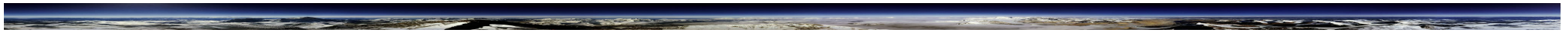
Support Vector Machines (SVM) and kernel methods



A. Cornuéjols

AgroParisTech – INRAé MIA 518

Outline

Supervised induction

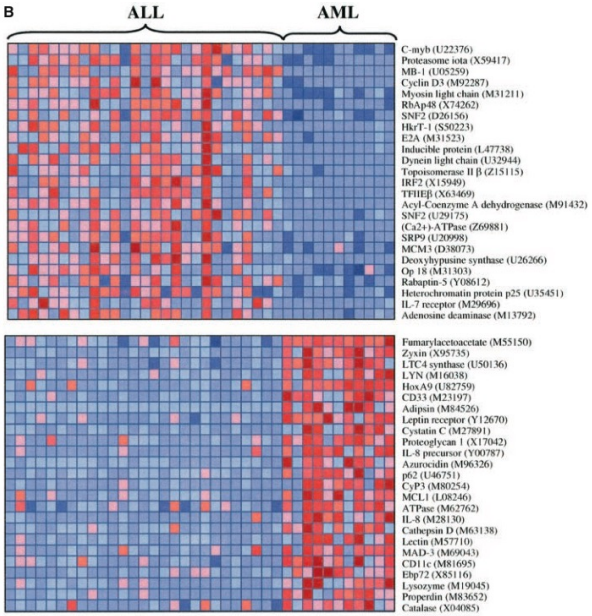
A reminder

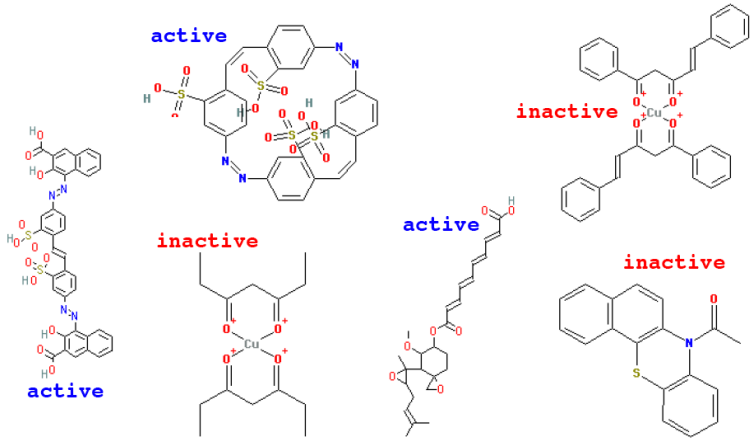
Supervised Learning

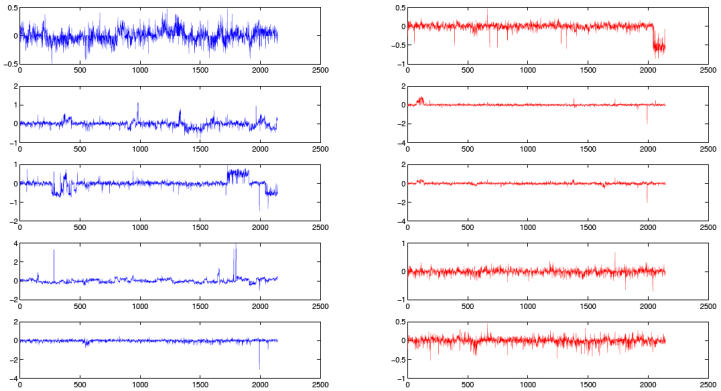
- Learning of a **function** *Input -> Output* $(\mathcal{X} \rightarrow \mathcal{Y})$
  - **Prediction**
    - Stock exchange prices $\Re^T \rightarrow \Re$
    - Predict when a mechanical piece will break down $\Re^n \rightarrow \Re$
    - Weather $\text{structure}^T \rightarrow \text{structure}$
  - **Classification**
    - Molecule $\rightarrow$ {active, not active}
    - Microarray $\rightarrow$ {cancer, not cancer}
    - Temporal series $\rightarrow$ {normal, abnormal}

Machine Learning

- From a training set $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{1 \leq i \leq m}$







La démarche conceptuelle

1. On vise la **performance en généralisation** or on ne mesure que la performance en apprentissage !?
   - D'où la question : comment choisir la meilleure hypothèse ?
     - Sans doute est-il bon qu'elle se comporte bien sur les données d'apprentissage, mais **parmi toutes celles qui se comportent bien, laquelle favoriser ?**

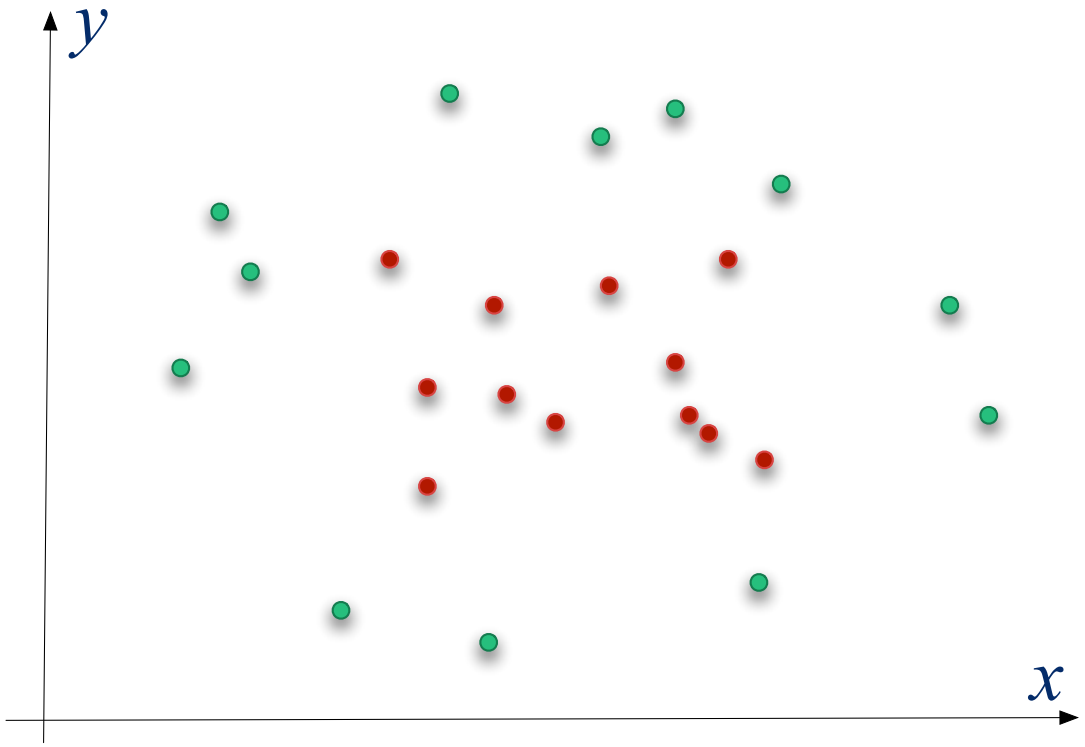
La démarche conceptuelle

1. On vise la **performance en généralisation** or on ne mesure que la performance en apprentissage !?
   - D'où la question : comment choisir la meilleure hypothèse ?
     - Sans doute est-il bon qu'elle se comporte bien sur les données d'apprentissage, mais **parmi toutes celles qui se comportent bien, laquelle favoriser ?**

2. Analyse de Vapnik sur le lien entre risque empirique mesuré ET servant de critère de sélection et le risque réel.
   - Il faut que la **richesse de l'espace d'hypothèse soit limitée** pour qu'un lien existe et ce lien est d'autant plus étroit que cette richesse est limitée.
   - Il faut donc en fait **optimiser un risque régularisé** dans lequel entre en compte la richesse de *H*.
     - Avantage, on peut biaiser le choix des hypothèses en fonction de connaissances a priori sur les concepts cibles.

Supposons un ensemble d'apprentissage dans R^2

- Et que $\mathcal{H}$ soit l'espace des rectangles

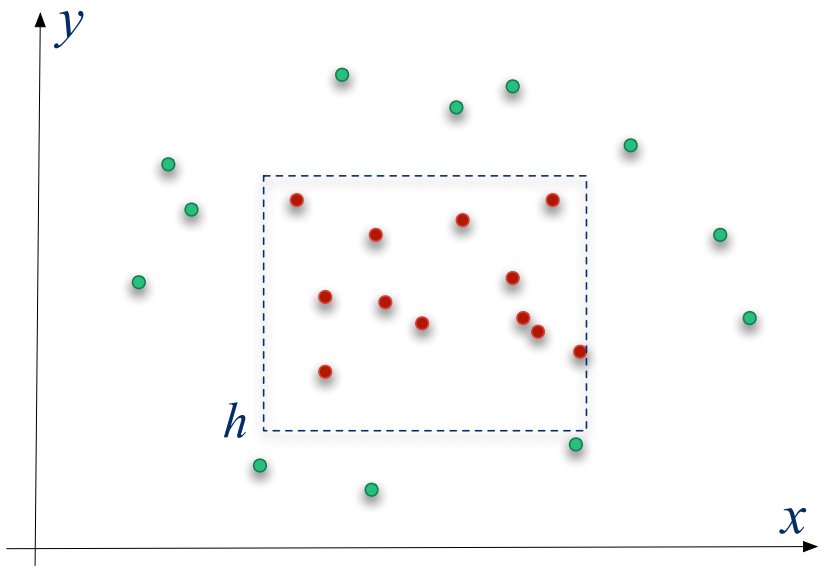


Comment choisir une hypothèse dans $\mathcal{H}$?

Quelle hypothèse choisir ?

Quelle qualité pour chaque hypothèse candidate ?

Quelle hypothèse choisir ?



Le « risque empirique »

$$\hat{R}(h) = \frac{1}{m} \sum_{i=1}^{m} \ell\big(h(\mathbf{x}_i), y_i\big)$$

...

Quelle hypothèse choisir ?

Quelle performance ?

- Coût d'**une erreur** de prédiction
  - La ***fonction de perte***

$$\ell\big(h(\mathbf{x}), y\big)$$

- Quel **coût à venir** (espérance) si je choisis *h* ?
  - Espérance de coût : le « ***risque réel*** »

$$R(h) = \int_{\mathcal{X}\times\mathcal{Y}} \ell\big(h(\mathbf{x}), y\big)\ \mathbf{p}_{\mathcal{X}\mathcal{Y}}(\mathbf{x}, y)\ d\mathbf{x}\, dy$$

Quelle hypothèse choisir ?

Comment trouver *h** (ou une bonne hypothèse) alors que l'on n'a accès qu'à un **échantillon d'apprentissage limité** ?

Quelle hypothèse choisir ?

Comment trouver h^* (ou une bonne hypothèse) alors que l'on n'a accès qu'à un **échantillon d'apprentissage limité** ?

- Critère inductif : $\mathcal{H}$ x S -> valeur(h)

- Le plus naturel : ERM
  - La Minimisation du Risque Empirique

A-t-on raison d'utiliser l'ERM ?

L'analyse « PAC learning »

- On arrive à :

$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad \mathbf{P}^m \left[R_{\text{Réel}}(h) \leq R_{\text{Emp}}(h) + \overbrace{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{m}}^{\varepsilon} \right] > 1 - \delta$$

Le principe de minimisation du risque empirique

n'est sain **que si** il y a des **contraintes sur l'espace des hypothèses**

Bounding the true risk with the empirical risk + …

- $\mathcal{H}$ finite, realizable case

$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad P^m \left[R_{\text{Réel}}(h) \leq R_{\text{Emp}}(h) + \frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{m} \right] > 1 - \delta$$

- $\mathcal{H}$ finite, **non** realizable case

$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad P^m \left[R_{\text{Réel}}(h) \leq R_{\text{Emp}}(h) + \sqrt{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{2\,m}} \right] > 1 - \delta$$

To sum up: for $|\mathcal{H}|$ finite

- **Non** realizable case

$$\varepsilon = \sqrt{\frac{\log|\mathcal{H}| + \log\frac{1}{\delta}}{2m}} \qquad \text{and} \qquad m \geq \frac{\log|\mathcal{H}| + \log\frac{1}{\delta}}{2\varepsilon^2}$$

- **Realizable** case

$$\varepsilon = \frac{\log|\mathcal{H}| + \log\frac{1}{\delta}}{m} \qquad \text{and} \qquad m \geq \frac{\log|\mathcal{H}| + \log\frac{1}{\delta}}{\varepsilon}$$

Bounds between the real risk and the empirical risk

- non realisable case and $\mathcal{H}$ not finite

How to proceed?

– General approach:

1. **Reduce** the study of the **infinite** case to the analysis of an **finite set of hypotheses**
2. **Measure** how much it is possible, **for any training set** S of labeled points, to find an **hypothesis** in $\mathcal{H}$ **that can fit** S

Bounds between the real risk and the empirical risk

- Measure using the ***Vapnik-Chervonenkis dimension***
  - Purely **combinatoric measure**, which does not depend on the number of examples
  - Size of the largest set of points points that can be labelled in any way by the hypotheses in H

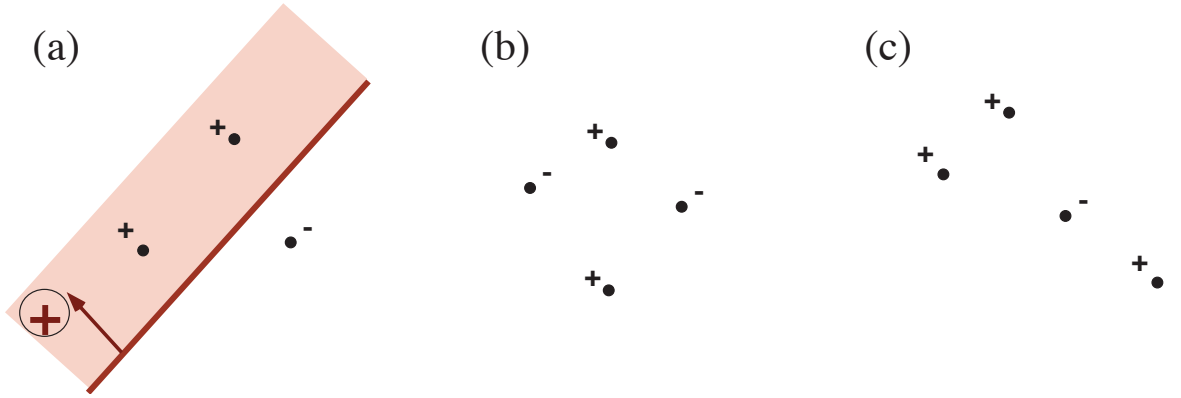
$$d_{VC}(\mathcal{H}) = \max\{m : \Pi_{\mathcal{H}}(m) = 2^m\}$$

One can compute the bound:

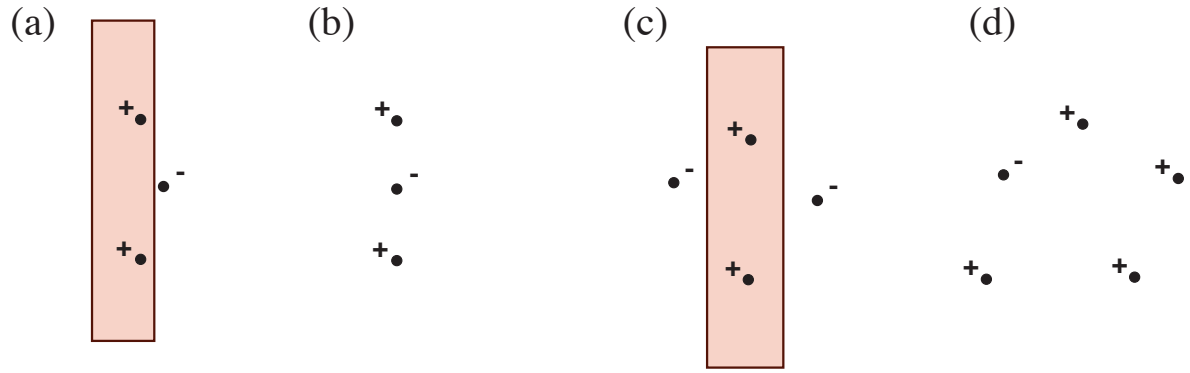
$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad P^m\left[R_{\text{Réel}}(h) \leq R_{\text{Emp}}(h) + \sqrt{\frac{8\, d_{VC}(\mathcal{H}) \log \frac{2\,e\,m}{d_{VC}(\mathcal{H})} + 8 \log \frac{4}{\delta}}{m}}\right] > 1 - \delta$$

VC dim: illustration

- d_{VC}(linear separators) = ?



- d_{VC}(rectangles) = ?



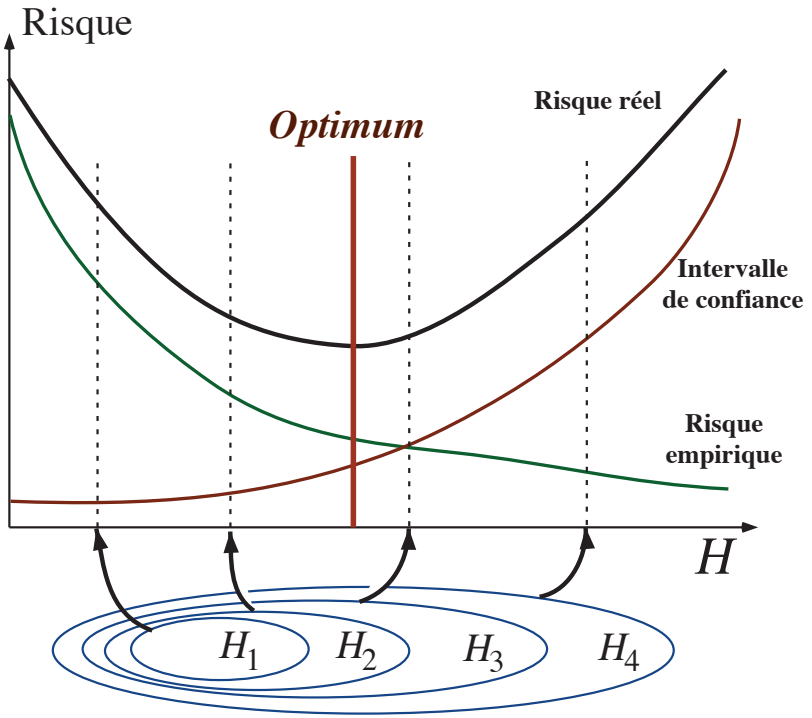
Théorie statistique de l'apprentissage

Le 3ème temps

Quel espace d'hypothèse $\mathcal{H}$?

SRM : Structural Risk Minimization

- **Stratification** des espaces d'hypothèses
  - Faite *a priori* (indépendam-ment des données)
  - Par exemple en utilisant la d_{VC}



L'analyse « PAC learning » ou statistique

$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad \mathbf{P}^m \left[R_{\text{Réel}}(h) \leq \underbrace{R_{\text{Emp}}(h) + \frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{m}}_{\text{Risque régularisé}} \right] > 1 - \delta$$

- *Nouveau critère inductif* :
  - Le **risque empirique régularisé**
    1. Satisfaire les contraintes posées par les **exemples**
    2. Choisir le meilleur **espace d'hypothèses** (capacité de *H*)

Recette pour … **concevoir des algorithmes** d'apprentissage

1. Définir un **critère inductif régularisé**
   a. Exprimer le coût d'erreur de prédiction en une fonction de perte
   b. Définir un **terme de régularisation** qui exprime
      les attendus sur les régularités du monde
   c. Si possible, rendre convexe le problème d'**optimisation** résultant

$$h_{opt} = \underset{h \in \mathcal{H}}{\text{ArgMin}} \left[\underbrace{\frac{1}{m} \sum_{i=1}^{m} l(h(\mathbf{x}_i), y_i)}_{\text{empirical risk}} + \lambda \underbrace{reg(\mathcal{H})}_{\text{bias on the world}} \right]$$

2. Utiliser ou développer un **algorithme d'optimisation efficace**

Learning **sparse linear** approximator

- The **hypothesis** is of the form $h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$
- A priori assumption: few non zero coefficients

Ridge regression

$$\mathbf{w}^*_{\text{ridge}} = \underset{\mathbf{w}}{\text{Argmin}} \left\{ \sum_{i=1}^{m} (y_i - \mathbf{w}\,\mathbf{x}_i)^2 + \lambda \, ||\mathbf{w}||_2^2 \right\}$$

Lasso regression

$$\mathbf{w}^*_{\text{lasso}} = \underset{\mathbf{w}}{\text{Argmin}} \left\{ \sum_{i=1}^{m} (y_i - \mathbf{w}\,\mathbf{x}_i)^2 + \lambda \, ||\mathbf{w}||_1 \right\}$$

La démarche conceptuelle (2)

1. Autre retombée de l'analyse de Vapnik :

   l'écart entre risque empirique et risque réel

   décroît plus rapidement **si** $f \in H$

   – D'où la question : peut-on s'arranger pour que $f \in H$?

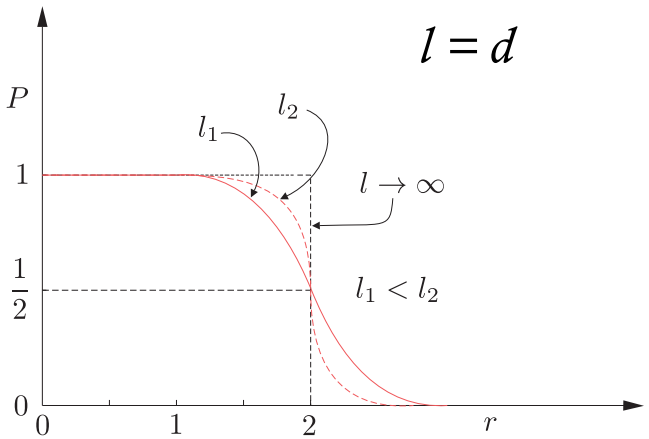
1. Le théorème de Cover pourrait jouer un rôle

Cover's theorem

Theorem

Given m points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m \in \mathbb{R}^d$. The number of groupings $\mathcal{O}(m, d)$ that can be formed by $(d$ - $1)$- dimensional hyperplanes to separate the m points in two classes, exploiting all possible combinations is given by:

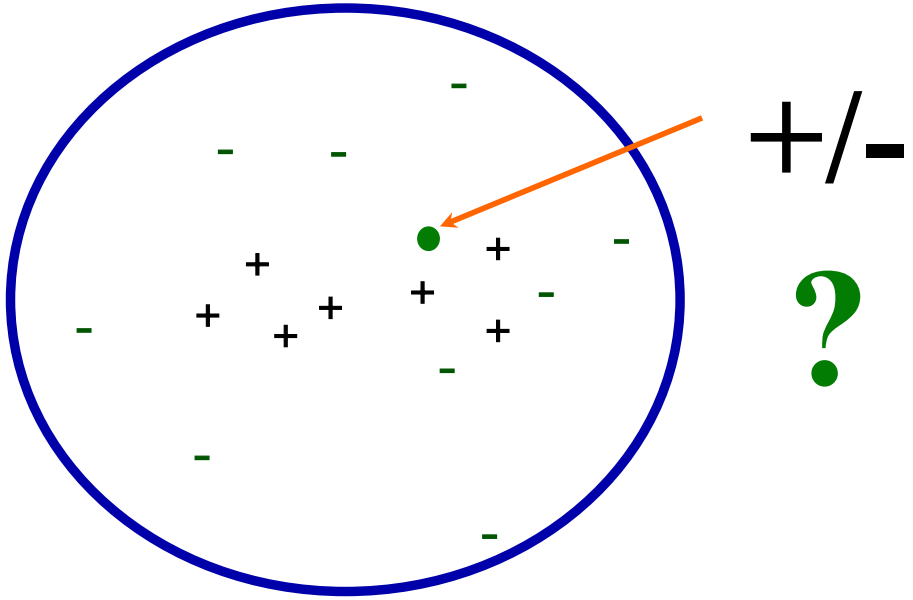
$$\mathcal{O}(m, d) = 2 \sum_{i=0}^{d} \binom{m-1}{i}$$



- For $m > 2(d + 1)$ the probability of linear separability becomes small.
- For large values of d, and provided that $m < 2\,(d + 1)$, the probability of any grouping of the data into two classes to be **linearly separable** tends to 1 !!

