

# Advanced Computational Econometrics: Machine Learning

## Chapter 6: Kernel-based classification and regression

Thierry Denœux

Université de technologie de Compiègne

<https://www.hds.utc.fr/~tdenoeux>

email: [tdenoeux@utc.fr](mailto:tdenoeux@utc.fr)

Spring 2022

# Support Vector classification and regression

- In this chapter we describe new methods for linear and nonlinear classification and regression.
- **Optimal separating hyperplanes** are first introduced for the case when two classes are linearly separable. Then we cover extensions to the nonseparable case, where the classes overlap.
- These techniques are then generalized to the **support vector machine (SVM)**, which produces nonlinear boundaries by constructing a linear boundary in a large, transformed version of the predictor space.
- Finally, we will transpose these ideas to regression, and introduce **support vector regression (SVR)**.

# Overview

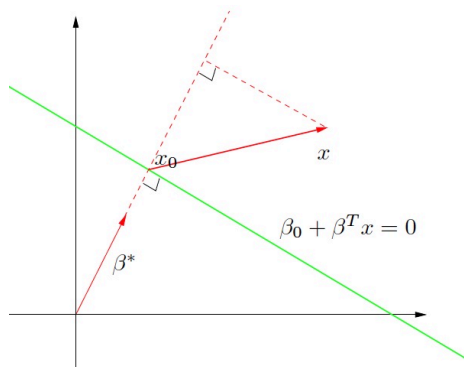
- 1 Optimal Separating hyperplane
  - Formalization
  - Solution in the separable case
  - Non-separable case
- 2 Support Vector Machines
  - The kernel trick
  - Kernel functions
  - SVM as a penalization method
- 3 Support Vector Regression
  - Loss function
  - Formalization
  - Solution and interpretation

# Overview

- 1 Optimal Separating hyperplane
  - Formalization
  - Solution in the separable case
  - Non-separable case
- 2 Support Vector Machines
  - The kernel trick
  - Kernel functions
  - SVM as a penalization method
- 3 Support Vector Regression
  - Loss function
  - Formalization
  - Solution and interpretation

# Hyperplane

In  $\mathbb{R}^p$ , a **hyperplane**  $H$  is defined by the equation  $g(x) = 0$  with  $g(x) = \beta_0 + \beta^T x$ . We have  $g(x) > 0$  on one side of  $H$  and  $g(x) < 0$  on the other side.



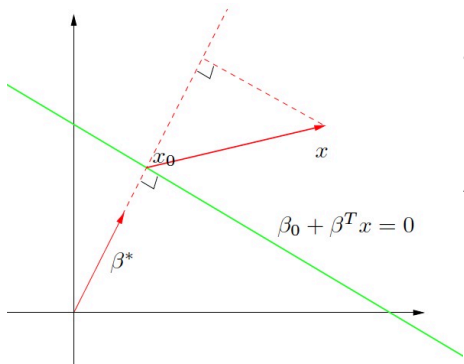
For any two points  $x_1$  and  $x_2$  lying in  $H$ , we have

$$\beta_0 + \beta^T x_1 = 0$$

$$\beta_0 + \beta^T x_2 = 0.$$

Consequently,  $\beta^T (x_1 - x_2) = 0$ , hence  $\beta^* = \beta / \|\beta\|$  is the vector normal to the surface of  $H$ .

## Hyperplane (continued)



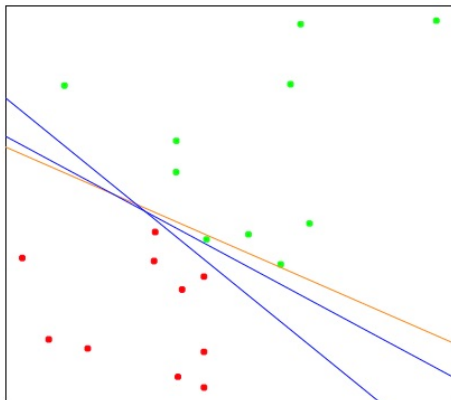
Let  $x_0 \in H$ . The **signed distance** of any point  $x$  to  $H$  is

$$d_s(x, H) = \beta^{*T}(x - x_0)$$

As  $\beta_0 = -\beta^T x_0$ , we have

$$\begin{aligned} d_s(x, H) &= \frac{\beta^T x - \beta^T x_0}{\|\beta\|} \\ &= \frac{\beta^T x + \beta_0}{\|\beta\|} \\ &= \frac{g(x)}{\|\beta\|} \end{aligned}$$

# Linearly separable data



- Consider a two-class data set  $\{(x_i, y_i)\}_{i=1}^n$  with  $y_i \in \{-1, 1\}$ .
- It is said to be **linearly separable** if there exists a hyperplane  $H : g(x) = 0$  that separates the two classes, i.e., such that

$$g(x_i)y_i > 0, \quad \forall i.$$

## Optimal separating hyperplane

Let  $H : g(x) = 0$  be a separating hyperplane. The distance between  $H$  and a learning vector  $x_i$  is

$$d(x_i, H) = \frac{g(x_i)y_i}{\|\beta\|}$$

### Definition (Margin)

The *margin* of  $H$  is the smallest distance between  $H$  and a learning vector  $x_i$ :

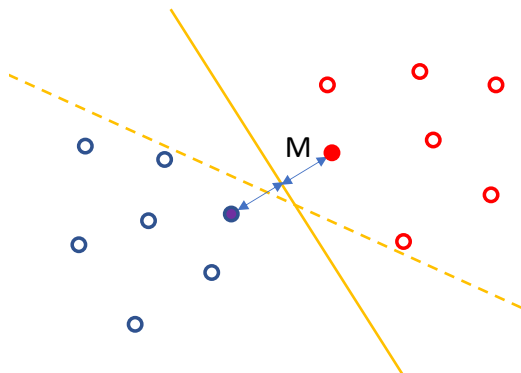
$$M = \min_i d(x_i, H).$$

### Definition (Optimal separating hyperplane, support vectors)

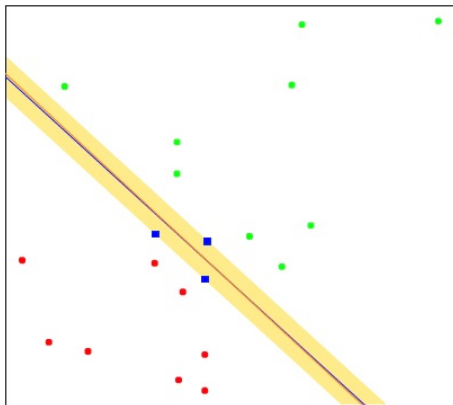
The *optimal separating hyperplane* (OSH) is the hyperplane with the largest margin. The learning vectors  $x_i$  such that  $d(x_i, H) = M$  are called the *support vectors* (SVs) of  $H$ .



