Machine learning 2
Support Vector Machines (SVM)

Pierre Chainais

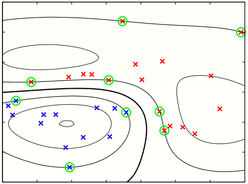
centralelille
ÉCOLE CENTRALE DE LILLE

Trajectory

1. SVM: Support Vector Machines
   - SVM in brief
   - Recall on linear classifiers
   - Maximizing the margin : the linearly separable case
   - Maximizing the margin: tne non separable case
   - SVM: the redescription space and the kernel
   - Complements
   - In summary...

SVM in brief

- $\mathbf{x} \in \mathbb{R}^D$ et 2 classes $y \in \{-1, 1\}$
- Classification algorithm: non-linear
- Embeds inputs $\mathbf{x} \in \mathbb{R}^D$ to a higher dimensional space (potentially infinite) $\mathcal{F}$: $\mathbf{x} \Rightarrow \phi(\mathbf{x})$
- Applies a linear classification classification in this space $\mathcal{F}$...
- ... without knowing explicitly functions ϕ = "kernel trick" !

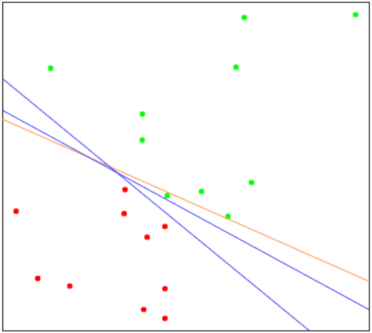


- $\mathbf{x} \in \mathbb{R}^D$ and 2 classes $y \in \{-1, 1\}$
- Training set (x_i, y_i), $i = 1, \ldots, N$
- First assume that the training set is linearly separable in $\mathbb{R}^D$
- Classification

  $= $ search for a hyperplane $\mathcal{H}$

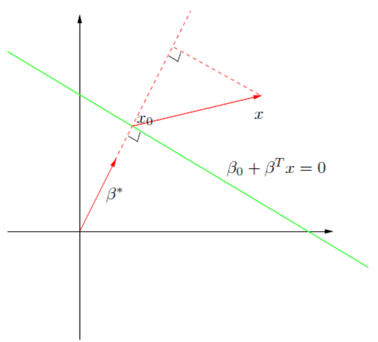
  $$h(\mathbf{x}) = \mathbf{x}^T\boldsymbol{\beta} + \beta_0 = 0, \quad \boldsymbol{\beta} \in \mathbb{R}^D, \beta_0 \in \mathbb{R}$$

  perfectly separating the 2 classes.
- If these classes are separable, $\exists$ an infinite set of planes



Recall on linear classifiers
Geométry

- For all points $(\mathbf{x}_1, \mathbf{x}_2)$ on $\mathcal{H}$, $\beta^T(\mathbf{x}_1 - \mathbf{x}_2) = 0$
- $\boldsymbol{\beta}^\star = \boldsymbol{\beta}/||\boldsymbol{\beta}||$ is orthogonal to the hyperplane $\mathcal{H}$,
- For all point $\mathbf{x}_0$ on $\mathcal{H}$, $\boldsymbol{\beta}^T\mathbf{x}_0 = -\beta_0$

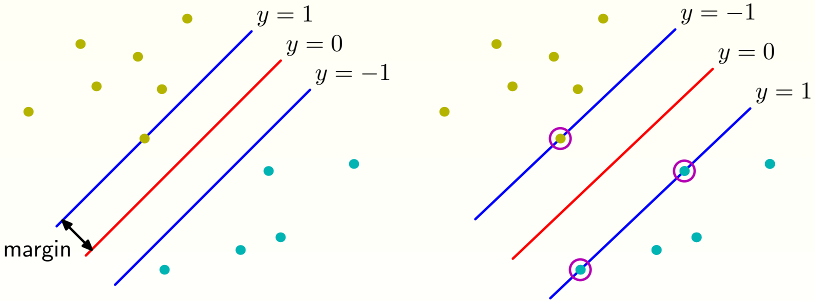


- The signed distance $d(\mathbf{x}, \mathcal{H})$ is

$$\boldsymbol{\beta}^{\star T}(\mathbf{x} - \mathbf{x}_0) = \frac{1}{||\boldsymbol{\beta}||}(\boldsymbol{\beta}^T\mathbf{x} + \beta_0)$$

