Support Vector Machine (SVM)

R. SUBLET [1], C. COURTES [1]

[1]IRMA
University of Strasbourg

PhD seminar: February 29, 2024

Introduction

SVM in the case of two classes $\{-1, +1\}$: find a function $f : \mathcal{X} \to \mathrm{R}$ which will be used to discriminate classes: $f(x) > 0$ will correspond to class $+1$ and $f(x) < 0$ to class -1.
We train the method on a training set $\{(x_i, y_i)\}_{i=1,..,n}$ where $x_i \in \mathcal{X} = \mathrm{R}^d$ and $y_i \in \{+1, -1\}$.

Summary

1. SVM with rigid margin
   - Theory
   - Programming
2. Generalization of the method
   - SVM with flexible margin
   - Non-linear case
3. Separation into several classes
   - Theory
   - Application

Summary

1. SVM with rigid margin
   - Theory
   - Programming

2. Generalization of the method
   - SVM with flexible margin
   - Non-linear case

3. Separation into several classes
   - Theory
   - Application

Linearly separable data

Here we suppose that the data are linearly separable, and only two classes. That means there is a hyperplane H separating our data in two classes.

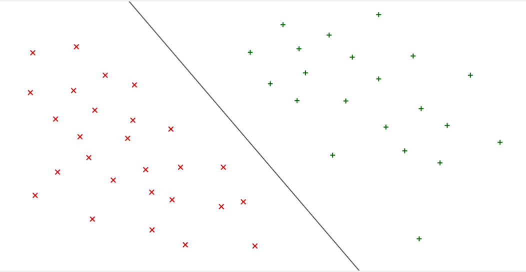


Figure: Two classes and a separator

Primal problem

H has equation $\langle w, x \rangle + b = 0$. The function f is therefore

$$f(x) = \text{sign}(\langle w, x \rangle + b)$$

Question : How to choose w and b? We introduce the Margin such as the distance off the hyperplane H to the closest point :

$$\text{Margin} = \min_{x \in \mathcal{X}} d(x, H)$$

Our goal will be to maximize the margin. We first note that only few points of $\mathcal{X}$ have importance for the calculus of the margin ($= \min_{x \in \mathcal{X}} d(x, H)$) : we call them **support vector**.

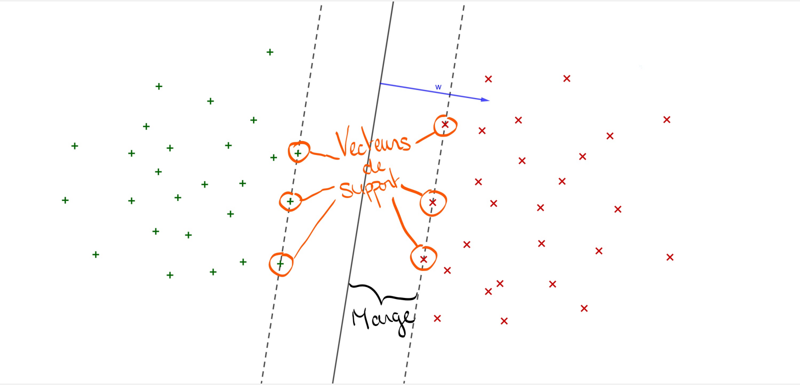


Figure: Support vector

We denote those support vectors by x_s, the margin is now given by:

$$\text{Margin} = d(x_s, H) = \frac{|\langle w, x_s \rangle + b|}{\|w\|}$$

where the chosen norm is the Euclidean norm on $\mathbb{R}^d$.
Note: w and b are not unique, since kw and kb are also solutions for any real k. To ensure uniqueness, we also impose the normalization condition: $|\langle w, x_s \rangle + b| = 1$ for support vectors x_s.
Finally,

$$\text{Margin} = \frac{1}{\|w\|}.$$

This brings us back to the mathematical problem of maximization of the margin. But we prefer write this problem as a minimization problem as follow :

$$\left\{ \begin{array}{l} \min\limits_{w \in \mathrm{R}^d, b \in \mathrm{R}} \dfrac{1}{2} \|w\|^2 \\ y_i(\langle w, x_i \rangle + b) \geq 1, i = 1, ..., n \end{array} \right. \qquad \text{(Pb primal)}$$

Lagrangian

We introduce the Lagrangian of this minimization problem

$$\begin{array}{rccl} \mathcal{L} & : & \mathrm{R}^d \times \mathrm{R} \times \mathrm{R}^n_+ & \rightarrow & \mathrm{R} \\ & & (w, b, \alpha) & \mapsto & \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(\langle w, x_i\rangle + b) - 1] \end{array}$$

Saddle point of the Lagrangian

A saddle point of this Lagrangian is $(w^*, b^*, \alpha^*) \in \mathrm{R}^d \times \mathrm{R} \times \mathrm{R}^n_+$ such as for all (w, b, α), we have $\mathcal{L}(w^*, b^*, \alpha) \leq \mathcal{L}(w^*, b^*, \alpha^*) \leq \mathcal{L}(w, b, \alpha^*)$.