# Example 1

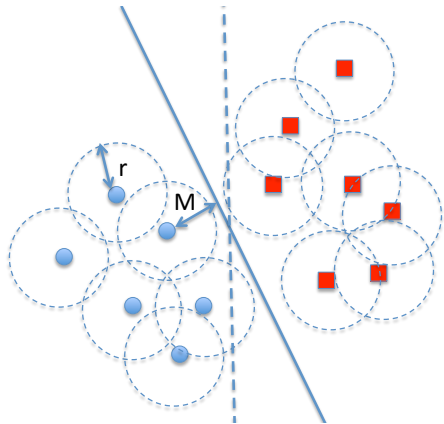


## Example 2



The shaded region delineates the maximum margin separating the two classes. There are 3 SVs, and the OSH is the blue line. The boundary found using logistic regression is the red line. In this case, it is very close to the OSH.

# The OSH is more likely to separate future data



- Future data can be assumed to be “close” to past data.
- Assume they will lie with a distance  $r$  of a past data point.
- If  $M > r$ , the hyperplane will classify future data perfectly.

# Overview

- 1 Optimal Separating hyperplane
  - Formalization
  - **Solution in the separable case**
  - Non-separable case
- 2 Support Vector Machines
  - The kernel trick
  - Kernel functions
  - SVM as a penalization method
- 3 Support Vector Regression
  - Loss function
  - Formalization
  - Solution and interpretation

# How to find the OSH?

- The OSH can be found by solving the following optimization problem:

$$\max_{\beta, \beta_0} M$$

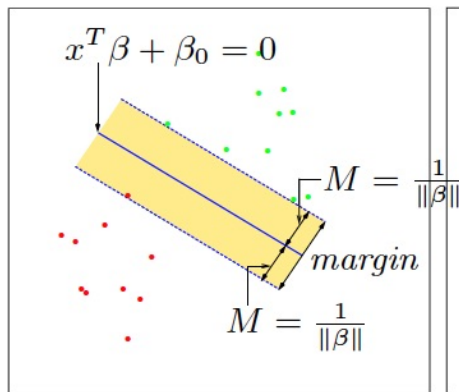
$$\text{subject to } \frac{y_i(\beta^T x_i + \beta_0)}{\|\beta\|} \geq M, \quad i = 1, \dots, n.$$

- If  $(\beta, \beta_0)$  is a solution, so is  $(\lambda\beta, \lambda\beta_0)$  for any  $\lambda$ . Hence, we can fix  $\|\beta\| = 1/M$  and reformulate the problem as

$$\min_{\beta, \beta_0} \frac{1}{2} \|\beta\|^2$$

$$\text{subject to } y_i(\beta^T x_i + \beta_0) \geq 1, \quad i = 1, \dots, n.$$

# Interpretation



- The constraints define an empty band or margin around the linear decision boundary of thickness  $1/\|\beta\|$ .
- The vectors  $x_i$  such that

$$\frac{y_i(\beta^T x_i + \beta_0)}{\|\beta\|} = \frac{1}{\|\beta\|},$$

i.e.,  $y_i(\beta^T x_i + \beta_0) = 1$ , are the SVs.

# Reminder on constrained optimization

## Lagrange function

Consider the following minimization problem:

$$\min_{\beta} f(\beta) \quad (1)$$

subject to the constraints  $c_i(\beta) \geq 0$ ,  $i = 1, \dots, n$ , where  $f$  and the  $c_i$ 's are differentiable functions.

### Definition (Lagrange function)

The *Lagrange function* is defined by

$$L(\beta, \alpha) = f(\beta) - \sum_{i=1}^n \alpha_i c_i(\beta),$$

where  $\alpha = (\alpha_1, \dots, \alpha_n)$  is the vector of *Lagrange multipliers*.

# Reminder on constrained optimization

## Karush-Kuhn-Tucker conditions

### Theorem (Karush-Kuhn-Tucker)

If function  $f$  has a minimum for some value  $\beta^*$  in the feasibility region, the following **Karush-Kuhn-Tucker (KKT) conditions** are verified for some vector  $\alpha^* = (\alpha_1^*, \dots, \alpha_n^*)$ :

$$\frac{\partial L}{\partial \beta}(\beta^*, \alpha^*) = 0 \quad (2a)$$

$$c_i(\beta^*) \geq 0, \quad i = 1, \dots, n \quad (2b)$$

$$\alpha_i^* c_i(\beta^*) = 0 \quad i = 1, \dots, n \quad (2c)$$

$$\alpha_i^* \geq 0 \quad i = 1, \dots, n. \quad (2d)$$

Remark: if  $\alpha_i^* > 0$ , then  $c_i(\beta^*) = 0$ : constraint  $i$  is active.



# Reminder on constrained optimization

## Wolfe dual

### Theorem (Wolfe dual)

*Problem (1) is equivalent to the following problem (Wolfe dual):*

$$\max_{\beta, \alpha} L(\beta, \alpha) \quad (3)$$

*subject to*

$$\frac{\partial L}{\partial \beta} = 0 \quad (4)$$

$$\alpha_i \geq 0 \quad i = 1, \dots, n. \quad (5)$$

# Lagrange function

- Let us come back to the problem  $\min_{\beta, \beta_0} \frac{1}{2} \|\beta\|^2$  subject to  $y_i(\beta^T x_i + \beta_0) \geq 1, i = 1, \dots, n$ .
- This is a **convex optimization problem** (quadratic criterion with linear inequality constraints), so the solution exists and it is unique.
- The Lagrange function is

$$L(\beta, \beta_0, \alpha) = \frac{1}{2} \|\beta\|^2 - \sum_{i=1}^n \alpha_i [y_i(\beta^T x_i + \beta_0) - 1] \quad (6)$$

- Setting the derivatives to zero, we obtain:

$$\frac{\partial L}{\partial \beta} = \beta - \sum_{i=1}^n \alpha_i y_i x_i = 0 \Rightarrow \beta = \sum_{i=1}^n \alpha_i y_i x_i \quad (7)$$

$$\text{and } \frac{\partial L}{\partial \beta_0} = - \sum_{i=1}^n \alpha_i y_i = 0$$



## Lagrangian of the dual problem

Substituting (7) and (8) in (6), we get

$$L_D(\alpha) = \frac{1}{2} \underbrace{\left( \sum_{i=1}^n \alpha_i y_i x_i \right)^T \left( \sum_{j=1}^n \alpha_j y_j x_j \right)}_{\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j} - \underbrace{\sum_i \alpha_i y_i \left( \sum_{j=1}^n \alpha_j y_j x_j \right)^T x_i}_{\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j} - \underbrace{\sum_i \alpha_i y_i \beta_0}_0 + \sum_i \alpha_i$$

which can be written as

$$L_D(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

# Solving the dual problem

- The solution is obtained by maximizing  $L_D(\alpha)$  subject to the constraints

$$\alpha_i \geq 0 \text{ and } \sum_{i=1}^n \alpha_i y_i = 0. \quad (9)$$

- This can be done using standard **quadratic programming** software. We will discuss a specialized optimization algorithm later.