Maximizing the margin : the linearly separable case

Notion of *margin*

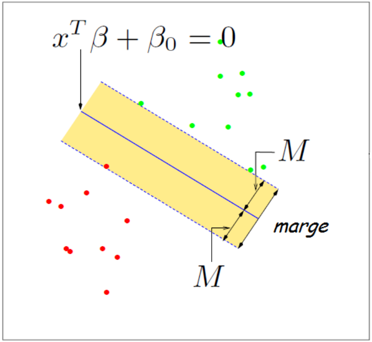


Support Vector Machine: primal problem

- One considers the optimisation problem

$$(\boldsymbol{\beta}, \beta_0) = \text{argmax}_{\boldsymbol{\beta}, \beta_0, ||\boldsymbol{\beta}||=1} M$$

$$\text{s.c. } y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) \geq M, i = 1, \ldots, N$$



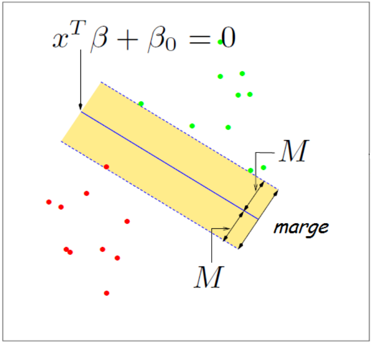
- These N conditions guarantee that all points are at least at a distance M of the hyperplane ($y_i = \pm 1$)
- We aims at maximizing the margin M

Support Vector Machine: primal problem

- One considers the optimisation problem

$$(\boldsymbol{\beta}, \beta_0) = \text{argmax}_{\boldsymbol{\beta}, \beta_0, ||\boldsymbol{\beta}||=1} M$$

$$\text{s.c. } y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) \geq M, i = 1, \ldots, N$$

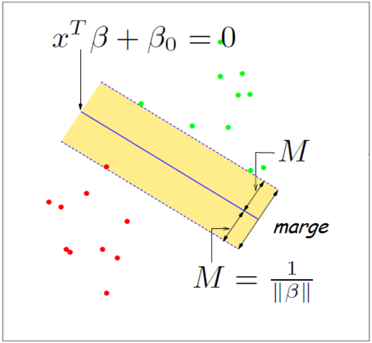


- $M = \min_{\mathbf{x}_i} \frac{1}{\|\boldsymbol{\beta}\|} y_i(\boldsymbol{\beta}^T \mathbf{x}_i + \beta_0)$
- Pb : $\text{argmax}_{\boldsymbol{\beta}, \beta_0} \frac{1}{\|\boldsymbol{\beta}\|} \min_{\mathbf{x}_i} [y_i(\boldsymbol{\beta}^T \mathbf{x}_i + \beta_0)]$

- Let $\min_{\mathbf{x}_i}[y_i(\boldsymbol{\beta}^T\mathbf{x}_i + \beta_0)] = 1$, one gets the equivalent formulation [*primal problem*]

$$(\boldsymbol{\beta}, \beta_0) = \operatorname{argmin}_{\boldsymbol{\beta}, \beta_0} \frac{1}{2}||\boldsymbol{\beta}||^2$$

such that $y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0) \geq 1, i = 1, \ldots, l$



- Quadratic criterion with linear equality constraints
- Convex optimisation problem