Thanks to Kuhn-Tucker theorem, we need to find a saddle point of the Lagrangian, and this point satisfies the primal problem. So first we need to minimize $\mathcal{L}$ over (w, b) :

$$\begin{cases} \nabla_w \mathcal{L}(w^*, b^*, \alpha^*) & = & 0 \\ \dfrac{\partial \mathcal{L}}{\partial b}(w^*, b^*, \alpha^*) & = & 0. \end{cases}$$

which leads to :

$$\begin{cases} \displaystyle\sum_{k=1}^{n} \alpha_k^* y_k x_k & = & w^* \\ \displaystyle\sum_{k=1}^{n} \alpha_k^* y_k & = & 0. \end{cases} \tag{1}$$

Re-injecting these expressions into the formula of the Lagrangian, for $\mathcal{L}(w, b, \alpha)$, we thus define the function $\theta(\alpha) = \mathcal{L}(w^*, b^*, \alpha)$:

$$\theta(\alpha) = \sum_{k=1}^{n} \alpha_k - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle.$$

We now have to maximize this function, so the problem is now write :

$$\begin{cases} \max\limits_{\alpha \in \mathrm{R}^n} \theta(\alpha) \\ \sum\limits_{k=1}^{n} \alpha_k y_k = 0 \\ \alpha_k \geq 0 \text{ for all } 1 \leq k \leq n. \end{cases}$$

To recover w^* with the formula $\sum_{k=1}^{n} \alpha_k^* y_k x_k = w^*$, so we need to have α^*. To find α^*, we solve the minimization problem :

$$\begin{cases} \min\limits_{\alpha \in \mathrm{R}^n} -\theta(\alpha) = \min\limits_{\alpha \in \mathrm{R}^n} \left(-\sum_{k=1}^{n} \alpha_k + \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \right) \\ \sum_{k=1}^{n} \alpha_k y_k = 0 \\ \alpha_k \geq 0 \text{ for all } 1 \leq k \leq n. \end{cases} \quad \text{(Pb dual)}$$

And we find b^* with : $y_s(\langle w^*, x_s \rangle + b^*) = 1$. To be sure we have a support vector, we have to chose x_s such as α_s be maximal.

We have to resolve this optimization problem. It is a minimization problem with constraints

$$\begin{cases} \min\limits_{w \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2} \|w\|^2 \\ y_i(\langle w, x_i \rangle + b) \geq 1, i = 1, ..., n \end{cases} \qquad \text{(Pb primal)}$$

We use the penalisation method. We introduce $\epsilon > 0$ and call the penalized function $\tilde{J}$ as follow :

$$\tilde{J}(w, b) = \frac{1}{2} \|w\|^2 + \frac{1}{\epsilon} \frac{1}{n} \|c(w, b)_+\|_1 \qquad (2)$$

where $c_i(w, b)_+ = \max(1 - y_i(\langle w, x_i \rangle + b) \; ; \; 0)$

$$\begin{cases} \min_{\alpha \in \mathrm{R}^n} \langle \alpha, \frac{1}{2} G\alpha - U \rangle \\ \sum_{k=1}^{n} \alpha_k y_k = 0 \\ \alpha_k \geq 0 \text{ for all } 1 \leq k \leq n. \end{cases}, \qquad \text{(Pb dual)}$$

where $G = (\langle x_i y_i, x_j y_j \rangle)_{i,j}$ and $U = (1, ..., 1)^T$. We will use an optimal (or constant) step gradient algorithm for the quadratic function $\alpha \mapsto \langle \alpha, \frac{1}{2} G\alpha - U \rangle$, coupled with a projection onto R_+^n and $\mathrm{Vect}(y)^\perp$ to take constraints into account.

Warning : The matrix G is not necessarily positive definite and therefore our function is not necessarily convex!

Convexification

Tikhonov's regularization method : We introduce $\nu > 0$ and define the function F :

$$F(\alpha) = \langle \alpha, \frac{1}{2}(G + \nu I_n)\alpha - U \rangle$$

We therefore have :

$$\nabla F(\alpha) = (G + \nu I_n)\alpha - U$$

We choose ν so that $G + \nu I_n$ is symmetrical positive definite.

