Introduction to Kernel Methods: Classification of Multivariate Data

Mathieu Fauvel

Introduction to Kernel Methods

Classification of multivariate data

Mathieu Fauvel

<2015-10-13 Tue>

Outline

Inner product

An *inner product* is a map $\langle .,. \rangle_{\mathcal{X}} : \mathcal{X} \times \mathcal{X} \to \mathcal{K}$ satisfying the following axioms:

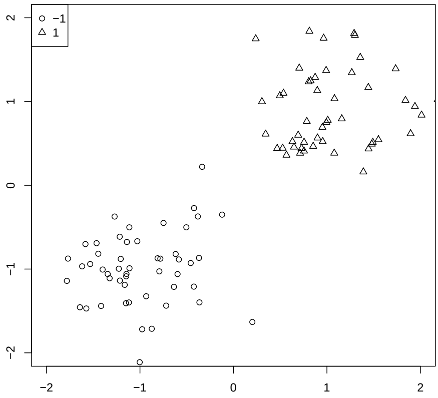
- Symmetry: $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{X}} = \langle \mathbf{y}, \mathbf{x} \rangle_{\mathcal{X}}$
- Bilinearity:
  $\langle a\mathbf{x} + b\mathbf{y}, c\mathbf{z} + d\mathbf{w} \rangle_{\mathcal{X}} = ac\langle \mathbf{x}, \mathbf{z} \rangle_{\mathcal{X}} + ad\langle \mathbf{x}, \mathbf{w} \rangle_{\mathcal{X}} + bc\langle \mathbf{y}, \mathbf{z} \rangle_{\mathcal{X}} + bd\langle \mathbf{y}, \mathbf{w} \rangle_{\mathcal{X}}$
- Non-negativity:$\langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{X}} \geq 0$
- Positive definiteness: $\langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{X}} = 0 \Leftrightarrow \mathbf{x} = \mathbf{0}$

The standard inner product in the Euclidean space, $\mathbf{x} \in \mathbb{R}^d$ and $d \in \mathbb{N}$, is called the dot product: $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbb{R}^n} = \sum_{i=1}^{n} x_i y_i$.

Toy data set

Suppose we want to classify the following data

$$\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}, \ (\mathbf{x}_i, y_i) \in \mathbb{R}^2 \times \{\pm 1\}$$



Simple classifier:

- Decision rule: *assigns a new sample* $\mathbf{x}$ *to the class whose mean is closer to* $\mathbf{x}$.

$$f(\mathbf{x}) = \operatorname{sgn}\left(||\mu_{-1} - \mathbf{x}||^2 - ||\mu_1 - \mathbf{x}||^2\right). \tag{1}$$

- Equation (1) can be written in the following way

$$f(\mathbf{x}) = \operatorname{sgn}\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right).$$

- Identify $\mathbf{w}$ and b. *Tips:* $||\mu_{-1} - \mathbf{x}||^2 = \langle \mu_{-1} - \mathbf{x}, \mu_{-1} - \mathbf{x} \rangle$ *and* $\mu_{-1} = \frac{1}{m_{-1}} \sum_{i=1}^{m_{-1}} \mathbf{x}_i$.

Solution 1/3

$$\|\mu_1 - \mathbf{x}\|^2 = \langle \frac{1}{m_1}\sum_{i=1}^{m_1}\mathbf{x}_i - \mathbf{x}, \frac{1}{m_1}\sum_{i=1}^{m_1}\mathbf{x}_i - \mathbf{x}\rangle$$

$$= \langle \frac{1}{m_1}\sum_{i=1}^{m_1}\mathbf{x}_i, \frac{1}{m_1}\sum_{i=1}^{m_1}\mathbf{x}_i\rangle + \langle \mathbf{x}, \mathbf{x}\rangle - 2\langle \frac{1}{m_1}\sum_{i=1}^{m_1}\mathbf{x}_i, \mathbf{x}\rangle$$

$$= \frac{1}{m_1^2}\sum_{\substack{i=1\\k=1}}^{m_1}\langle \mathbf{x}_i, \mathbf{x}_k\rangle + \langle \mathbf{x}, \mathbf{x}\rangle - 2\langle \frac{1}{m_1}\sum_{i=1}^{m_1}\mathbf{x}_i, \mathbf{x}\rangle \qquad (2)$$

$$\|\mu_{-1} - \mathbf{x}\|^2 = \frac{1}{m_{-1}^2}\sum_{\substack{j=1\\l=1}}^{m_{-1}}\langle \mathbf{x}_j, \mathbf{x}_l\rangle + \langle \mathbf{x}, \mathbf{x}\rangle - 2\langle \frac{1}{m_{-1}}\sum_{j=1}^{m_{-1}}\mathbf{x}_j, \mathbf{x}\rangle \qquad (3)$$

