Optimization Project: Support Vector Machine

K. Kamtue & Cl. Réda

ENS Cachan

January 12th, 2017

Project

Support Machine Vector

Objective

Classify data

Project

Support Machine Vector

Objective

Classify data

- Applied to **binary classification** ($y_i \in \{1, -1\}$);

Project

Support Machine Vector

Objective

Classify data

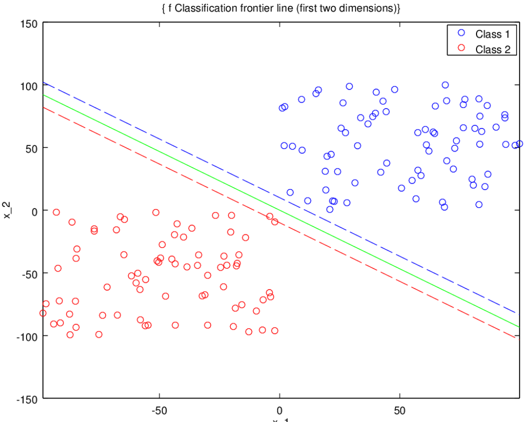
- Applied to **binary classification** ($y_i \in \{1, -1\}$);
- Looking for a **hyperplane** $f : x \rightarrow \omega^T x$ such as:

$$\forall i, f(x_i) = \begin{cases} < 0 & \text{si y=-1} \\ > 0 & \text{si y=1} \end{cases} \Leftrightarrow \forall i, y_i \times f(x_i) > 0 \qquad (1)$$

Project

Support Machine Vector

Figure: Example with two classes (**red** and **blue**)



Optimization problem

Looking for the optimization problem

Naive optimization problem

γ: distance between the lines $f(x) = 1$ and $f(x) = -1$.

Optimization problem

Looking for the optimization problem

Naive optimization problem

γ: distance between the lines $f(x) = 1$ and $f(x) = -1$.

$$max_w \ \gamma = \frac{2}{\|w\|}$$
$$\text{subject to } \forall i, y_i \times f(x_i) > 0$$

Optimization problem

Looking for the optimization problem

Naive optimization problem

γ: distance between the lines $f(x) = 1$ and $f(x) = -1$.

$$max_w \ \gamma = \frac{2}{\|w\|}$$
$$\text{subject to } \forall i, y_i \times f(x_i) > 0$$

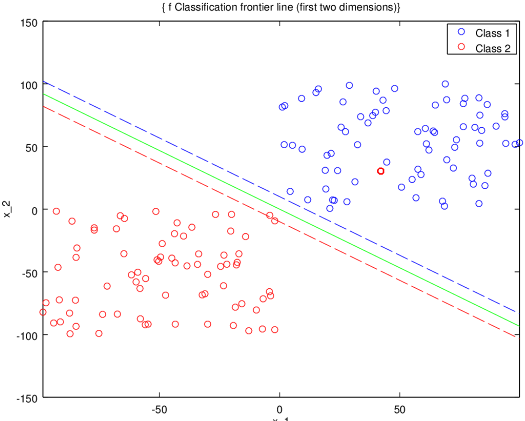
$$\Leftrightarrow min_w \ \frac{1}{2}\|w\|^2$$
$$\text{subject to } \forall i, y_i \times f(x_i) > 0$$

Beware: if the data set is not linearly separable!

Optimization

Looking for the optimization problem

Figure: Example with two classes (red and blue)



Optimization problem

Adapting the problem to **non-separable sets**

Let z_i be $max(0, 1 - y_i \times f(x_i))$ (**Hinge loss**).

Optimization problem

Adapting the problem to **non-separable sets**

Let z_i be $max(0, 1 - y_i \times f(x_i))$ (**Hinge loss**).

Having the problem convex and always feasible

Penalty for **classification errors** with $(z_i)_i$ and C :

$$min_{w,z} \ \frac{1}{2}\|w\|^2 + C \sum_{i \leq m} z_i$$
subject to
$$\forall i, z_i \geq 0$$
$$\forall i, y_i \times (\omega^T x_i) \geq 1 - z_i$$

Implementation

Solving the optimization problem

- Use **Newton's method** to find ω :

Reminder: Update of ω with Newton's method

$$\omega_{n+1} \leftarrow \omega_n + s \times \nabla^2 obj(\omega_n)^{-1} \nabla obj(\omega_n)$$

(finding **step size** value s by **backtracking line search**)

Implementation

Solving the optimization problem

- Use **Newton's method** to find ω :

Reminder: Update of ω with Newton's method

$$\omega_{n+1} \leftarrow \omega_n + s \times \nabla^2 obj(\omega_n)^{-1} \nabla obj(\omega_n)$$

(finding **step size** value s by **backtracking line search**)

- Make the problem independant from **dimension**;

Implementation

Solving the optimization problem

- Use **Newton's method** to find ω :

Reminder: Update of ω with Newton's method

$$\omega_{n+1} \leftarrow \omega_n + s \times \nabla^2 obj(\omega_n)^{-1} \nabla obj(\omega_n)$$

(finding **step size** value s by **backtracking line search**)

- Make the problem independant from **dimension**;
- Use **logarithmic barrier method**.