Method : Gradient algorithm on the quadratic function F coupled with a projection on $\mathrm{Vect}(y)^{\perp}$ and on R_{+}^{n}.
If we denote ρ^k the descent step, the method at iteration k is written :

$$\alpha^{k+1} = \alpha^k - \rho^k \nabla F(\alpha^k)$$

Then for projection:
$\lambda^k := \alpha^k - \langle \alpha^k, \frac{y}{\|y\|} \rangle \frac{y}{\|y\|}$
$\alpha_i^k = \max(0, \lambda_i^k)$

Choice of ρ^k :

- fixed-step gradient method $\rho^k = \rho$ fixed.
- optimal step gradient method.

Property

Let $Q(x) = \frac{1}{2}\langle Ax, x\rangle - \langle b, x\rangle$ be a quadratic function where $A \in \mathcal{S}n^{++}(\mathrm{R})$ and $b \in \mathrm{R}^n$, then the optimal step at iteration k, denoted ρ_o^k is given by

$$\rho_o^k = \frac{\|Ax_k - b\|^2}{\langle A(Ax_k - b), Ax_k - b\rangle} \tag{3}$$

For our quadratic function, we have the optimal step given by :

$$\rho^k = \frac{\|d^k\|^2}{\langle d^k, (G + \nu I_n)d^k\rangle}$$

where $d^k = (G + \nu I_n)\alpha^k - U$

Model comparison

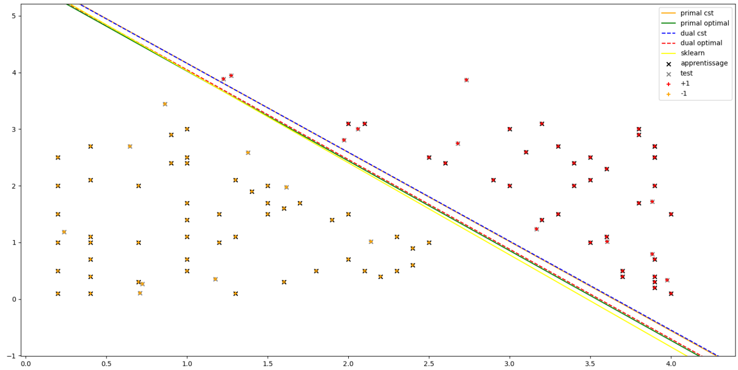


Figure: Different results compare with sklearn: optimal step are the best.

Summary

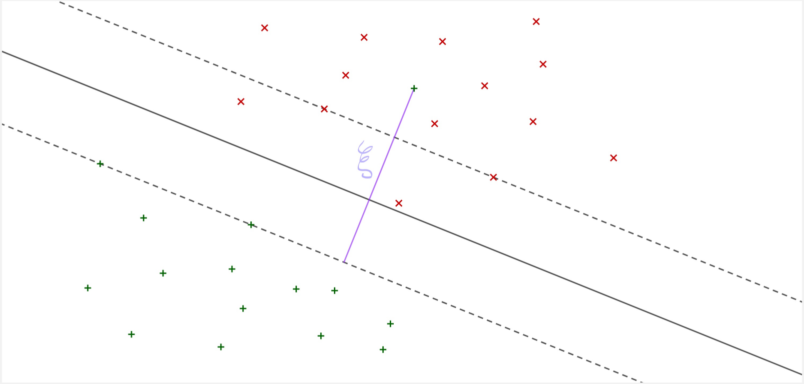
1. SVM with rigid margin
   - Theory
   - Programming

2. Generalization of the method
   - SVM with flexible margin
   - Non-linear case

3. Separation into several classes
   - Theory
   - Application

Data cannot be strictly separated

We add constraint release variables $\xi_i = \max(0, 1 - y_i(\langle w, x_i \rangle + b))$ for each constraint, called the hinge loss.



Optimization problem

Our primal problem become:

$$\begin{cases} \min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \\ y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i, \ i = 1, ..., n \\ \xi_i \geq 0, \ i = 1, ..., n \end{cases} \tag{4}$$

(where $C > 0$ is a balancing variable)

No-linear case

In many cases we can't find a linear separator. But SVM can deal with that with the kernel tricks.