Plugging (2) and (3) into (1) leads to

$$\langle \mathbf{w}, \mathbf{x}\rangle + b = \langle 2(\frac{1}{m_1}\sum_{i=1}^{m_1}\mathbf{x}_i - \frac{1}{m_{-1}}\sum_{j=1}^{m_{-1}}\mathbf{x}_j), \mathbf{x}\rangle - \frac{1}{m_1^2}\sum_{\substack{i=1\\k=1}}^{m_1}\langle \mathbf{x}_i, \mathbf{x}_k\rangle + \frac{1}{m_{-1}^2}\sum_{\substack{j=1\\l=1}}^{m_{-1}}\langle \mathbf{x}_j, \mathbf{x}_l\rangle$$

Solution 2/3

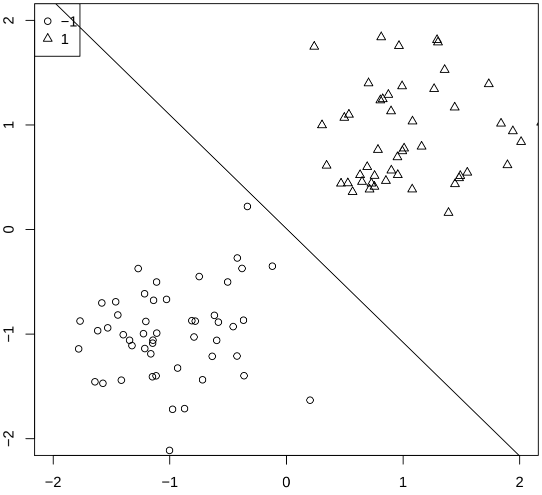
$$\mathbf{w} = 2 \sum_{i=1}^{m_1+m_{-1}} \alpha_i y_i \mathbf{x}_i \tag{4}$$

$$y_i = 1 \text{ or } -1$$

$$\alpha_i = \frac{1}{m_1} \text{ or } \frac{1}{m_{-1}}$$

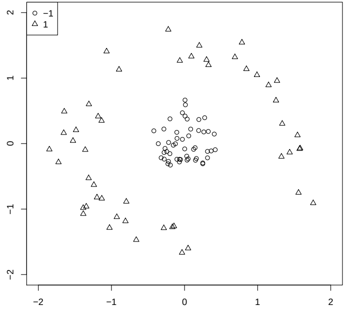
$$b = -\frac{1}{m_1^2} \sum_{\substack{i=1\\k=1}}^{m_1} \langle \mathbf{x}_i, \mathbf{x}_k \rangle + \frac{1}{m_{-1}^2} \sum_{\substack{j=1\\l=1}}^{m_{-1}} \langle \mathbf{x}_j, \mathbf{x}_l \rangle \tag{5}$$

Solution 3/3



Another toy data set

- Now the data are distributed a bit differently:



- However, if we can find a feature space where the data are linearly separable, it can still be applied.
- **Questions**:
  1. Find a feature space where the data are linearly separable.
  2. Try to write the dot product in the feature space in terms of input space variables.

Feature space

Two simple feature spaces are possible:

1. Projection in the polar domain

$$\rho = x_1^2 + x_2^2$$
$$\theta = \arctan\left(\frac{x_2}{x_1}\right)$$

2. Projection in the space of monomials of order 2.

$$\phi : \mathbb{R}^2 \to \mathbb{R}^3$$
$$\mathbf{x} \mapsto \phi(\mathbf{x})$$
$$(x_1, x_2) \mapsto (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