Decision functions

Classification by nearest neighbours

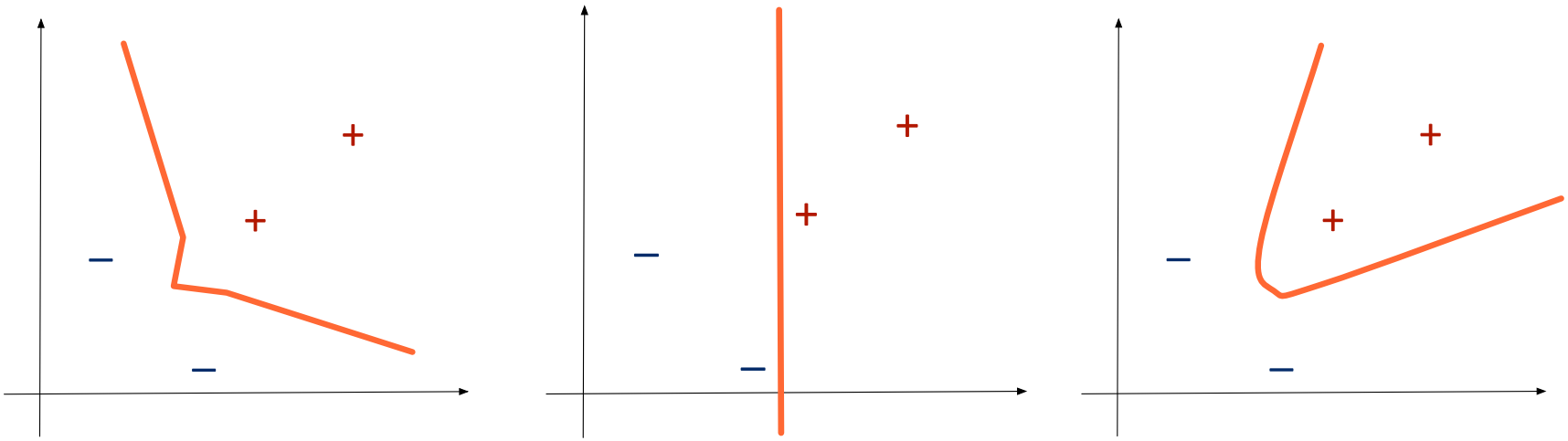


Example space: $\mathcal{X}$

Three questions:

- How to choose the **relevant points**?
- How to **weight** the points?
- Which **distance** to use?

Learning a decision boundary



k-Nearest neighbours | Decision tree | Multi-layer perceptron

Outline

1. Supervised induction: a reminder
2. Perceptron with widest margin
   1. Derivation
   2. The margin: its signification and importance
   3. Soft margins
3. SVM and the kernel trick
4. In practice
5. Kernels engineering
6. Conclusion

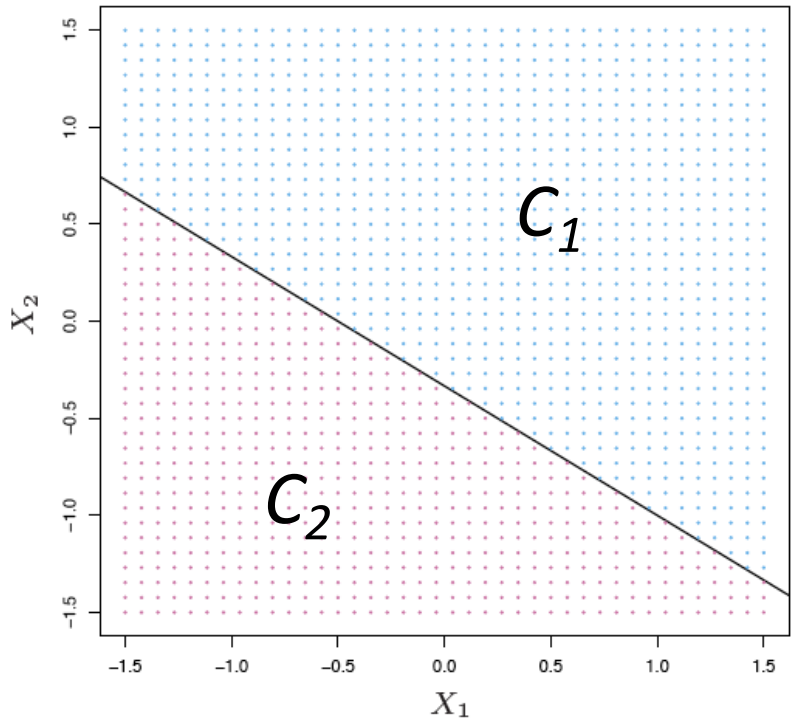
The Perceptron

The perceptron: elementary algebra

- In the plan

$$1 + 2\,x_1 + 3\,x_2 = 0$$

or: $$x_2 = -\frac{2}{3}\,x_1 - \frac{1}{3}$$



Given a new point: $\mathbf{x}^* = [x_1^*, x_2^*]^\top$

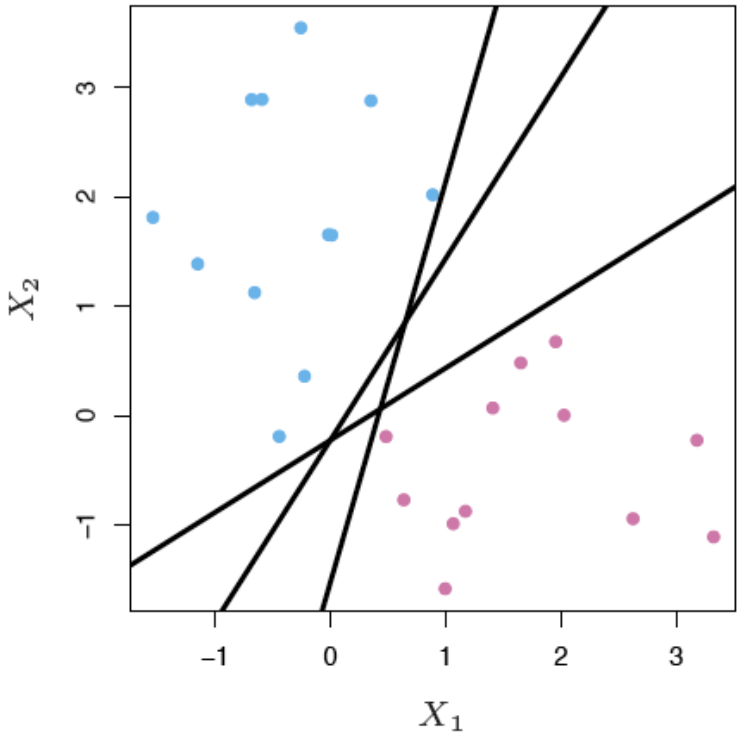
$$1 + 2\,x_1^* + 3\,x_2^* = \begin{cases} \geq 0, & \mathbf{x}^* \in \mathcal{C}_1 \\ \leq 0, & \mathbf{x}^* \in \mathcal{C}_2 \\ = 0, & \mathbf{x}^* \text{indéterminé} \end{cases}$$

The perceptron: elementary algebra

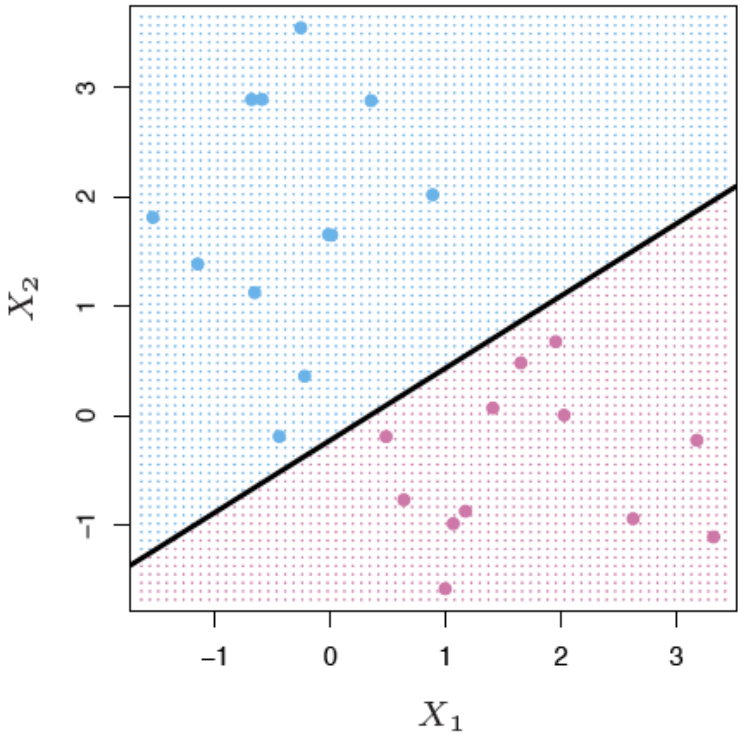
- In a space of dimension *d*
  - Equation of an *hyperplan* (of dimension *d*-1) $\sum_{j=1}^{d} w_j x_j + w_0 = 0$
  - Expression of an **hypothesis** (perceptron) :

$$h(\mathbf{x}) = \sum_{j=1}^{d} w_j x_j + w_0$$

How to choose the right perceptron?



Given a learning set,
there exists a set of solutions.



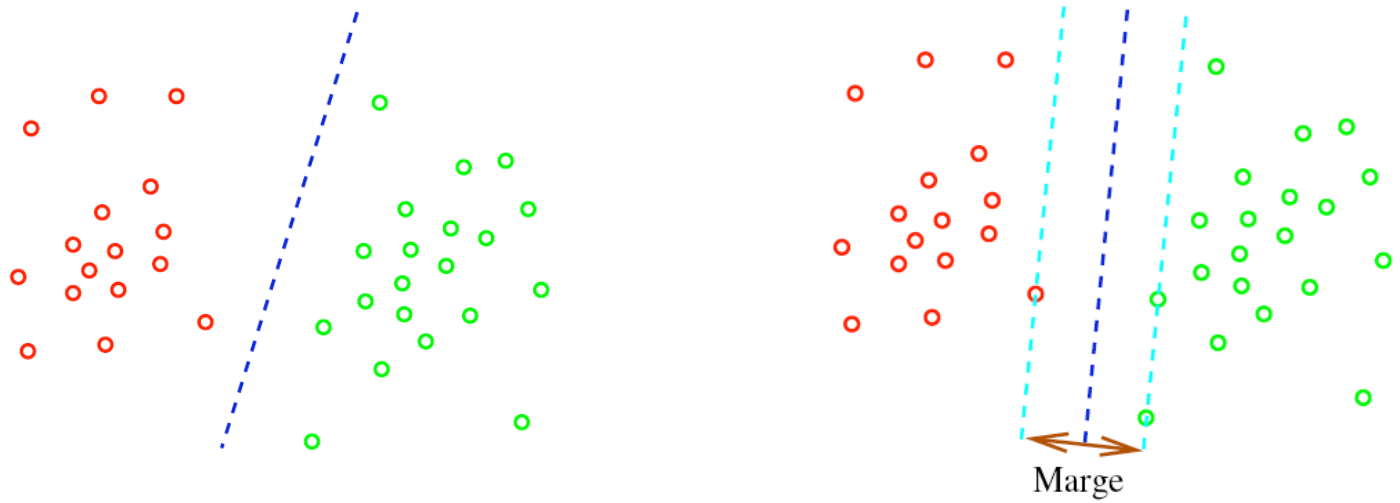
Here is one of them with the
induced classification of points.

Perceptron

with largest margin

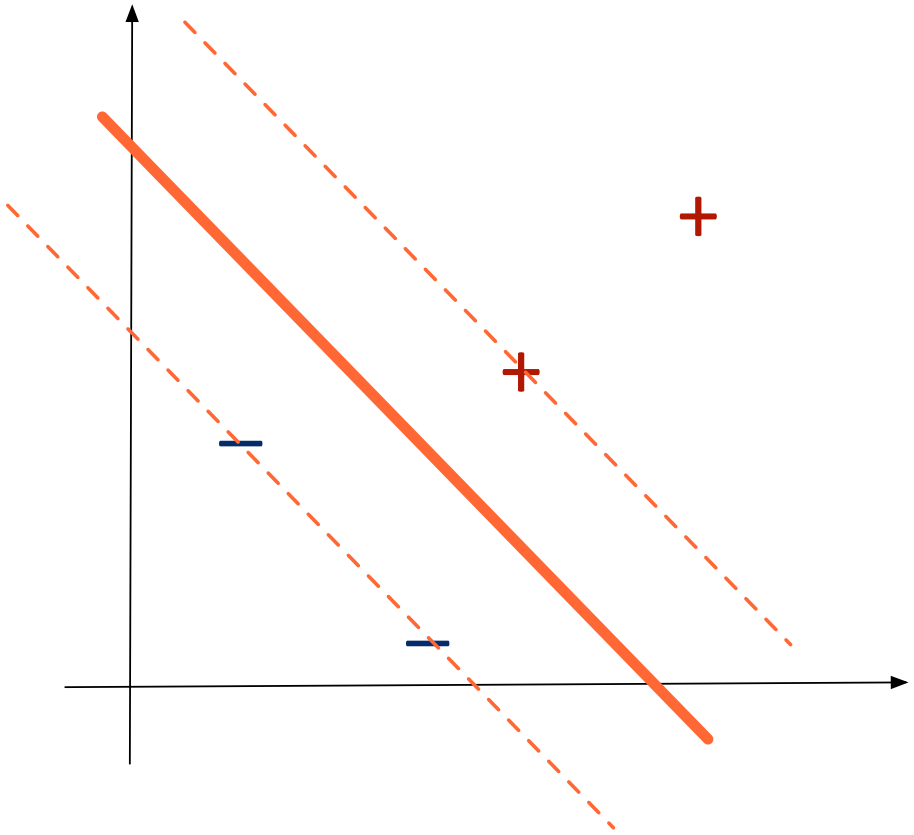
Perceptron: why select the largest margin?

- Assume that the two target classes can be linearly separated.
  - There generally exist an infinite number of solutions
  - We want to select the one that maximizes the distance to the nearest examples of the two classes: this is **classifier with maximal margin**



- Intuitive justification: it is the **linear classifier** that satisfies the constraints set by the training instances and is **the most robust** to possible variations of*S.*

SVM « Séparateurs à **vastes marges** »



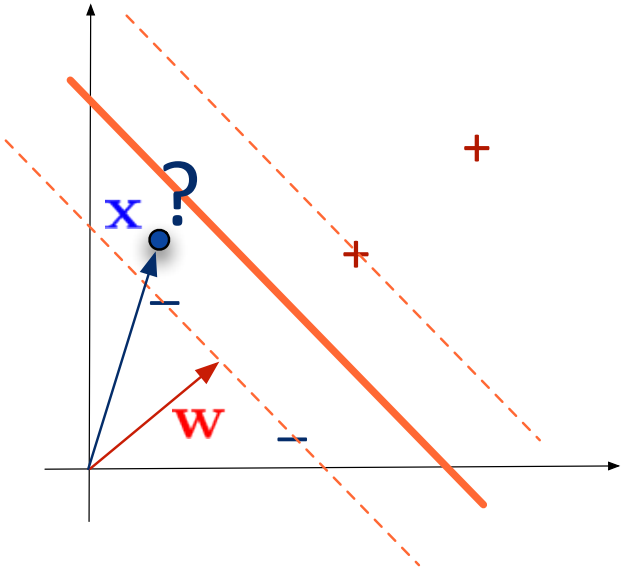
- Vapnik (1992)

SVM « Séparateurs à Vastes Marges »

- On veut : $\mathbf{w} \cdot \mathbf{x} \geq C^{te}$
- Fonction hypothèse :

$$\mathbf{w} \cdot \mathbf{x} - w_0 \begin{cases} > 0 & \mathbf{x} \in \text{class '+'} \\ < 0 & \mathbf{x} \in \text{class '-'} \end{cases}$$

?



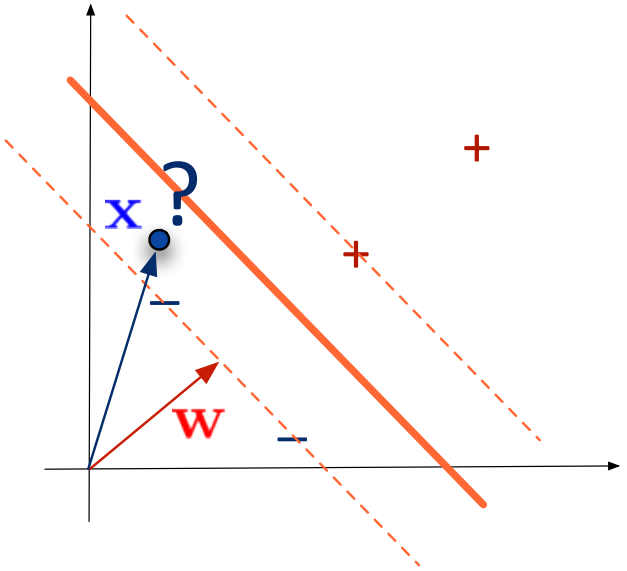
SVM « Séparateurs à Vastes Marges »

- On veut $\mathbf{w} \cdot \mathbf{x} \geq C^{te}$

- Fonction hypothèse :

$$\mathbf{w} \cdot \mathbf{x} - w_0 \begin{cases} > 0 & \mathbf{x} \in \text{class '+'} \\ < 0 & \mathbf{x} \in \text{class '-'} \end{cases}$$

?



- On décide de respecter les contraintes suivantes (nous sommes libres de fixer le référentiel)

y_i = +1 si $\mathbf{x}_i$ est un `+'

y_i = -1 si $\mathbf{x}_i$ est un `-'

$$y_i \left(\mathbf{w} \cdot \mathbf{x}_i - w_0\right) \geq 1 \qquad \forall\, \mathbf{x}_i$$

et

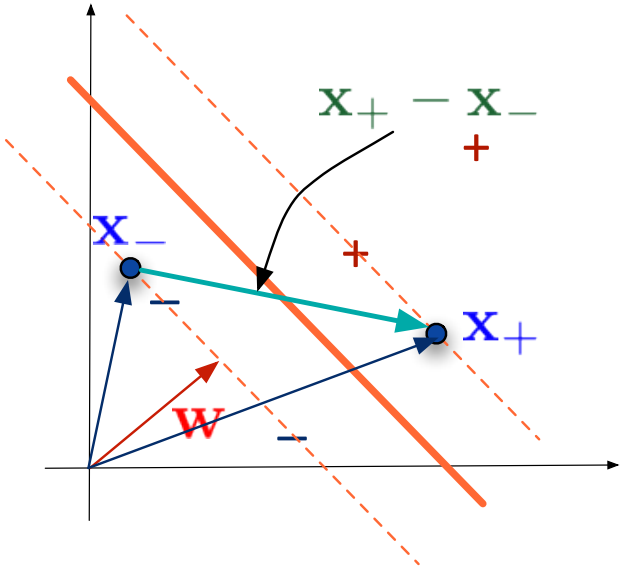
$$y_i \left(\mathbf{w} \cdot \mathbf{x} - w_0\right) = 1 \qquad \forall\, \mathbf{x} \text{ on the "margin"}$$

SVM « Séparateurs à Vastes Marges »

$$y_i\,(\mathbf{w}\,.\,\mathbf{x} - w_0) = 1 \qquad \forall\, \mathbf{x} \text{ on the "margin"}$$

$$\mathbf{w}.\mathbf{x}_+ = 1 + w_0 \qquad \text{therefore} \qquad \mathbf{x}_+ = \frac{1 + w_0}{\mathbf{w}}$$

$$-\,\mathbf{w}.\mathbf{x}_- = 1 - w_0 \qquad \text{therefore} \qquad \mathbf{x}_- = \frac{-1 + w_0}{\mathbf{w}}$$



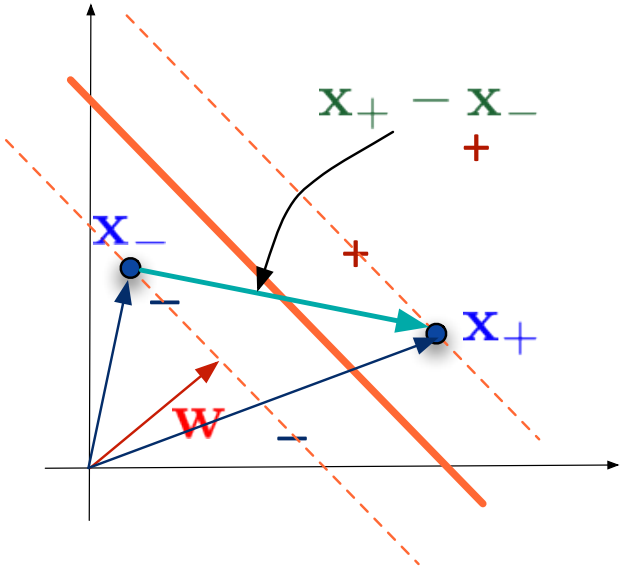
SVM « Séparateurs à Vastes Marges »

$$y_i\,(\mathbf{w}\,.\,\mathbf{x} - w_0) = 1 \qquad \forall\, \mathbf{x} \text{ on the "margin"}$$

$$\mathbf{w}.\mathbf{x}_+ = 1 + w_0 \qquad \text{therefore} \qquad \mathbf{x}_+ = \frac{1 + w_0}{\mathbf{w}}$$

$$-\,\mathbf{w}.\mathbf{x}_- = 1 - w_0 \qquad \text{therefore} \qquad \mathbf{x}_- = \frac{-1 + w_0}{\mathbf{w}}$$

$$\text{width} = (\mathbf{x}_+ - \mathbf{x}_-)\,.\,\frac{\mathbf{w}}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}$$



should be maximized

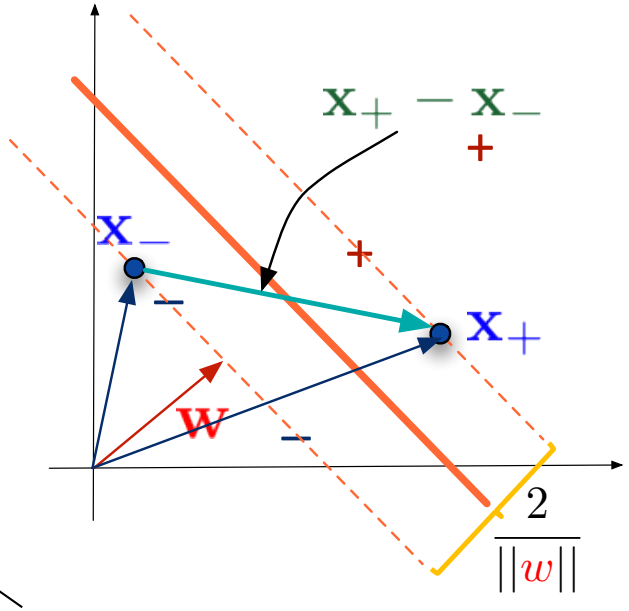
SVM « Séparateurs à Vastes Marges »

$$y_i\,(\mathbf{w}\,.\,\mathbf{x} - w_0) = 1 \qquad \forall\, \mathbf{x} \text{ on the "margin"}$$

$$\mathbf{w}.\mathbf{x}_+ = 1 + w_0 \qquad \text{therefore} \qquad \mathbf{x}_+ = \frac{1 + w_0}{\mathbf{w}}$$

$$-\,\mathbf{w}.\mathbf{x}_- = 1 - w_0 \qquad \text{therefore} \qquad \mathbf{x}_- = \frac{-1 + w_0}{\mathbf{w}}$$

$$\text{width} = (\mathbf{x}_+ - \mathbf{x}_-)\,.\,\frac{\mathbf{w}}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}$$



should be maximized

minimize $||\mathbf{w}||$ or, equivalently $\frac{1}{2}||\mathbf{w}||^2$

SVM: the linear case (**primal expression**)