Support Vector Machine: dual problem

- Lagrangian formulation

$$L(\boldsymbol{\beta}, \beta_0, \alpha) = \frac{1}{2}||\boldsymbol{\beta}||^2 - \sum_{i=1}^{N} \alpha_i [y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) - 1]$$

  où $\alpha_i \geq 0$, $i = 1, \ldots, N$ sont les multiplicateurs de Lagrange

- Zeroes of the derivatives w.r.t. $\boldsymbol{\beta}$ et β_0 yield:

$$\boldsymbol{\beta} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i, \quad 0 = \sum_{i=1}^{N} \alpha_i y_i$$

- By substitution, on gets the dual formulation

$$L_{dual} = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \underbrace{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i y_i \alpha_j y_j \mathbf{x}_i^T \mathbf{x}_j}_{||\boldsymbol{\beta}||^2}$$

Support Vector Machine: dual problem

- This is a quadratic programming problem of dimension n [*dual problem*]

$$\text{argmax}_\alpha \left[\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i y_i \alpha_j y_j \, \mathbf{x}_i^T \mathbf{x}_j \right]$$

$$\text{such that (constraints)} \quad \forall i \quad \alpha_i \geq 0, \quad \sum_{i=1}^{N} \alpha_i y_i = 0$$

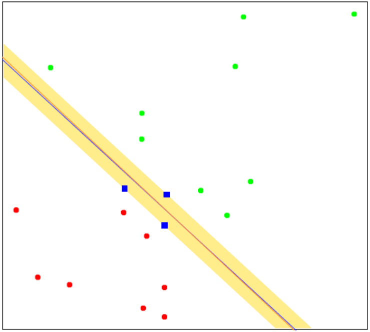
- This is a convex optimization problem ⇒ Sequential Minimal (SMO) Optimization [Platt, 1999]
- Remark : depends on $\mathbf{x}_i^T \mathbf{x}_j$ only

SVM : properties of the solution

- On a

$$\widehat{\boldsymbol{\beta}} = \sum_{i=1}^{N} \widehat{\alpha}_i y_i \, \mathbf{x}_i$$

- Only few $\widehat{\alpha}_i$, corresponding to points x_i on the margin, are non zero
- The corresponding x_i are called **support vectors**
- Sparse solution ("parcimonieuse" in french)



SVM: solution to the dual formulation

Decision

- One uses the function

$$h(\mathbf{x}) = \mathbf{x}^T \widehat{\boldsymbol{\beta}} + \widehat{\beta}_0 = \sum_{i=1}^{N} \widehat{\alpha}_i y_i \, \mathbf{x}_i^T \mathbf{x}$$

  to classify new elements

$$f(\mathbf{x}) = \widehat{y}(\mathbf{x}) = \text{signe}[h(\mathbf{x})]$$

- Remark : depends on $\mathbf{x}_i^T \mathbf{x}$ only.

- One uses the function

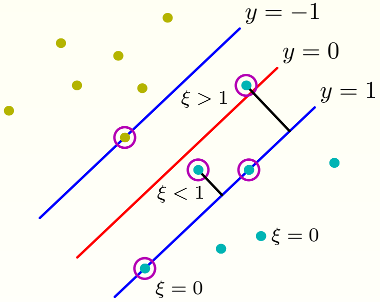
$$h(\mathbf{x}) = \mathbf{x}^T\widehat{\boldsymbol{\beta}} + \widehat{\beta}_0 = \sum_{i=1}^{N} \widehat{\alpha}_i y_i \ \mathbf{x}_i^T\mathbf{x}$$

to classify new elements

$$f(\mathbf{x}) = \widehat{y}(\mathbf{x}) = \mathrm{signe}[h(\mathbf{x})]$$

- Remark : depends on $\mathbf{x}_i^T\mathbf{x}$ only.