Feature space associated to monomials of order 2

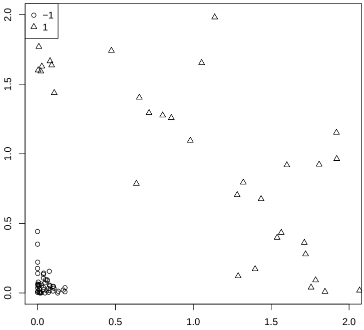
- In $\mathbb{R}^3$, the inner product can be expressed as

$$\begin{aligned}
\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathbb{R}^3} &= \sum_{i=1}^{3} \phi(\mathbf{x})_i \phi(\mathbf{x}')_i \\
&= \phi(\mathbf{x})_1 \phi(\mathbf{x}')_1 + \phi(\mathbf{x})_2 \phi(\mathbf{x}')_2 + \phi(\mathbf{x})_3 \phi(\mathbf{x}')_3 \\
&= \mathbf{x}_1^2 \mathbf{x}'^2_1 + \mathbf{x}_2^2 \mathbf{x}'^2_2 + 2\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}'_1 \mathbf{x}'_2 \\
&= (\mathbf{x}_1 \mathbf{x}'_1 + \mathbf{x}_2 \mathbf{x}'_2)^2 \\
&= \langle \mathbf{x}, \mathbf{x}' \rangle^2_{\mathbb{R}^2} \\
&= k(\mathbf{x}, \mathbf{x}').
\end{aligned}$$

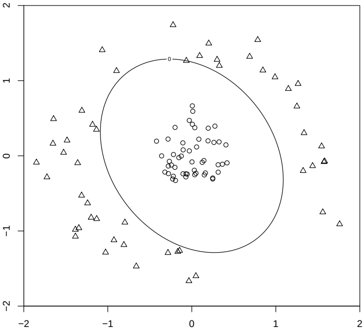
- The decision rule can be written in the input space thanks to the function k.

$$f(\mathbf{x}) = 2 \sum_{i=1}^{m_1 + m_{-1}} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) - \frac{1}{m_1^2} \sum_{\substack{i=1 \\ k=1}}^{m_1} k(\mathbf{x}_i, \mathbf{x}_k) + \frac{1}{m_{-1}^2} \sum_{\substack{j=1 \\ l=1}}^{m_{-1}} k(\mathbf{x}_j, \mathbf{x}_l)$$

Non linear decision function



Feature space (x_1^2, x_2^2)



Separating function

Conclusion

- A linear algorithm can be turned to a non linear one, simply by exchanging the dot product by an appropriate function.
- This function has to be equivalent to a dot product in a feature space.
- It is called a *kernel function* or just *kernel*.
- What are the properties of *kernel* ?

Definition

Definition (Positive semi-definite kernel)

$k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is positive semi-definite is

- $\forall (\mathbf{x}, \mathbf{x}') \in \mathbb{R}^d \times \mathbb{R}^d, k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i)$.
- $\forall n \in \mathbb{N}, \forall \xi_1 \dots \xi_n \in \mathbb{R}, \forall \mathbf{x}_1 \dots \mathbf{x}_n \in \mathbb{R}^d, \sum_{i,j}^{n} \xi_i \xi_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$.

Theorem (Moore-Aronsjan (1950))

To every positive semi-definite kernel k, there exists a Hilbert space $\mathcal{H}$ and a feature map $\phi : \mathbb{R}^d \to \mathcal{H}$ such that for all $\mathbf{x}_i, \mathbf{x}_j$ we have $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{H}}$.

Operations on kernels

Let k_1 and k_2 be positive semi-definite, and $\lambda_{1,2} > 0$ then:

1. $\lambda_1 k_1$ is a valid kernel
2. $\lambda_1 k_1 + \lambda_2 k_2$ is positive semi-definite.
3. $k_1 k_2$ is positive semi-definite.
4. $\exp(k_1)$ is positive semi-definite.
5. $g(\mathbf{x}_i)g(\mathbf{x}_j)$ is positive semi-definite, with $g : \mathbb{R}^d \rightarrow \mathbb{R}$.

Polynomial kernel

The polynomial kernel of order p and bias q is defined as

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + q)^p$$
$$= \sum_{l=1}^{p} \binom{p}{l} q^{p-l} \langle \mathbf{x}_i, \mathbf{x}_j \rangle^l.$$

It correspond to the feature space of monomials *up to* degree p. Depending on $q \gtrless 0$, the relative weights of the higher order monomial is inscreased/deacreased.

