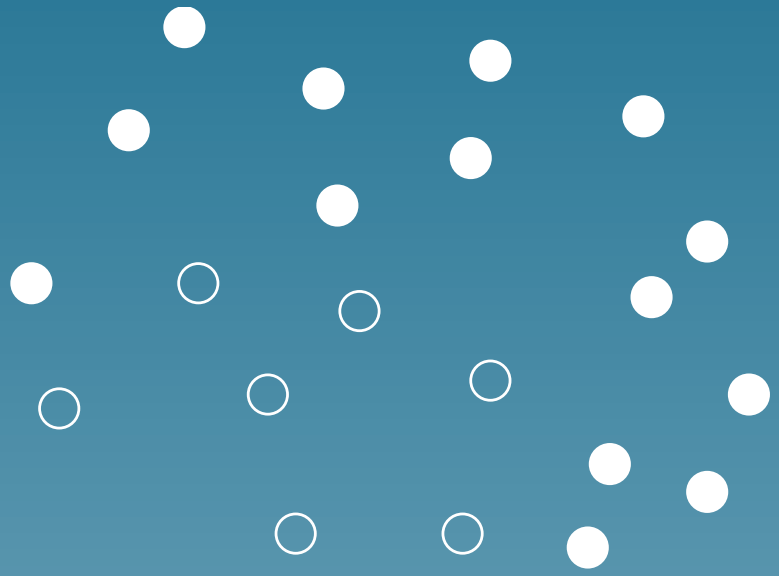
Overview

1. Statistical learning theory
2. Support vector machines
3. Computational biology
4. Short example: virtual screening for drug design

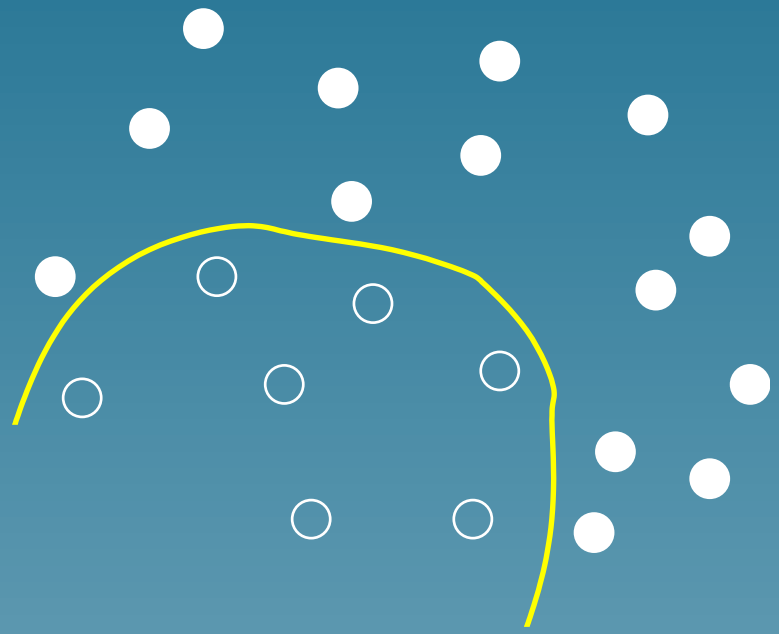
Partie 1

Statistical learning theory

The pattern recognition problem

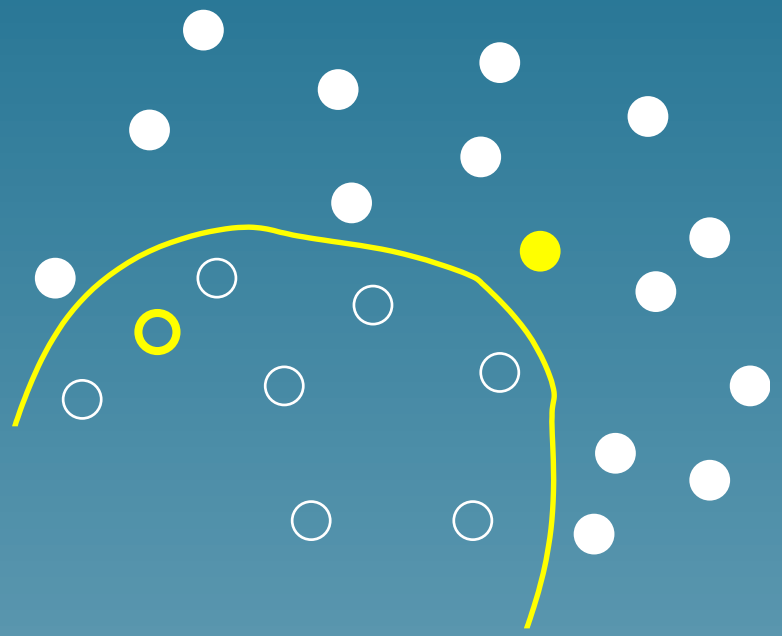


The pattern recognition problem



- Learn from labelled examples a discrimination rule

The pattern recognition problem



- Learn from labelled examples a discrimination rule
- Use it to predict the class of new points

Pattern recognition applications

- Multimedia (OCR, speech recognition, spam filter,..)
- Finance, Marketing
- Bioinformatics, medical diagnosis, drug design,...

Formalism

- Object: $x \in \mathcal{X}$

Formalism

- Object: $x \in \mathcal{X}$
- Label: $y \in \mathcal{Y}$ (e.g., $\mathcal{Y} = \{-1, 1\}$)

Formalism

- Object: $x \in \mathcal{X}$
- Label: $y \in \mathcal{Y}$ (e.g., $\mathcal{Y} = \{-1, 1\}$)
- Training set: $S = (z_1, \ldots, z_N)$ with $z_i = (x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$

Formalism

- Object: $x \in \mathcal{X}$
- Label: $y \in \mathcal{Y}$ (e.g., $\mathcal{Y} = \{-1, 1\}$)
- Training set: $S = (z_1, \ldots, z_N)$ with $z_i = (x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$
- A classifier is any $f : \mathcal{X} \to \mathcal{Y}$
- A learning algorithm is:
  - ⋆ a set of classifiers $\mathcal{F} \subset \mathcal{Y}^{\mathcal{X}}$
  - ⋆ a learning procedure: $S \in (\mathcal{X} \times \mathcal{Y})^N \to f \in \mathcal{F}$

Example: linear disrimination

- Objects: $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{-1, 1\}$
- Classifiers: $\mathcal{F} = \left\{ f_{w,b} : (w, b) \in \mathbb{R}^d \times \mathbb{R} \right\}$, where

$$f_{w,b}(x) = \begin{cases} 1 & \text{if } w.x + b > 0, \\ -1 & \text{if } w.x + b \leq 0, \end{cases}$$

- Algorithm: linear perceptron, Fisher discriminant, ...

Questions

- How to analyse/understand learning algorithms?
- How to design "good" learning algorithms?

Useful hypothesis: probabilistic setting

- Learning is only possible if the future is related to the past

Useful hypothesis: probabilistic setting

- Learning is only possible if the future is related to the past
- Mathematically: $\mathcal{X} \times \mathcal{Y}$ is a measurable set endowed with a probability measure P

Useful hypothesis: probabilistic setting

- Learning is only possible if the future is related to the past
- Mathematically: $\mathcal{X} \times \mathcal{Y}$ is a measurable set endowed with a probability measure P
- Past observations: $Z_1, \ldots, Z_N$ are N independent and identically distributed (according to P) random variables

Useful hypothesis: probabilistic setting

- Learning is only possible if the future is related to the past
- Mathematically: $\mathcal{X} \times \mathcal{Y}$ is a measurable set endowed with a probability measure P
- Past observations: $Z_1, \ldots, Z_N$ are N independent and identically distributed (according to P) random variables
- Future observations: a random variable Z_{N+1} also distributed according to P

Useful hypothesis: probabilistic setting

- Learning is only possible if the future is related to the past
- Mathematically: $\mathcal{X} \times \mathcal{Y}$ is a measurable set endowed with a probability measure P
- Past observations: $Z_1, \ldots, Z_N$ are N independent and identically distributed (according to P) random variables
- Future observations: a random variable Z_{N+1} also distributed according to P
- The future is related to the past by P

How good is a classifier?

- Let $l : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ a loss function (e.g., $0/1$ loss)

How good is a classifier?