# Interpreting the solution

## Support vectors

- The solution  $\alpha^*$  must satisfy the KKT conditions, which include (7), (8), (9) and

$$\alpha_i^* [y_i(\beta^{*T} x_i + \beta_0^*) - 1] = 0, \quad i = 1, \dots, n. \quad (10)$$

- From these we can see that, if  $\alpha_i^* > 0$ , then  $y_i(\beta^{*T} x_i + \beta_0^*) = 1$ , i.e.,  $x_i$  is a SV.
- The SV's are the input vectors  $x_i$  such that  $\alpha_i^* > 0$ .

# Interpreting the solution

## Computing $\beta^*$ and $\beta_0^*$

- From (7) we see that the solution vector  $\beta^*$  is defined in terms of a linear combination of the SVs:

$$\beta^* = \sum_{i=1}^n \alpha_i^* y_i x_i = \sum_{i \in \mathcal{S}} \alpha_i^* y_i x_i \quad (11)$$

with  $\mathcal{S} = \{i : \alpha_i^* > 0\}$ .

- The intercept  $\beta_0^*$  can be found from (10): for any SV  $x_i$ , we have

$$y_i(\beta^{*T} x_i + \beta_0^*) = 1,$$

from which we can get  $\beta_0^*$ .

# SVM classifier

- The equation of the OSH is

$$g^*(x) = \beta^{*T} x + \beta_0^* = \sum_{i \in S} \alpha_i^* y_i x_i^T x + \beta_0^* = 0$$

- The corresponding classifier is

$$D(x) = \text{sign } g^*(x).$$

- The classifier is based only on SVs, which are close to the boundary between classes.

# Overview

- 1 Optimal Separating hyperplane
  - Formalization
  - Solution in the separable case
  - **Non-separable case**
- 2 Support Vector Machines
  - The kernel trick
  - Kernel functions
  - SVM as a penalization method
- 3 Support Vector Regression
  - Loss function
  - Formalization
  - Solution and interpretation



# Extension to non-separable data

- Until now, we have assumed that the data are linearly separable.
- This will generally not be the case with real data, so the technique derived so far is not really useful in practice.
- We need to propose an alternative formulation for the **non-separable case**.

## Weakening the constraints

- Suppose that the classes overlap in predictor space.
- One way to deal with the overlap is to still maximize the margin  $M$ , but allow for some points to be on the wrong side of the margin.
- Define the **slack variables**  $\xi = (\xi_1, \xi_2, \dots, \xi_n)$  with  $\xi_i \geq 0$ . The constraints can be modified as

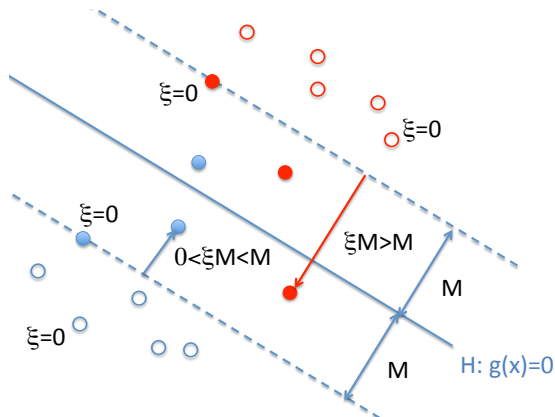
$$\frac{y_i(\beta^T x_i + \beta_0)}{\|\beta\|} \geq M(1 - \xi_i), \quad i = 1, \dots, n.$$

- As before, fixing  $\|\beta\| = 1/M$ , this is equivalent to

$$y_i(\beta^T x_i + \beta_0) \geq 1 - \xi_i, \quad i = 1, \dots, n.$$

- The value  $\xi_i$  is the proportional amount by which vector  $x_i$  is on the wrong side of its margin.

# Interpretation



- The filled points are on the wrong side of their margin by an amount  $M\xi_i$ .
- Points on the correct side have  $\xi_i = 0$ .
- Misclassified points have  $\xi_i > 1$ .

# Optimization problem

The optimization problem now becomes:

$$\min_{\beta, \beta_0, \{\xi_i\}} \frac{1}{2} \|\beta\|^2 + \frac{C}{n} \sum_{i=1}^n \xi_i, \quad (12)$$

subject to

$$\begin{aligned} \xi_i &\geq 0, \quad i = 1, \dots, n \\ y_i(\beta^T x_i + \beta_0) &\geq 1 - \xi_i, \quad i = 1, \dots, n \end{aligned}$$

where  $C$  is a hyperparameter.

# Lagrange function

- The Lagrange function is

$$L(\beta, \beta_0, \xi, \alpha, \mu) = \frac{1}{2} \|\beta\|^2 + \frac{C}{n} \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i [y_i(\beta^T x_i + \beta_0) - (1 - \xi_i)] - \sum_{i=1}^n \mu_i \xi_i.$$

- Setting the derivatives w.r.t.  $\beta$ ,  $\beta_0$  and  $\xi$  to zero, we get, as before,

$$\beta = \sum_{i=1}^n \alpha_i y_i x_i, \quad \sum_{i=1}^n \alpha_i y_i = 0, \quad (13)$$

and

$$\frac{C}{n} - \alpha_i - \mu_i = 0 \Rightarrow \alpha_i = \frac{C}{n} - \mu_i$$



(14)

Université de Technologie Compiègne

## Dual formulation

- By substituting (13), we obtain the Lagrangian dual objective function

$$L_D(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^n \underbrace{\left( \frac{C}{n} - \alpha_i - \mu_i \right)}_0 \xi_i, \quad (15)$$

which has exactly the same form as in the previous problem.

- We maximize  $L_D$  subject to  $0 \leq \alpha_i \leq \frac{C}{n}$  and  $\sum_{i=1}^n \alpha_i y_i = 0$ .
- The **sequential minimal optimization (SMO)** algorithm gives an efficient way of solving this problem.

# SMO algorithm

- The SMO algorithm is a **grouped coordinate ascent** procedure.
- Maximizing  $L_D(\alpha)$  one  $\alpha_i$  at a time does not work, because due to the constraint

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

variable  $\alpha_i$  is uniquely determined from the other  $\alpha_j$ 's through the equation

$$\alpha_i = -y_i \sum_{j \neq i} \alpha_j y_j.$$

- Instead, the SMO algorithm maximizes  $L_D(\alpha)$  w.r.t. to **each pair of variables**  $(\alpha_i, \alpha_j)$  sequentially.

## SMO algorithm (continued)

Repeat until convergence {

- 1 Select some pair  $\alpha_i$  and  $\alpha_j$  to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum).
- 2 Reoptimize  $L_D(\alpha)$  with respect to  $\alpha_i$  and  $\alpha_j$ , while holding all the other  $\alpha_k$ 's ( $k \neq i, j$ ) fixed.

}

To test for convergence of this algorithm, we can check whether the KKT conditions are satisfied to within some tolerance (see next slide).