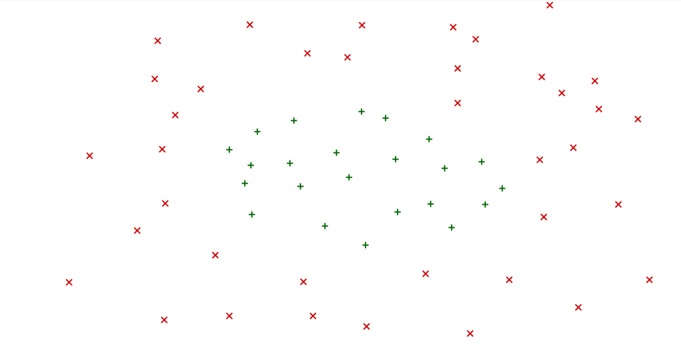


Figure: No linear separation

Principle: We solve the SVM problem in a feature space $\mathcal{H}$ where data are linearly separable. $\mathcal{H}$ is an Hilbert space with the scalar product $\langle . \, , . \rangle_{\mathcal{H}}$. We introduce the representative function $\phi : \mathcal{X} \to \mathcal{H}$. We have to solve the following dual problem

$$\begin{cases} \min_{\alpha \in \mathrm{R}^n} \left(- \sum_{k=1}^{n} \alpha_k + \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}} \right) \\ \sum_{k=1}^{n} \alpha_k y_k = 0 \\ \alpha_k \geq 0 \text{ for all } 1 \leq k \leq n. \end{cases} \qquad \text{(Pb dual)}$$

Kernel Trick

Warning ! The scalar product $\langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}}$ can be time-consuming, both computationally and memory-intensive to evaluate.

We call the **kernel function** $k : \mathcal{X} \times \mathcal{X} \to \mathrm{R}$ such that there exists a representation function ϕ and a representation space $\mathcal{H}$ such that $k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}\ \forall (x, x') \in \mathcal{X}$.

Kernel trick

So the dual problem become :

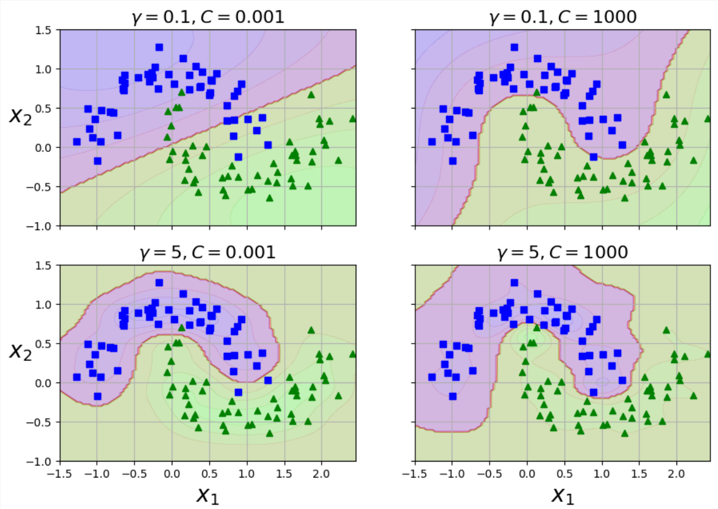
$$\begin{cases} \min\limits_{\alpha \in \mathrm{R}^n} \left(-\sum_{k=1}^{n} \alpha_k + \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \right) \\ \sum_{k=1}^{n} \alpha_k y_k = 0 \\ \alpha_k \geq 0 \text{ for all } 1 \leq k \leq n. \end{cases} \qquad \text{(Pb dual)}$$

Example of classic kernel :

1. Polynomial kernel : $k(x, x') = (\langle x, x' \rangle + c)^p$.
2. Radial basis function (RBF) : $k(x, x') = \exp \frac{\|x - x'\|^2}{2\sigma^2}$.

Example of RBF kernel

With the kernel $k(x, x') = \exp(\gamma \|x - x'\|^2)$ and flexible margin :



Summary

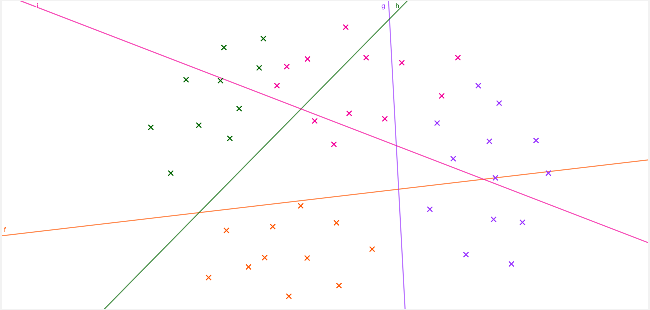
1. SVM with rigid margin
   - Theory
   - Programming