- Let $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ a loss function (e.g., $0/1$ loss)
- The risk of a classifier $f : \mathcal{X} \to \mathcal{Y}$ is the average loss:

$$R(f) = E_{(X,Y)\sim P}\left[l(f(X), Y)\right]$$

How good is a classifier?

- Let $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ a loss function (e.g., $0/1$ loss)
- The risk of a classifier $f : \mathcal{X} \to \mathcal{Y}$ is the average loss:

$$R(f) = E_{(X,Y)\sim P}\left[l(f(X), Y)\right]$$

- Ideal goal: for a given (unknown) P, find

$$f^* = \arg\inf_f R(f)$$

How to learn a good classifier?

- P is unknown, we must learn a classifier f from the training set S.

How to learn a good classifier?

- P is unknown, we must learn a classifier f from the training set S.
- For any classifier $f : \mathcal{X} \to \mathcal{Y}$, we can compute the empirical risk:

$$R_N(f) = \frac{1}{N} \sum_{i=1}^{N} l(f(x_i), y_i)$$

- Obviously, $R(f) = E_{S \sim P}\left[R_N(f)\right]$ for any $f \in \mathcal{F}$

Empirical risk minimization

- For a given class $\mathcal{F} \subset \mathcal{Y}^{\mathcal{X}}$, chose f that minimizes the empirical risk:

$$\hat{f}_N = \arg\min_{f \in \mathcal{F}} R_N(f)$$

Empirical risk minimization

- For a given class $\mathcal{F} \subset \mathcal{Y}^{\mathcal{X}}$, chose f that minimizes the empirical risk:

$$\hat{f}_N = \arg\min_{f \in \mathcal{F}} R_N(f)$$

- The best choice, if P was known, would be

$$f_0 = \arg\min_{f \in \mathcal{F}} R(f)$$

Empirical risk minimization

- For a given class $\mathcal{F} \subset \mathcal{Y}^{\mathcal{X}}$, chose f that minimizes the empirical risk:

$$\hat{f}_N = \arg\min_{f \in \mathcal{F}} R_N(f)$$

- The best choice, if P was known, would be

$$f_0 = \arg\min_{f \in \mathcal{F}} R(f)$$

- Central question: is $R(\hat{f}_N)$ close to $R(f_0)$?

Consistency issue

- An algorithm is called consistent iff

$$\lim_{N\to\infty} R(\hat{f_N}) = R(f_0)$$

- $R(\hat{f_N})$ is random, so we need a notion of convergence for random variable, such as convergence in probability:

$$\lim_{N\to\infty} P\left\{|R(\hat{f_N}) - R(f_0)| > \epsilon\right\} = 0, \forall\epsilon > 0.$$

Consistency and law of large numbers

- Classical law of large numbers: for any $f \in \mathcal{F}$,

$$\lim_{N \to \infty} P\left\{|R_N(f) - R(f)| > \epsilon\right\} = 0, \forall \epsilon > 0.$$

Consistency and law of large numbers

- Classical law of large numbers: for any $f \in \mathcal{F}$,

$$\lim_{N \to \infty} P\left\{|R_N(f) - R(f)| > \epsilon\right\} = 0, \forall \epsilon > 0.$$

- $0 \leq R(\hat{f_N}) - R(f_0) \leq \left[R(\hat{f_N}) - R_N(\hat{f_N})\right] + \left[R_N(f_0) - R(f_0)\right]$

- The second term converges to 0 by classical LLN applied to f_0.

- What about the first one?

Uniform law of large numbers

- Classical LLN is not enough to ensure that:

$$R(\hat{f}_N) - R_N(\hat{f}_N) \underset{N \to \infty}{\to} 0$$

Uniform law of large numbers

- Classical LLN is not enough to ensure that:

$$R(\hat{f}_N) - R_N(\hat{f}_N) \underset{N\to\infty}{\to} 0$$

- Theorem: the ERM principle is consistent if and only if the following uniform law of large numbers holds:

$$\lim_{N\to\infty} P\left\{\sup_{f\in\mathcal{F}} (R(f) - R_N(f)) > \epsilon\right\} = 0, \forall\epsilon > 0.$$

Vapnik-Chervonenkis entropy and consistency

- For any N, S and $\mathcal{F}$, let

$$G_{\mathcal{F}}(N, S) = card\{(f(x_1), \ldots, f(x_N)) : f \in \mathcal{F}\}$$

- Obviously, $G_{\mathcal{F}}(N, S) \leq 2^N$

Vapnik-Chervonenkis entropy and consistency

- For any N, S and $\mathcal{F}$, let

$$G_{\mathcal{F}}(N,S) = card\left\{(f(x_1),\ldots,f(x_N)) : f \in \mathcal{F}\right\}$$

- Obviously, $G_{\mathcal{F}}(N,S) \leq 2^N$
- Theorem: the ULLN holds if and only if:

$$\lim_{N\to\infty} \frac{E_{S\sim P}\left[\ln G_{\mathcal{F}}(N,S)\right]}{N} = 0.$$

Distribution-independent bound

- Theorem (Vapnik-Chervonenkis): For any $\delta > 0$, the following holds with P-probability at least $1 - \delta$:

$$\forall f \in \mathcal{F}, R(f) \leq R_N(f) + \sqrt{\frac{\ln \sup_S G_{\mathcal{F}}(N, S) + \ln 1/\delta}{8N}}.$$

Distribution-independent bound

- Theorem (Vapnik-Chervonenkis): For any $\delta > 0$, the following holds with P-probability at least $1 - \delta$:

$$\forall f \in \mathcal{F}, R(f) \leq R_N(f) + \sqrt{\frac{\ln \sup_S G_{\mathcal{F}}(N, S) + \ln 1/\delta}{8N}}.$$

- This is valid for any P!
- A sufficient condition for distribution-independent fast ULLN is:

$$\lim_{N \to \infty} \frac{\ln \sup_S G_{\mathcal{F}}(N, S)}{N} = 0$$

VC dimension

- The VC dimension of $\mathcal{F}$ is the largest number of points that can be shattered by $\mathcal{F}$, i.e., such that there exists a set S with

$$G_{\mathcal{F}}(N, S) = 2^N$$

VC dimension

- The VC dimension of $\mathcal{F}$ is the largest number of points that can be shattered by $\mathcal{F}$, i.e., such that there exists a set S with

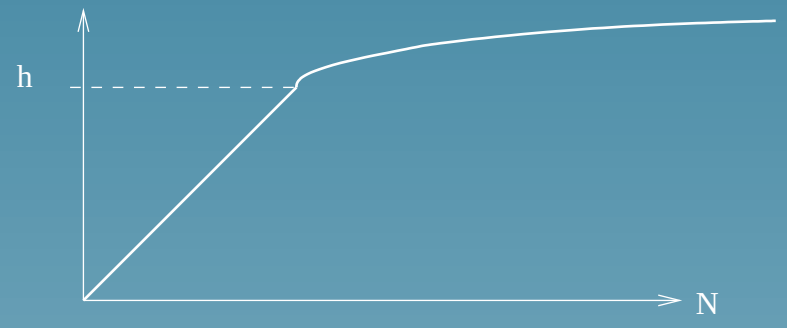
$$G_{\mathcal{F}}(N, S) = 2^N$$

- Example: for hyperplanes in $\mathbb{R}^d$, the VC dimension is $d + 1$

Sauer lemma

Let $\mathcal{F}$ be a set with finite VC dimension h. Then

$$\ln \sup_S G_{\mathcal{F}}(N, S) \begin{cases} = N & \text{if } N \leq h, \\ \leq h \ln \frac{eN}{h} & \text{if } N \geq h \end{cases}$$



VC dimension and learning

- Finiteness of the VC-dimension is a necessary and sufficient condition for uniform convergence independant of P

VC dimension and learning

- Finiteness of the VC-dimension is a necessary and sufficient condition for uniform convergence independant of P
- If $\mathcal{F}$ has finite CV dimension h, then the following holds with probability at least $1 - \delta$:

$$\forall f \in \mathcal{F}, R(f) \leq R_N(f) + \sqrt{\frac{h \ln \frac{2eN}{h} + \ln \frac{4}{\delta}}{8N}}.$$