# Interpretation of the solution

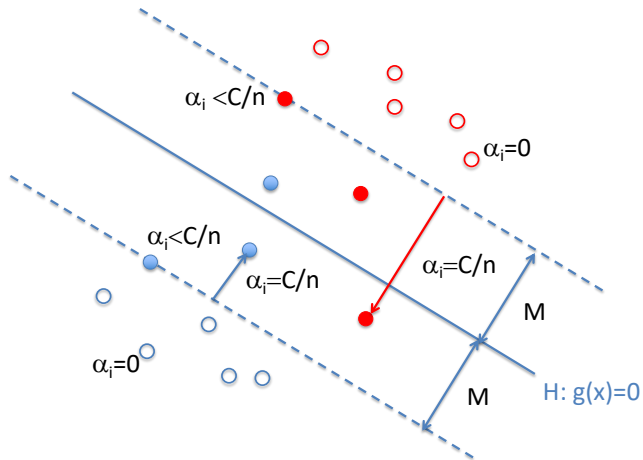
- The solution verifies the KKT conditions (13)-(14) and

$$\alpha_i^* [y_i(\beta^{*T} x_i + \beta_0^*) - (1 - \xi_i^*)] = 0, \quad i = 1, \dots, n \quad (16)$$

$$\mu_i^* \xi_i^* = 0, \quad i = 1, \dots, n \quad (17)$$

- As before, the SVs are defined as the points such that  $\alpha_i^* > 0$ .
- From (14) and (17), the SVs such that  $\alpha_i^* < C/n$  verify  $\mu_i^* > 0$  and  $\xi_i^* = 0$ : they lie on the edge of the margin (“in-bound SVs”). The remainder ( $\xi_i^* > 0$ ) have  $\alpha_i^* = C/n$  and usually lie inside the margin (“margin errors”).
- The SVs such that  $\xi_i^* > 1$  are misclassified.
- From (16) we can see that any of the in-bound SVs ( $\alpha_i^* > 0, \xi_i^* = 0$ ) can be used to solve for  $\beta_0^*$ , and we typically use an average of all the solutions for numerical stability.

# Interpretation



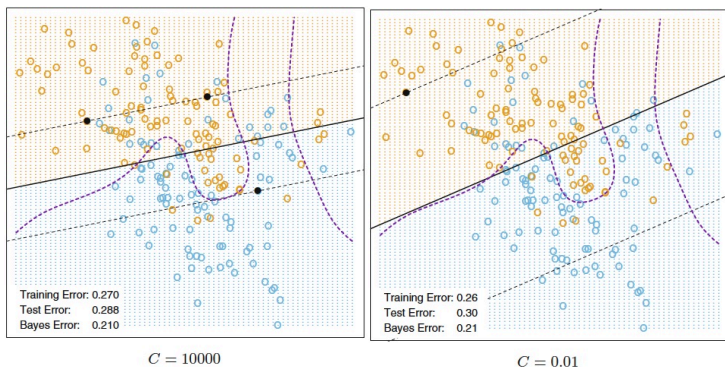
# Tuning $C$

- The tuning parameter of this procedure is the cost parameter  $C$ .
- The optimal value for  $C$  can be estimated by **cross-validation**.
- From (12), the margin is smaller for larger  $C$ . Hence larger values of  $C$  focus attention more on points near the decision boundary, while smaller values involve data further away.

## Bound on the LOO error

- The LOO cross-validation error can be bounded above by the proportion of SVs in the data.
- The reason is that leaving out an observation that is not a SV will not change the solution. Hence these observations, being classified correctly by the original boundary, will be classified correctly in the cross-validation process.
- However this bound tends to be too high, and not generally useful for choosing  $C$ .

# Example



The SVs ( $\alpha_i^* > 0$ ) are all the points on the wrong side of their margin. The black solid dots are in-bound SVs ( $\alpha_i^* < C/n$ ). In the left (resp., right) panel 62% (resp., 85%) of the observations are SVs.

# Application in R

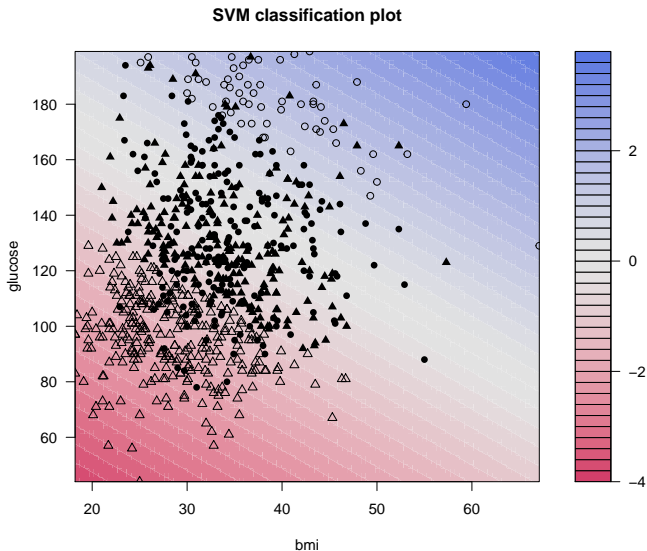
```
library("kernlab")

ii<-which((pima$glucose>0) & (pima$bmi>0))

svmfit<-ksvm(as.factor(class)~ glucose+bmi,data=pima[ii,],
             type="C-svc",kernel="vanilladot",C=10)

plot(svmfit,data=pima[ii,],grid=100)
```

## Result

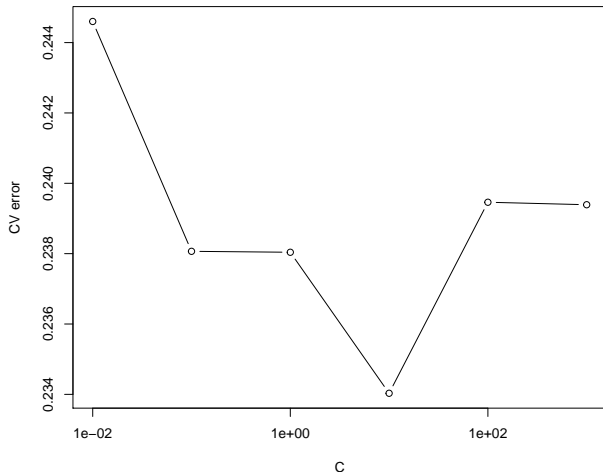


# Selection of $C$ by cross-validation

```
CC<-c(0.01,0.1,1,10,100,1000)
N<-length(CC)
err<-rep(0,N)
for(i in 1:N){
  err[i]<-cross(ksvm(as.factor(class)~glucose+bmi,data=pima[ii,],
                    type="C-svc",kernel="vanilladot",C=CC[i],cross=5))
}
plot(CC,err,type="b",log="x",xlab="C",ylab="CV error")
```



# Cross-validation result



# Overview

- 1 Optimal Separating hyperplane
  - Formalization
  - Solution in the separable case
  - Non-separable case
- 2 Support Vector Machines
  - The kernel trick
  - Kernel functions
  - SVM as a penalization method
- 3 Support Vector Regression
  - Loss function
  - Formalization
  - Solution and interpretation

# Overview

- 1 Optimal Separating hyperplane
  - Formalization
  - Solution in the separable case
  - Non-separable case
- 2 Support Vector Machines
  - The kernel trick
  - Kernel functions
  - SVM as a penalization method
- 3 Support Vector Regression
  - Loss function
  - Formalization
  - Solution and interpretation

# Extension to non-linear classification

- The support vector classifier described so far finds **linear boundaries** in the predictor space.
- As with other linear methods, we could make the procedure more flexible by **enlarging the predictor space** using basis expansions such as, e.g., polynomials or splines.
- Linear boundaries in the enlarged space generally achieve better training-class separation, and translate to nonlinear boundaries in the original space.