- The normalized margin is equal to $2/||\mathbf{w}||$
- So that an equivalent problem is (***primal expression***):

$$\begin{cases} \text{Minimiser} & \frac{1}{2}\| \mathbf{w} \|^2 \\ \text{sous les contraintes} & u_i \left(\mathbf{w}^\top \mathbf{x}_i + w_0\right) \geq 1 \quad i = 1, \ldots, m \end{cases}$$

SVM: the linear case (**primal expression**)

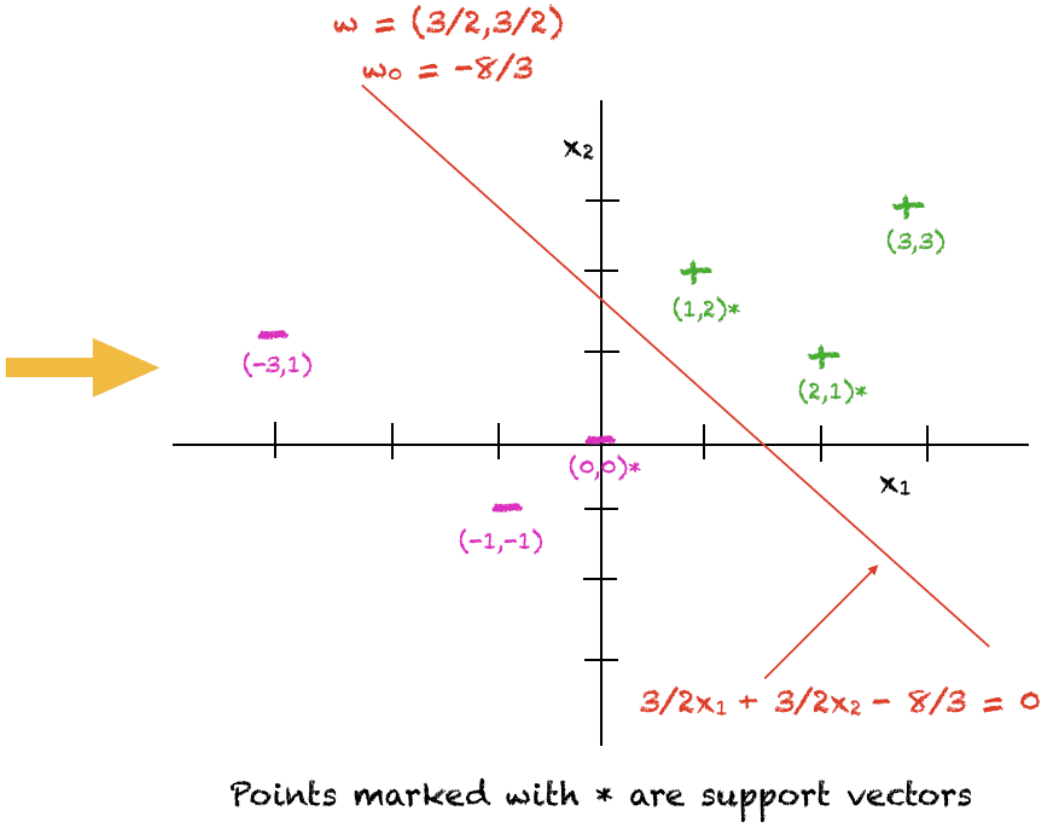
- The normalized margin is equal to $2/||\mathbf{w}||$
- So that an equivalent problem is (***primal expression***):

$$\begin{cases} \text{Minimiser} & \frac{1}{2}\| \mathbf{w} \|^2 \\ \text{sous les contraintes} & u_i \left(\mathbf{w}^\top \mathbf{x}_i + w_0\right) \geq 1 \quad i = 1, \ldots, m \end{cases}$$

- **It is a quadratic optimization problem with linear constraints**. There exists a whole set of efficient algorithms. One can thus obtain the solution vector **w*** plus w_0* that define **the decision function**:

$$h(\mathbf{x}) = \text{sign}\left\{\mathbf{w}^{\star\top}\mathbf{x} + w_0^\star\right\}$$

i	Data point x	label t	α
0	(1,2)	+1	0.5
1	(2,1)	+1	0.5
2	(3,3)	+1	0
3	(0,0)	-1	1
4	(-1,-1)	-1	0
5	(-3,1)	-1	0



Points marked with * are support vectors

...

SVM: the linear case (**the dual form**)

- An optimization problem has a **dual form** if the function to be optimized and the constraints are strictly convex. Then the solution of the dual form is also a solution of the primal form of the optimization problem.

SVM: the linear case (**the dual form**)

- An optimization problem has a **dual form** if the function to be optimized and the constraints are strictly convex. Then the solution of the dual form is also a solution of the primal form of the optimization problem.

$$\begin{cases} \text{Minimiser} & \frac{1}{2}\| \mathbf{w} \|^2 \\ \text{sous les contraintes} & u_i \left(\mathbf{w}^\top \mathbf{x}_i + w_0\right) \geq 1 \quad i = 1, \ldots, m \end{cases}$$

- First, define the **Lagrangian**

$$L(\mathbf{w}, w_0, \alpha) = \frac{1}{2} ||\mathbf{w}||^2 - \sum_{i=1}^{m} \alpha_i \left(y_i \left(\mathbf{w} \,.\, \mathbf{x}_i + w_0\right)\right) - 1)$$

SVM: the linear case (**the dual form**)

- An optimization problem has a **dual form** if the function to be optimized and the constraints are strictly convex. Then the solution of the dual form is also a solution of the primal form of the optimization problem.

- First, define the **Lagrangian**

$$L(\mathbf{w}, w_0, \alpha) = \frac{1}{2} ||\mathbf{w}||^2 - \sum_{i=1}^{m} \alpha_i \left(y_i \left(\mathbf{w} \,.\, \mathbf{x}_i + w_0 \right) \right) - 1 \Big)$$

- The solution corresponds to a saddle point of the Lagrangian :

$$\frac{\partial L}{\partial \mathbf{w}}(\mathbf{w}, w_0, \alpha) = 0, \quad \frac{\partial L}{\partial w_0}(\mathbf{w}, w_0, \alpha) = 0$$

SVM: the linear case (**the dual form**)

- The **Lagrangian**

$$L(\mathbf{w}, w_0, \alpha) = \frac{1}{2} ||\mathbf{w}||^2 - \sum_{i=1}^{m} \alpha_i \left(y_i \left(\mathbf{w} \,.\, \mathbf{x}_i + w_0 \right) \right) - 1 \Big)$$

- Derivatives:

$$\frac{\partial L(\mathbf{w}, w_0, \alpha)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{m} \alpha_i \, y_i \, \mathbf{x}_i = 0$$

$$\frac{\partial L(\mathbf{w}, w_0, \alpha)}{\partial w_0} = - \sum_{i=1}^{m} \alpha_i \, y_i = 0$$

- Hence:

$$\sum_{i=1}^{m} \alpha_i \, y_i = 0 \quad \text{and} \quad \mathbf{w}^* = \sum_{i=1}^{m} \alpha_i \, y_i \, \mathbf{x}_i$$

Representer theorem

SVM: the linear case (**the dual form**)

$$\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i \, y_i \, \mathbf{x}_i$$

$$h^*(\mathbf{x}) = \text{sign}\left\{ (\mathbf{w}^{\star\top} \mathbf{x}) + w_0^\star \right\} = \text{sign}\left\{ \sum_{i \in \mathcal{P}_S} \alpha_i^\star \, u_i \, . \, \langle \mathbf{x}_i \, , \, \mathbf{x} \rangle + w_0^\star \right\}$$

- In order to find the α coefficients, one must solve the quadratic optimization problem:

$$\begin{cases} \text{Max}_\alpha \left[\sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j u_i u_j \langle \mathbf{x}_i \, , \, \mathbf{x}_j \rangle \right] \\ \text{avec} \quad \begin{cases} \alpha_i \geq 0, \quad i = 1, \ldots, m \\ \sum_{i=1}^m \alpha_i u_i = 0 \end{cases} \end{cases}$$

SVM: the linear case (**the dual form**)

- Under the **Karush-Kuhn-Tucker conditions**, only the points on the margin (hyperplanes) $(\mathbf{w}^* \mathbf{x}_i^\top) + w_0^* = \pm 1$ are involved.
- These points are known as *support vectors*

SVM: the linear case (**the dual form**)

- Under the **Karush-Kuhn-Tucker conditions**, only the points on the margin (hyperplanes) $(\mathbf{w}^* \mathbf{x}_i^\top) + w_0^* = \pm 1$ are involved.
- These points are known as *support vectors*
- So the solution can be **sparse**:

$$h^*(\mathbf{x}) = \text{sign}\left\{(\mathbf{w}^{\star\top}\ \mathbf{x}) + w_0^\star\right\} = \text{sign}\left\{\sum_{i \in \mathcal{P}_S} \alpha_i^\star\, u_i\, .\, \langle \mathbf{x}_i\ ,\ \mathbf{x}\rangle + w_0^\star\right\}$$

- And $w_0 = y_c - \mathbf{w}^\top \mathbf{x}_c = y_c - \sum_{i=1}^{m_c} y_i \alpha_i \mathbf{x}_i^\top \mathbf{x}_c$

(x_c,y_c) is one of the support vectors

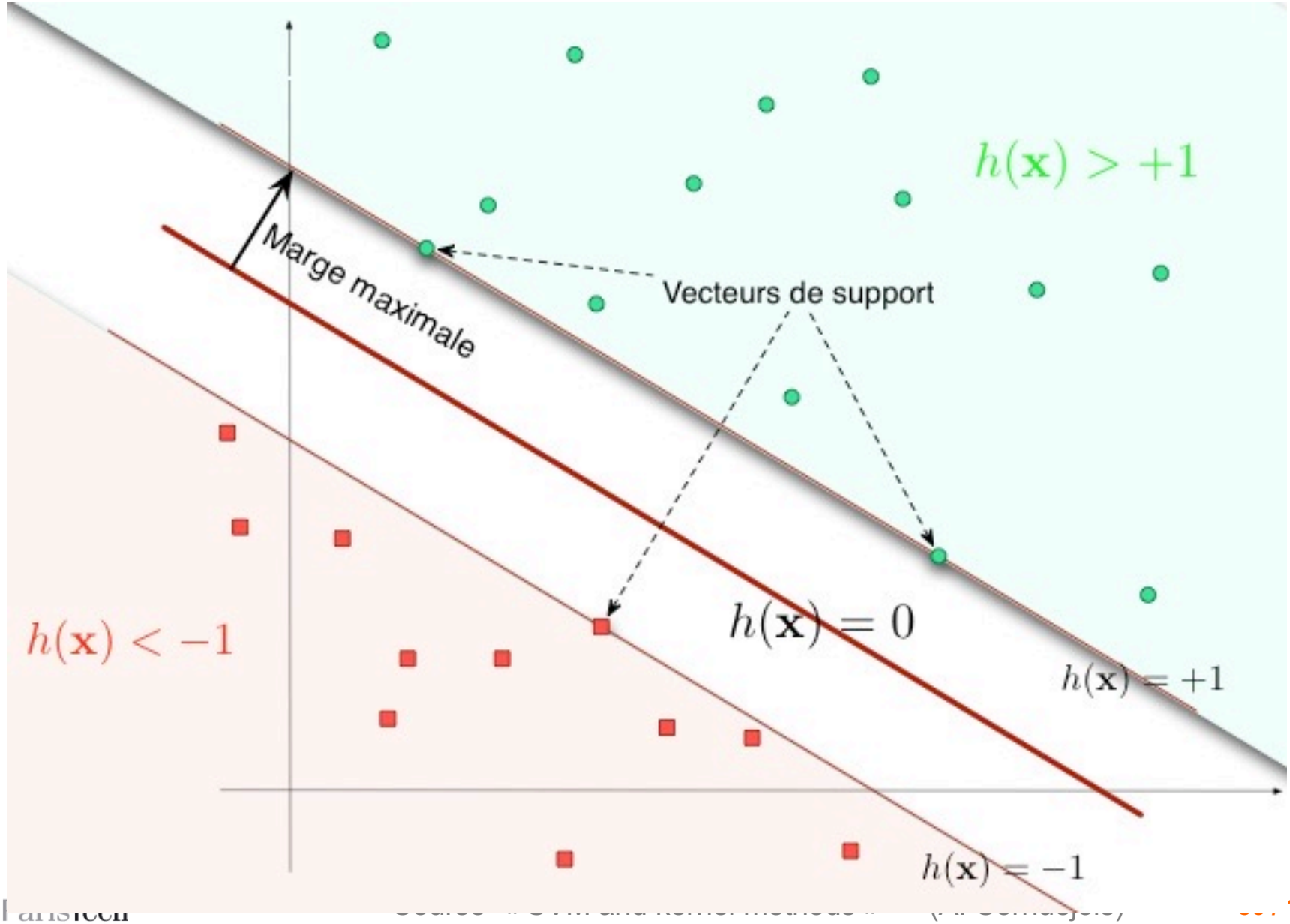
$$\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i\, y_i\, \mathbf{x}_i$$

Computing the threshold

Taking any two support vectors from opposite classes $\boldsymbol{x}_A \in \{\boldsymbol{x}_i : y_i = 1\,,\, \alpha_i^* > 0\}$ and $\boldsymbol{x}_B \in \{\boldsymbol{x}_i : y_i = -1\,,\, \alpha_i^* > 0\}$, calculate the threshold:

$$w_0^\star = -\frac{1}{2}\sum_{i=1}^{n} y_i \alpha_i^* (\boldsymbol{x}_i^T \boldsymbol{x}_A + \boldsymbol{x}_i^T \boldsymbol{x}_B)\,.$$

SVM: the linear case (**the dual form**)



Analysis

- In a way, this is not so different from a **nearest neighbor classifier**
  - The **relevant neighbors** are the support vectors
  - They are **weighted** with the α_i coefficients
  - The **similarity function** is given by **the dot product**
- The solution is entirely determined by the **Gram matrix**
  - Its characteristics tell a lot about the problem
- The solution $h(x)$ is a **linear combination**

Theoretical analysis

Very briefly

Bound on the generalization error

- If we were to use the Vapnik-Cervonenkis dimension
  - Which is *d*+1 for perceptrons in *d*-dimension
- We would get

$$R(h) \leq \widehat{R}(h) + \sqrt{\frac{2(d+1)\log\frac{e\,m}{d+1}}{m}} + \sqrt{\frac{\log\frac{1}{\delta}}{2\,m}}$$

Clearly uninformative when *d* is large compared to *m*

Bound on the generalization error if $f \in H$

- Instead, we can get

Radius of the smallest sphere enclosing all the training vectors

$$R(h) \leq \mathcal{O}\left(\frac{(\frac{R}{\gamma})^2 \ln^2 m + \ln(\frac{1}{\delta})}{m}\right)$$

margin

!!!

which **does not** depend on *d*!

Bound on the generalization error if $f \notin H$

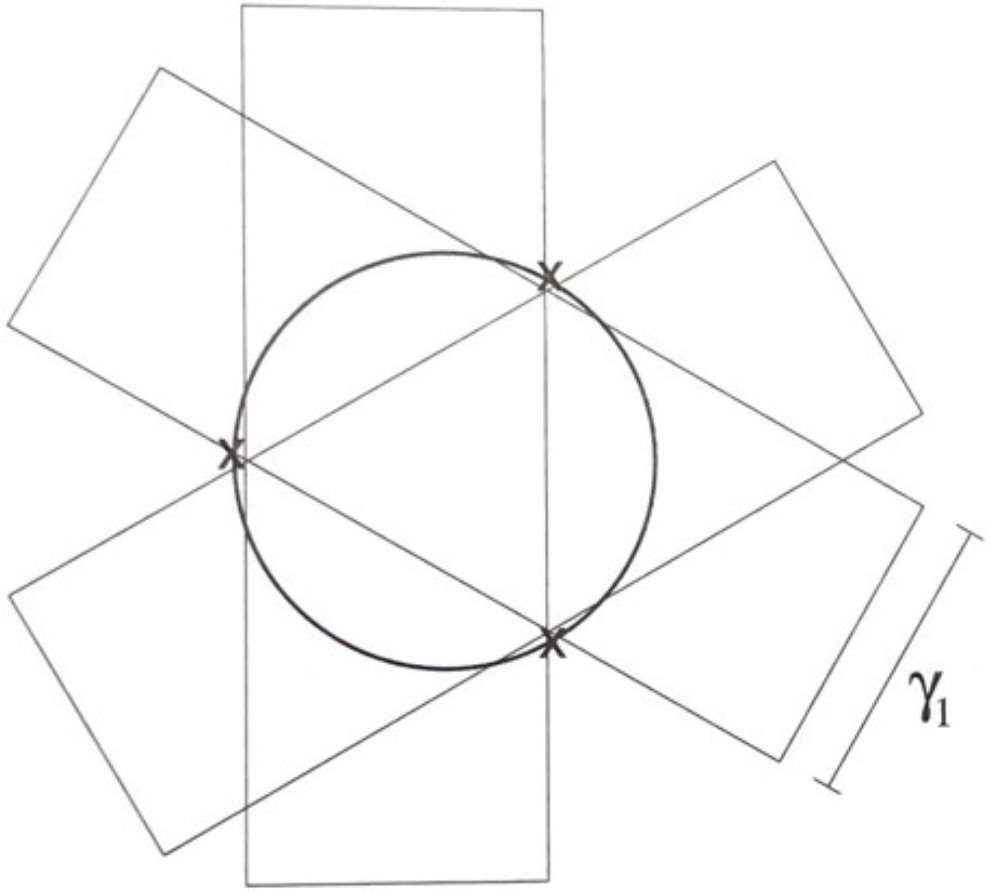
Radius of the smallest sphere enclosing all the training vectors

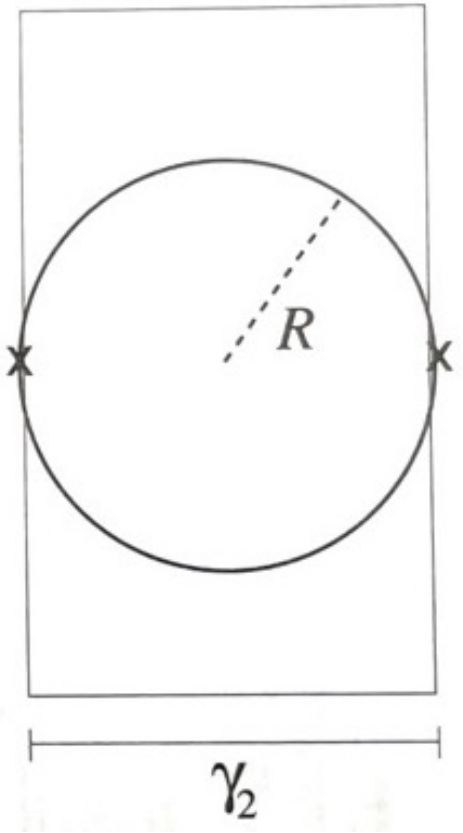
$$R(h) \leq \widehat{R}(h) + \mathcal{O}\left(\sqrt{\frac{(\frac{R}{\gamma})^2 \ln^2 m + \ln(\frac{1}{\delta})}{m}}\right)$$

margin

Note the **square root**

La marge comme contrôle de la capacité de *H*



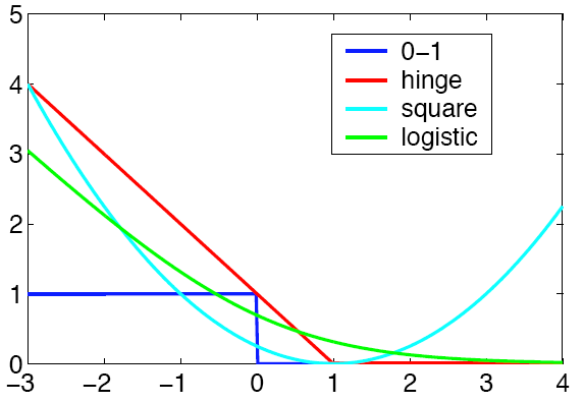


The margin

- Controls the convergence rate

- Automatically controls the capacity of the hypothesis
  - **Not** determined a priori (as in the SRM)!!
  - But **after** observing the learning examples

- The hinge loss acts as a regularizer



Lessons

- Capacity measure that is independent of the dimension
  - Even for linear separator
- Can be applied to infinite dimensional space!!
- Central for the application of **kernelized algorithms**

SVM et **méthodes à noyaux**

- Expression de **l'hypothèse** (fameuse « forme duale »)

$$h^*(\mathbf{x}) = \text{sign}\left\{ \sum_{i \in \mathcal{P}_S} \alpha_i^\star \; y_i \; \langle \mathbf{x}_i \, , \, \mathbf{x} \rangle + w_0^\star \right\}$$

- Trois idées
  - Hypothèses comme combinaison **linéaire**
  - Directement fonction des exemples (*exemples support*)
  - Minimise un risque régularisé dans lequel **la marge** mesure la versatilité de l'hypothèse

SVM

and **kernel** functions

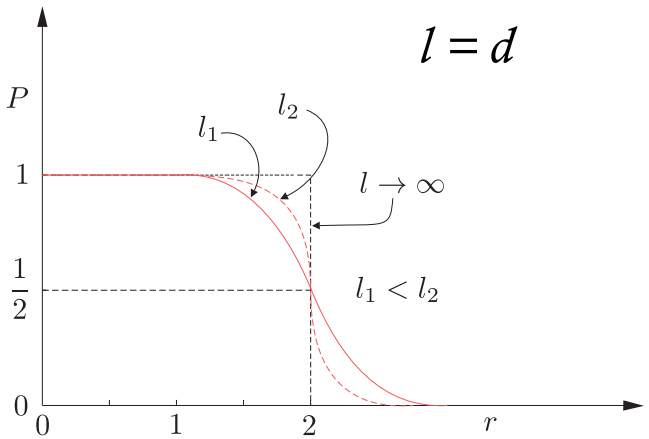
Outline

Cover's theorem

Theorem

Given m points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m \in \mathbb{R}^d$. The number of groupings $\mathcal{O}(m, d)$ that can be formed by $(d$ - $1)$- dimensional hyperplanes to separate the m points in two classes, exploiting all possible combinations is given by:

$$\mathcal{O}(m, d) = 2 \sum_{i=0}^{d} \binom{m-1}{i}$$



- For $m > 2(d + 1)$ the probability of linear separability becomes small.
- For large values of d, and provided that $m < 2\,(d + 1)$, the **probability** of any grouping of the data into two classes to be **linearly separable tends to 1**.