Maximizing the margin: tne non separable case



- Tolerance of the superposition of 2 classes
- Soft margin that tolerates small classification errors in the training set

SVM: the non separable case

- Introduction of spring variables $\xi_i > 0$, $i = 1, \ldots, N$
- Primal problem where C is a constant ≥ 0

$$(\boldsymbol{\beta}, \beta_0, \boldsymbol{\xi}) = \text{argmin}_{\boldsymbol{\beta}, \beta_0, \boldsymbol{\xi}} \frac{1}{2}||\boldsymbol{\beta}||^2 + C \sum_{i=1}^{N} \xi_i$$

under conditions

$$\begin{cases} \xi_i \geq 0 \\ y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i, i = 1, \ldots, N \end{cases}$$

SSVM: the non separable case

- Dual formulation

$$\max_{\alpha} \left[\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i y_i \alpha_j y_j \, \mathbf{x}_i^T \mathbf{x} \right]$$

with constraints

$$\forall i \;\; 0 \leq \alpha_i \leq C, \;\; \sum_{i=1}^{N} \alpha_i y_i = 0$$

- Difference : upper bound C on the α_i
- Problem depending on the $\mathbf{x}_i^T \mathbf{x}_j$ only...
- Convex problem - solution by quadratic programming
- Read Bishop pp. 331-334 for primal $\Rightarrow$ dual

SSVM: the non separable case

- Dual formulation

$$\max_{\alpha} \left[\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i y_i \alpha_j y_j \; \mathbf{x}_i^T \mathbf{x} \right]$$

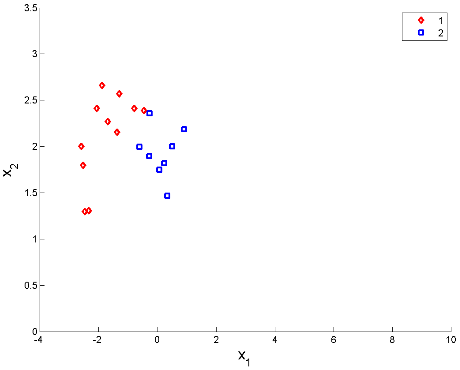
with constraints

$$\forall i \;\; 0 \leq \alpha_i \leq C, \;\; \sum_{i=1}^{N} \alpha_i y_i = 0$$

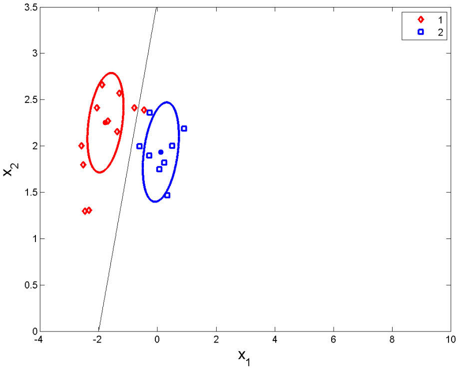
- Difference : upper bound C on the α_i
- Problem depending on the $\mathbf{x}_i^T \mathbf{x}_j$ only...
- Convex problem - solution by quadratic programming
- Read Bishop pp. 331-334 for primal $\Rightarrow$ dual

- SVM & LDA

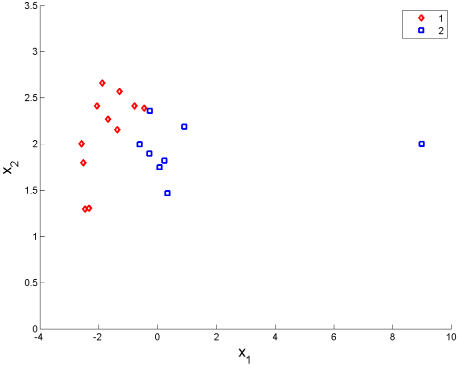
  = linear classification
- LDA = generative model (gaussian) for each class
- Atypical points, even far from the boundary ⇒ critical influence on the classification rule
- Points that are far from the margin ⇒ no influence on SVM