## Extension to non-linear classification (continued)

- Once the basis functions  $\Phi_j(x)$ ,  $j = 1, \dots, J$  are selected, the procedure is the same as before:
  - We fit the SV classifier using predictors

$$\Phi(x_i) = (\Phi_1(x_i), \Phi_2(x_i), \dots, \Phi_J(x_i)), \quad i = 1, \dots, n,$$

and produce the (nonlinear) function  $g^*(x) = \Phi(x)^T \beta^* + \beta_0$ .

- The classifier is  $D^*(x) = \text{sign}(g^*(x))$  as before.
- In SVM, the mapping  $x \rightarrow \Phi(x)$  will be defined **implicitly**, and  $J$  will be potentially very large (even infinite!).

# The OSH depends only on dot-products

A key feature of the OSH is that it depends only on the **dot products** between input vectors:

- The solution is found by maximizing

$$L_D(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j, \quad (18)$$

subject to  $0 \leq \alpha_i \leq C/n$  and  $\sum_{i=1}^n \alpha_i y_i = 0$ .

- The optimal discriminant function is

$$g^*(x) = \sum_{i \in S} \alpha_i^* y_i x_i^T x + \beta_0^* = 0,$$

where  $\beta_0^*$  also depends only on the dot products  $x_i^T x_j$ .

## Dot-products in the transformed input space

- Assume that the input vector  $x$  is replaced by  $\Phi(x)$  for some transformation  $\Phi : \mathbb{R}^p \rightarrow \mathcal{H}$ .
- The objective function will become

$$L_D(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \Phi(x_i), \Phi(x_j) \rangle \quad (19)$$

and the optimal discriminant function will be

$$g^*(x) = \sum_{i \in \mathcal{S}} \alpha_i^* y_i \langle \Phi(x_i), \Phi(x) \rangle + \beta_0^* = 0,$$

where  $\langle \cdot, \cdot \rangle$  denotes the dot-product in  $\mathcal{H}$ .

- All we need is a method to compute dot-products in  $\mathcal{H}$ .

# The “kernel trick”

- If there exists a **kernel function**  $\mathcal{K} : \mathbb{R}^p \times \mathbb{R}^p \rightarrow \mathbb{R}_+$  such that

$$\mathcal{K}(x, x') = \langle \Phi(x), \Phi(x') \rangle,$$

then the transformation  $\Phi$  will be defined **implicitly**.

- This is the “kernel trick”.



## Example

- Assume  $p = 2$  and  $\mathcal{K}(x, x') = (x^T x')^2$ .
- We have

$$\begin{aligned}\mathcal{K}(x, x') &= (x_1 x'_1 + x_2 x'_2)^2 \\ &= x_1^2 (x'_1)^2 + 2x_1 x_2 x'_1 x'_2 + x_2^2 (x'_2)^2 \\ &= \Phi(x)^T \Phi(x')\end{aligned}$$

with

$$\Phi : x \longrightarrow \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1 x_2 \\ x_2^2 \end{pmatrix}$$

- Function  $\Phi$  is **defined implicitly** by the kernel function  $\mathcal{K}$ .

# Overview

- 1 Optimal Separating hyperplane
  - Formalization
  - Solution in the separable case
  - Non-separable case
- 2 Support Vector Machines
  - The kernel trick
  - **Kernel functions**
  - SVM as a penalization method
- 3 Support Vector Regression
  - Loss function
  - Formalization
  - Solution and interpretation

# Mercer condition

## Theorem

A kernel function  $\mathcal{K}$  corresponds to a dot-product in some space  $\mathcal{H}$  iff it verifies the following *Mercer condition*:

$$\forall f : \mathbb{R}^p \rightarrow \mathbb{R} \text{ s.t. } \int f(x)^2 dx < \infty, \quad \int \mathcal{K}(x, x') f(x) f(x') dx dx' \geq 0.$$

- If the Mercer condition is not verified, the Wolfe dual problem may not have a solution.
- In practice, the method may still work most of the time with a kernel function that does not meet this condition.

# Popular kernel functions

- Three popular choices for  $\mathcal{K}$  in the SVM literature are

$$\mathcal{K}(x, x') = (a + b \cdot x^T x')^d, \quad d > 0 \quad (\text{polynomial kernel})$$

$$\mathcal{K}(x, x') = \exp[-\sigma \|x - x'\|^2], \quad \sigma > 0 \quad (\text{RBF or Gaussian kernel})$$

$$\mathcal{K}(x, x') = \tanh(a + b \cdot x^T x') \quad (\text{MLP kernel}).$$

- The polynomial and Gaussian verify the Mercer condition, but the MLP kernel does not.
- With the MLP kernel, the discriminant function is

$$g(x) = \sum_{i \in \mathcal{S}} \alpha_i^* y_i \tanh(a + b \cdot x_i^T x) + \beta_0^*.$$

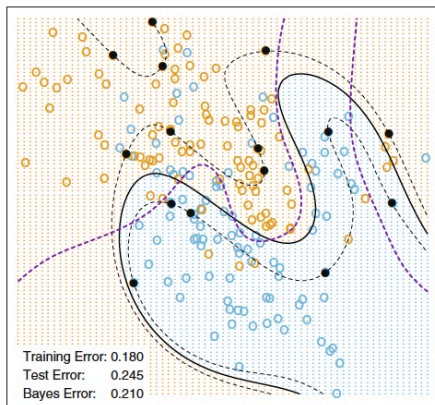
It is the transfer function of a neural network with  $n_S = \text{card}(\mathcal{S})$  hidden units (see chapter on neural networks).

# Influence of $C$

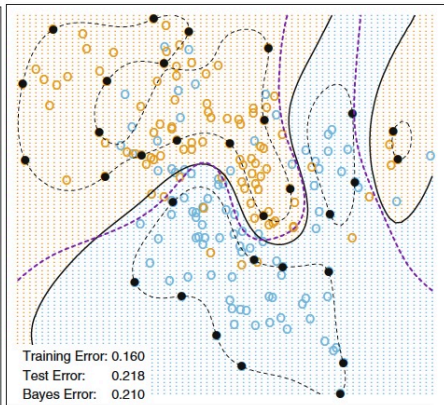
- The role of parameter  $C$  is clearer in an enlarged predictor space, since perfect separation is often achievable there. (The dimension of  $\mathcal{H}$  may be very large and even infinite.)
- A small value of  $C$  will encourage a small value of  $\|\beta\|$ , which in turn causes  $g(x)$  and hence the boundary to be smoother.
- Both  $C$  and the kernel parameters ( $a, b, d, \sigma$ , etc.) are usually tuned by cross-validation.