Partie 2

Support vector machines

Structural risk minimization

- Define a nested family of function sets:

$$\mathcal{F}_1 \subset \mathcal{F}_2 \subset \ldots \subset \mathcal{Y}^{\mathcal{X}}$$

  with increasing VC dimensions:

$$h_1 \leq h_2 \leq \ldots$$

- In each class, the ERM algorithm finds a classifier $\hat{f}_i$ that satisfies:

$$R(\hat{f}_i) \leq \inf_{f \in \mathcal{F}_i} R_N(f) + \sqrt{\frac{h_i \ln \frac{2eN}{h_i} + \ln \frac{4}{\delta}}{8N}}.$$

Structural risk minimization (2)

- SRM principle: choose $\hat{f} = \hat{f}_i$ that minimizes the upper bound

Structural risk minimization (2)

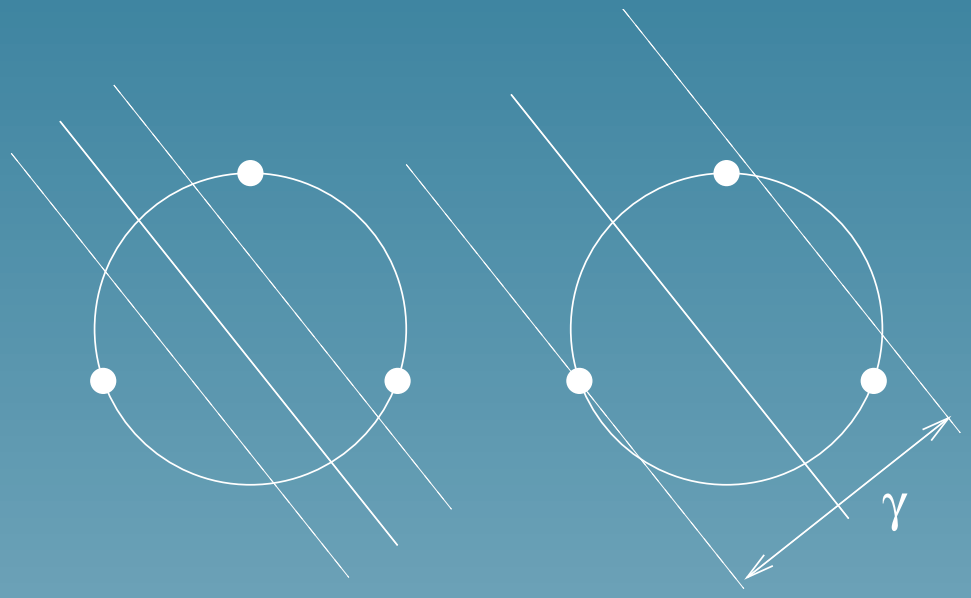
- SRM principle: choose $\hat{f} = \hat{f}_i$ that minimizes the upper bound
- The validity of this prinple can also be justified mathematically

Curse of dimension?

- Remember the VC dim of the class of hyperplanes in $\mathbb{R}^d$ is $d+1$
- Can not learn in large dimension?

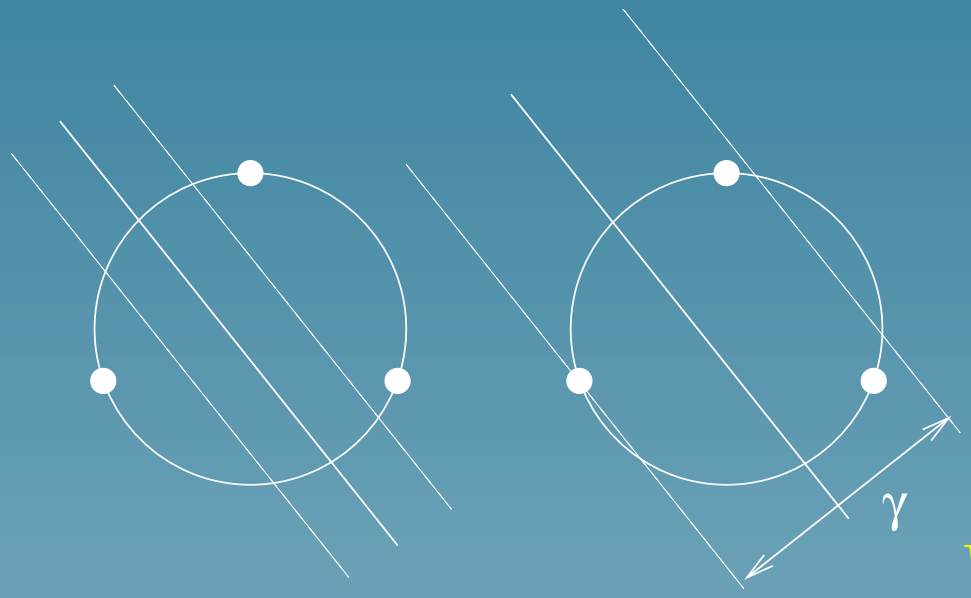
Large margin hyperplanes

For a given set of points S in a ball of radius R, consider only the hyperplanes $\mathcal{F}_\gamma$ that correctly separate the points with margin at least γ:



Large margin hyperplanes

For a given set of points S in a ball of radius R, consider only the hyperplanes $\mathcal{F}_\gamma$ that correctly separate the points with margin at least γ:



$$VC(\mathcal{F}_\gamma) \leq \frac{R^2}{\gamma^2}$$

(independent of the dimension!)

SRM on hyperplanes

Intuitively, select an hyperplane with:

- small empirical risk
- large margin

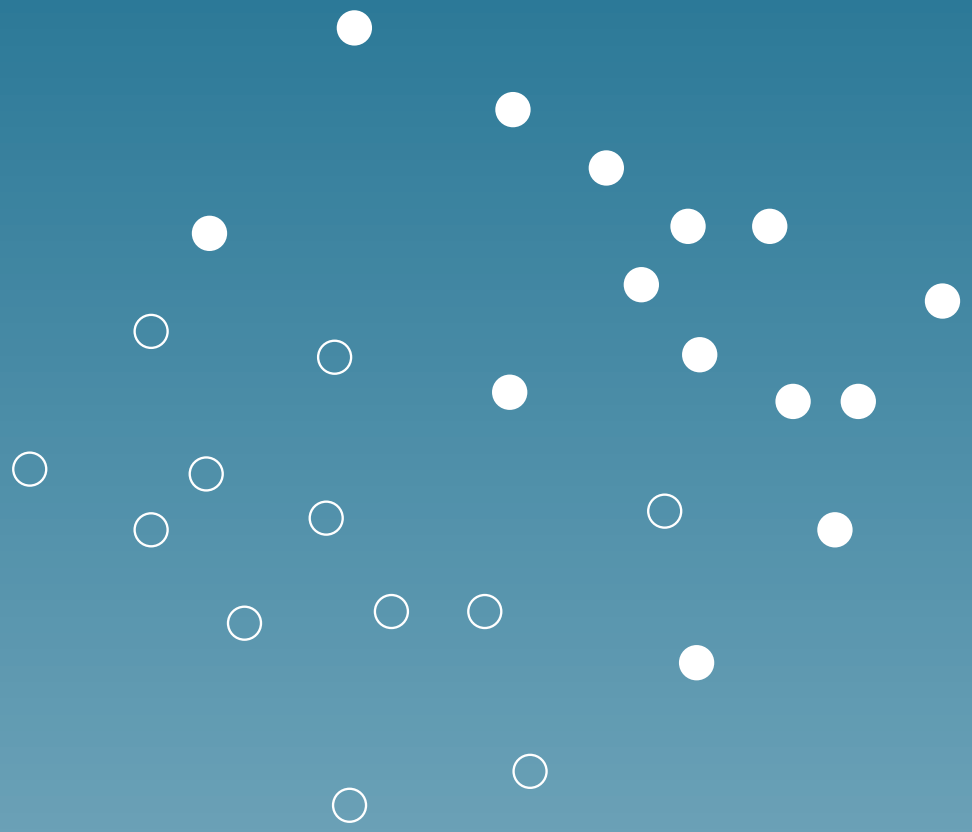
SRM on hyperplanes

Intuitively, select an hyperplane with:

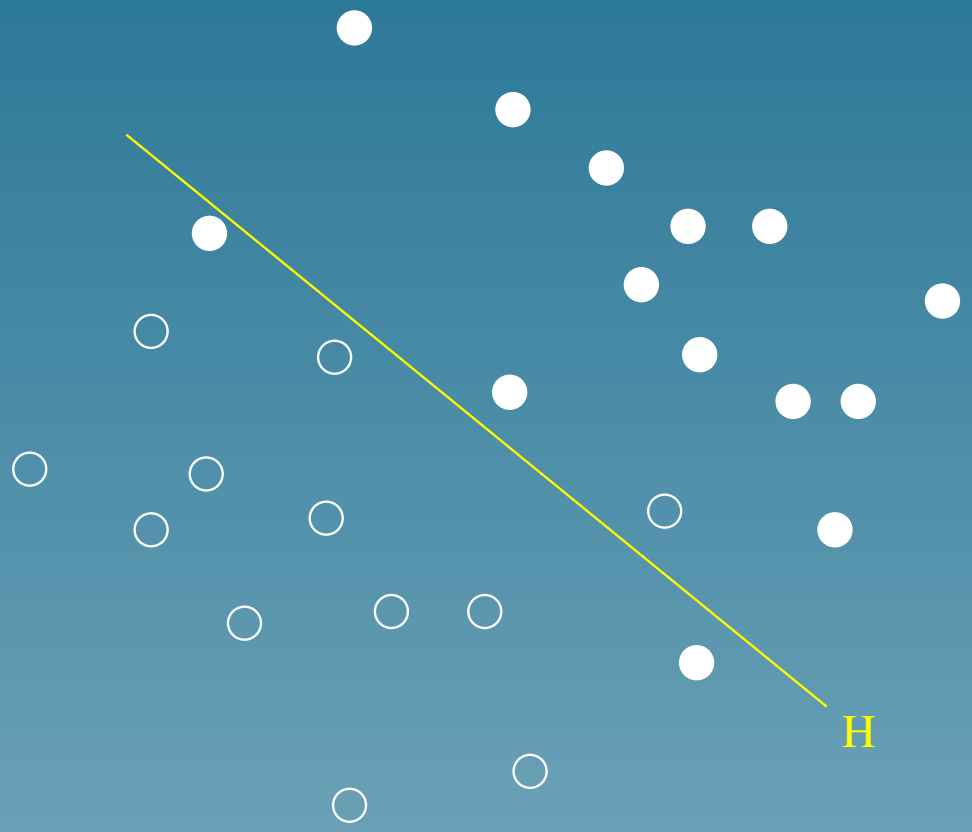
- small empirical risk
- large margin

Support vector machines implement this principle.