From the linear **to** the non linear case

- **Idea**: to increase the dimension of the input space

- By **generating** a whole lot of new dimensions
  - That result from combinations and functions of the initial dimensions (descriptive features)

- And **search a linear discriminant function** in this **redescription space**

Kernels vs. manually engineered Features

- Problems
  - Needs an **expert** in the domain (e.g. chinese characters)
  - Features may not be robust
  - Can be **expensive** to compute (e.g. control of the central board in chess)

- Solution
  - Use « **shotgun** » approach

    Compute a LARGE number of features and hope a good one is among them
  - Do this **efficiently**

Feature space mapping

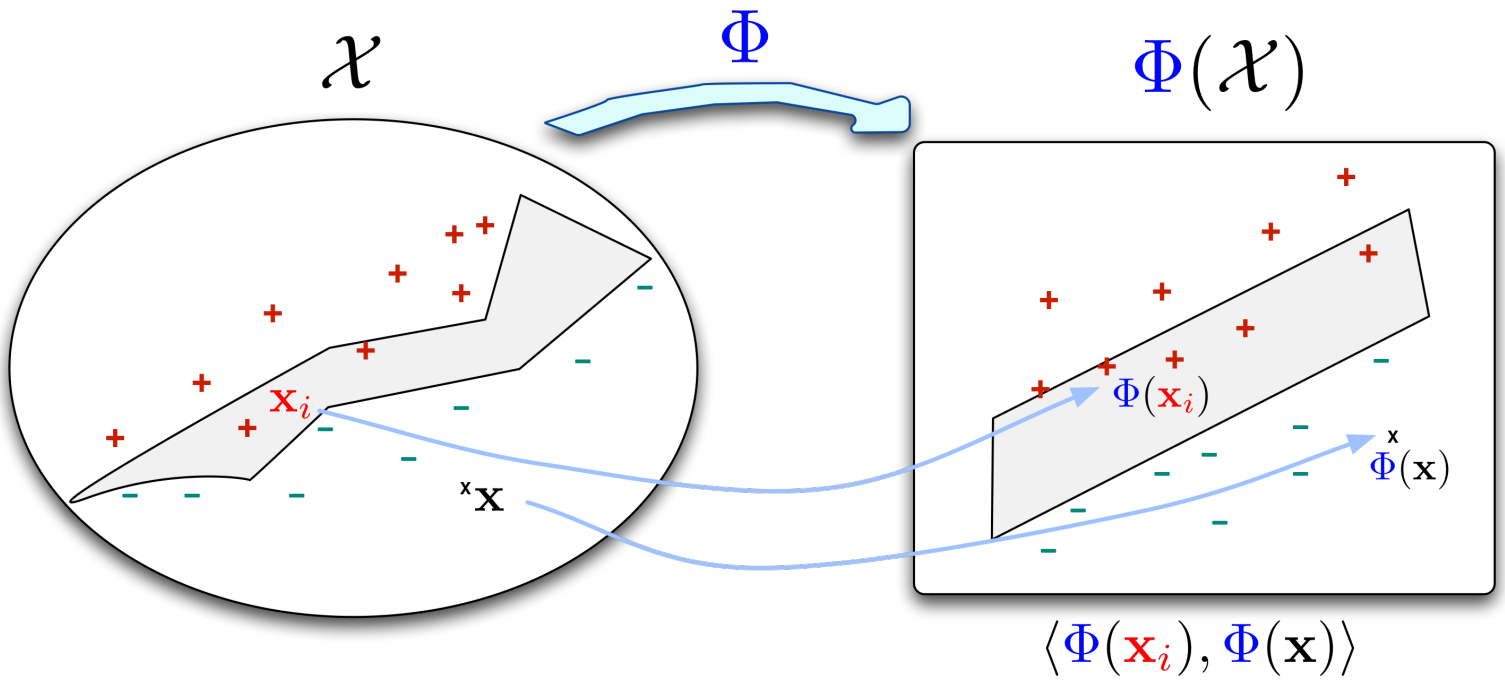
- Naive approach
  - Express data *x* in terms of features $\Phi(x)$
  - Solve problem in the feature space

  BUT requires **explicit feature** computation
- Use of Kernel trick
  - Write algorithm in terms of **inner products**
  - Replace $\langle x, x' \rangle$ by $K(x, x') = \langle \Phi(x), \Phi(x') \rangle$

  Provides a dimension-insensitive method

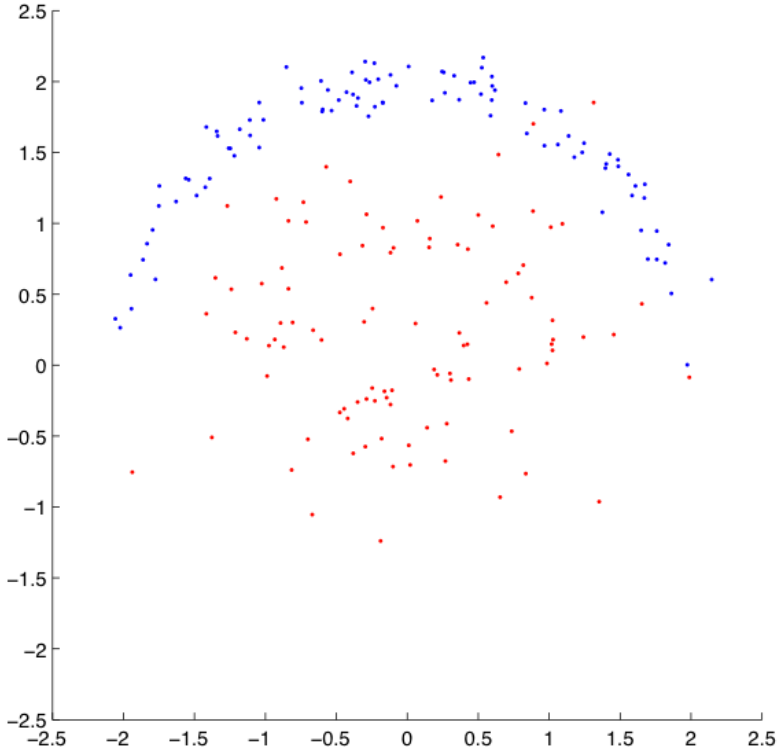
  Requires that kernel matrix *K* is positive semidefinite

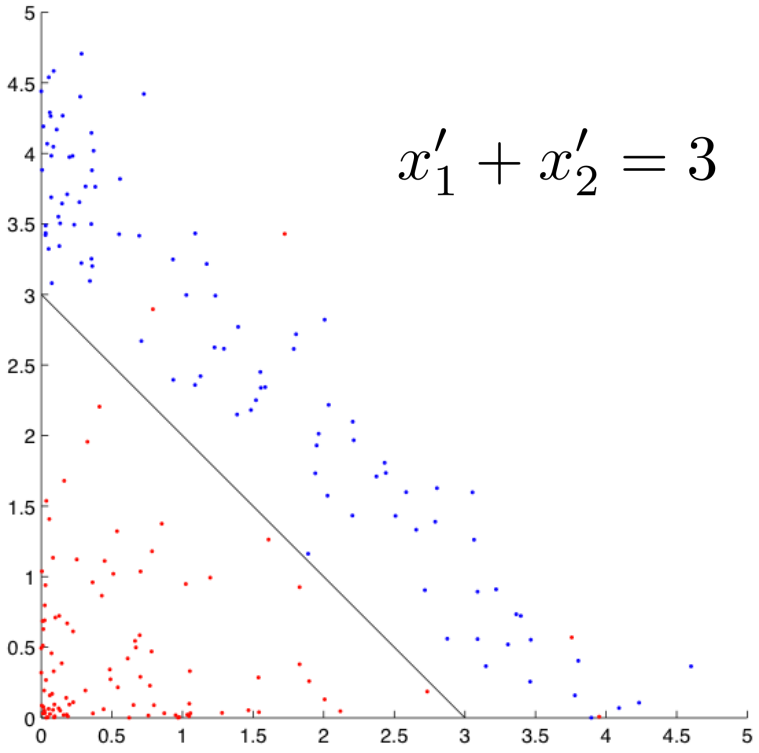
Projection in a redescription space



Change of representation

- From the non linear to the linear case, thanks to …

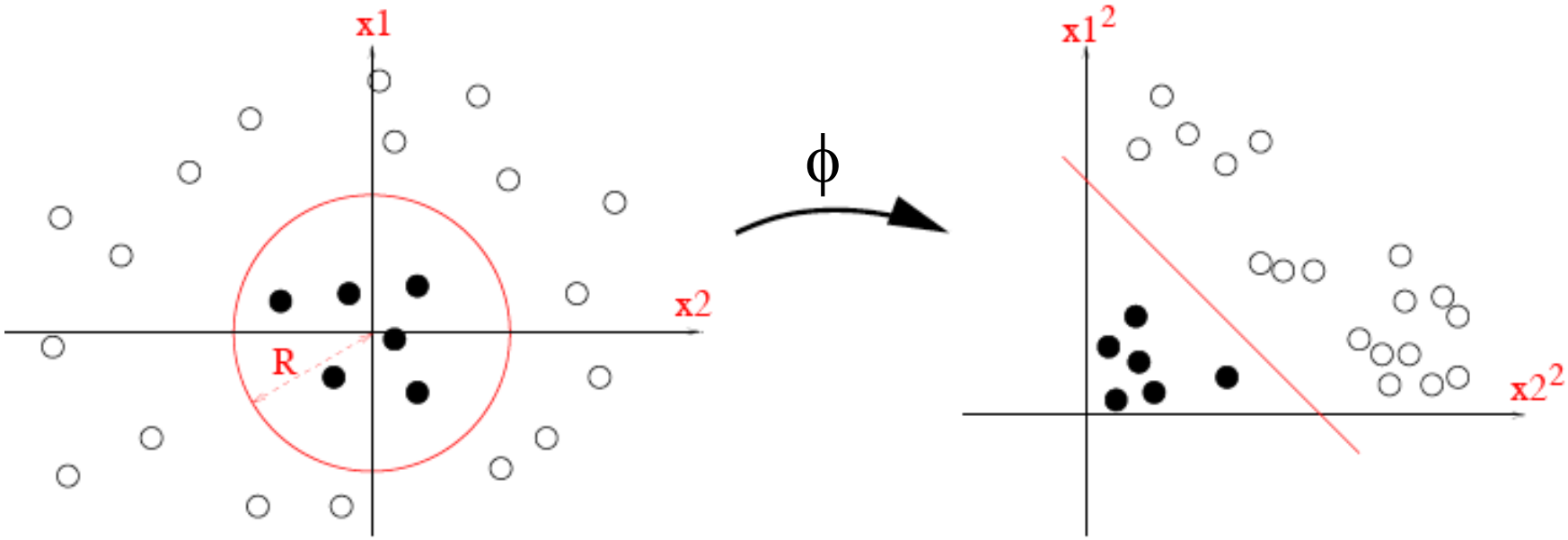




$$(x_1, x_2) \longrightarrow (x'_1, x'_2) = (x_1^2, x_2^2)$$

Change of representation

- Change of representation: the ideal case

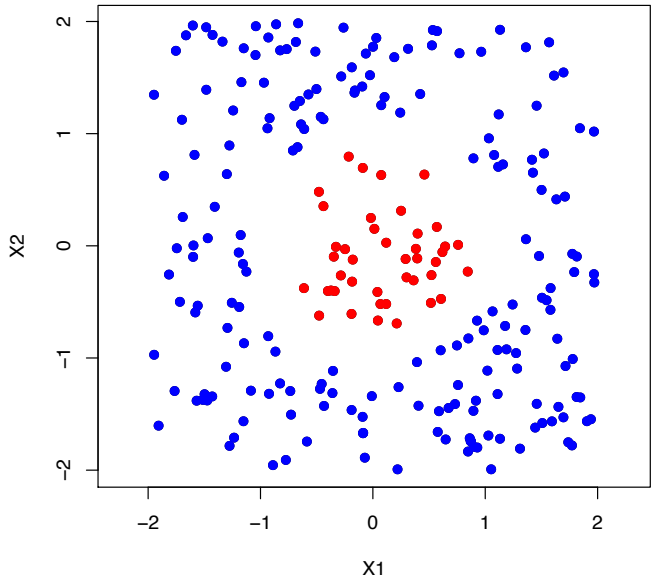


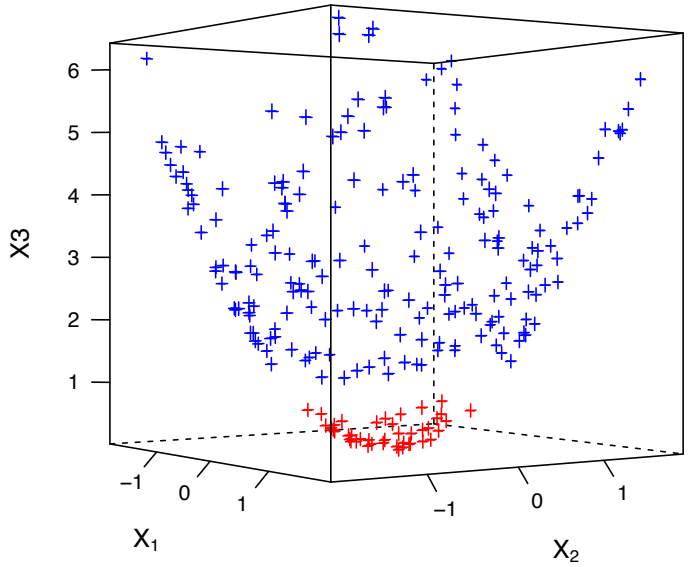
Let $\phi(\mathbf{x}) = (x_1^2, x_2^2)'$, $\mathbf{w} = (1,1)'$ and $b = 1$. Then the decision function is:

$$f(\mathbf{x}) = x_1^2 + x_2^2 - R^2 = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b,$$

Change of representation

- Change of representation: the ideal case

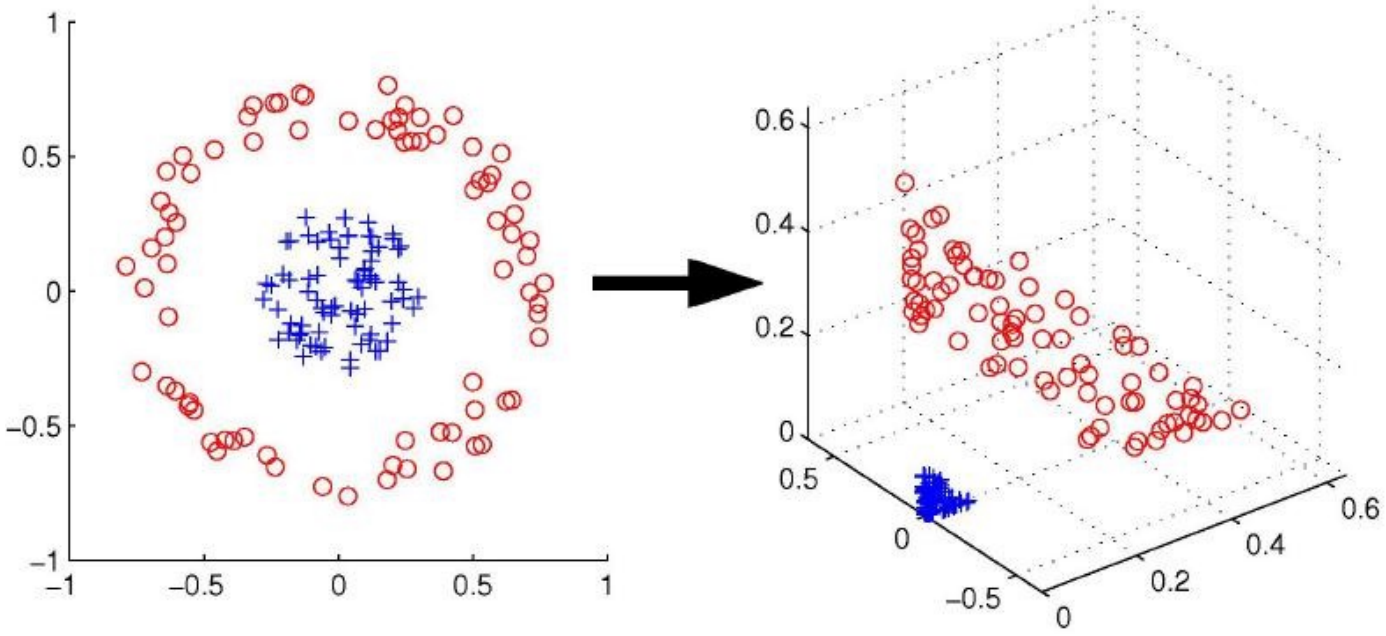




$$(x_1, x_2) \mapsto (\phi(x_1, x_2)) = (x_1, x_2, x_1^2 + x_2^2)$$

Change of representation

- Change of representation: the ideal case



(x_1, x_2) $(x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$

Change of representation

- **How to operate?**
- We are looking for:
  - A change of representation such that
    - A linear decision function exists between the classes
    - Obeying the true « similarity » between the data points
- Translation
  - Find a redescription space $\phi(\mathcal{X})$ (*feature expansion map*) with (very) high dimension
    - But **how to find $\phi(\mathcal{X})$?**
    - Under the additional constraint that computations are **tractable?**

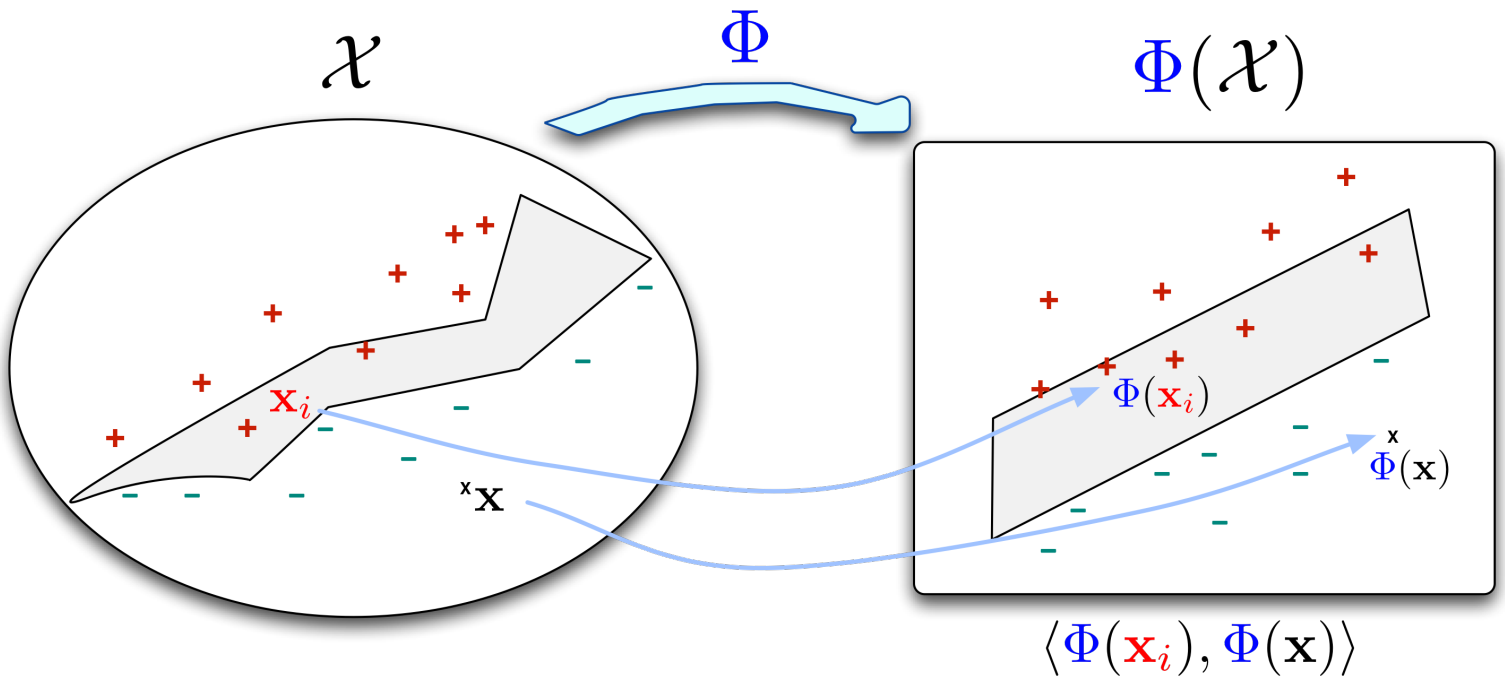
SVM: the linear case (**the dual form**)

$$h^*(\mathbf{x}) = \text{sign}\left\{(\mathbf{w}^{\star\top}\ \mathbf{x}) + w_0^\star\right\} = \text{sign}\left\{\sum_{i\in\mathcal{P}_S} \alpha_i^\star\, u_i\, .\, \langle \mathbf{x}_i\ ,\ \mathbf{x}\rangle + w_0^\star\right\}$$

- In order to find the α coefficients, one must solve the quadratic optimization problem:

$$\begin{cases} \text{Max}_\alpha \left[\sum_{i=1}^m \alpha_i \ -\ \frac{1}{2}\ \sum_{i,j=1}^m \alpha_i\alpha_j u_i u_j \langle \mathbf{x}_i\ ,\ \mathbf{x}_j\rangle\right] \\ \text{avec} \quad \begin{cases} \alpha_i \geq 0, \quad i = 1,\ldots,m \\ \sum_{i=1}^m \alpha_i u_i \ = \ 0 \end{cases} \end{cases}$$

Projection in a redescription space



SVM: the linear case in the feature space (**the dual form**)

$$h^*(\mathbf{x}) = (\mathbf{w}^\star \; \phi(\mathbf{x})) + w_0^\star = \sum_{i \in \mathcal{P}_S} \alpha_i^\star \, u_i \, . \, \langle \phi(\mathbf{x}_i) \, , \, \phi(\mathbf{x}) \rangle + w_0^\star$$

- In order to find the α coefficients, one must solve the quadratic optimization problem:

$$\begin{cases} \text{Max}_\alpha \left[\sum_{i=1}^m \alpha_i \; - \; \frac{1}{2} \; \sum_{i,j=1}^m \alpha_i \alpha_j u_i u_j \langle \phi(\mathbf{x}_i) \, , \, \phi(\mathbf{x}_j) \rangle \right] \\ \text{avec} \quad \begin{cases} \alpha_i \geq 0, \quad i = 1, \ldots, m \\ \sum_{i=1}^m \alpha_i u_i \; = \; 0 \end{cases} \end{cases}$$