## Example

SVM - Degree-4 Polynomial in Feature Space



SVM - Radial Kernel in Feature Space



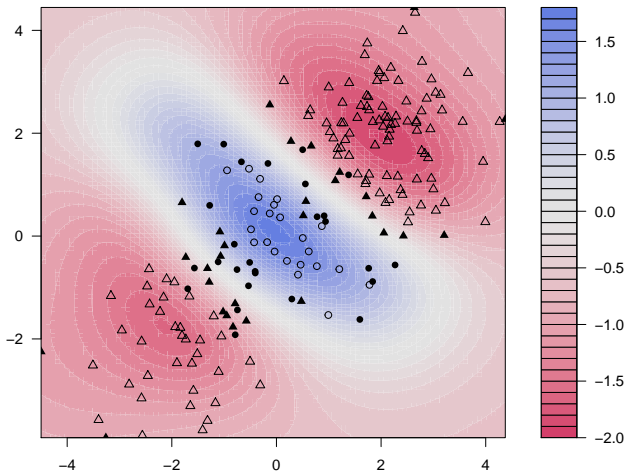
# Application in R

```
x<-matrix(rnorm(200*2),ncol=2)
y<-as.factor(c(rep(-1,150),rep(1,50)))
x[1:100,]<- x[1:100,]+2
x[101:150,]<- x[101:150,]-2

svmfit<-ksvm(x,y,type="C-svc",kernel="rbfdot",
             kpar=list(sigma=1),C=1)
plot(svmfit,data=x,grid=100)
```

## Result

SVM classification plot





# Estimation of posterior probabilities

- The SVM classifier gives us a decision function, but it does not provide estimates of conditional class probabilities

$$P(x) = \mathbb{P}(Y = +1 \mid X = x).$$

- One approach to estimate these probabilities is to use **logistic regression**, with the output  $g(x)$  of the SVM classifier as the predictor. We then have

$$\hat{P}(x) = \frac{1}{1 + \exp[-(a + b \cdot g(x))]}$$

- To avoid overfitting, it is preferable to estimate the additional parameters  $a$  and  $b$  from a **validation dataset** or using **cross-validation**.

# Overview

- 1 Optimal Separating hyperplane
  - Formalization
  - Solution in the separable case
  - Non-separable case
- 2 Support Vector Machines
  - The kernel trick
  - Kernel functions
  - SVM as a penalization method
- 3 Support Vector Regression
  - Loss function
  - Formalization
  - Solution and interpretation

# Unconstrained formulation

- Let  $g(x_i) = \langle \beta, \Phi(x_i) \rangle + \beta_0$ . The problem

$$\min_{\beta, \beta_0, \{\xi_i\}} \frac{1}{2} \|\beta\|^2 + \frac{C}{n} \sum_{i=1}^n \xi_i,$$

subject to  $\xi_i \geq 0$  and  $y_i g(x_i) \geq 1 - \xi_i$ ,  $i = 1, \dots, n$  is equivalent to the unconstrained optimization problem:

$$\min_{\beta, \beta_0} \sum_{i=1}^n \underbrace{[1 - y_i g(x_i)]_+}_{\text{"hinge" loss}} + \underbrace{\frac{\lambda}{2} \|\beta\|^2}_{\text{penalty}},$$

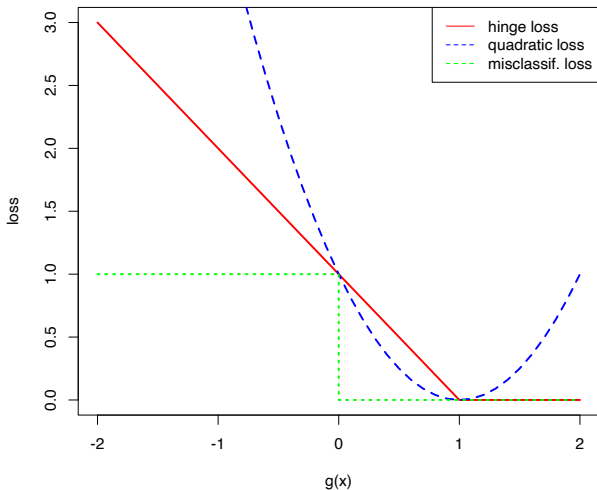
where  $[\cdot]_+$  denotes the positive part, with  $\lambda = n/C$ .

- Proof: see next slide.

# Hinge loss

- Proof: from  $\xi_i \geq 0$  and  $\xi_i \geq 1 - y_i g(x_i)$ , we get equivalently  $\xi_i \geq \max(0, 1 - y_i g(x_i)) = [1 - y_i g(x_i)]_+$ . Since we minimize  $\xi_i$ , we set  $\xi_i = [1 - y_i g(x_i)]_+$ .
  - The hinge loss can be compared to other loss functions such as:
    - Misclassification:  $I(\text{sign}(g(x)) \neq y)$
    - Squared error loss:  $(y - g(x))^2$
- (see next slide)

# Hinge loss (case $y = +1$ )



# Overview

- 1 Optimal Separating hyperplane
  - Formalization
  - Solution in the separable case
  - Non-separable case
- 2 Support Vector Machines
  - The kernel trick
  - Kernel functions
  - SVM as a penalization method
- 3 Support Vector Regression
  - Loss function
  - Formalization
  - Solution and interpretation

# Overview

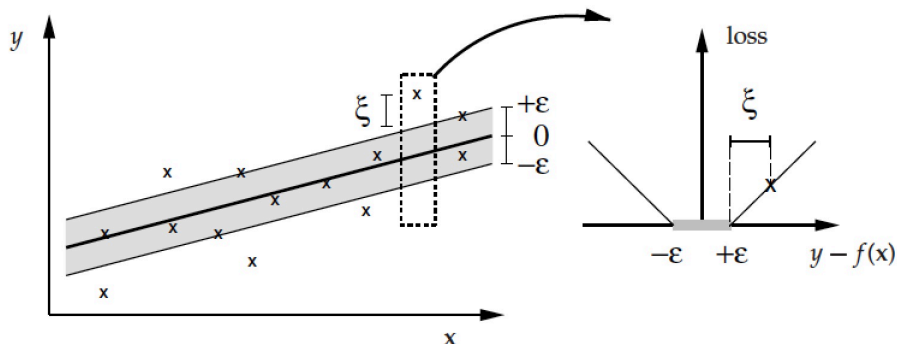
- 1 Optimal Separating hyperplane
  - Formalization
  - Solution in the separable case
  - Non-separable case
- 2 Support Vector Machines
  - The kernel trick
  - Kernel functions
  - SVM as a penalization method
- 3 Support Vector Regression
  - **Loss function**
  - Formalization
  - Solution and interpretation

# From classification to regression

- SVMs were first developed for classification.
- As described in the previous chapter, they represent the decision boundary in terms of a typically small subset of all training examples – the **Support Vectors**.
- To generalize the SV algorithm to regression, we need to find a way of retaining this feature. This can be achieved using the  **$\epsilon$ -insensitive loss function**

$$\begin{aligned} |f(x) - y|_{\epsilon} &= \begin{cases} 0 & \text{if } |f(x) - y| \leq \epsilon, \\ |f(x) - y| - \epsilon & \text{otherwise.} \end{cases} \\ &= [|f(x) - y| - \epsilon]_{+} \end{aligned}$$



$\epsilon$ -insensitive loss function

The  $\epsilon$ -insensitive loss function does not penalize errors below some  $\epsilon$ , chosen a priori. The  $\epsilon$ -insensitive zone is sometimes referred to as the  $\epsilon$ -tube.