Training set without oulier

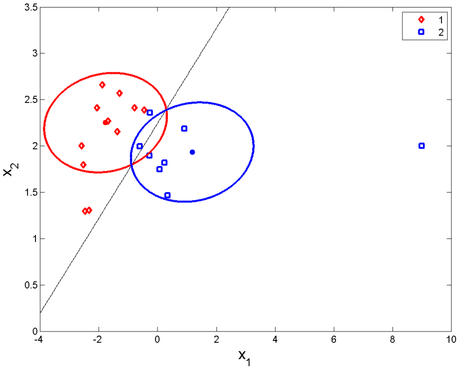


Linear discriminant analysis (without outlier)

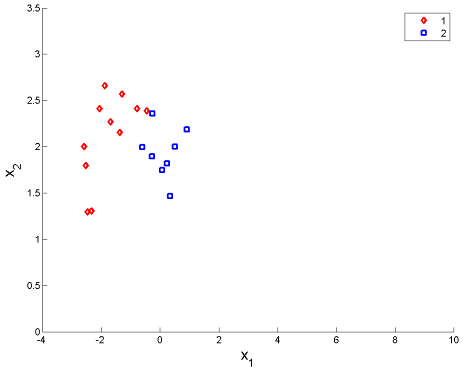


Training set with outlier

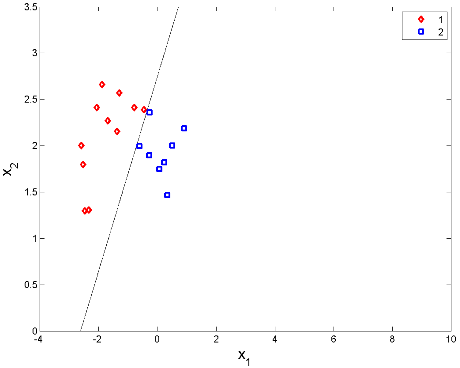
Linear discriminant analysis: influence of an outlier



Linear discriminant analysis (with outlier)



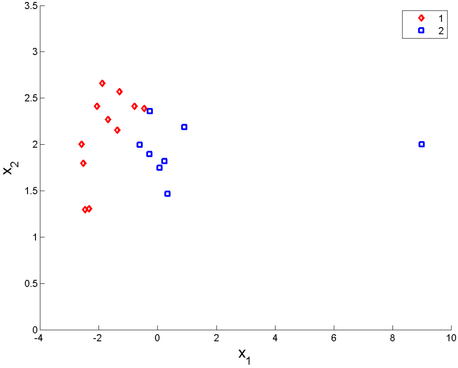
Training set without outlier



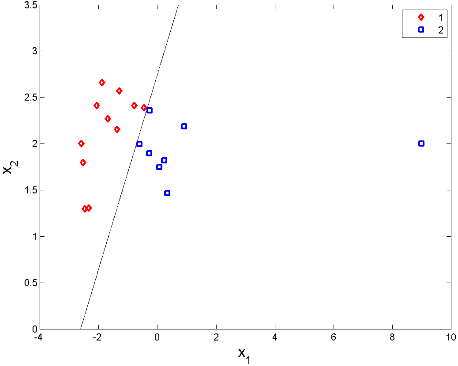
SVM (without outlier)

SVM & LDA

SVM: no influence of an outlier



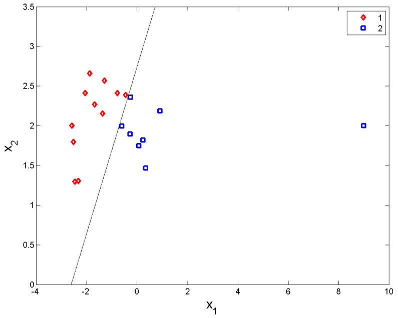
Training set with outlier



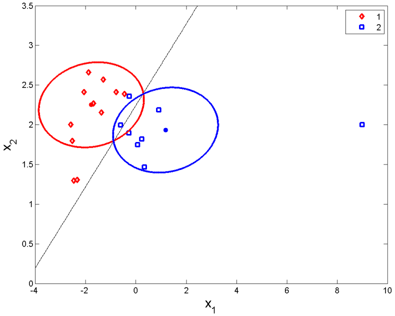
SVM (with outlier)