Change of representation and kernel functions

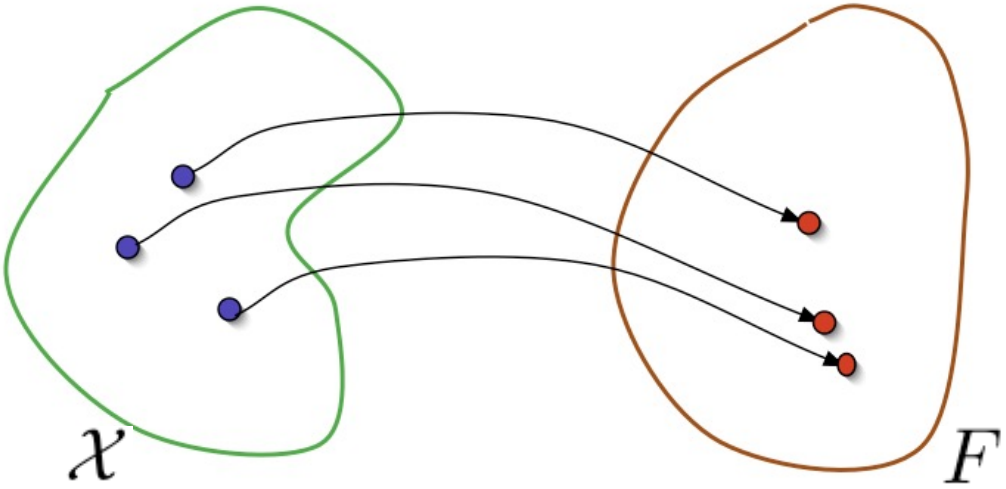
- The **kernel function** trick

  - Function ***k*** such that:

$$k : \mathcal{X} \times \mathcal{X} \rightarrow \Re$$
$$\forall \boldsymbol{x}, \boldsymbol{z} \in \mathcal{X} : \quad k(\boldsymbol{x}, \boldsymbol{z}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle$$

where: $\phi : \boldsymbol{x} \mapsto \phi(\boldsymbol{x}) \in F$

Redescription space with a dot product



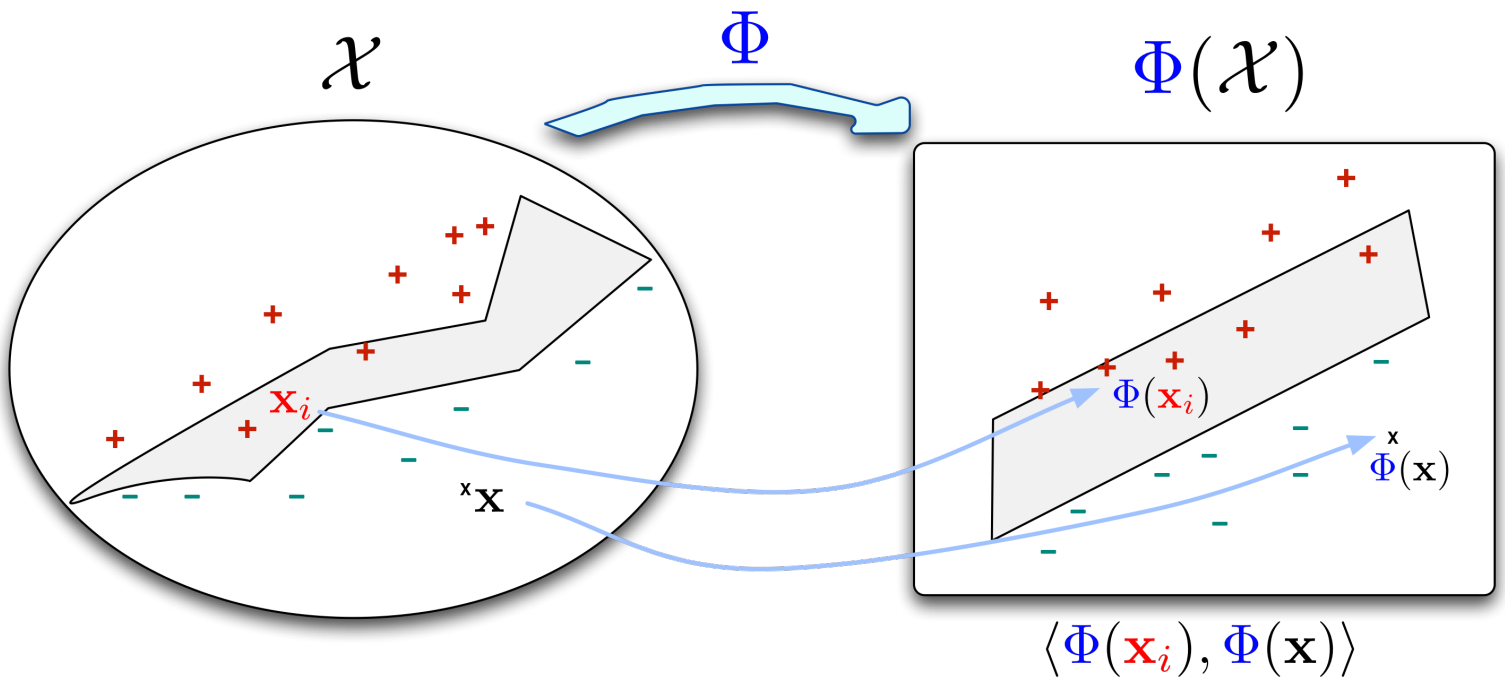
SVM: the linear case in the **input** space (**the dual form**)

$$h^*(\mathbf{x}) = (\mathbf{w}^\star \; \phi(\mathbf{x})) + w_0^\star = \sum_{i \in \mathcal{P}_S} \alpha_i^\star \, u_i \, . \, \kappa(\mathbf{x}_i \, , \, \mathbf{x}) + w_0^\star$$

- In order to find the α coefficients, one must solve the quadratic optimization problem:

$$\begin{cases} \text{Max}_\alpha \; \left[\sum_{i=1}^m \alpha_i \; - \; \frac{1}{2} \; \sum_{i,j=1}^m \alpha_i \alpha_j u_i u_j \kappa(\mathbf{x}_i \, , \, \mathbf{x}_j) \right] \\ \text{avec} \quad \begin{cases} \alpha_i \geq 0, \quad i = 1, \ldots, m \\ \sum_{i=1}^m \alpha_i u_i \; = \; 0 \end{cases} \end{cases}$$

SVM et méthodes à noyaux



$$h^*(\mathbf{x}) = \text{sign}\left\{ \sum_{i \in \mathcal{P}_S} \alpha_i^\star \; y_i \; \kappa(\mathbf{x}_i \, , \, \mathbf{x}) + w_0^\star \right\}$$

SVM: the gist

- **Support Vector Machines** (SVM)

$$h^*(\mathbf{x}) = (\mathbf{w}^\star \; \phi(\mathbf{x})) + w_0^\star = \sum_{i \in \mathcal{P}_S} \alpha_i^\star \, u_i \, . \, \kappa(\mathbf{x}_i \, , \, \mathbf{x}) + w_0^\star$$

- Représentation ***non paramétrique*** : fonction des exemples d'apprentissage
  - La complexité de l'hypothèse dépend de $|\mathcal{S}|$
- **Parcimonieux**
  - Seulement les $m' \leq m$ exemples support
  - Solution la plus robuste (large marge) $\Rightarrow$ donc nécessite moins de précision (le moins de bits)
- Ne dépend pas de d, mais de m'

Examples of kernel functions

Soit : $\mathcal{X} \subseteq \Re^2$

$$\phi : \quad \boldsymbol{x} = (x_1, x_2) \;\mapsto \phi(\boldsymbol{x}) = (x_1^2, x_2^2, \sqrt{2}\, x_1 x_2) \in F = \Re^3$$

$$h(\boldsymbol{x}) = w_{11}\, x_1^2 + w_{22}\, x_2^2 + w_{12}\, \sqrt{2}\, x_1 x_2$$

$$\begin{aligned} \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle &= \langle (x_1^2, x_2^2, \sqrt{2}\, x_1 x_2), (z_1^2, z_2^2, \sqrt{2}\, z_1 z_2) \\ &= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 x_2 z_1 z_2 \\ &= (x_1 z_1 + x_2 z_2)^2 = \langle x, z \rangle^2 \end{aligned}$$

$$k(\boldsymbol{x}, \boldsymbol{z}) = \langle \boldsymbol{x}, \boldsymbol{z} \rangle^2$$

is a kernel function

- Rk: The space *F* defined by Φ is not unique! :

$$\phi : \quad \boldsymbol{x} = (x_1, x_2) \;\mapsto \phi(\boldsymbol{x}) = (x_1^2, x_2^2, x_1 x_2, x_2 x_1) \in F = \Re^4$$

(the very same kernel function corresponds also to the dot product in this other space as well)

Kernel methods

Kernel functions for vectors

- **Polynomial kernels**

$$k_{\texttt{poly1}}(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x}^\top \boldsymbol{z})^d$$

All products of exactly d variables

$$k_{\texttt{poly2}}(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x}^\top \boldsymbol{z} + c)^d$$

All products of at most d variables

- **Gaussian kernels**

$$k_G(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\frac{d(\boldsymbol{x}, \boldsymbol{z})^2}{2\,\sigma^2}\right)$$

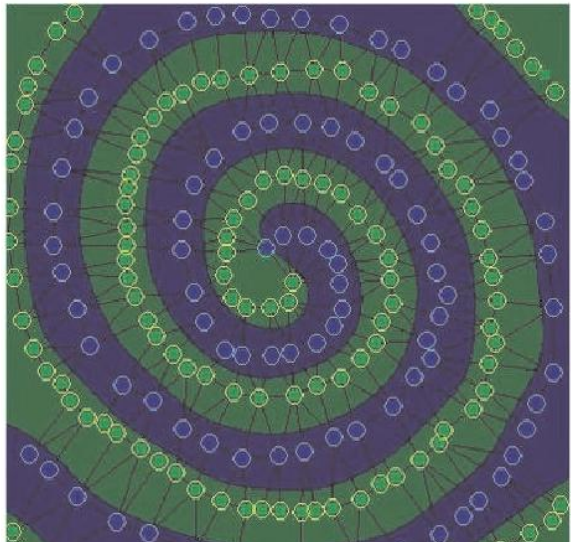
Related to a decomposition in Fourier series

- **Sigmoïd "kernel"**

$$k(\boldsymbol{x}, \boldsymbol{z}) = \tanh(\kappa\, \boldsymbol{x}^\top \boldsymbol{z} + \theta)$$

Not positive definite. But close to decision functions used in neural networks

Radius-Basis function kernel (RBF)



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

- No closed form Φ
- $\Phi(X)$ of infinite dimension
  For x in $\mathbb{R}$

$$\Phi(x) = exp\left(-\gamma x^2\right)\left[1, \sqrt{\frac{2\gamma}{1!}}x, \sqrt{\frac{(2\gamma)^2}{2!}}x^2, \sqrt{\frac{(2\gamma)^3}{3!}}x^3, \ldots\right]$$

- Choice of γ ? (intuition: think of H, $H_{i,j} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$)

Constructing kernel functions

- **Composition** rules

$$
\begin{aligned}
&(i) && \kappa(\mathbf{x},\mathbf{x}') = c\,\kappa_1(\mathbf{x},\mathbf{x}') \\
&(ii) && \kappa(\mathbf{x},\mathbf{x}') = f(\mathbf{x})\,\kappa_1(\mathbf{x},\mathbf{x}')f(\mathbf{x}') \\
&(iii) && \kappa(\mathbf{x},\mathbf{x}') = \mathit{poly}\big(\kappa_1(\mathbf{x},\mathbf{x}')\big) \\
&(iv) && \kappa(\mathbf{x},\mathbf{x}') = \exp\big(\kappa_1(\mathbf{x},\mathbf{x}')\big) \\
&(v) && \kappa(\mathbf{x},\mathbf{x}') = \kappa_1(\mathbf{x},\mathbf{x}') + \kappa_2(\mathbf{x},\mathbf{x}') \\
&(vi) && \kappa(\mathbf{x},\mathbf{x}') = \kappa_1(\mathbf{x},\mathbf{x}')\,\kappa_2(\mathbf{x},\mathbf{x}') \\
&(vii) && \kappa(\mathbf{x},\mathbf{x}') = \kappa_3\big(\Phi(\mathbf{x}),\Phi(\mathbf{x}')\big) \\
&(viii) && \kappa(\mathbf{x},\mathbf{x}') = \mathbf{x}^\top \mathbf{A}\,\mathbf{x}' \\
&(ix) && \kappa(\mathbf{x},\mathbf{x}') = \kappa_a(\mathbf{x}_a,\mathbf{x}'_a) + \kappa_b(\mathbf{x}_b,\mathbf{x}'_b) \\
&(x) && \kappa(\mathbf{x},\mathbf{x}') = \kappa_a(\mathbf{x}_a,\mathbf{x}'_a)\,\kappa_b(\mathbf{x}_b,\mathbf{x}'_b)
\end{aligned}
$$

Illustration

- Given 5 points on the line:

  $\{(x_1=1, u_1=1), (x_2=2, u_2=2), (x_3=4, u_3=-1), (x_4=5, u_4=-1), (x_5=6, u_5=1)\}$

- Let us use a **polynomial kernel of degree 2**
  - $k(x_i, x_j) = (x_i\, x_j + 1)^2$
  - C = 100
- One finds the α_i through:

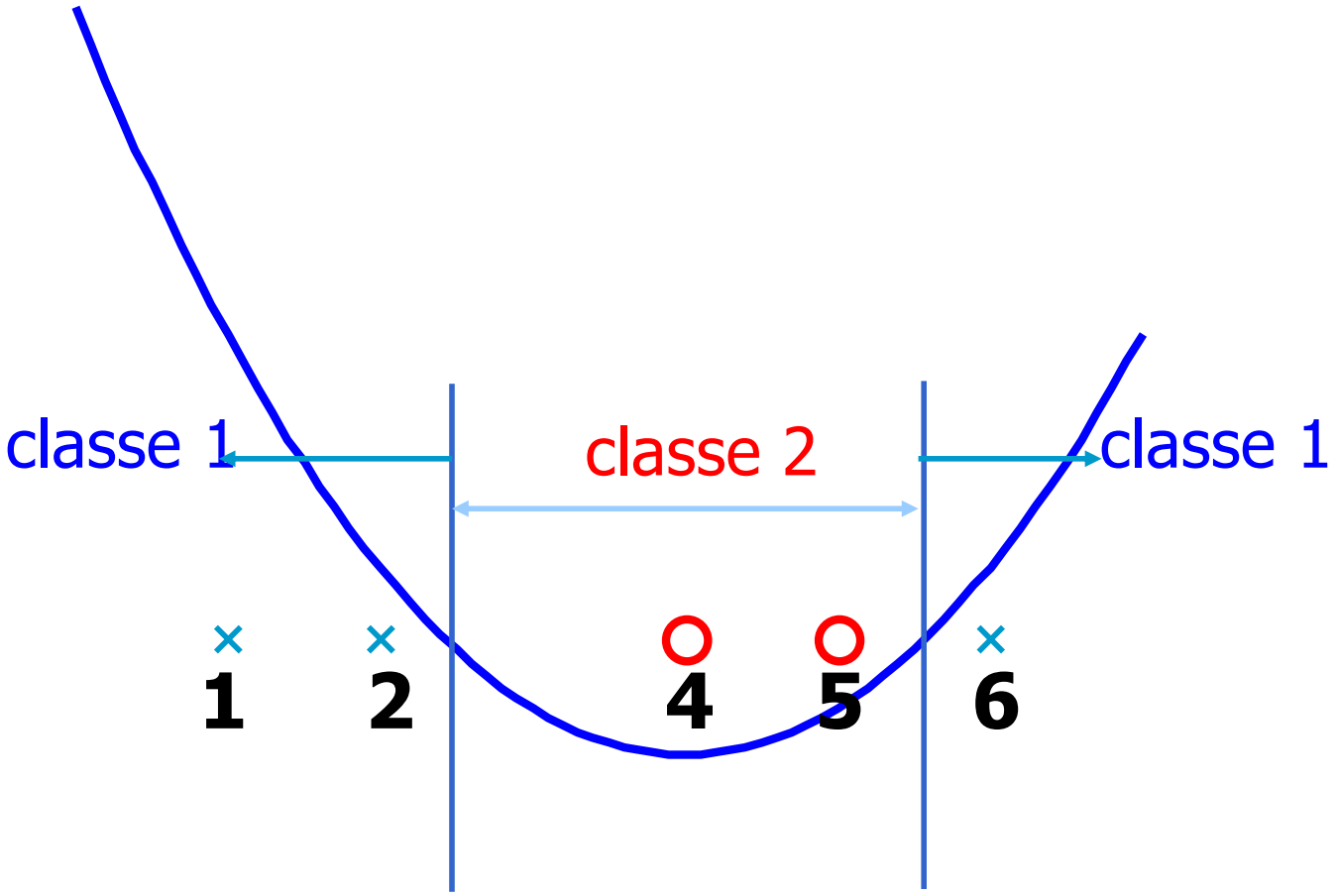
$$\begin{cases} \max\limits_{\alpha} \sum\limits_{i=1}^{5} \alpha_i - \frac{1}{2} \sum\limits_{i,j=1}^{5} \alpha_i \alpha_j u_i u_j \, (\boldsymbol{x}_i \cdot \boldsymbol{x}_j + 1)^2 \\ \sum\limits_{i,j=1}^{5} \alpha_i u_i = 0 \\ 0 \leq \alpha_i \leq 100 \quad (\forall i) \end{cases}$$

Illustration

- An algorithm that solves quadratic optimization problems will find
  - $\alpha_1 = 0, \quad \alpha_2 = 2.5, \quad \alpha_3 = 0, \quad \alpha_4 = 7.333, \quad \alpha_5 = 4.833$
  - The support vectors are: $\{ x_2 = 2, \; x_4 = 5, \; x_5 = 6\}$
- Giving the *decision function*:
  - $h(x) = (2.5)(1)(2x+1)^2 + 7.333(1)(5x+1)^2 + 4.833(1)(6x+1)^2+b$
    $= 0.6667\, x^2 - 5.333\, x + b$
  - With b obtained thanks to $h(2)=1$ or to $h(5)=-1$ or to $h(6)=1$, since x_2, x_4 and x_5 are on the line $u_i(w^T\Phi(x)+b) = 1$
    which gives $b = 9$
- Hence:

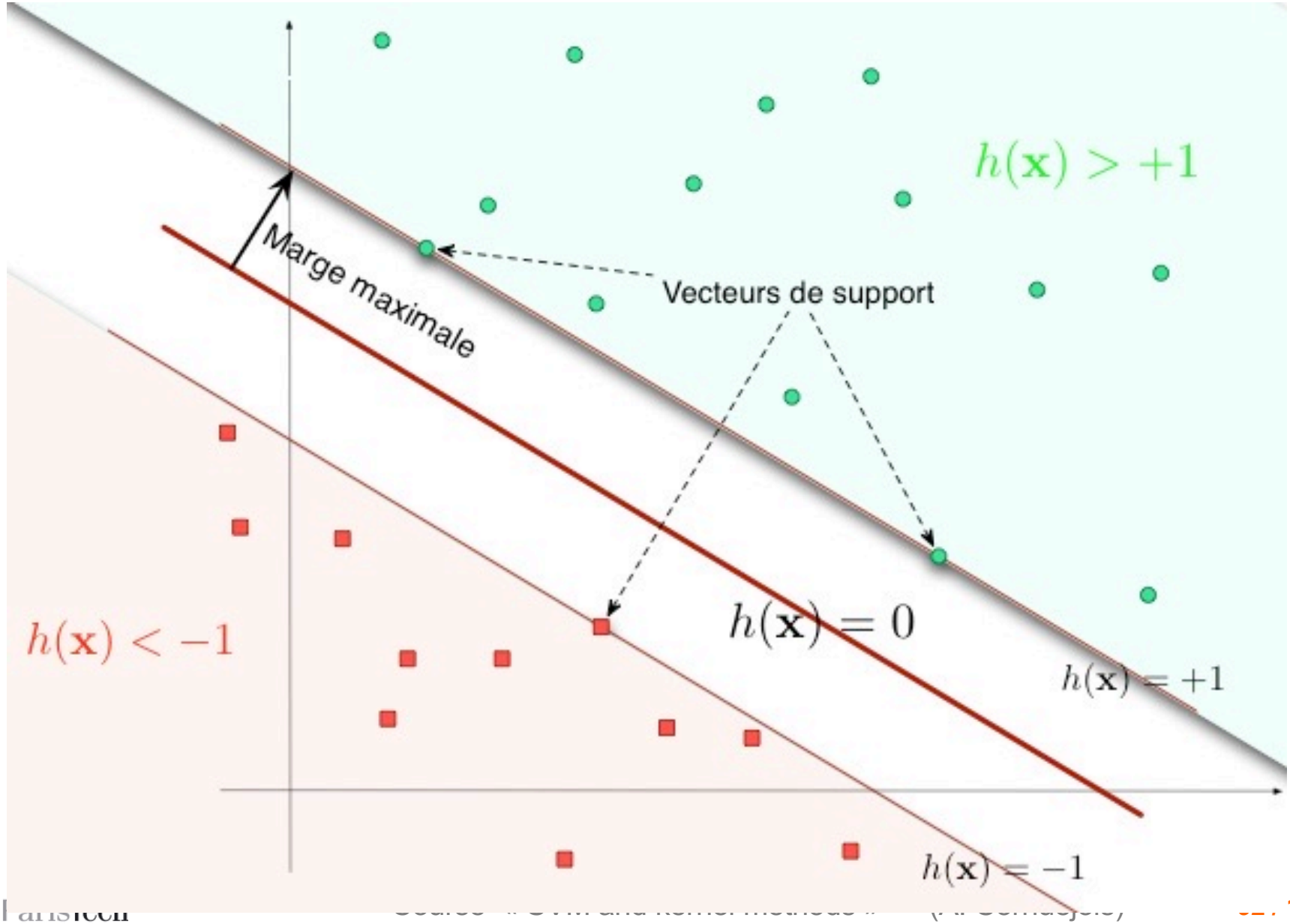
$$h(x) = 0.6667\, x^2 - 5.333\, x + 9$$

Illustration



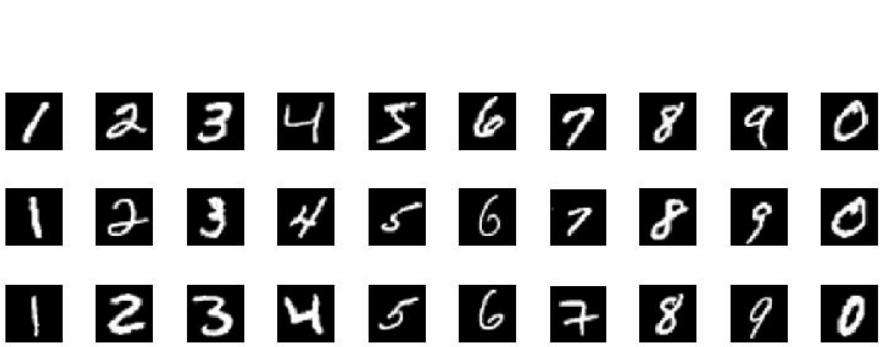
{$x = 2$, $x = 5$, $x = 6$} are the **support vectors**

SVM: the linear case (**the dual form**)