Gaussian kernel

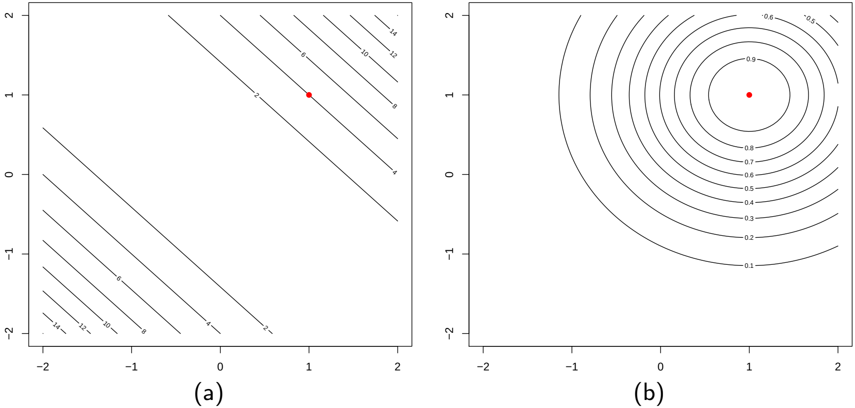
The Gaussian kernel with paramater σ is defined as

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right).$$

More generally, any distance can be used in the exponential rather than the Euclidean distance. For instance, the spectral angle is a valid distance:

$$\Theta(\mathbf{x}_i, \mathbf{x}_j) = \frac{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}.$$

Kernel values in $\mathbb{R}^2$



- (a) Polynomial kernel values for $p = 2$ and $q = 0$ and $\mathbf{x} = [1, 1]$,
- (b) Gaussian kernel values for $\sigma = 2$ and $\mathbf{x} = [1, 1]$.

How to construct kernel for my data ?

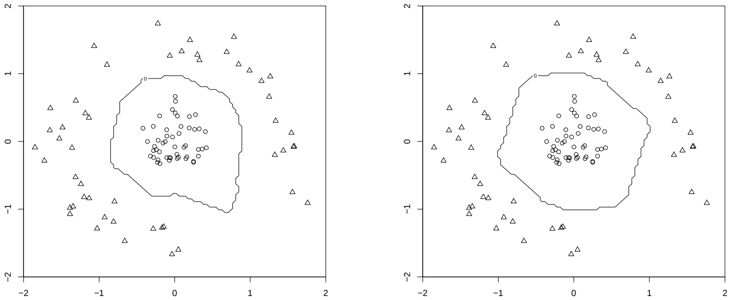
- A kernel is usually seen as a measure of similarity between two samples. It reflects in some sens, how two samples are similar.
- In practice, it is possible to define kernels using some *a priori* of our data.
- For instance: in image classification. It is possible to build kernels that includes information from the spatial domain.
  - Local correlation
  - Spatial position
  - ...

Compute distance in feature space 1/2

- The K-NN decision rule is based on the distance between two samples. In the feature space, the distance is computed as $\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2_{\mathcal{H}}$.
- Write this equation in terms of kernel function.
- Fill the function R `labwork_knn.R` by adding the construction of the kernel function. Then run it.

Compute distance in feature space 2/2

$$\begin{aligned}\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2_{\mathcal{H}} &= \langle \phi(\mathbf{x}_i) - \phi(\mathbf{x}_j), \phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\rangle_{\mathcal{H}} \\ &= \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_i)\rangle_{\mathcal{H}} + \langle \phi(\mathbf{x}_j), \phi(\mathbf{x}_j)\rangle_{\mathcal{H}} - 2\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle_{\mathcal{H}} \\ &= k(\mathbf{x}_i, \mathbf{x}_i) + k(\mathbf{x}_j, \mathbf{x}_j) - 2k(\mathbf{x}_i, \mathbf{x}_j)\end{aligned}$$



(a) (b)

- (a) KNN classification
- (b) Kernel KNN classification with a polynomial kernel of order 2

Learning problem

- Given a training set $\mathcal{S}$ and a loss function L, we want to find a function f from a set of functions $\mathcal{F}$ that minimizes its expected loss, or *risk*, $R(f)$:

$$R(f) = \int_{\mathcal{S}} L(f(\mathbf{x}), y) d\mathcal{P}(\mathbf{x}, y). \tag{6}$$

- But $\mathcal{P}(\mathbf{x}, y)$ is unknown !
- The *empirical risk*, $R_{emp}(f)$ can be computed:

$$R_{emp}(f) = \frac{1}{n} \sum_{i=1}^{n} L(f(\mathbf{x}^i), y_i) \tag{7}$$

- Convergence ?
  - f_1 minimizes R_{emp}, then $R_{emp}(f_1) \longrightarrow R(f_1)$ as n tends to infinity
  - But f_1 is not necessarily a minimizer of R.