SVM

Comparison with linear discriminant analysis



SVM



Linear discriminant analysis

Redescription space and SVM

- In place of searching for a separating hyperplane in the inputs space, we embed the data in a redescription (feature space) of higher dimension

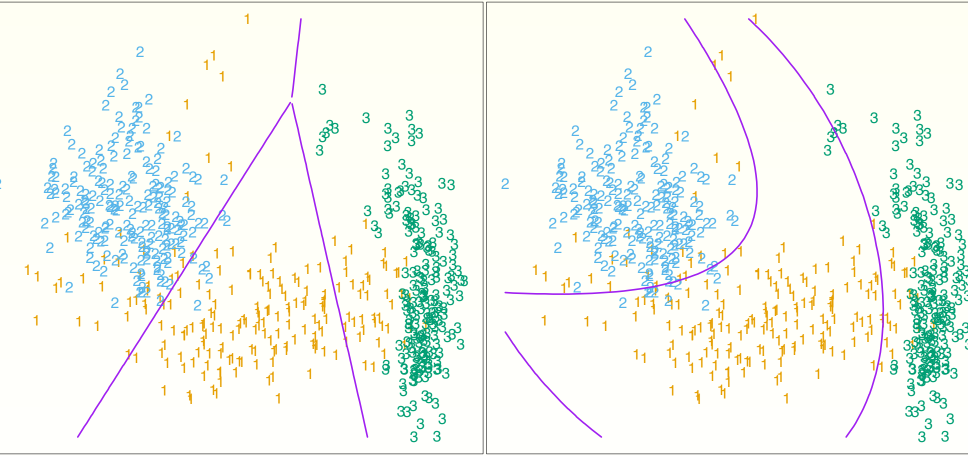
$$\begin{aligned}\phi :& \mathbb{R}^D \to \mathcal{F} \\ x &\to \phi(\mathbf{x})\end{aligned}$$

- Linear separation in $\mathcal{F}$ yields a non-linear separation in the inputs space $\mathbb{R}^D$

Back to LDAe

Application to $\phi_j(\mathbf{x})$: example, $x_1, x_2, x_1^2, x_1x_2, x_2^2$

$$\mathbf{x} \in \mathbb{R}^2,\ \phi(\mathbf{x}) = (x_1, x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2)$$



Linear classification in the redescription

$\Longleftrightarrow$ non linear classification in the input space

- Dual formulation

$$\max_{\alpha} \left[\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i y_i \alpha_j y_j \, \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \right]$$

with constraints

$$\forall i \quad 0 \le \alpha_i \le C, \quad \sum_{i=1}^{N} \alpha_i y_i = 0$$

- Solution given by

$$\begin{aligned} h(\mathbf{x}) &= \widehat{\boldsymbol{\beta}}^T \phi(x) + \widehat{\beta}_0 \\ &= \sum_{i=1}^{N} \widehat{\alpha}_i y_i \underbrace{\phi(\mathbf{x}_i)^T \phi(\mathbf{x})}_{k(\mathbf{x}_i, \mathbf{x})} + \widehat{\beta}_0 \end{aligned}$$

- The probelm and its solution depend on the scalar product $\phi(\mathbf{x})^T\phi(\mathbf{x}')$ only
- One denotes by $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ the kernel function defined by

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T\phi(\mathbf{x}')$$

- Kernel $\Rightarrow$ computing in the input space
  wihtout paking explicit the embedding $\mathbf{x} \to \phi(\mathbf{x})$
- This is the *kernel trick*)

- $\mathbf{x} \in \mathbb{R}^2$, $\phi(\mathbf{x}) = (x_1, x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2)$
- The scalar product in the redescription space is given by

$$\begin{aligned}\phi(\mathbf{x})^T\phi(\mathbf{x}') &= x_1x_1' + x_2x_2' + x_1^2x_1'^2 + 2x_1x_1'x_2x_2' + x_2^2x_2'^2 \\ &= (x_1x_1' + x_2x_2') + (x_1x_1' + x_2x_2')^2 \\ &= (x^Tx') + (x^Tx')^2 = k(x, x')\end{aligned}$$