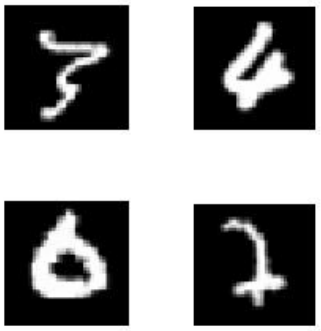


Written digits recognition

MNIST data



Data

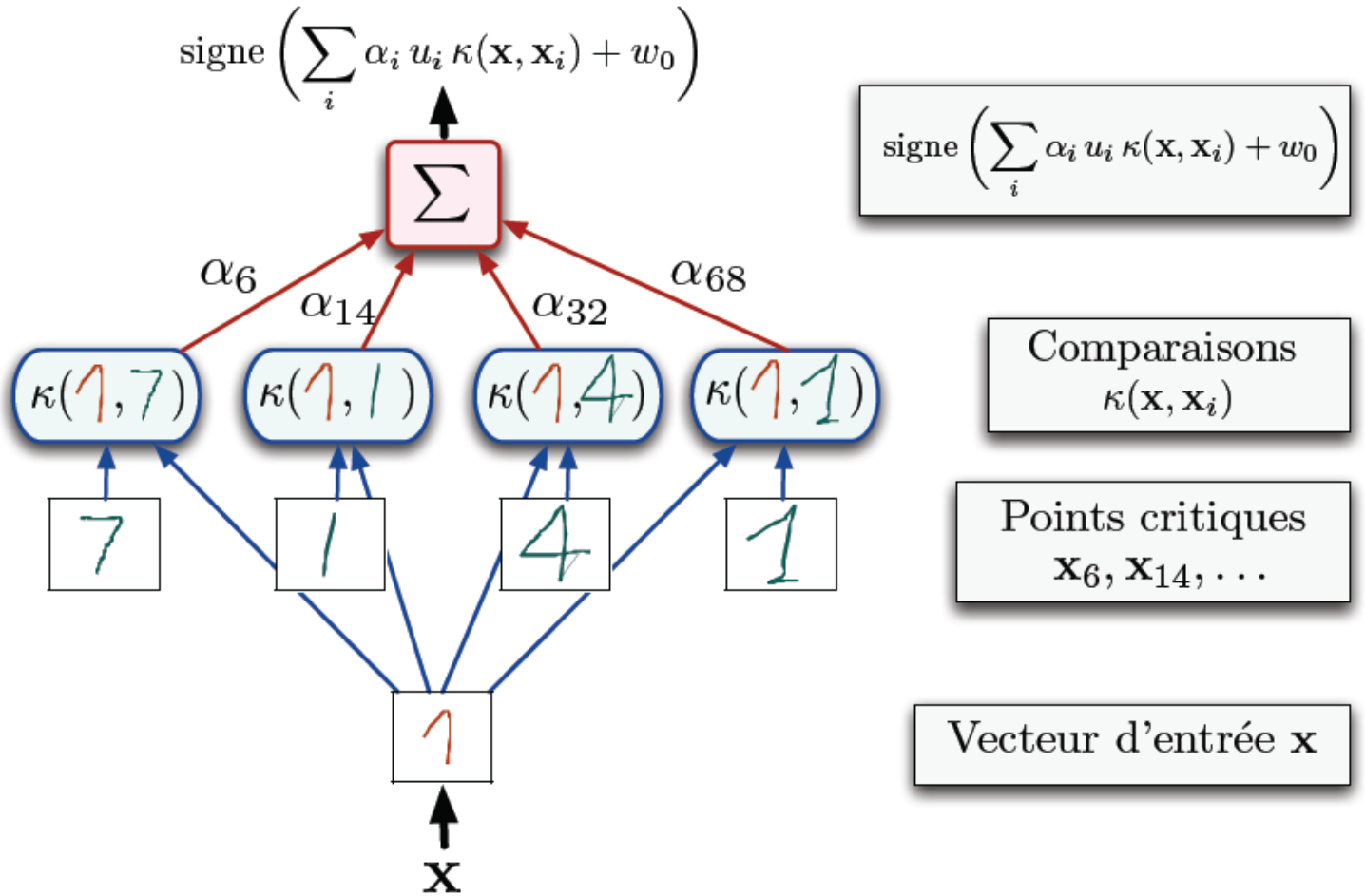


Support vectors

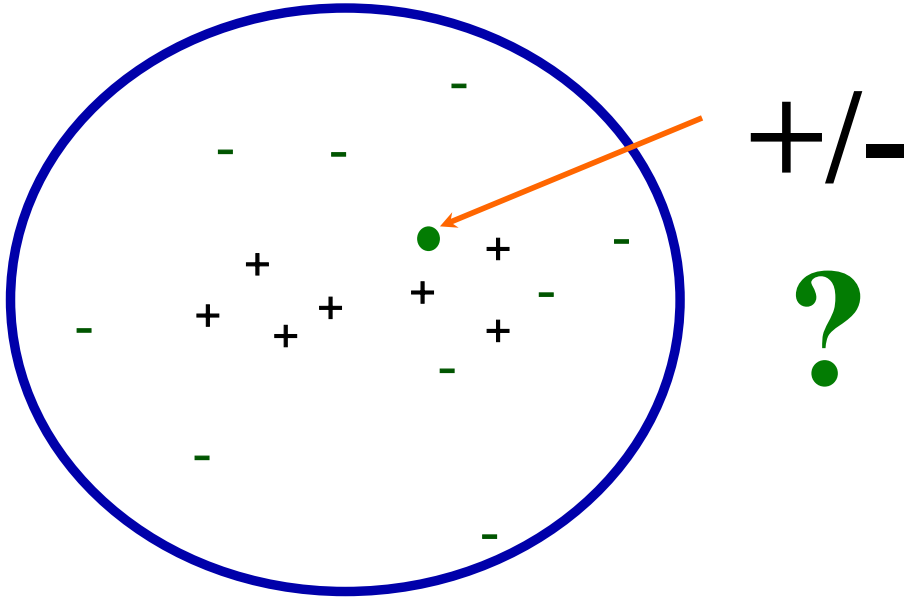
- Separate the 4s from the 7s

Another perspective

- **Support Vector Machines** (SVM)



Classification by nearest neighbours



Example space: $\mathcal{X}$

Three questions:

- How to choose the **relevant points**?
- How to **weight** the points?
- Which **distance** to use?

Questions and the "kernel trick"

- Is it more likely that the training set be **linearly separable** when it is projected in a high dimensional space?
  - Yes
- How to **chose** the new dimensions?
  - Automatically
- How to avoid the risk of **over-fitting**?
  - Simple
- How to get **tractable computations** in such a high dimensional space?
  - Not even a problem

Outline

In practice

In practice

- One must chose:
  - The **type of kernel function *k***
    - Its form
    - Its parameters
  - The **value of constant *C***

- Selecting these (meta) parameters is usually done using cross-validation

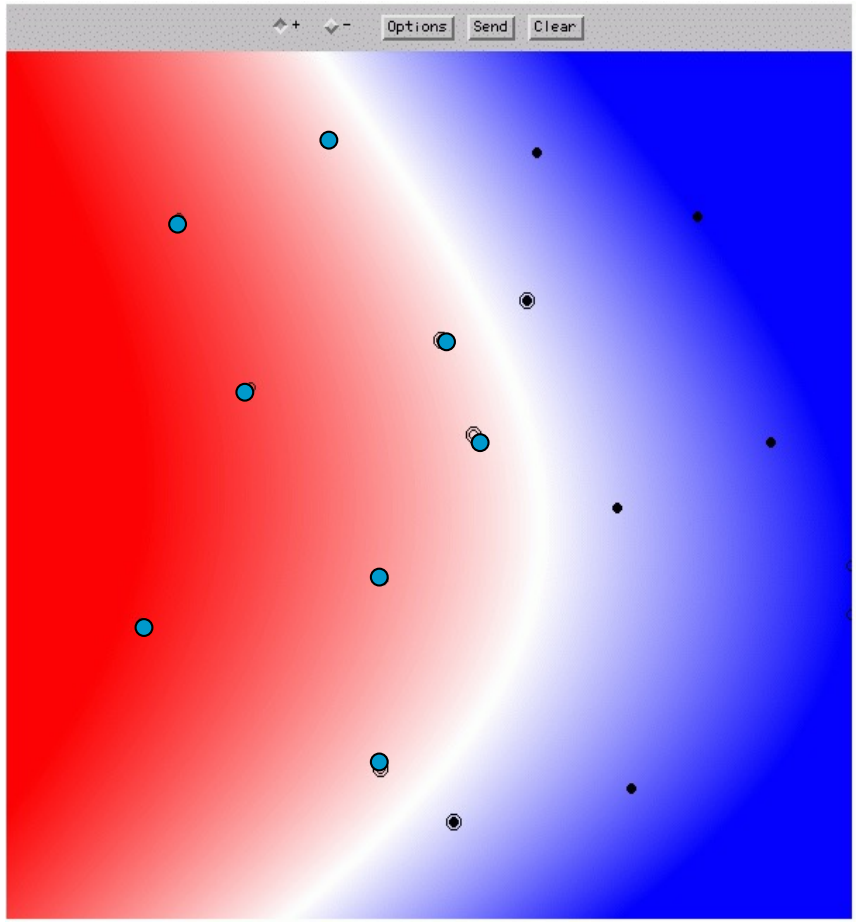
Example

: example +

: example -

Circles : support vectors

Kernel function: polynomial of degree 3



Démo :

http://svm.research.bell-labs.com/

http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml

A learning set

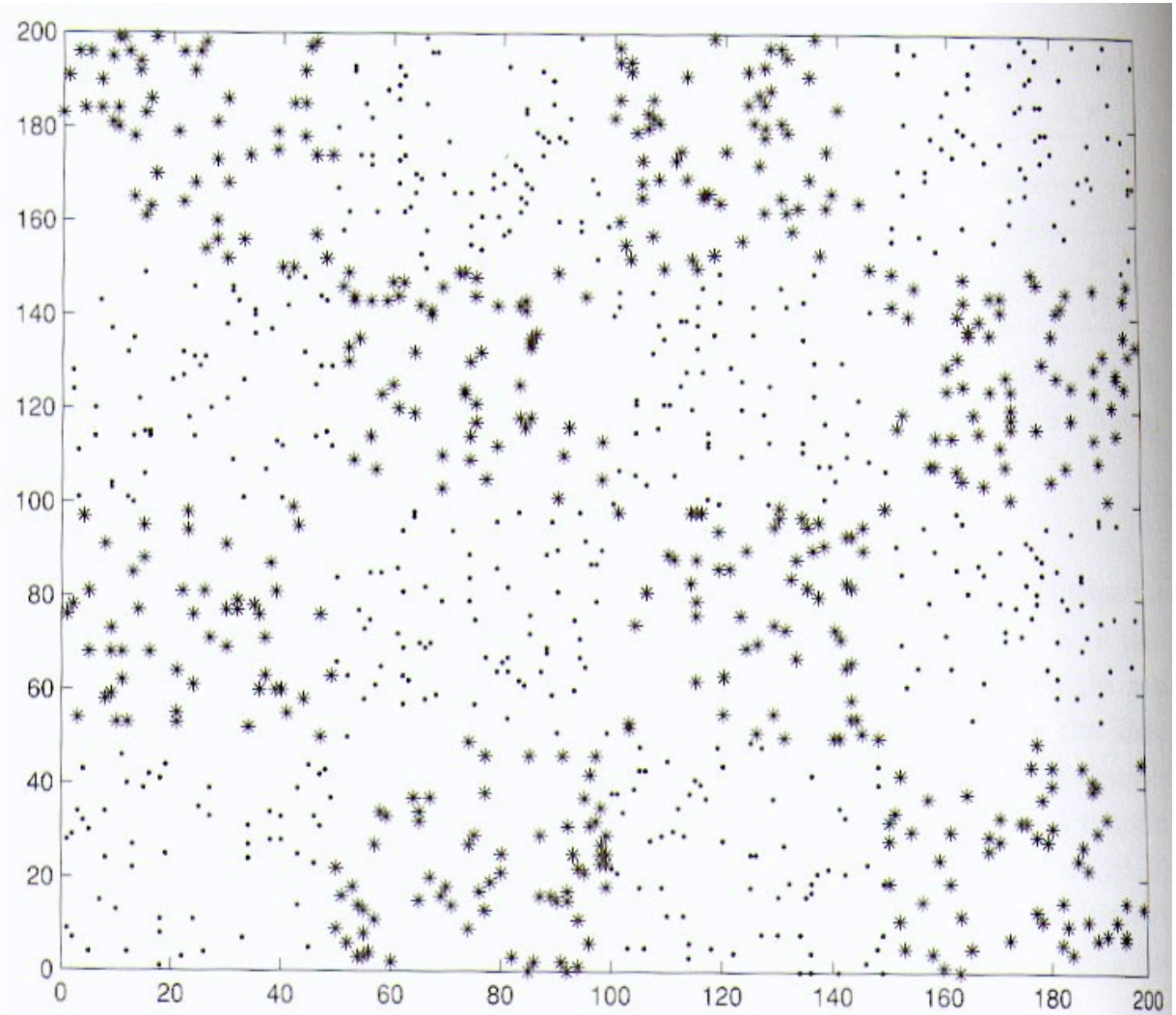
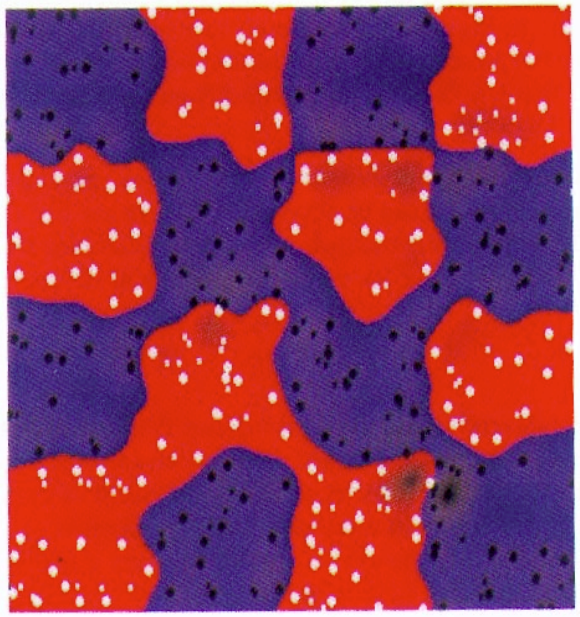


Figure 10.2 Training points for checkerboard pattern

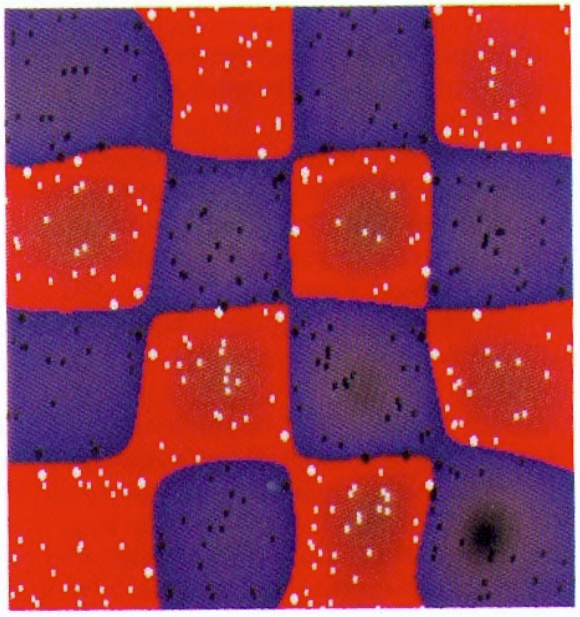
Impact of the control parameters

- Learning two classes
  - examples uniformly drawn on a checkerboard
- SVM à Gaussian kernels

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

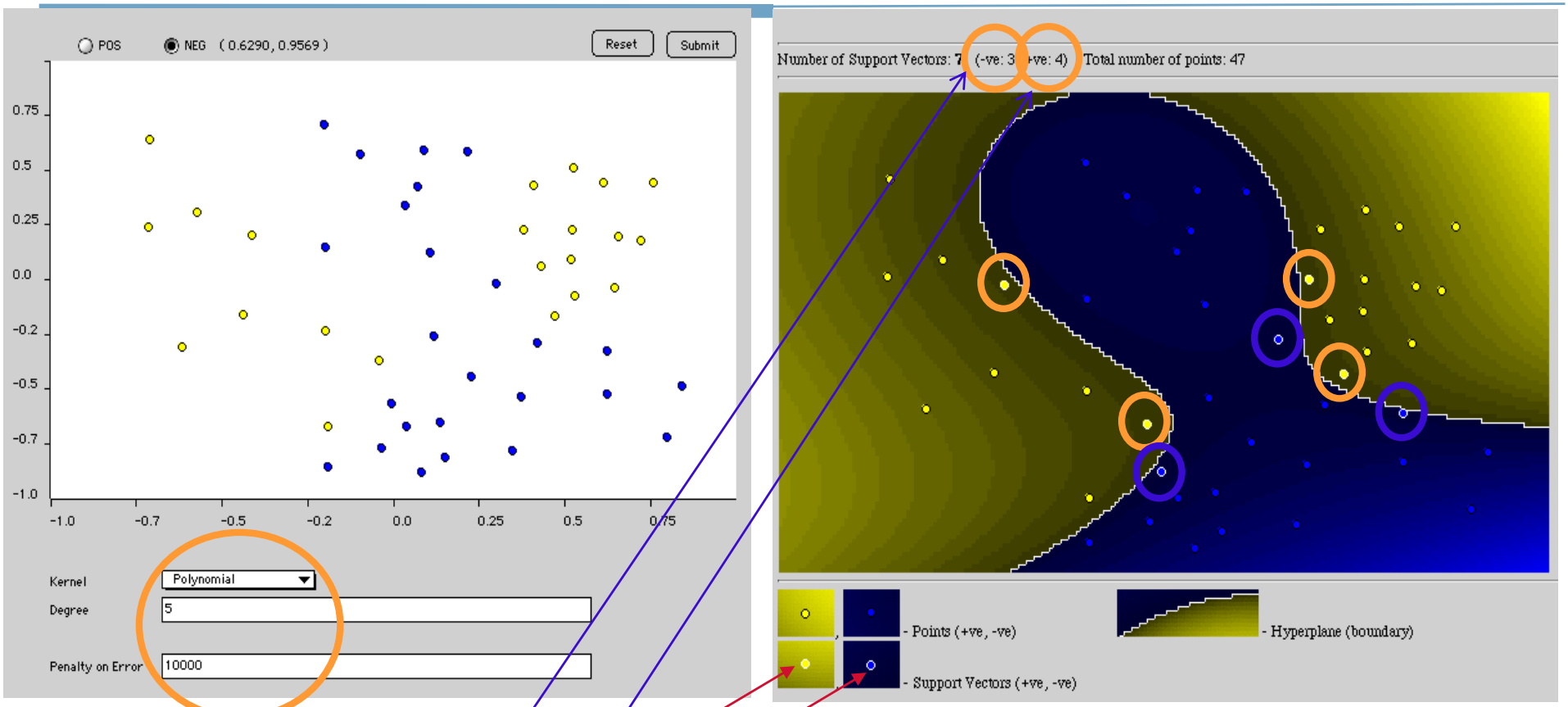


- Here two values of s are tested
  - **Top** **: a small value**
  - **Bottom** **: a large value**



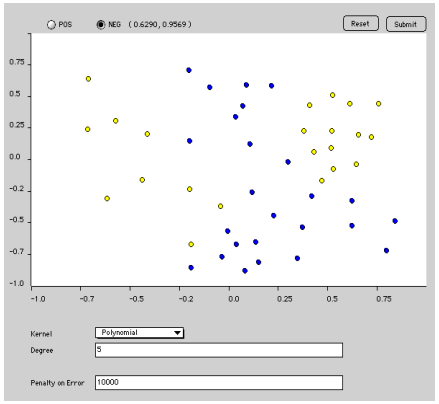
- The large points are support vectors
  - **There are more support vectors at the top than at the bottom**
- In both cases: $R_{emp} = 0$

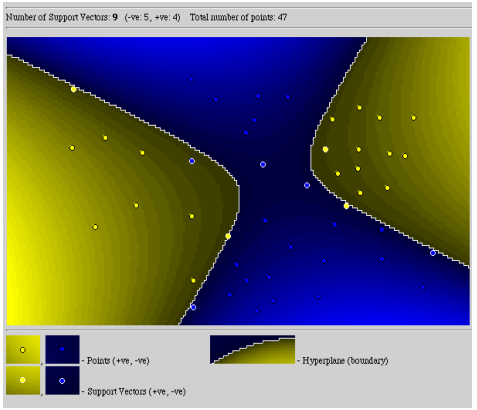
Control parameters: the kernel function

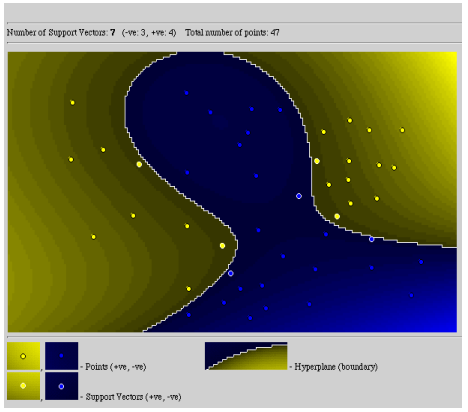


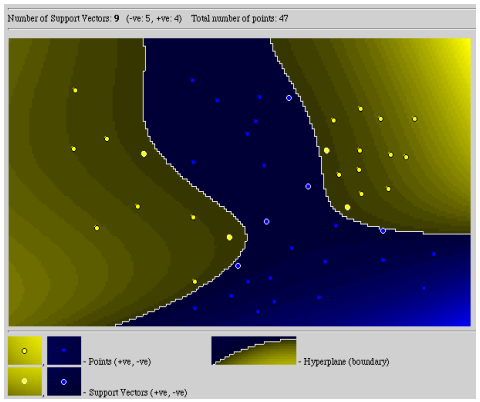
- http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml
- 47 examples (22 +, 25 -)
- *Support vectors*: 4 + et 3 –
- A *polynomial kernel* of *degree 5* and *C = 10000*