# Basic approach

- The regression algorithm is then developed in close analogy to the case of classification.
- Again, we estimate linear functions, use a  $\|\beta\|^2$  regularizer, and rewrite everything in terms of dot products to generalize to the nonlinear case.

# Overview

- 1 Optimal Separating hyperplane
  - Formalization
  - Solution in the separable case
  - Non-separable case
- 2 Support Vector Machines
  - The kernel trick
  - Kernel functions
  - SVM as a penalization method
- 3 Support Vector Regression
  - Loss function
  - **Formalization**
  - Solution and interpretation

# Problem formulation

- We search for the linear function  $f(x) = \beta^T x + \beta_0$  minimizing the following criterion:

$$\underbrace{\frac{1}{2} \|\beta\|^2}_{\text{regularization}} + \frac{C}{n} \underbrace{\sum_{i=1}^n |f(x_i) - y_i|}_{\text{loss function}}.$$

- $C$  is a **hyperparameter**, which balances training error and model complexity.
- To solve this problem, we transform it into an equivalent constrained optimization problem.

# Reformulation as a constrained optimization problem

- We have

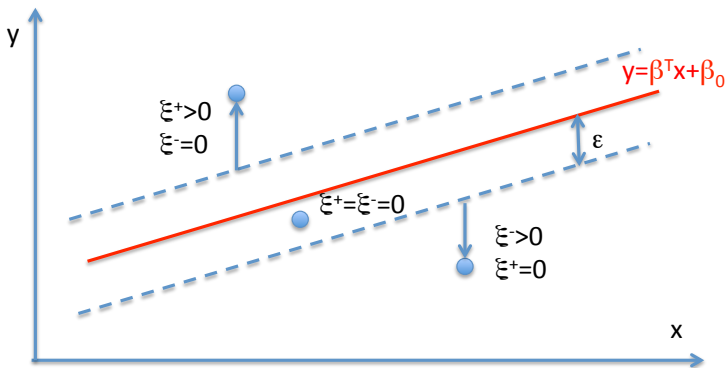
$$\begin{aligned}
 |f(x_i) - y_i|_\epsilon &= [|f(x_i) - y_i| - \epsilon]_+ \\
 &= \begin{cases} [f(x_i) - y_i - \epsilon]_+ = \xi_i^- & \text{if } f(x_i) \geq y_i \\ [y_i - f(x_i) - \epsilon]_+ = \xi_i^+ & \text{if } f(x_i) < y_i \end{cases}
 \end{aligned}$$

- Furthermore,  $\xi_i^- = 0$  if  $f(x_i) < y_i$ , and  $\xi_i^+ = 0$  if  $f(x_i) \geq y_i$ . We can thus write

$$|f(x_i) - y_i|_\epsilon = \xi_i^+ + \xi_i^-$$

- The quantities  $\xi_i^+$  and  $\xi_i^-$  are called “slack variables” (they will become slack variables in the constrained optimization formulation of the problem)

## Representation of the slack variables



# Primal objective function

Using the slack variables, the previous problem can be reformulated as a quadratic optimization problem:

$$\min_{\beta, \beta_0, \xi^-, \xi^+} \frac{1}{2} \|\beta\|^2 + \frac{C}{n} \sum_{i=1}^n (\xi_i^- + \xi_i^+)$$

subject to:

$$\xi_i^+ \geq y_i - \beta^T x_i - \beta_0 - \epsilon$$

$$\xi_i^+ \geq 0$$

$$\xi_i^- \geq \beta^T x_i + \beta_0 - y_i - \epsilon$$

$$\xi_i^- \geq 0$$

for  $i = 1, \dots, n$ .

# Lagrange function

The Lagrange function is

$$\begin{aligned}
 L(\beta, \beta_0, \alpha_i^-, \alpha_i^+, \eta_i^-, \eta_i^+) &= \frac{1}{2} \|\beta\|^2 + \frac{C}{n} \sum_{i=1}^n (\xi_i^- + \xi_i^+) \\
 &\quad - \sum_{i=1}^n (\eta_i^- \xi_i^- + \eta_i^+ \xi_i^+) - \sum_{i=1}^n \alpha_i^- (\epsilon + \xi_i^- + y_i - \beta^T x_i - \beta_0) \\
 &\quad \quad \quad - \sum_{i=1}^n \alpha_i^+ (\epsilon + \xi_i^+ - y_i + \beta^T x_i + \beta_0),
 \end{aligned}$$

where  $\alpha_i^-$ ,  $\alpha_i^+$ ,  $\eta_i^-$ ,  $\eta_i^+$  are Lagrange multipliers.



# Derivatives of the Lagrange function

- Setting the derivatives to zero, we obtain:

$$\frac{\partial L}{\partial \beta} = \beta - \sum_{i=1}^n (\alpha_i^+ - \alpha_i^-) x_i = 0,$$

$$\frac{\partial L}{\partial \beta_0} = \sum_{i=1}^n (\alpha_i^- - \alpha_i^+) = 0,$$

$$\frac{\partial L}{\partial \xi_i^-} = \frac{C}{n} - \alpha_i^- - \eta_i^- = 0, \quad i = 1, \dots, n$$

$$\frac{\partial L}{\partial \xi_i^+} = \frac{C}{n} - \alpha_i^+ - \eta_i^+ = 0, \quad i = 1, \dots, n.$$

- We use these relations to simplify the expression of the Lagrange function (see next slide).