- Can be computed without using ϕ (potentially unknown)

Theorem: Mercer's conditions

A symmetric function $k : \mathbb{R}^D \times \mathbb{R}^D$ is a kernel if for all $\mathbf{x}_i$, $(k(\mathbf{x}_i, \mathbf{x}_j))_{i,j}$ is a positive definite matrix. Then, there exists a space $\mathcal{F}$ and a (vectorial) function ϕ such that

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$$

- This condition is difficult to check
- It does not permit to determine neither $\mathcal{F}$ nor ϕ

- Linear

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

- Polynomials of order d

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}')^d \text{ ou } (1 + \mathbf{x}^T \mathbf{x}')^d$$

- Gaussian (radial basis)

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

- Neural network

$$k(\mathbf{x}, \mathbf{x}') = \tanh(\kappa_1 \mathbf{x}^T \mathbf{x}' + \kappa_2)$$

Kkernel trick

On reprend tout en remplaçant partout :

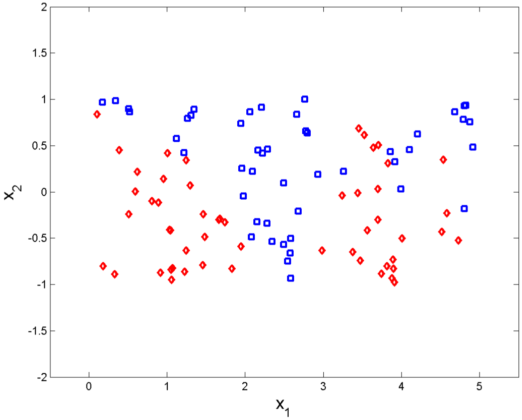
$$\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = \kappa(\mathbf{x}_i, \mathbf{x}_j)$$

$\Rightarrow$ Gram matrix
and one gets:

$$\hat{\alpha} = \mathrm{argmax}_{\alpha} \left[\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i y_i \alpha_j y_j \; k(\mathbf{x}_i, \mathbf{x}_j) \right]$$

$$h(\mathbf{x}) = \sum_{i=1}^{N} \hat{\alpha}_i y_i \; k(\mathbf{x}_i, \mathbf{x}) + \hat{\beta}_0$$