Control parameters: the kernel function





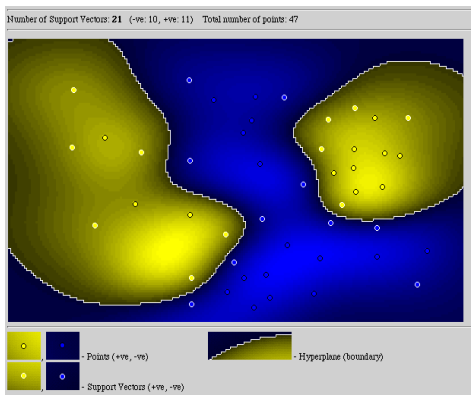


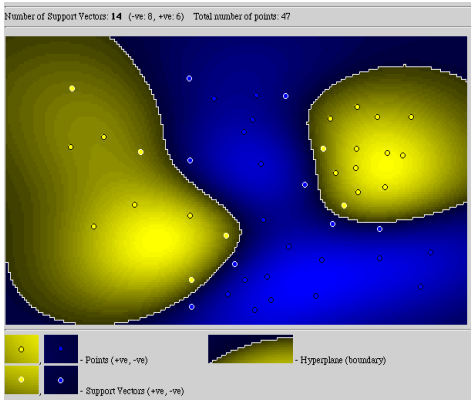


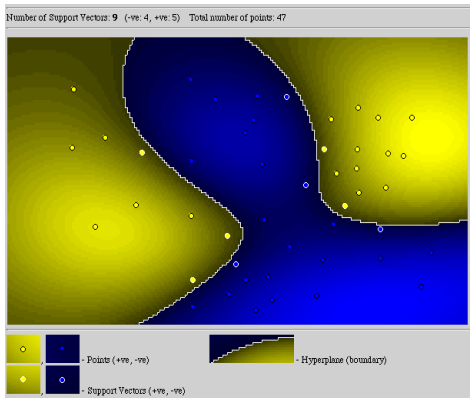
(5-, 4+) **(3-, 4+)** **(5-, 4+)**

- 47 examples (22 +, 25 -)
- *Support vectors*: 4 + et 3 -

Polynomial kernels **of *degree 2, 5* or 8 and $C = 10000$**



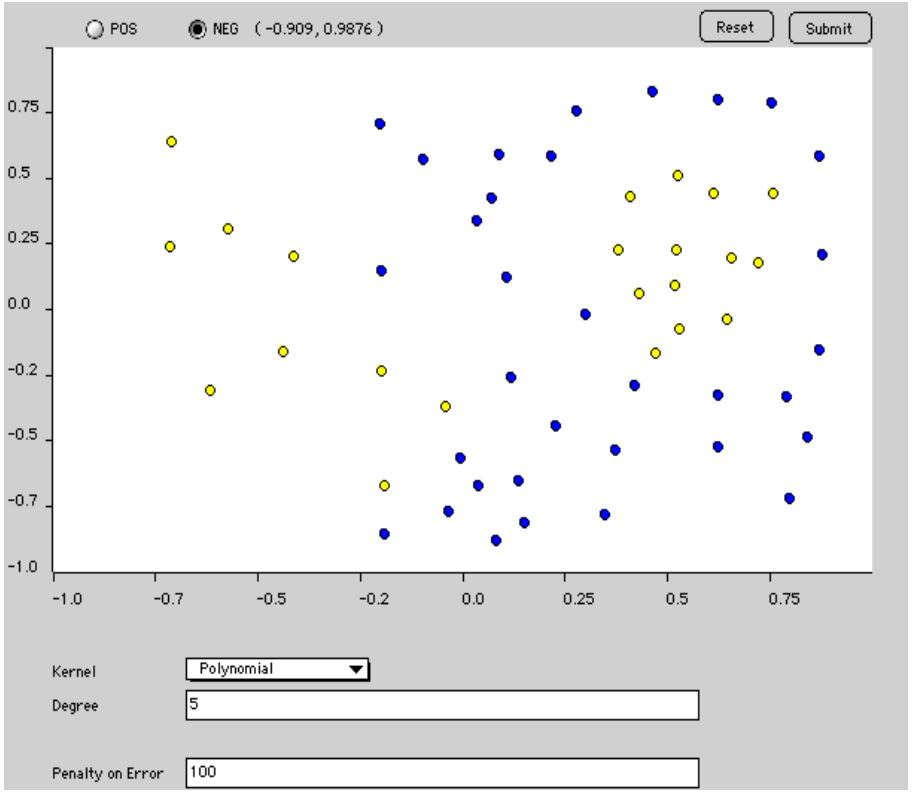


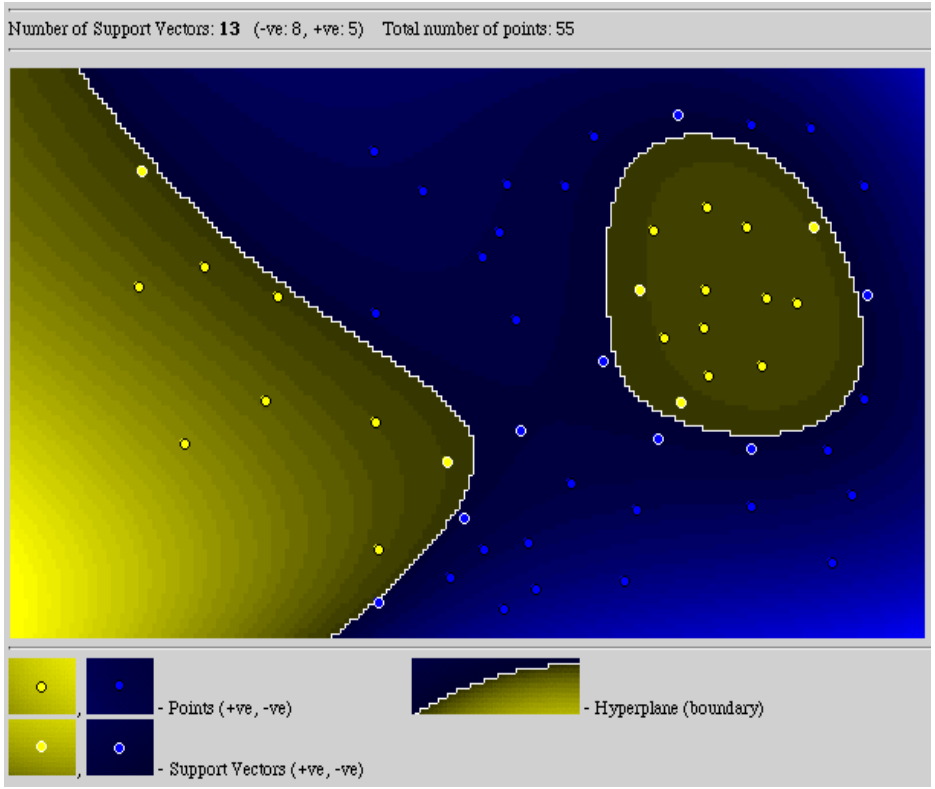


(10-, 11+) **(8-, 6+)** **(4-, 5+)**

Gaussian kernel **with $\sigma = 2, 5, 10$ or $C = 10000$**

Adding a few points ...





- http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml
  - **47 + 8 examples (30 +, 25 -)**
  - ***Support vectors*: 5 + and 8 –**
  - ***Polynomial kernel* of *degréee 5* and *C = 10000***

Controlling the C parameter

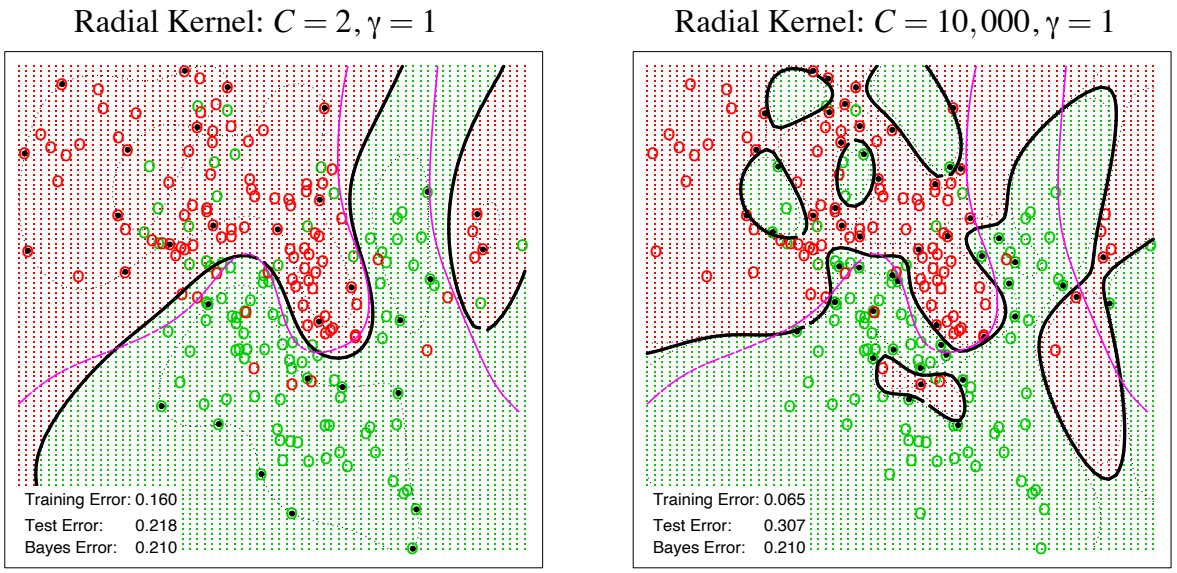


Figure 3: Simulated data illustrate the need for regularization. The 200 data points are generated from a pair of mixture densities. The two SVM models used radial kernels with the scale and cost parameters as indicated at the top of the plots. The thick black curves are the decision boundaries, the dotted curves the margins. The less regularized fit on the right overfits the training data, and suffers dramatically on test error. The broken purple curve is the optimal Bayes decision boundary.

Hastie, T., Rosset, S., Tibshirani, R., & Zhu, J. (2004). **The entire regularization path for the support vector machine**. *Journal of Machine Learning Research*, *5*(Oct), 1391-1415.

Bound on the generalization error

- For **any probability distribution** *D* on $\mathcal{X}$ x {-1, +1}, with probability 1 - δ, over *m* random examples in *S*, the maximal margin hyperplane has **generalization error** no more than

$$R(h) \leq \frac{1}{m - \#sv}\left(\#sv \log \frac{e\,m}{\#sv} + \log \frac{m}{\delta}\right)$$

#sv = nb of support vectors

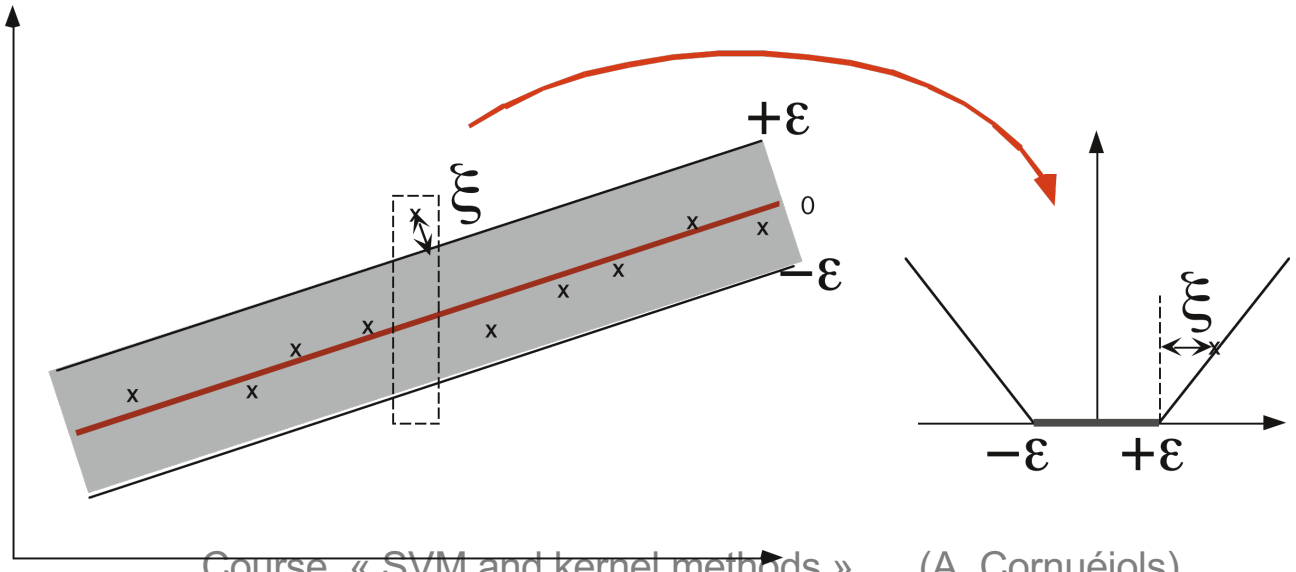
Estimation of the performance

- *Empirically*: using cross-validation

- *Heuristically* (but well-funded theoretically)
  - **Number of support vectors**
    - The less, the better
  - **Characteristics of the Gram matrix *G***
    - If there is no structure in *G*, no regularities can be found in the data
    - E.g.
      - If terms off-diagonal are very small: **over-fitting**
      - If the matrix is uniform: **under-fitting** (all points are labeled in the same class)

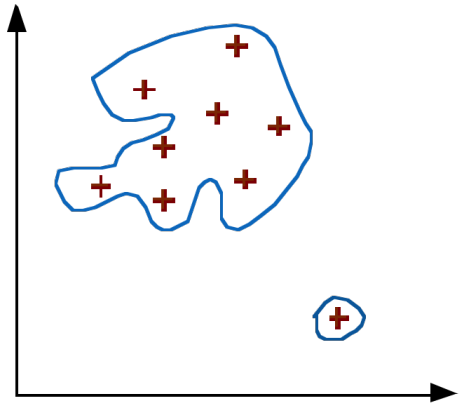
SVM et régression

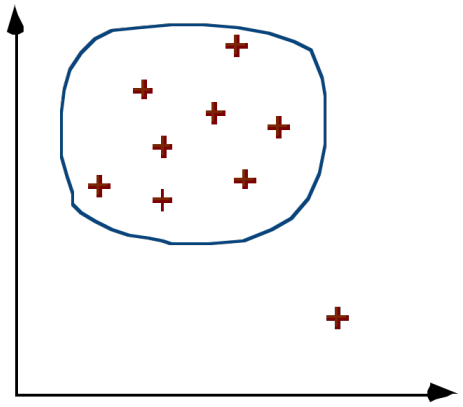
- Fonction de perte : $|y - f(\boldsymbol{x})|_\varepsilon = \max\{0, |y - f(\boldsymbol{x})| - \varepsilon\}$
  - **Régression linéaire :** $f(\boldsymbol{x}) := (\boldsymbol{w} \cdot \boldsymbol{x}) + w_0$
  - **Soit à minimiser :** $\frac{1}{2}\| \boldsymbol{w} \|^2 + C \sum_{i=1}^{m} |y_i - f(\boldsymbol{x}_i)|_\varepsilon$
  - **Généralisation :** $f(\boldsymbol{x}) = \sum_{i=1}^{m} (\alpha_i^\star - \alpha_i) K(\boldsymbol{x}_i, \boldsymbol{x}) + w_0$



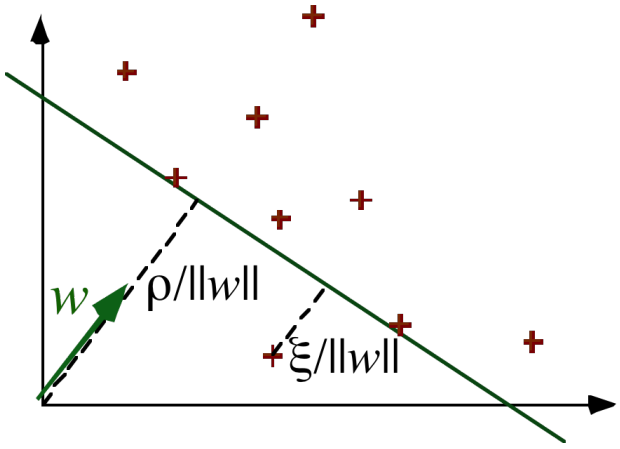
SVM et apprentissage non supervisé

- Détection de « nouveautés »





On cherche à séparer au maximum le nuage de points de l'origine

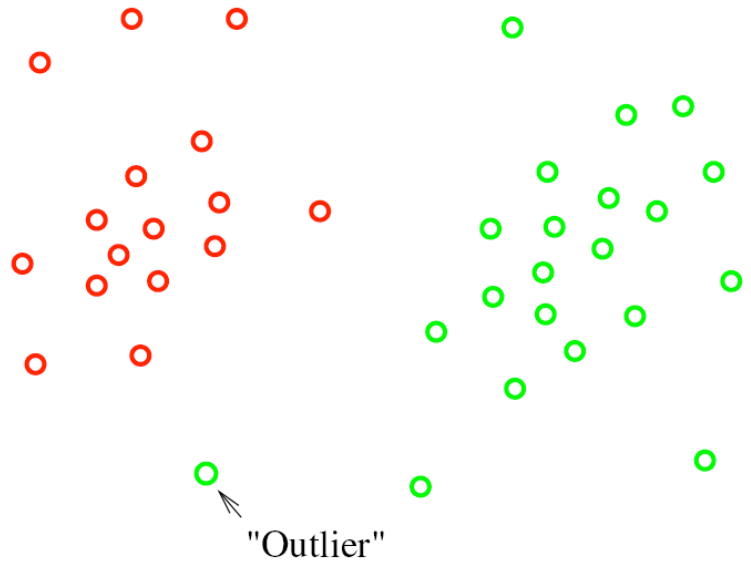


Soft-margin SVM

Dealing with **noisy** data

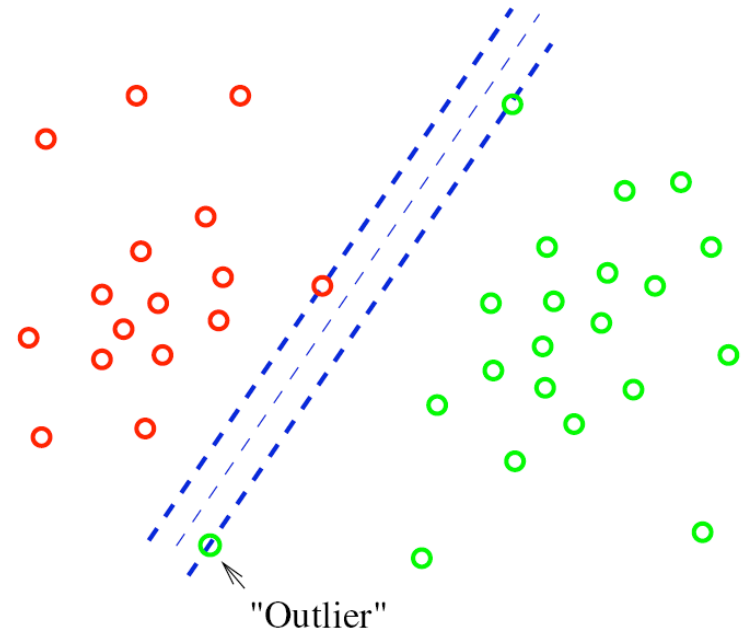
The perceptron: robustness

- But what if there is **noise** in the data or if **outliers** are present?



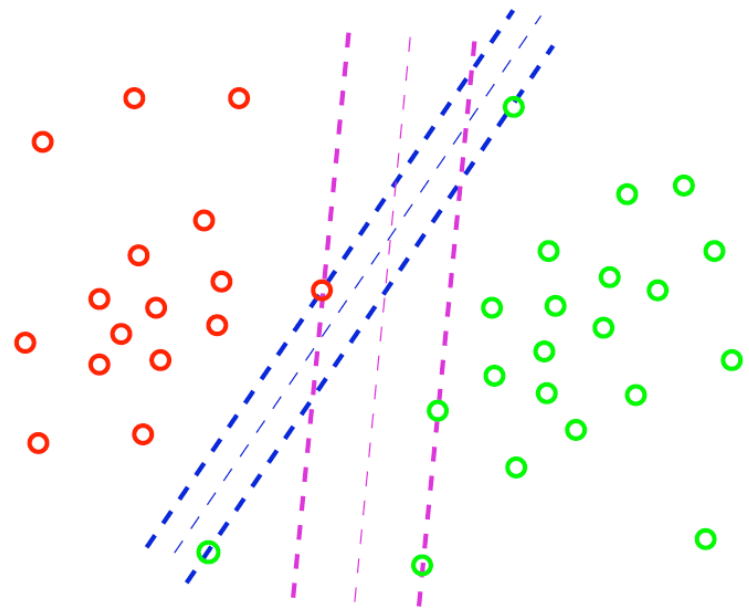
The perceptron: robustness

- Mais si il y a du **bruit** dans les données ou des **points aberrants** ?



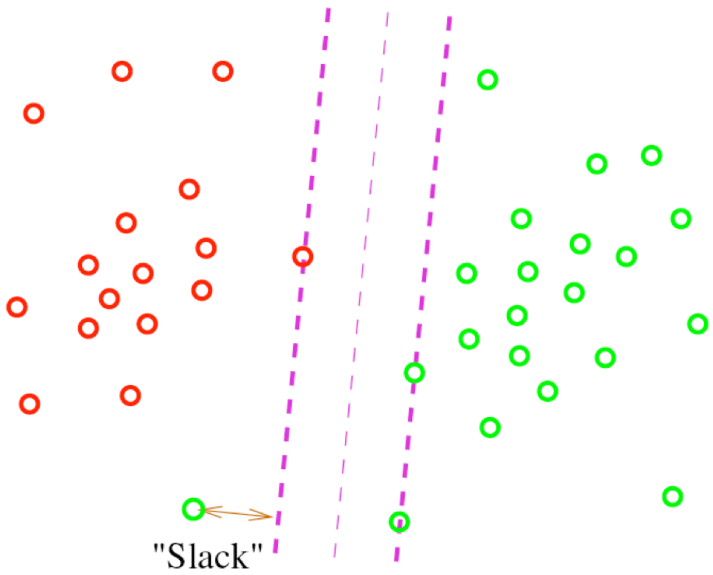
The perceptron: robustness

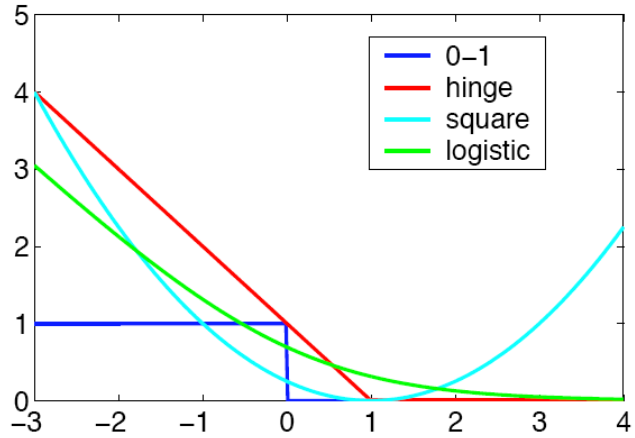
- One is ready to tolerate violations of the constraints
  (up to a certain limit set by a constant *C*)



SVM with soft margins

- We are ready to **tolerate violations of the constraints** set by the learning examples (up to a limit set by the constant *C*)





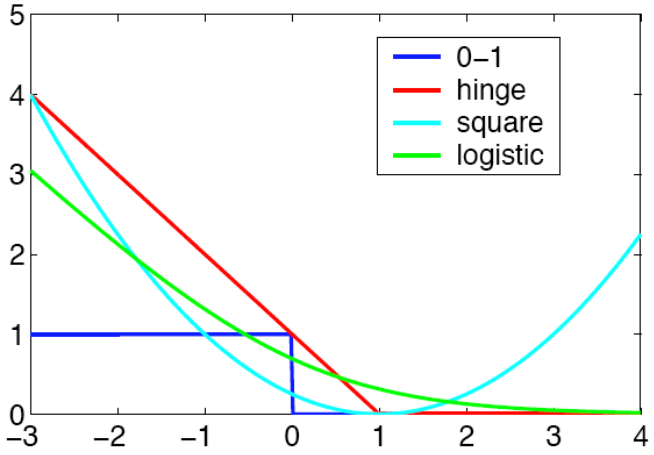
- « *surrogate loss functions* »
- The **hinge loss** is the most used
  - For some good theoretical reasons

SVM

- La recherche de la marge maximale conduit au **critère** :

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\text{ArgMin}} \Big[\underbrace{\sum_{i=1}^{m} |1 - y_i h(\mathbf{x}_i)|_+}_{\text{Risque empirique}} + \underbrace{\frac{1}{2}\mathbf{w}^\top \mathbf{w}}_{\text{Marge}} \Big]$$

Fonction de *perte de substitution* (surrogate loss)



Le critère inductif

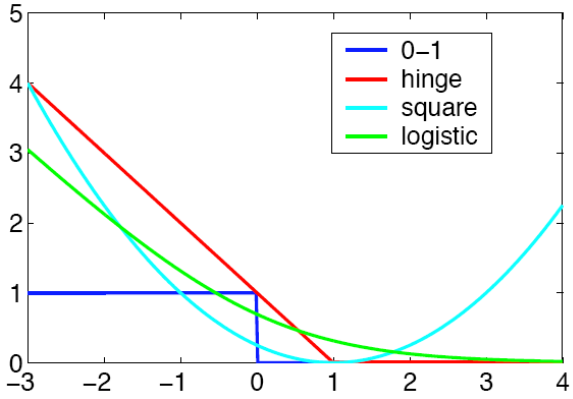
- **Critère de substitution** à la place du risque réel

1. **Risque empirique régularisé**

$$h^\star \;=\; \underset{h\in\mathcal{H}}{\mathrm{ArgMin}} \left[R_{\mathrm{Emp}}(h) \;+\lambda\, \mathrm{reg}(h) \right]$$

2. **Fonction de perte de substitution** (surrogate function)

$$\begin{aligned} \ell(y, h(\mathbf{x})) \;&=\; \max(0, 1 - y.h(\mathbf{x})) \\ &=\; (1 - y.h(\mathbf{x}))_+ \;=\; \xi \end{aligned}$$

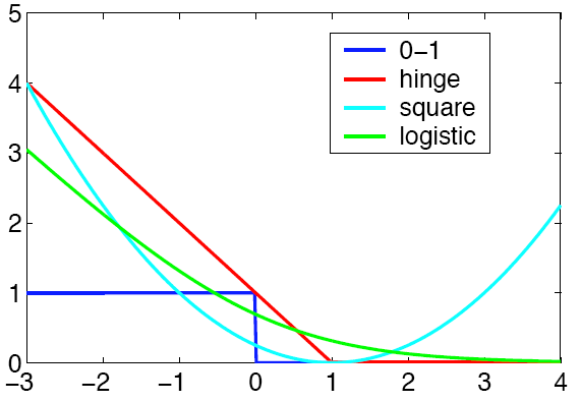


3. Contrainte sur l'**espace des hypothèses** considéré $\mathcal{H}$

- C'est la forme de la **fonction de coût** qui induit la **régularisation**

$$h^\star = \underset{h \in \mathcal{H}}{\text{ArgMin}} \left[R_{\text{Emp}}(h) + \lambda \, \text{reg}(h) \right]$$

$$\begin{aligned} \ell(y, h(\mathbf{x})) &= \max(0, 1 - y.h(\mathbf{x})) \\ &= (1 - y.h(\mathbf{x}))_+ = \xi \end{aligned}$$



SVM : marges poreuses

La version relaxée revient à trouver le minimum de

$$\|w\|^2 + C \sum_{i=1}^{n} \xi_i \,,$$

sous les contraintes $\forall i, \, (\langle w, X_i \rangle + b) \, Y_i \geq 1 - \xi_i \,.$

Note: la constante C est maintenant un paramètre de l'algorithme. Il faut donc en pratique un moyen de la choisir de façon adéquate (très souvent, on utilise la cross-validation ou des approximations de celle-ci).