## Simplification of the Lagrange function

$$\begin{aligned}
L = & \frac{1}{2} \left( \sum_i (\alpha_i^+ - \alpha_i^-) x_i \right)^T \left( \sum_j (\alpha_j^+ - \alpha_j^-) x_j \right) \\
& + \sum_{i=1}^n \xi_i^- \underbrace{\left( \frac{C}{n} - \eta_i^- - \alpha_i^- \right)}_0 + \sum_{i=1}^n \xi_i^+ \underbrace{\left( \frac{C}{n} - \eta_i^+ - \alpha_i^+ \right)}_0 \\
& - \epsilon \sum_i (\alpha_i^+ + \alpha_i^-) - \beta_0 \underbrace{\sum_i (\alpha_i^+ - \alpha_i^-)}_0 + \sum_i y_i (\alpha_i^+ - \alpha_i^-) \\
& - \sum_{i=1}^n (\alpha_i^+ - \alpha_i^-) \left( \sum_j (\alpha_j^+ - \alpha_j^-) x_j \right)^T x_i
\end{aligned}$$

# Dual problem

$$L_D(\alpha_i^-, \alpha_i^+) = -\frac{1}{2} \sum_{i,j} (\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-) x_i^T x_j \\ - \epsilon \sum_{i=1}^n (\alpha_i^+ + \alpha_i^-) + \sum_{i=1}^n y_i (\alpha_i^+ - \alpha_i^-),$$

to be maximized subject to

$$\sum_{i=1}^n (\alpha_i^+ - \alpha_i^-) = 0$$

$$0 \leq \alpha_i^- \leq \frac{C}{n}, \quad i = 1, \dots, n,$$

$$0 \leq \alpha_i^+ \leq \frac{C}{n}, \quad i = 1, \dots, n.$$

# Overview

- 1 Optimal Separating hyperplane
  - Formalization
  - Solution in the separable case
  - Non-separable case
- 2 Support Vector Machines
  - The kernel trick
  - Kernel functions
  - SVM as a penalization method
- 3 Support Vector Regression
  - Loss function
  - Formalization
  - Solution and interpretation

# Support vectors

- As in the case of SVMs, the dual problem can be solved using any quadratic programming solver.
- Let  $\alpha_i^{-*}, \alpha_i^{+*}, i = 1, \dots, n$  be the solution.
- The learning vectors  $x_i$  such that  $\alpha_i^{-*} > 0$  or  $\alpha_i^{+*} > 0$  are called the **support vectors**. They lie outside the tube (or at the border).
- Let  $\mathcal{S}$  be the set of support vectors. We have

$$\beta^* = \sum_{i \in \mathcal{S}} (\alpha_i^{+*} - \alpha_i^{-*}) x_i$$

and

$$f^*(x) = \sum_{i \in \mathcal{S}} (\alpha_i^{+*} - \alpha_i^{-*}) x_i^T x + \beta_0^*$$

# Sparsity of the SV expansion

- We thus have a **sparse expansion** of  $\beta$  in terms of  $x_i$  (we do not need all  $x_i$  to compute  $\beta^*$ ).
- The points inside the tube (i.e., which are not support vectors) do not contribute to the solution: we could remove any one of them, and still obtain the same solution.

# Karush-Kuhn-Tucker conditions

- The solution  $\alpha_i^{-*}, \alpha_i^{+*}, i = 1, \dots, n$  must satisfy the KKT conditions

$$\alpha_i^{-*}(\epsilon + \xi_i^{-*} + y_i - \beta^{*T} x_i - \beta_0^*) = 0 \quad (20a)$$

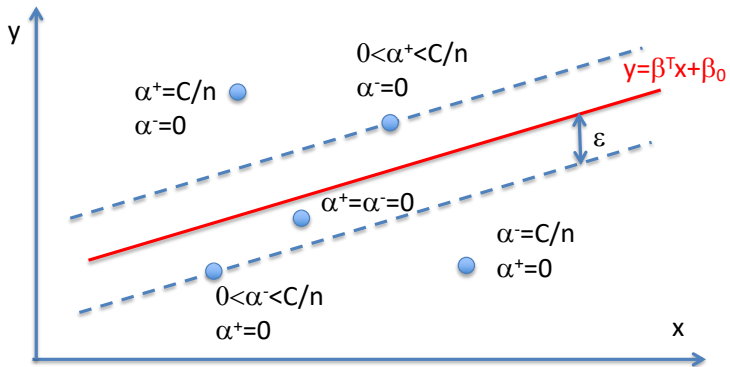
$$\alpha_i^{+*}(\epsilon + \xi_i^{+*} - y_i + \beta^{*T} x_i + \beta_0^*) = 0 \quad (20b)$$

$$\left(\frac{C}{n} - \alpha_i^{-*}\right) \xi_i^{-*} = 0, \quad \left(\frac{C}{n} - \alpha_i^{+*}\right) \xi_i^{+*} = 0 \quad (20c)$$

- Consequences:

- Only examples  $(x_i, y_i)$  with corresponding  $\alpha_i^{-*} = C/n$  or  $\alpha_i^{+*} = C/n$  can lie outside the tube (i.e.,  $\xi_i^{-*} > 0$  or  $\xi_i^{+*} > 0$ ).
- When  $\alpha_i^{+*} \in (0, C/n)$  or  $\alpha_i^{-*} \in (0, C/n)$ , we have  $\xi_i^{+*} = \xi_i^{-*} = 0$ .  
The corresponding SVs lie at the border of the tube (see next slide).

# Interpretation of $\alpha_i^+$ and $\alpha_i^-$





# Calculation of $\beta_0$

- $\beta_0^*$  can be calculated from (20a) or (20b) for SVs at the border of the tube as

$$\beta_0^* = \begin{cases} \epsilon + y_i - \beta^{*T} x_i & \text{for } \alpha_i^{-*} \in (0, C/n) \\ y_i - \beta^{*T} x_i - \epsilon & \text{for } \alpha_i^{+*} \in (0, C/n) \end{cases}$$

- Theoretically, it suffices to use any Lagrange multiplier in  $(0, C/n)$ .
- If given the choice between several such multipliers in  $(0, C/n)$ , it is safer to use one that is not too close to 0 or  $C/n$ .

# Parameter tuning

The solution depends on two parameters,  $\epsilon$  and  $C$ . These play different roles:

- Parameter  $\epsilon$  in the loss function specifies the desired accuracy of the approximation. If we scale our response, then we might consider using preset values for  $\epsilon$ .
- The quantity  $C$  is a more traditional regularization parameter. It can be estimated, for example, by cross-validation.

# Nonlinear extension

- As in the classification case, the complete algorithm can be described in terms of **dot products** between the data.
- This makes it possible to formulate a nonlinear extension using **kernels**, replacing dot products  $x_i^T x_j$  in  $\mathcal{X}$  with dot products

$$\langle \Phi(x_i), \Phi(x_j) \rangle = \mathcal{K}(x_i, x_j)$$

in  $\mathcal{H}$ .

- Additional kernel parameters may be determined by cross-validation.

# Application in R

```
library('kernlab')
library('MASS')
mcycle.data<-data.frame(mcycle)
mcycle.data$accel<-scale(mcycle.data$accel)
t<- seq(min(mcycle.data$times),max(mcycle.data$times),0.5)
testdat<-data.frame(times=t)

svmfit<-ksvm(accel~.,data=mcycle.data,scaled=TRUE,type="eps-svr",
             kernel="rbfdot",C=100,epsilon=0.1,kpar=list(sigma=1))

yhat<-predict(svmfit,newdata=testdat)
plot(mcycle.data$times,mcycle.data$accel)
lines(t,yhat)
```

## Result