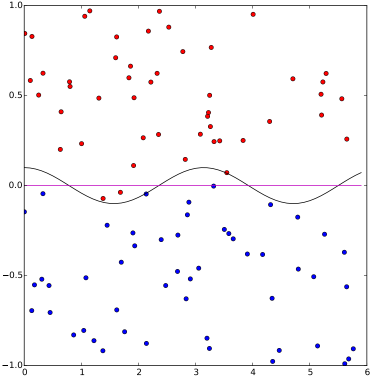
Non parametric classification

- *Bayesian* approach consists of selecting a distribution *a priori* for $\mathcal{P}(\mathbf{x}, y)$ (*GMM*)
- In *machine learning*, no assumption is made as to the distribution, but only about the *complexity* of the class of functions $\mathcal{F}$.
- Favor *simple* functions to
  - discard over-fitting problems,
  - to achieve a good generalization ability.
- Vapnik-Chervonenkis (VC) theory: the *complexity* is related to the number of points that can be separated by a function.

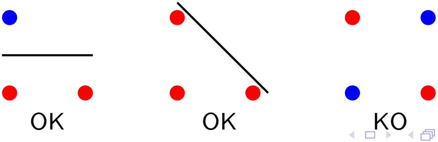
$$R(f) \leq R_{emp}(f, n) + \mathcal{C}(f, n)$$

Illustration

- Trade-off between R_{emp} and complexity



- VC dimension

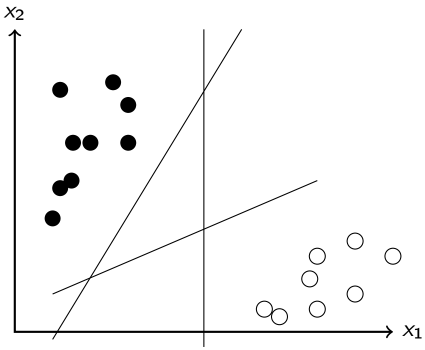


Separating hyperplane

- A separating hyperplane $H(\mathbf{w}, b)$ is a linear decision function that separate the space into two half-spaces, each half-space corresponding to the given class, *i.e.*, $\text{sgn}\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) = y_i$ for all samples from $\mathcal{S}$.
- The condition of correct classification is

$$y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) \geq 1. \tag{8}$$

- Many hyperplanes ?



Optimal separating hyperplane

- From the VC theory, the optimal one is the one that maximize the *margin*
- The *margin* is inversely proportional to $\|\mathbf{w}\|^2 = \langle \mathbf{w}, \mathbf{w} \rangle$.
- Optimal parameters are found by solving the **convex** optimization problem

$$\begin{aligned} \text{minimize } & \frac{\langle \mathbf{w}, \mathbf{w} \rangle}{2} \\ \text{subject to } & y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) \geq 1, \ \forall i \in 1, \ldots, n. \end{aligned}$$

- The problem is traditionally solved by considering *soft margin* constraints: $y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) \geq 1 + \xi_i$

$$\begin{aligned} \text{minimize } & \frac{\langle \mathbf{w}, \mathbf{w} \rangle}{2} + C \sum_{i=1}^{n} \xi_i \\ \text{subject to } & y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) \geq 1 - \xi_i, \ \forall i \in 1, \ldots, n \\ & \xi_i \geq 0, \ \forall i \in 1, \ldots, n. \end{aligned}$$

Quadratic programming

- The previous problem is solved by considering the Lagrangian

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{\langle \mathbf{w}, \mathbf{w} \rangle}{2} + C \sum_{i=1}^{n} \xi_i + \sum_{i=1}^{n} \alpha_i \left(1 - \xi_i - y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right)\right) - \sum_{i=1}^{n} \beta_i \xi_i$$