Le développement du vecteur w comme combinaison de vecteurs de support reste valide dans cette situation.

SVM with soft margins

- Control of the bias-variance tradeoff
  - ***C* large**: Errors are costly
    - Low **variance**: less sensitivity to variations of *S*
    - Stronger **Bias** : we want hypotheses with large margins
  - ***C* small**: one is tolerant to errors
    - Large **variance**: high sensitivity to the characteristics of *S*
    - Weaker **Biais**: we are less demanding on the size of the margin

→ *C* controls this tradeoff

- It is often set using cross-validation

Bound on the generalization error

- For **any probability distribution** *D* on $\mathcal{X}$ x {-1, +1}, with probability 1 - δ, over *m* random examples in *S* within ball of radius *R*, the maximal margin hyperplane with unit weight vector *w* and margin γ has **generalization error** no more than

$$R(h) \leq \frac{c}{m}\left(\frac{R^2 + ||\xi||_2^2}{\gamma^2}\log^2 m + \log\frac{1}{\delta}\right)$$

$\xi = (\xi_1, \xi_2, \ldots, \xi_m)$ how much violation on each example

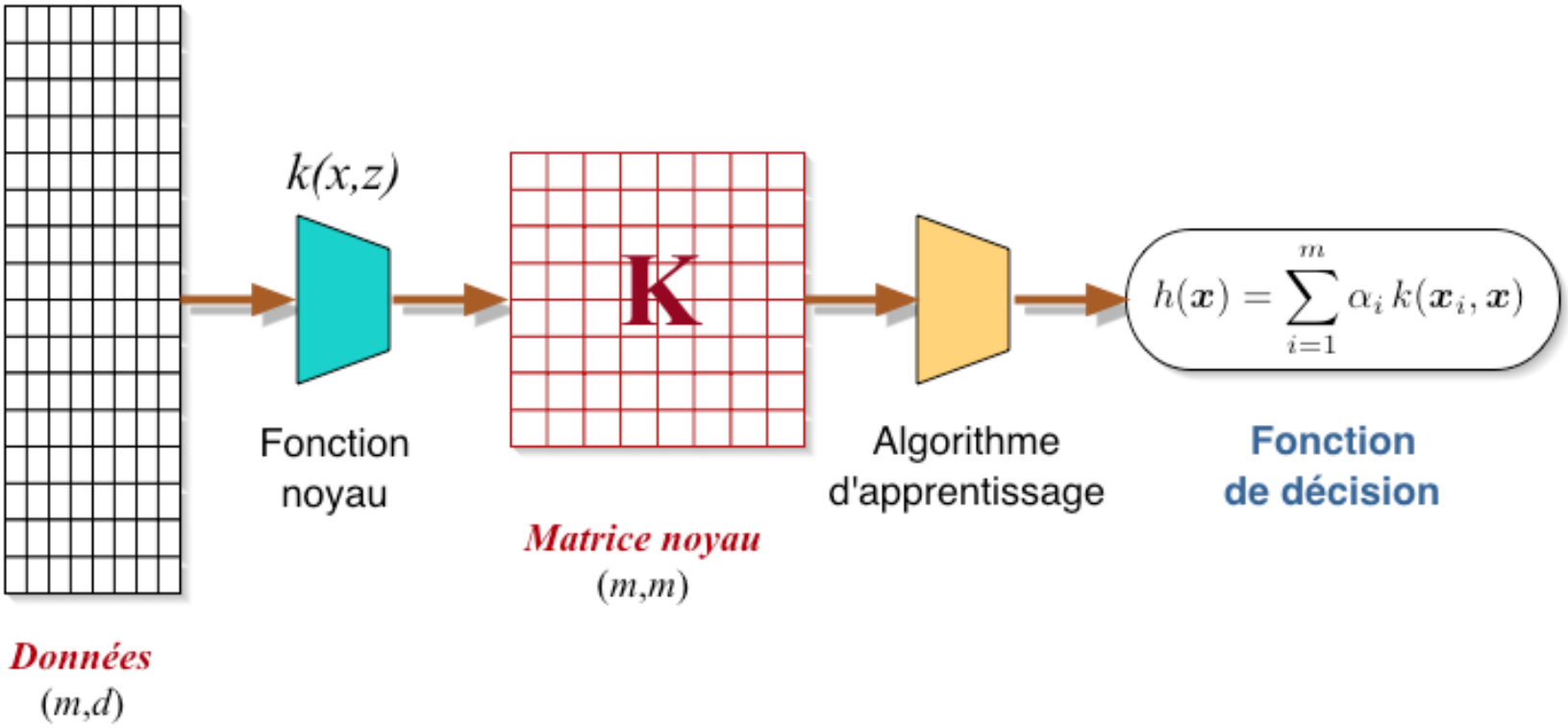
the margin slack vector with respect to *h* and γ

Outline

Kernel functions Engineering

Les méthodes à noyau

- Modularité
  - Découplage entre
    - Les **algorithmes** (linéaires)
    - La **description des données**



Les méthodes à noyau

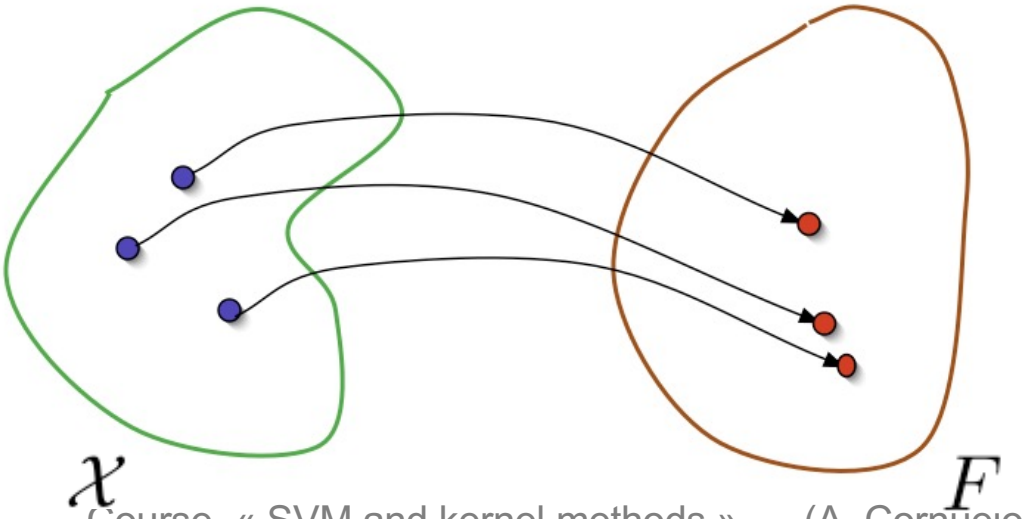
Les fonctions noyau

- Fonction ***k*** telle que :

$$k : \mathcal{X} \times \mathcal{X} \rightarrow \Re$$
$$\forall \boldsymbol{x}, \boldsymbol{z} \in \mathcal{X} : \quad k(\boldsymbol{x}, \boldsymbol{z}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle$$

où : $\phi : \boldsymbol{x} \mapsto \phi(\boldsymbol{x}) \in F$

Espace de redescription muni d'un produit interne

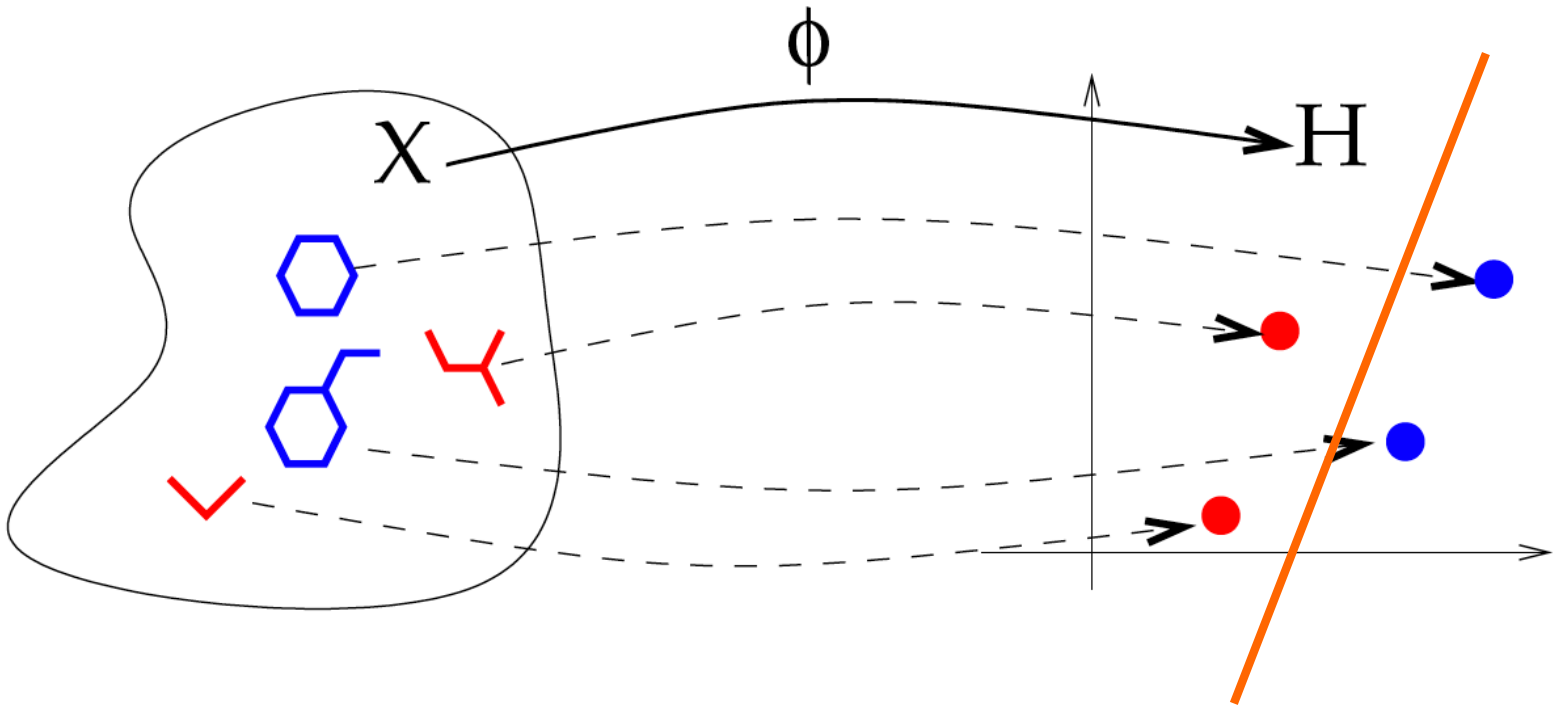


Kernel functions exist for non vectorial spaces

- Genomes
- Texts
- Images
- Video
- Social graphs
- …

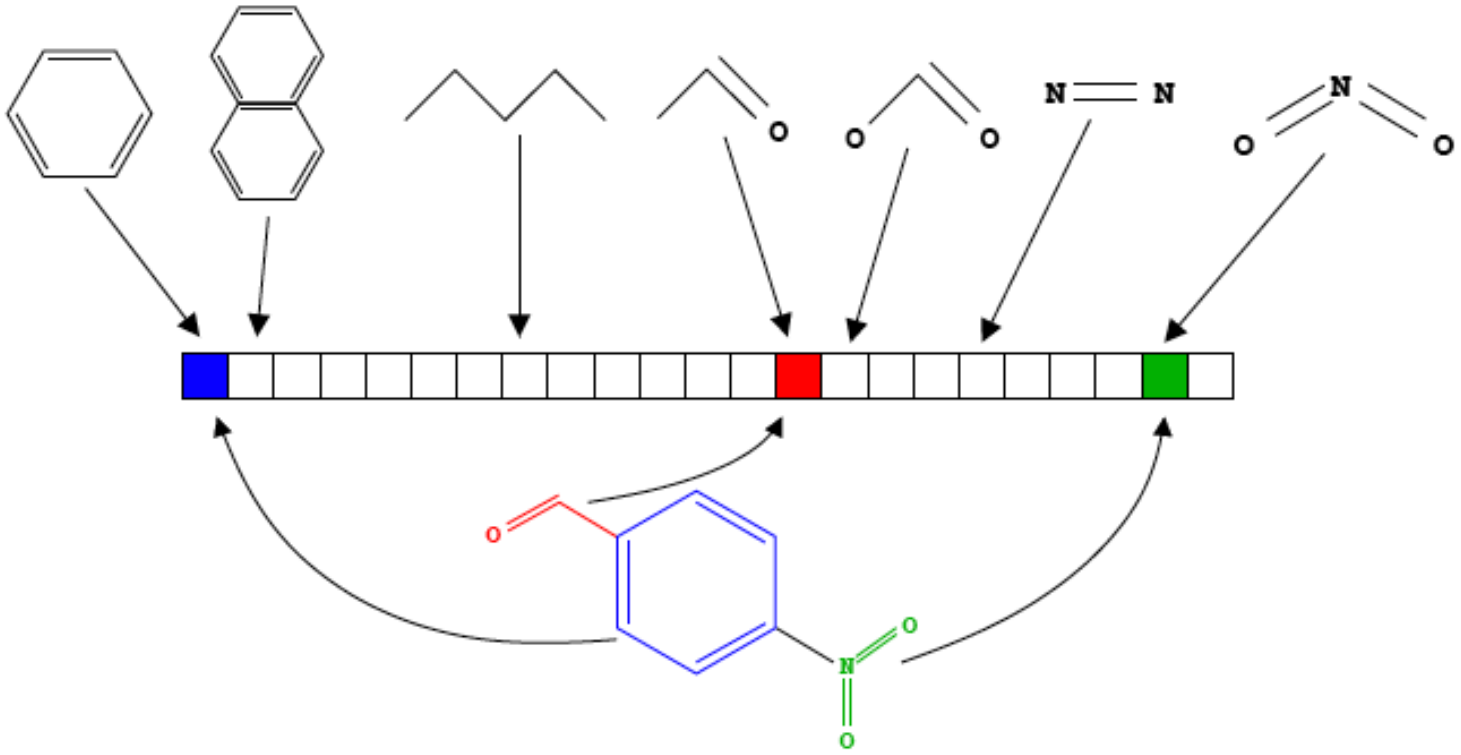
Change of representation

- Change of representation: the ideal case



Change of representation

- Change of representation



String kernels (2)

Φ: projection on $\mathbb{R}^D$ où $D = |\Sigma|^n$

	CH	CA	CT	AT
CHAT	ε^2	ε^3	ε^4	ε^2
CARTOON	0	ε^2	ε^4	ε^3

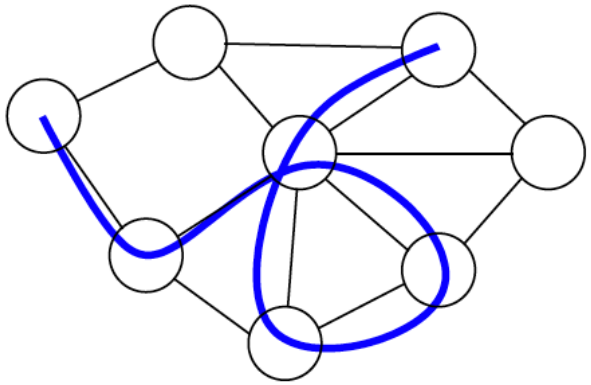
$$K(CHAT, CARTON) = 2\varepsilon^5 + \varepsilon^8$$

Prefer the normalized version

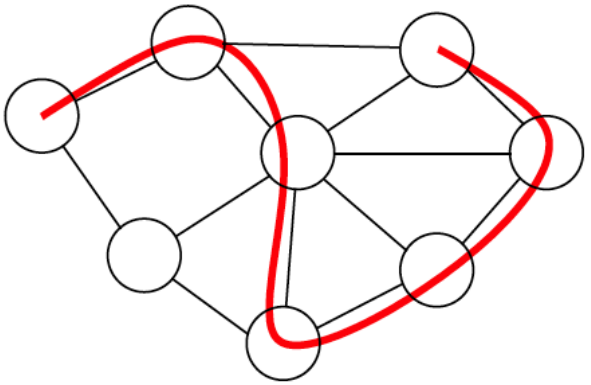
$$\kappa_(s, s') = \frac{K(s, s')}{\sqrt{K(s, s)K(s's')}}$$

Comparaison de graphes

- Fonction noyau sur des parcours (« *walks* »)



Parcours



Chemin

Proposition

These three kernels (*n*th-order, random and geometric walk kernels) can be computed efficiently in polynomial time.

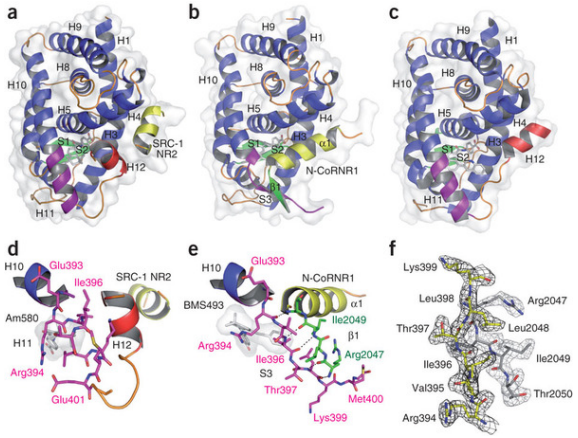
Other kernels and kernel applications

Application 1 Document mining

- Pre-processing matters a lot (stop-words, stemming)
- Multi-lingual aspects
- Document classification
- Information retrieval

Application 2, Bio-informatics

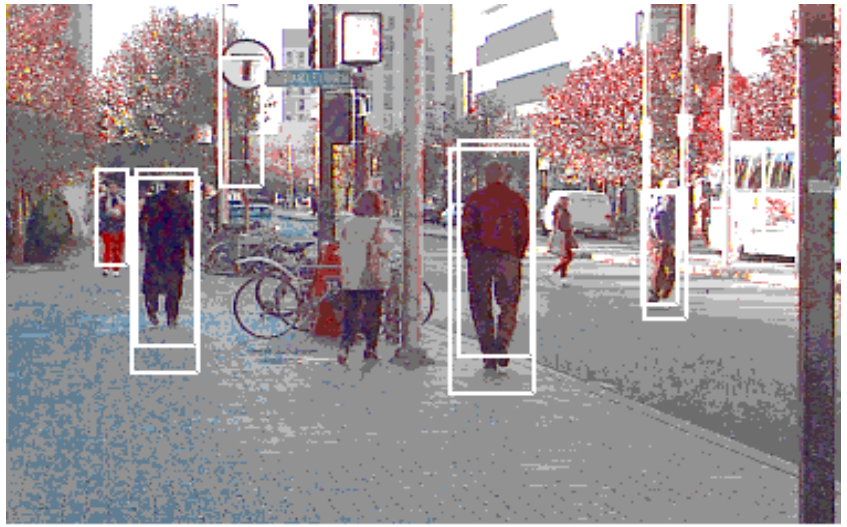
- Pre-processing matters a lot
- Classification (secondary structures)

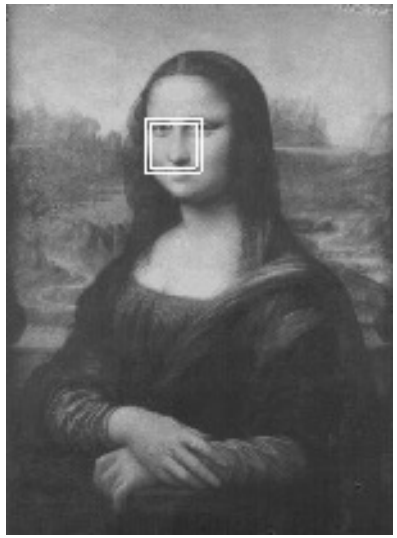


- Extension to **graph kernels**, see: http://videolectures.net/gbr07_vert_ckac/

Applications

- **Text** categorization
- **Hand-written character** recognition
- **Visage detection**
- **Diagnostic** (e.g. breath cancer)
- **Classification** of proteins
- **Forecast** of electrical consumption
- Indexing videos by text (keywords)
- …





Trained SVM classifiers for pedestrian and face object detection (Papageorgiou, Oren, Osuna and Poggio, 1998)

Kernel functions can be applied in many tasks

- Multi-class Classification
- Regression
- « novelty » detection
- Non linear principal component analysis

Outline

Assessment

and perspectives

Une méthode très intéressante

- Recherche un séparateur linéaire : un **optimum global**
- Méthode adaptative avec changement automatique d'espace de description (*embedding space*)
  - Séparateur défini par un ensemble d'**exemples**
    Mais seulement ceux sur la **marge** (contrairement à un kNN)
  - Combine les **avantages** des méthodes
    - **non paramétriques** : s'ajustent automatiquement aux données
    - et **paramétriques** : résiste bien au surajustement

Conclusions

- From a **conceptual point of view**
  - Introduces the **margin** as a **measure of "capacity"**
  - The **kernel trick** opens **non linear** problems to well controlled **linear methods** thanks to virtual redescriptions

Conclusions

- Quite a **black box**
  - It is very **difficult** to **interpret** what has been learned
    (support vectors (and their weights) in a unknown feature space)
  - **A priori knowledge** can only be expressed through the choice of the kernel function
- Not well adapted to problem exhibiting **"long distance" dependencies**
  - Only one level of non linearities (vs. "deep belief networks")
- Current **research**
  - Learning kernels
  - Combining kernels

References

V. Barra & A. Cornuéjols & L. Miclet

Apprentissage Artificiel. Concepts et algorithmes. De Bayes et Hume au Deep Learning

Eyrolles (4° éd.), 2021

Lutz Hamel

Knowledge discovery with Support Vector Machines

Wiley, 2009

J. Shawe-Taylor & N. Cristianini

Kernel methods for pattern analysis

Cambridge University Press, 2004

B. Schölkopf & A. Smola

Learning with kernels. Support Vector Machines, Regularization, Optimization, and Beyond

MIT Press, 2002

B. Schölkopf, K. Tsuda & J-Ph. Vert (Eds)

Kernel methods for computational biology

MIT Press, 2004