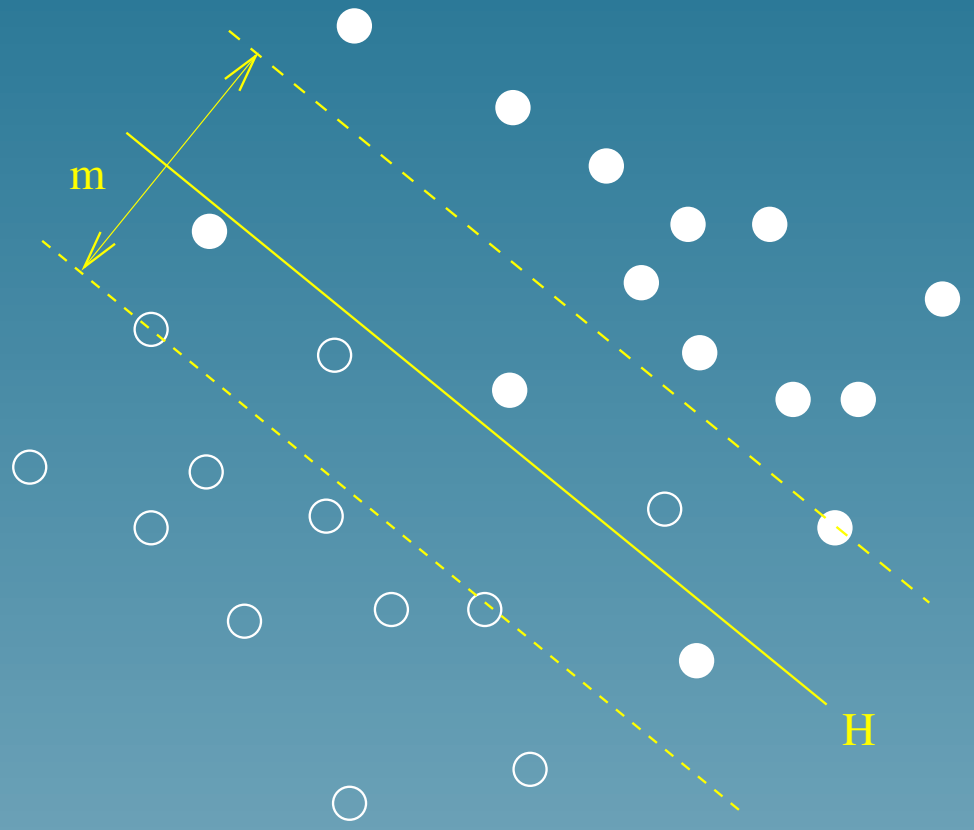
Linear SVM



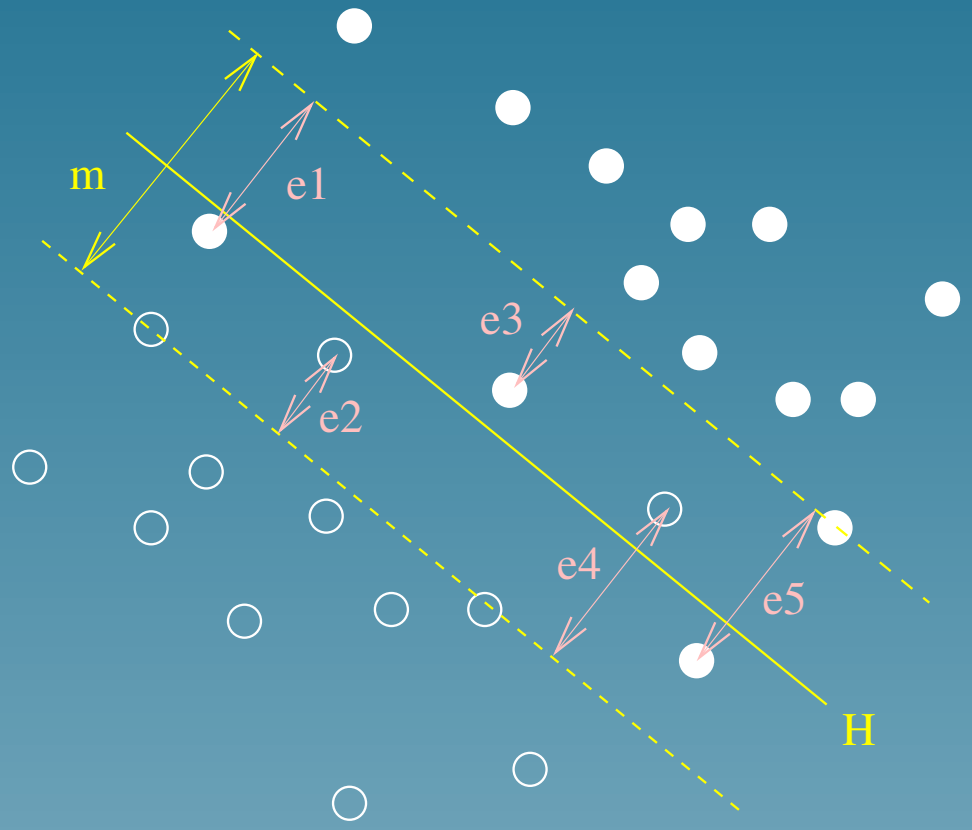
Linear SVM



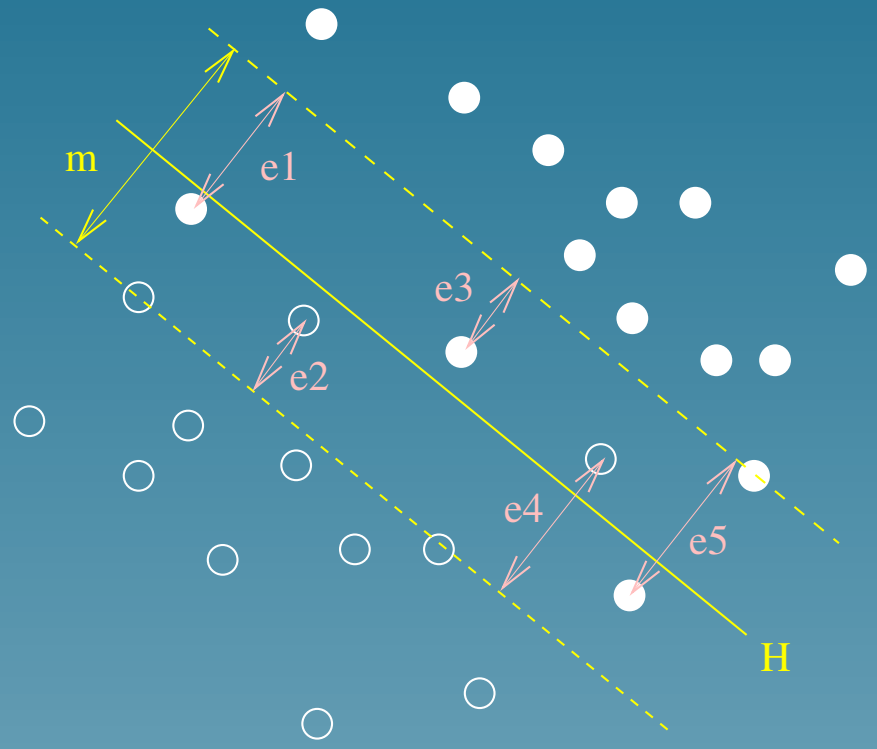
Linear SVM



Linear SVM



Linear SVM



$$\min_{H,m} \left\{ \frac{1}{m^2} + C \sum_i e_i \right\}$$

Dual formulation

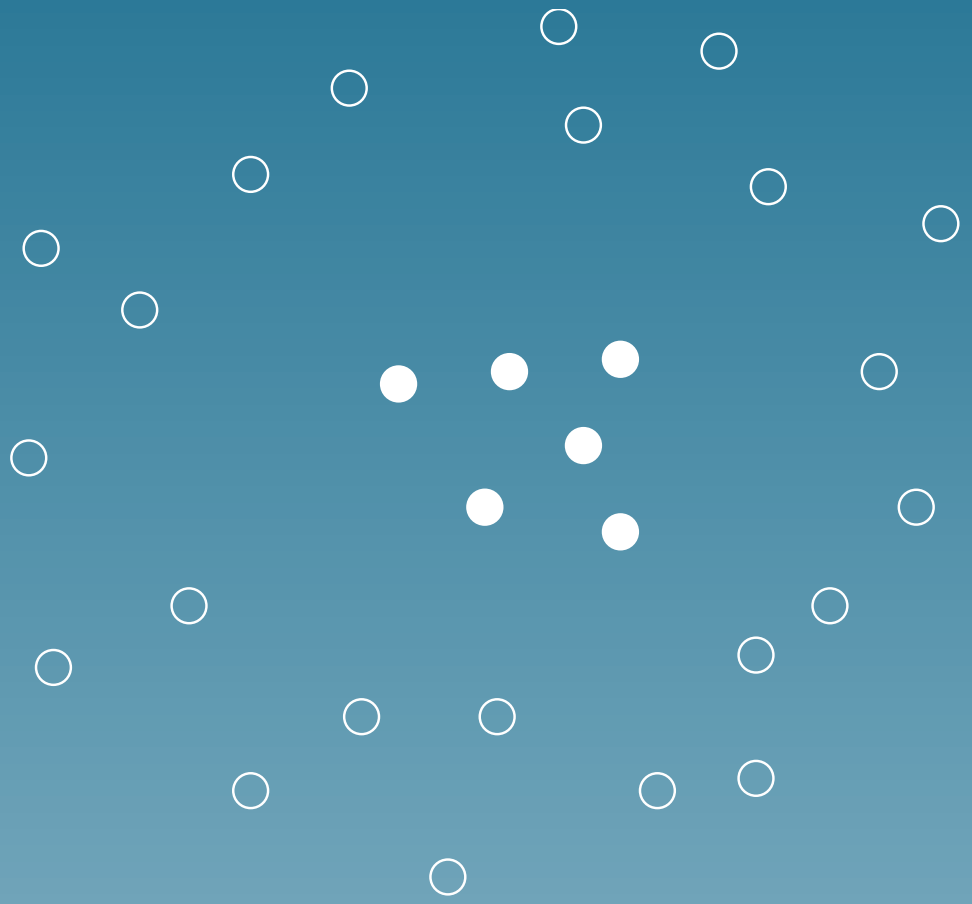
The classification of a new point x is the sign of:

$$f(x) = \sum_i \alpha_i \vec{x}.\vec{x_i} + b,$$

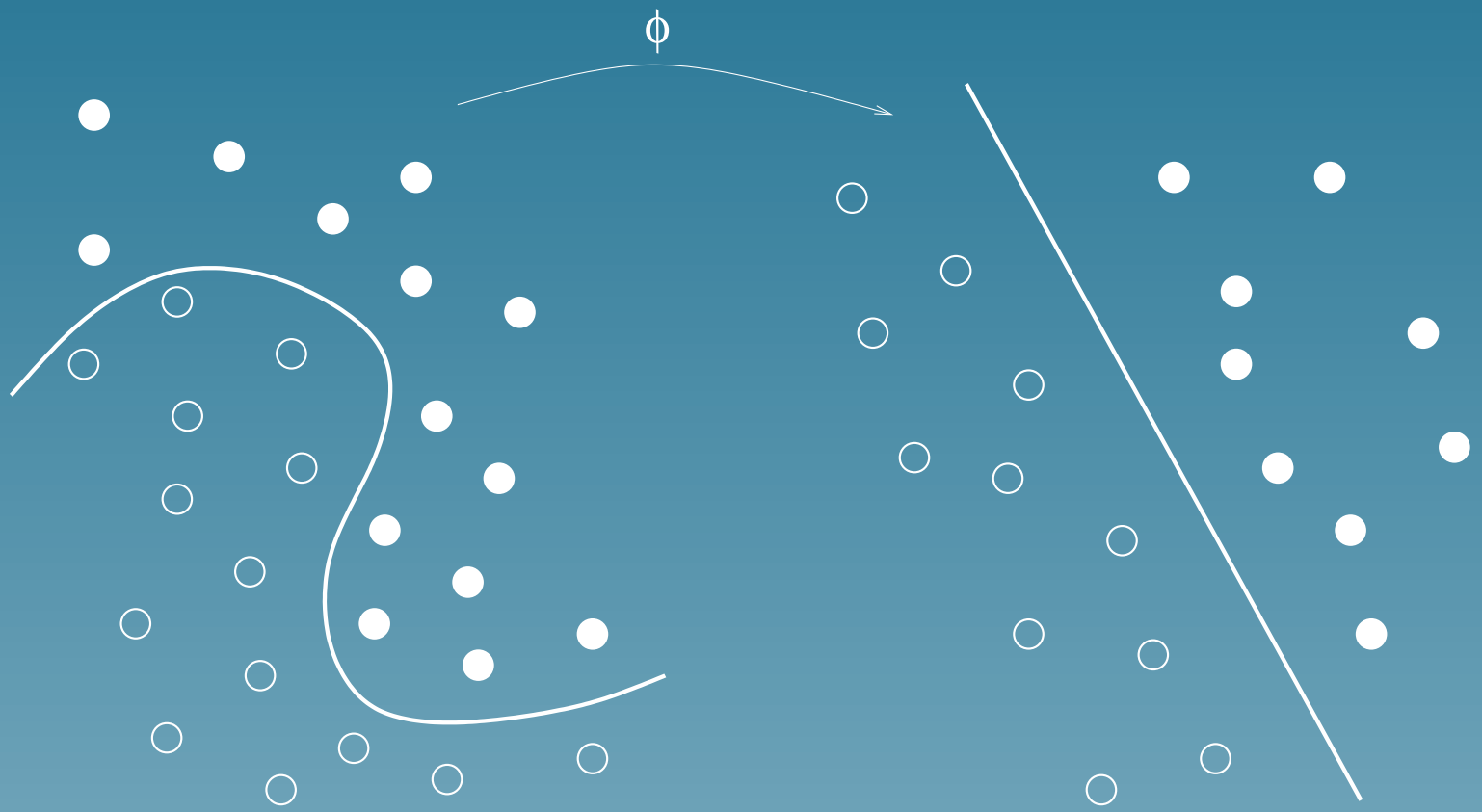
where α_i solves:

$$\begin{cases} \max_{\vec{\alpha}} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \vec{x_i}.\vec{x_j}) \\ \forall i = 1, \ldots, n \quad 0 \leq \alpha_i \leq C \\ \sum_{i=1}^n \alpha_i y_i = 0 \end{cases}$$