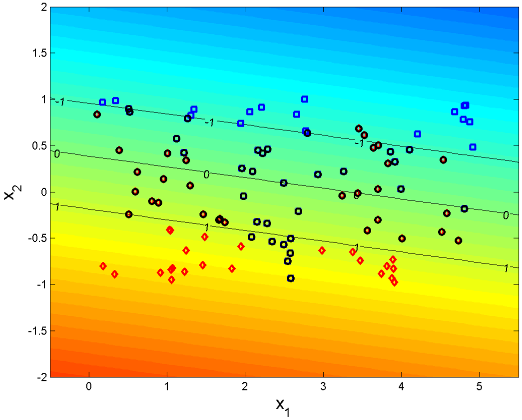
Examples



Training set

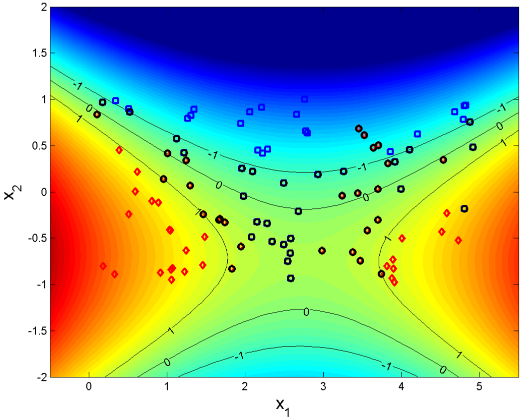
Support vector machines

Example

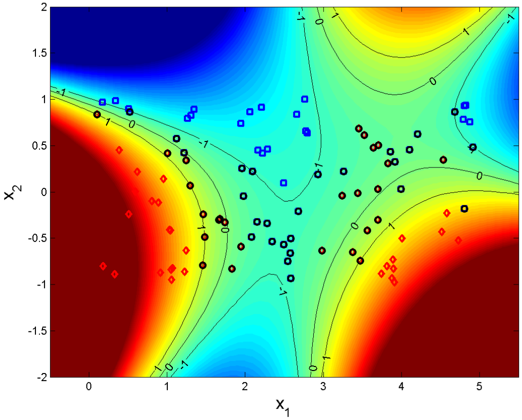


Linear kernel

Support vector machines
Example



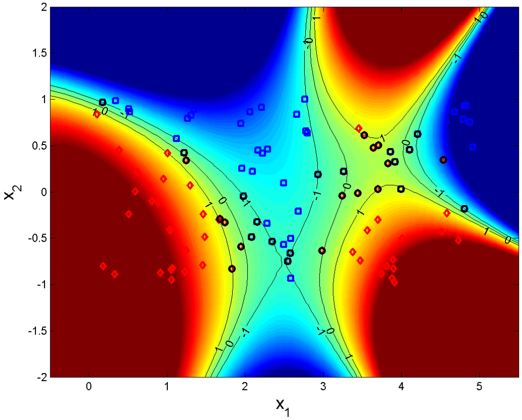
Polynomial kernel of order 2



Polynomial kernel of order 3

Support vector machines

Example



Polynomial kernel of order 4

Computational cost

- Dimension of input data: D
- Size of the training set: N
- Between $O(DN^2)$ and $O(DN^3)$
- Therefore no too big N, but maybe high dim. D

- $y \in \{1, \ldots, K\}$, K the number of classes
- Using several binary SVMs
  - One against all: K classifiers using one class vs others
  - One against one: $K(K-1)/2$ classifiers using all possible pairs of classes
- Global objective function: K SVM simultaneously (expensive...)
- 1 class only: for anomaly detection

- One considers the following optimization problem

$$\underbrace{\min_{\boldsymbol{\beta},\beta_0} \sum_{i=1}^{N} [1 - y_i h(\mathbf{x}_i)]_+}_{\text{training error}} + \underbrace{\frac{\lambda}{2} ||\boldsymbol{\beta}||^2}_{\text{Regularisation}}$$

where $[x]_+ = \max(x, 0)$, $h(\mathbf{x}) = \phi(\mathbf{x})^T \boldsymbol{\beta} + \beta_0$ and $\lambda = \frac{1}{C}$

- One searches for a solution of the form $\sum_i \alpha_i k(\mathbf{x}_i, \cdot)$
- Solution = SVM $\Rightarrow$ representation $h(\mathbf{x}) = \sum_i \hat{\alpha}_i k(\mathbf{x}_i, \mathbf{x})$

Complements

Kernels for any kind of data

- Texts,
- Trees,
- Graphs,
- DNA sequences...

Kernel methods:

- Kernel PCA,
- Kernel FLD,
- Kernel clustering,
- One class SVM...

http://rvlasveld.github.io/blog/2013/07/12/introduction-to-one-class-support-vector-machines/

In summary

4 important ideas :

1. data $\in \mathcal{X}$: **implicit non-linear proj.** on a vectorial space $\mathcal{F}$,
2. **linear classification** in $\mathcal{F}$ (quadratic optimisation)
3. algorithms use **the kernel** $k(\mathbf{x}_i, \mathbf{x}_j)$ **only**
4. **support vectors** $\Rightarrow$ a sparse representation of the classifier

Propriétés :

- No local minima,
- Kernel trick: computations in the input space,
- Control of the risk of overfitting (spring variables),
- Not too many parameters to tune ($\neq$ neural networks),
- Robustness: support vectors,
- Very good results in general (often state of the art)
- Pb : computational cost if large N (big data)