- Minimizing with respect to the primal variables and maximizing w.r.t the dual variables leads to the so-called dual problem:

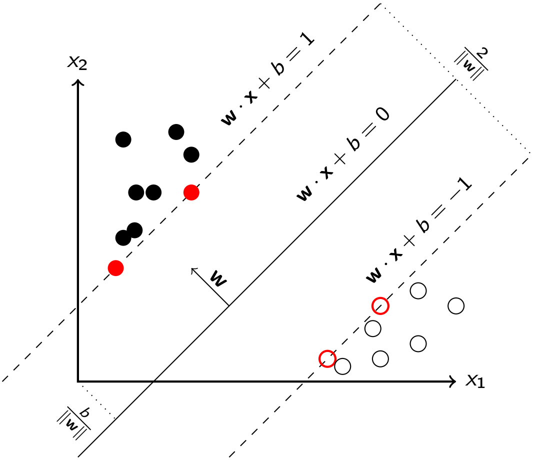
$$\max_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

$$\text{subject to } 0 \leq \alpha_i \leq C$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$

- $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$, only some of the α_i are non zero. Thus $\mathbf{w}$ is supported by some training samples – those with non-zero optimal α_i. These are called the *support vectors*.

Visual solution of the SVM



Kernelization of the algorithm

It is possible to extend the linear SVM to non linear SVM by switching the dot product to a kernel function:

$$\max_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

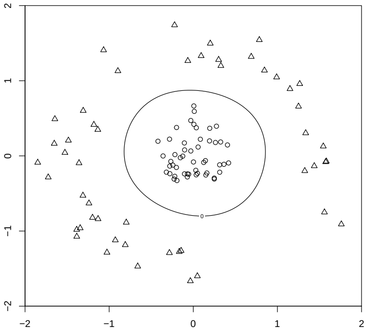
$$\text{subject to } 0 \leq \alpha_i \leq C$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$

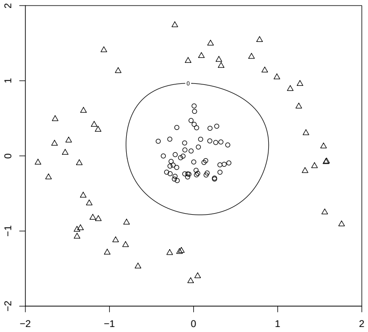
Now, the SVM is a non-linear classifier in the input space $\mathbb{R}^d$, but is still linear in the feature space – the space induced by the kernel function. The decision function is simply:

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}, \mathbf{x}_i) + b \right)$$

Toy example with the Gaussian kernel

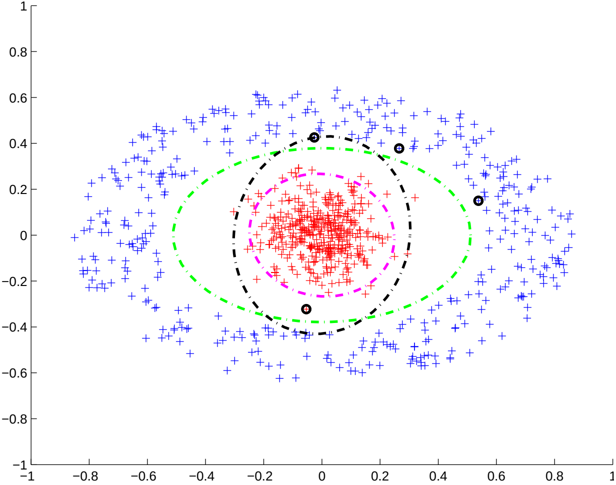


$C = 1$



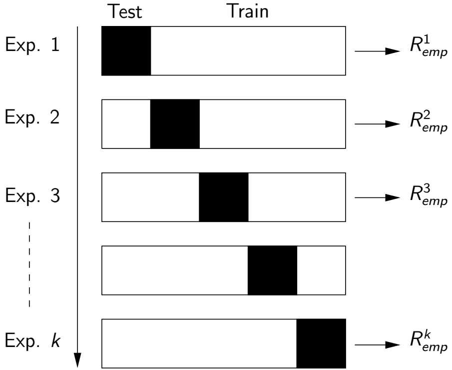
$C = 100$

Comparison with GMM



Cross-validation

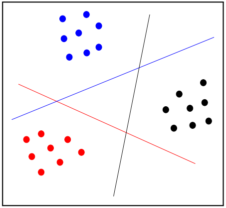
- Crucial step: improve or decrease drastically the performances of SVM
- Cross validation is conventionally used. CV estimates the expected error R.