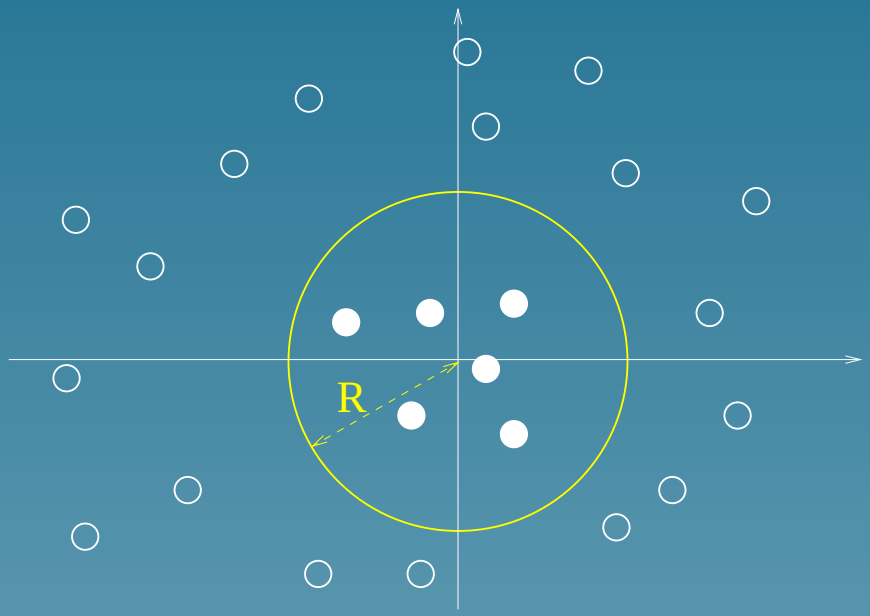
Sometimes linear classifiers are not interesting



Solution: non-linear mapping to a feature space



Example



Let $\Phi(\vec{x}) = (x_1^2, x_2^2)'$, $\vec{w} = (1, 1)'$ and $b = 1$. Then the decision function is:

$$f(\vec{x}) = x_1^2 + x_2^2 - R^2 = \vec{w}.\Phi(\vec{x}) + b,$$

Kernel (*simple but important*)

For a given mapping Φ from the space of objects $\mathcal{X}$ to some feature space, the kernel of two objects x and x' is the inner product of their images in the features space:

$$\forall x, x' \in \mathcal{X}, \quad K(x, x') = \vec{\Phi}(x).\vec{\Phi}(x').$$

Example: if $\vec{\Phi}(\vec{x}) = (x_1^2, x_2^2)'$, *then*

$$K(\vec{x}, \vec{x'}) = \vec{\Phi}(\vec{x}).\vec{\Phi}(\vec{x'}) = (x_1)^2(x_1')^2 + (x_2)^2(x_2')^2.$$

Training a SVM in the feature space

Replace each $\vec{x}.\vec{x'}$ in the SVM algorithm by $K(x, x')$

The dual problem is to maximize

$$L(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j),$$

under the constraints:

$$\begin{cases} 0 \leq \alpha_i \leq C, \quad \text{for } i = 1, \ldots, N \\ \sum_{i=1}^{N} \alpha_i y_i = 0. \end{cases}$$

Predicting with a SVM in the feature space

The decision function becomes:

$$\begin{aligned} f(x) &= \vec{w}^*.\vec{\Phi}(x) + b^* \\ &= \sum_{i=1}^{N} \alpha_i K(x_i, x) + b^*. \end{aligned} \tag{1}$$

The kernel trick

- The explicit computation of $\vec{\Phi}(x)$ is not necessary. The kernel $K(x, x')$ is enough. SVM work implicitly in the feature space.
- It is sometimes possible to easily compute kernels which correspond to complex large-dimensional feature spaces.

Kernel example

For any vector $\vec{x} = (x_1, x_2)'$, consider the mapping:

$$\Phi(\vec{x}) = \left(x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1\right)'.$$

The associated kernel is:

$$\begin{aligned} K(\vec{x}, \vec{x}') &= \Phi(\vec{x}).\Phi(\vec{x}') \\ &= (x_1x_1' + x_2x_2' + 1)^2 \\ &= (\vec{x}.\vec{x}' + 1)^2 \end{aligned}$$

Classical kernels for vectors

- Polynomial:

$$K(x, x') = (x.x' + 1)^d$$

- Gaussian radial basis function

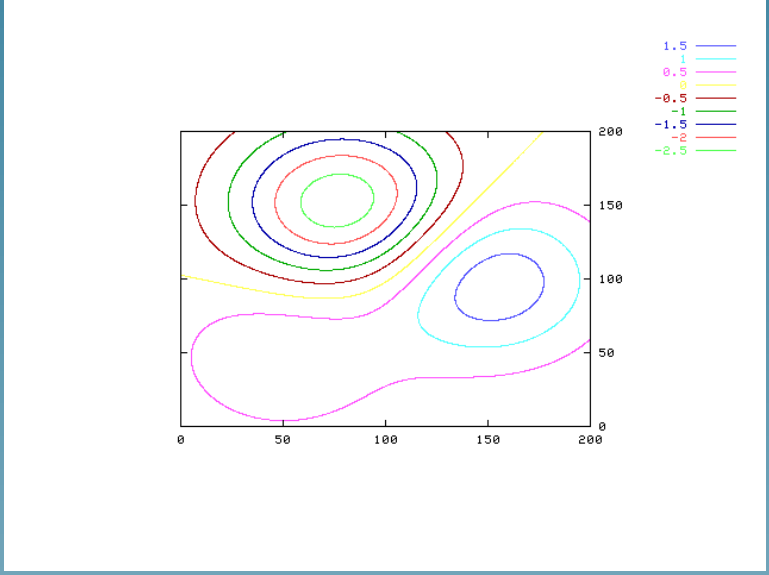
$$K(x, x') = \exp\left(\frac{||x - x'||^2}{2\sigma^2}\right)$$

- Sigmoid

$$K(x, x') = \tanh(\kappa x.x' + \theta)$$

Example: classification with a Gaussian kernel

$$f(\vec{x}) = \sum_{i=1}^{N} \alpha_i \exp\left(\frac{||\vec{x} - \vec{x}_i||^2}{2\sigma^2}\right)$$



Kernels

For any set $\mathcal{X}$, a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel iff:

- it is symetric :

$$K(x, y) = K(y, x),$$

- it is positive semi-definite:

$$\sum_{i,j} a_i a_j K(x_i, x_j) \geq 0$$

for all $a_i \in \mathbb{R}$ and $x_i \in \mathcal{X}$

Kernel properties

- The set of kernels is a convex cone closed under the topology of pointwise convergence
- Closed under the Schur product $K_1(x,y)K_2(x,y)$
- Bochner theorem: in $\mathbb{R}^d$,

$$K(x,y) = \phi(x-y)$$

is a kernel iff ϕ has a non-negative Fourier transform (generalization to more general groups and semi-groups)