2. Generalization of the method
   - SVM with flexible margin
   - Non-linear case

3. Separation into several classes
   - Theory
   - Application

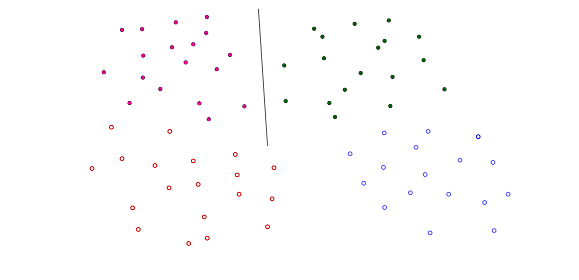
OVR (One Versus Rest)

This involves creating a function for each k classes, which separates the elements of that class from all other elements. We call g_k the function linked to the following class k : $g_k(x) = \langle w_k, x \rangle + b_k$. The class of a point will be given by $\tilde{k} = \arg\min_k g_k(x)$.



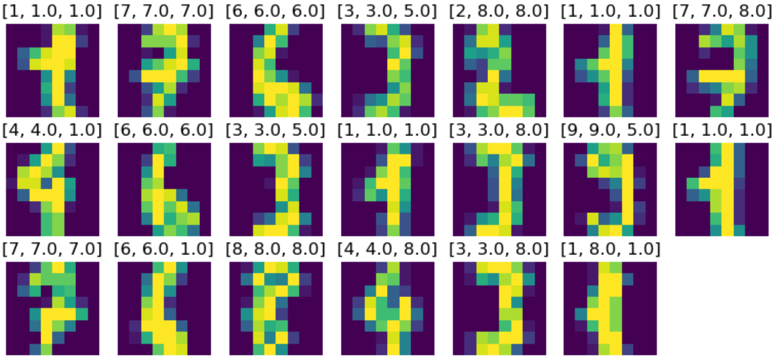
OVO (One Versus One)

This involves creating a classifying function for each class, separates it from every other class. In this way, the classes are tested two by two. If we have n classes, we will build $n(n-1)/2$ classifying functions. We then make a majority vote.



Handwriting recognition

Her we take 80 learning data and 20 test data :



legend : [reference,OVR,OVO]

Proportion OVO

For 20 test data

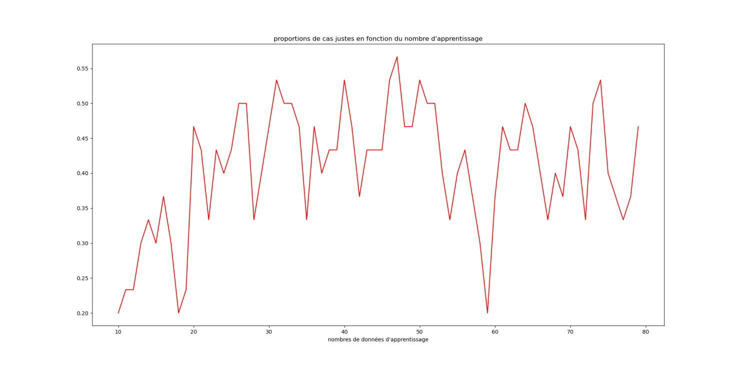


Figure: Percentage of correct estimation by number of learning data

Proportion OVR

For 20 test data

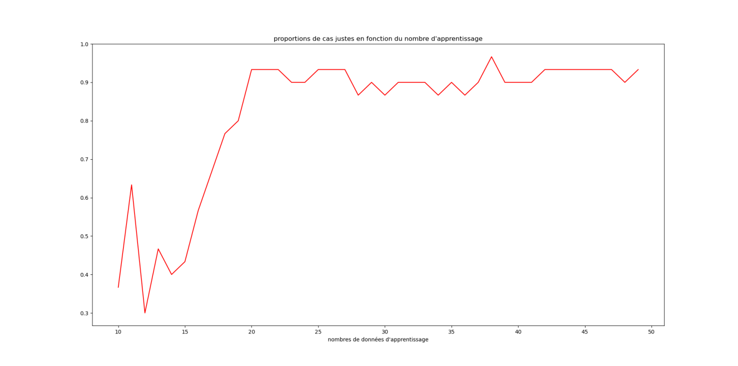


Figure: Percentage of correct estimation by number of learning data

Conclusion

The SVM method can be used to find the class of an object, provided we have enough training data and find the right kernel to use.

Conclusion

The SVM method can be used to find the class of an object, provided we have enough training data and find the right kernel to use.

Thanks for your attention