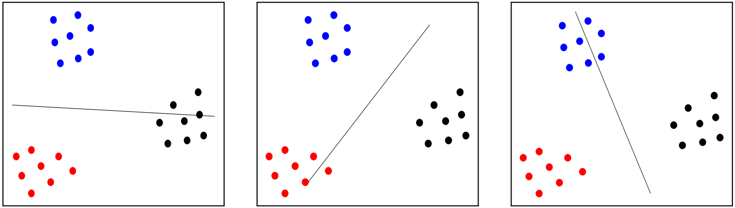
- $R(\mathbf{p}) \approx \frac{1}{k} \sum_{i=1}^{k} R^i_{emp}$
- Good behavior in various supervised learning problem but **high computational load**. Test 10 values with $k = 5 \Rightarrow$ **50 learning steps**. *But it can be perform in parallel...*

Collection of binary classifiers

- One versus All: m binary classifiers



- One versus One: m(m-1)/2 classifiers

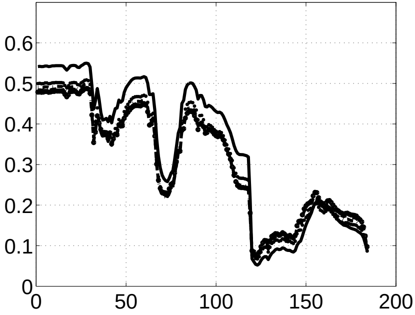


Toy non linear data set

- Run the code `toy_svm.R`
- The default classification is done with a Gaussian kernel: try to do it with a polynomial kernel
- Check the influence of each hyperparameters for the Gaussian and Polynomial kernel
- For the Gaussian kernel and a given set of hyperparameters
  - Get the number of support vectors
  - Plot them
  - Conclusions ?

Simulated data

- Simulated reflectance of Mars surface (500 to 5200 nm)
- The model has 5 parameters (Sylvain Douté): the grain size of water and CO_2 ice, the proportion of water, CO_2 ice and dust.
- $\mathbf{x} \in \mathbb{R}^{184}$ and $n = 31500$.
- Fives classes according to the grain size of water.



In this labwork, we are going to use the R package *e1071* that use the C++ library *libsvm*, the state of the art QP solver.

Questions

- Using the file `main_svm.R`, classify the data set with
  - SVM with a Gaussian kernel,
  - K-NN and Kernel KNN (with a polynomial kernel)
- Implement the cross-validation for SVM, to select the optimal hyperparameters (C,σ)
- Compute the confusion matrix for each methods and look at the classification accuracy

Load the data

```
## Load some library
library("e1071")

load("astrostat.RData")

n=nrow(x)
d=ncol(x)
C = max(y)

numberTrain = 100 # Select "numberTrain" per class for training
numberTest = 6300-numberTrain # The remaining is for validation

## Initialization of the training/validation sets
xt = matrix(0,numberTrain*C,d)
yt = matrix(0,numberTrain*C,1)
xT = matrix(0,numberTest*C,d)
yT = matrix(0,numberTest*C,1)

for (i in 1:C)
{
    t = which(y==i)
    ts = sample(t) # Permute ramdomly the samples for class i
    xt[(1+numberTrain*(i-1)):(numberTrain*i),]=x[ts[1:numberTrain],]
    yt[(1+numberTrain*(i-1)):(numberTrain*i),]=y[ts[1:numberTrain],]

    xT[(1+numberTest*(i-1)):(numberTest*i),]=x[ts[(numberTrain+1):6300],]
    yT[(1+numberTest*(i-1)):(numberTest*i),]=y[ts[(numberTrain+1):6300],]
}
```

Perform a simple classification

```
## Learn the model
model = svm(xt,yt,cost=1,gamma=0.001,type="C",cachesize=512)

## Predict the validation samples
yp = predict(model,xT)

## Confusion matrix
confu = table(yT,yp)
OA = sum(diag(confu))/sum(confu)
```