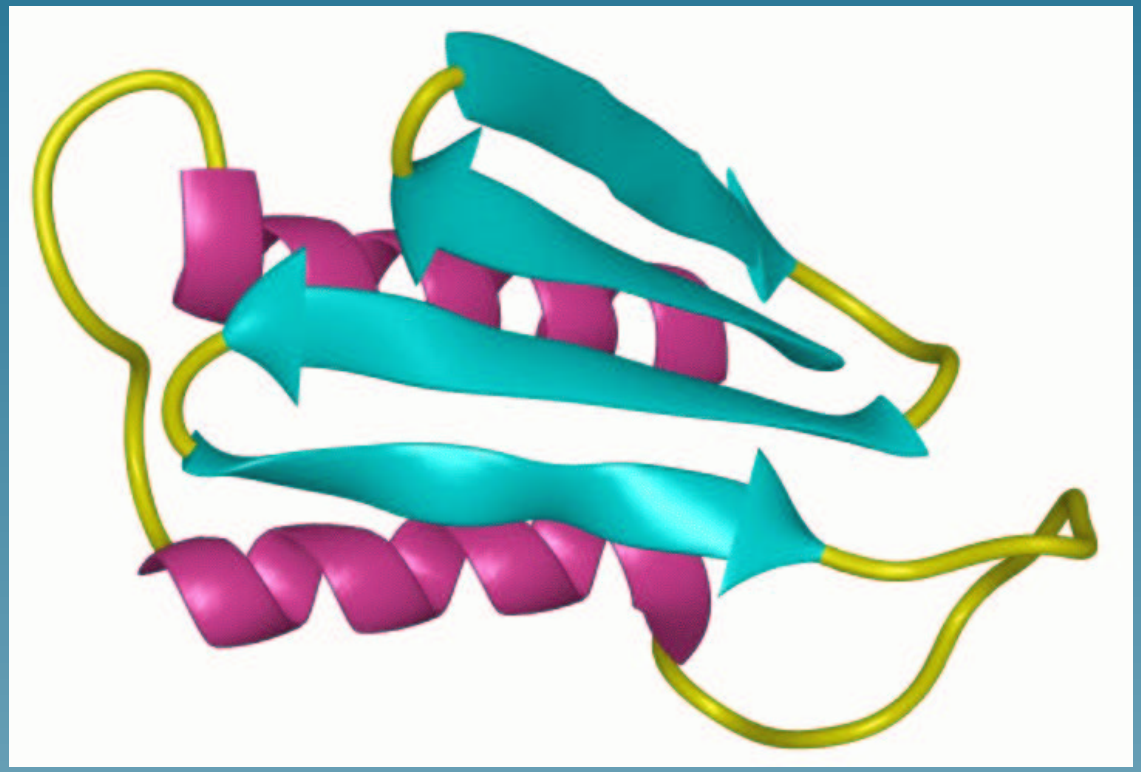
Partie 3

Computational biology

Overview

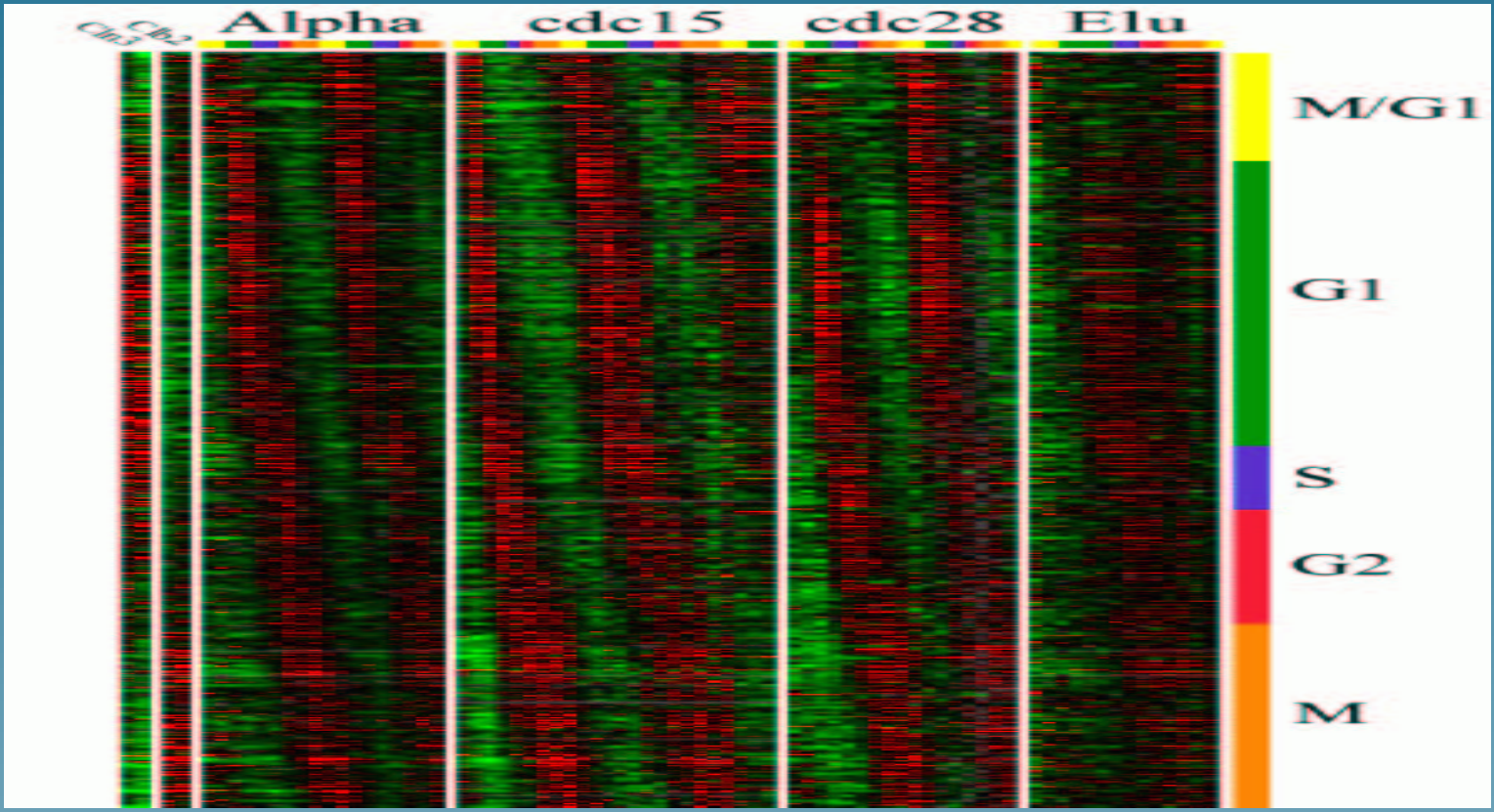
- 1990-95: DNA chips
- 2003: completion of the Human Genome Projects
- Gene sequences, structures, expression, localization, phenotypes, SNP, ... many many data
- Pharmacy, environment, food, ... many applications

Data examples



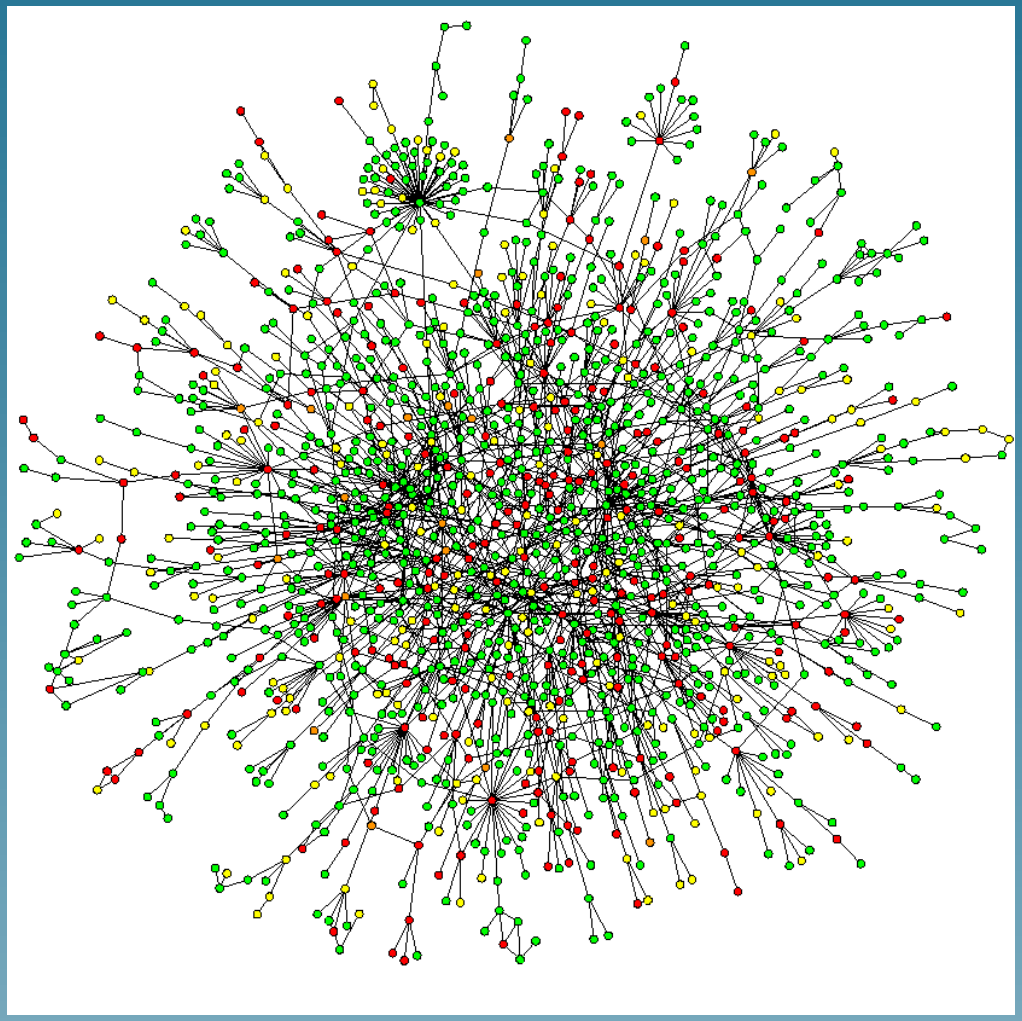
2D and 3D structure of prion

Data examples



(From Spellman et al., 1998)

Data examples



Pattern recognition examples

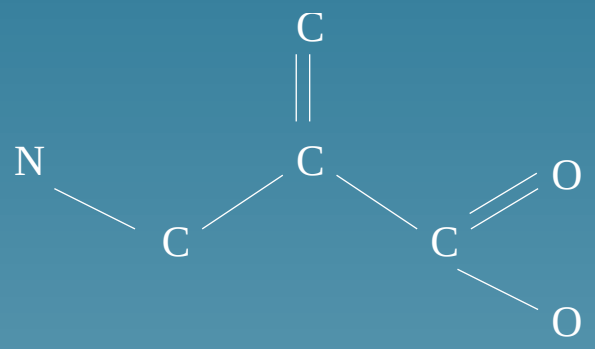
- Medical diagnosis from gene expression
- Functional genomics
- Structural genomics
- Drug design
- ... SVM are being applied

Partie 4

Virtual screening of small molecules

The problem

- Objects = chemical compounds (formula, structure..)



- Problem = predict their:
  - ⋆ drugability
  - ⋆ pharmacocinetics
  - ⋆ activity on a target etc...

Classical approaches

- Use molecular descriptors to represent the compouds as vectors
- Select a limited numbers of relevant descriptors
- Use linear regression, NN, nearest neighbour etc...

SVM approach

- We need a kernel $K(c_1, c_2)$ between compounds

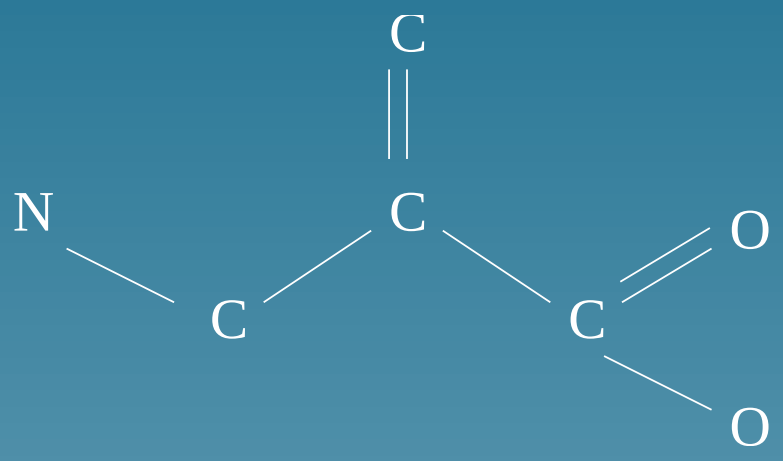
SVM approach

- We need a kernel $K(c_1, c_2)$ between compounds
- One solution: inner product between vectors

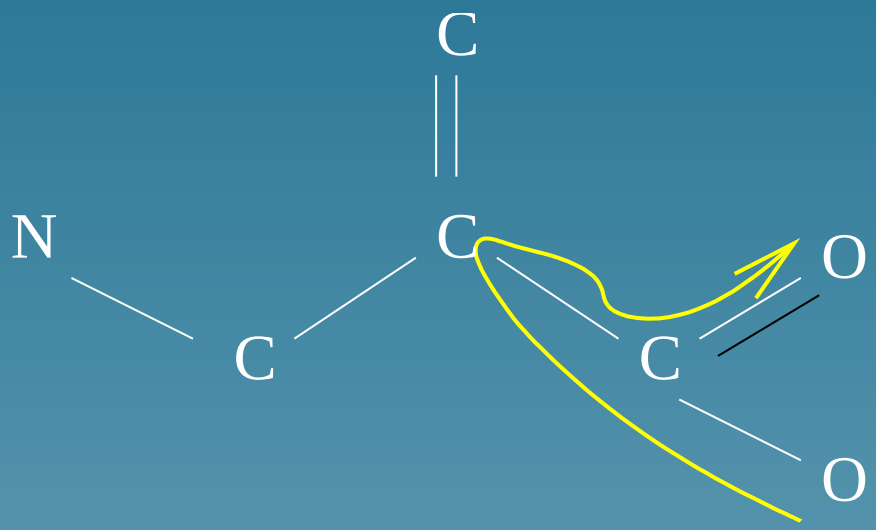
SVM approach

- We need a kernel $K(c_1, c_2)$ between compounds
- One solution: inner product between vectors
- Alternative solution: define a kernel directly using graph comparison tools

Example: graph kernel (Kashima et al., 2003)

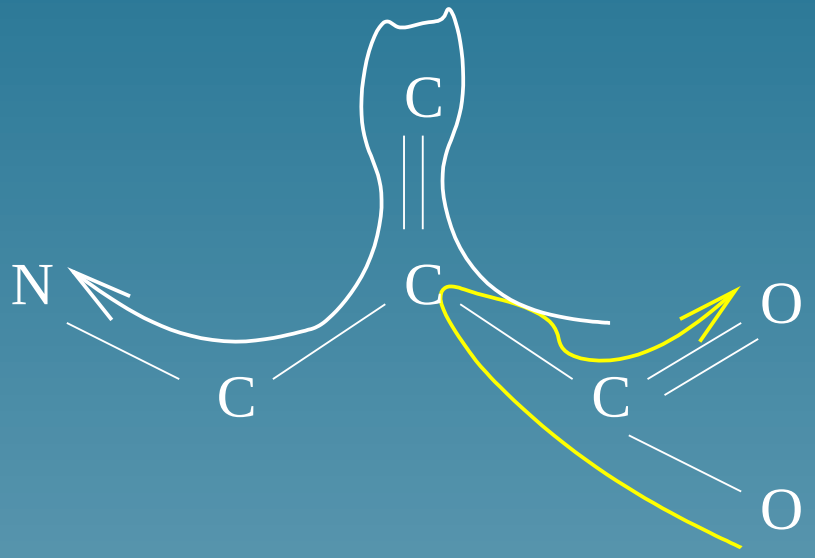


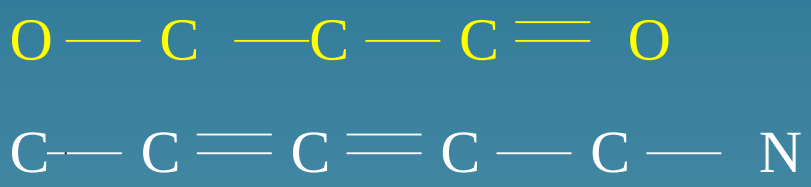
Example: graph kernel (Kashima et al., 2003)



Extract random paths

Example: graph kernel (Kashima et al., 2003)





Extract random paths

Example: graph kernel (Kashima et al., 2003)

- Let H_1 be a random path of a compound c_1
- Let H_2 be a random path of a compound c_2
- The following is a valid kernel:

$$K(c_1, c_2) = \mathsf{Prob}(H_1 = H_2).$$

Remarks

- The feature space is infinite-dimensional (all possible paths)

Remarks

- The feature space is infinite-dimensional (all possible paths)
- The kernel can be computed efficiently

Remarks

- The feature space is infinite-dimensional (all possible paths)
- The kernel can be computed efficiently
- Two compounds are compared in terms of their common substructures

Remarks

- The feature space is infinite-dimensional (all possible paths)
- The kernel can be computed efficiently
- Two compounds are compared in terms of their common substructures
- What about kernels for the 3D structure?

Conclusion

Conclusion

- Important developments of statistical learning theory recently
- Involve several fields: probability/statistics, functional analysis, optimization, harmonic analysis on semigroups, differential geometry...
- Results in useful state-of-the-art algorithms in many fields
- Computational biology directly benefits from these developments... but it's only the begining