Implementation

Independance from dimension: **dual problem**

After Lagrangian calculus and minimization in ω:

Implementation

Independance from dimension: **dual problem**

After Lagrangian calculus and minimization in ω:

Dual problem

$$max_{\lambda \in \mathbb{R}^{+m}} - \frac{1}{2} \| \sum_i \lambda_i y_i x_i \|_2^2 + \mathbf{1}^T \lambda$$
$$\text{subject to } \forall i, 0 \leq \lambda_i \leq C$$
(KKT conditions)

Implementation

Independance from dimension: **dual problem**

After Lagrangian calculus and minimization in ω:

Dual problem

$$max_{\lambda \in \mathbb{R}^{+m}} -\frac{1}{2}\|\sum_i \lambda_i y_i x_i\|_2^2 + \mathbf{1}^T\lambda$$

subject to $\forall i, 0 \leq \lambda_i \leq C$

(KKT conditions)

Get **primal solution** from **dual solution**

$$\omega^* = \sum_i \lambda_i^* y_i x_i$$

Implementation

Make the problem independant from dimension

Use the **kernel trick** :

Dual problem

Let K be $X^T X$ (**linear kernel**):

$$\max -\frac{1}{2}\lambda^T diag(y) K diag(y)\lambda + \mathbf{1}^T\lambda$$
$$\text{subject to } \forall i, 0 \leq \lambda_i \leq C$$

Implementation

Delete inequality constraints

Use the **logarithmic barrier method** :

Implementation

Delete inequality constraints

Use the **logarithmic barrier method** :

Barrier function

$$\begin{aligned}\Phi(\lambda) &= \sum_i(-log(C-\lambda_i) - log(\lambda_i)) \\ &= -\sum_i log((C-\lambda_i)\lambda_i)\end{aligned}$$

Implementation

Delete inequality constraints

Use the **logarithmic barrier method** :

Barrier function

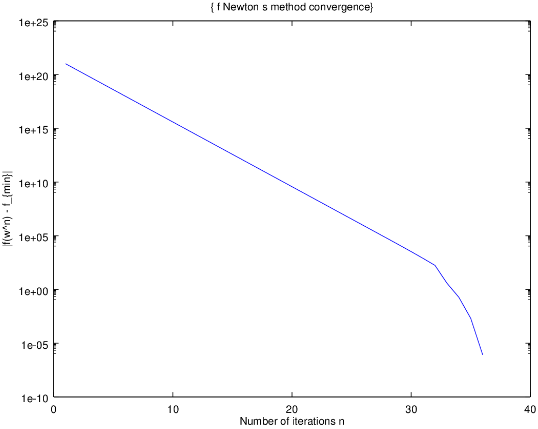
$$\Phi(\lambda) = \sum_i (-log(C - \lambda_i) - log(\lambda_i)) \\ = -\sum_i log((C - \lambda_i)\lambda_i)$$

Final optimization problem

$$max \; -\frac{1}{2}\lambda^T diag(y) K diag(y) \lambda + \mathbf{1}^T \lambda + \Phi(\lambda)$$

Testing the implementation

Newton's method convergence



Testing the implementation

Dependance on the sample size

Table: Time complexity dependance

Set	C	d	n	Iteration number	Time
1	5	40000	10	11	0.315
1	5	40	100	12	0.715
1	5	40	1000	large	$>$ 1,000

Testing the implementation

Speeding of convergence when C increases

Performed on the same sample set:

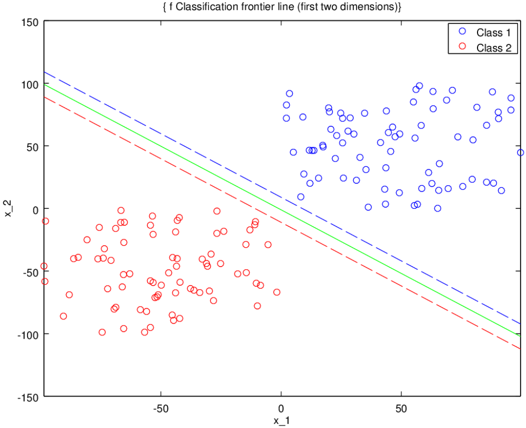
Table: Computation time in function of C

Test Set	C	d	n	#iterations	Time	Fail (%)
1	1	40	10	11	25.414	0
1	5	40	10	11	0.177	0
1	10	40	10	11	0.168	0

Plotting the **classification frontier**

Pour $C = 5, n = 150, d = 200$

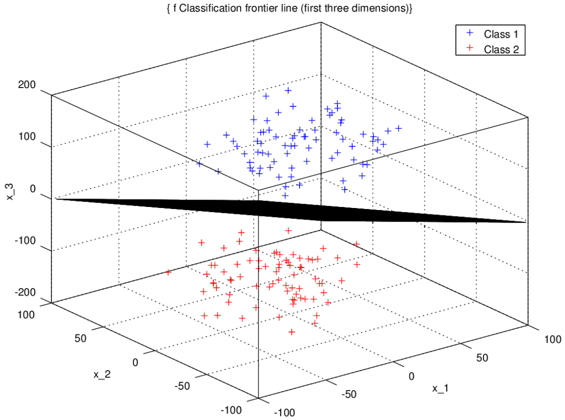
Points centrés réduits avec des fonctions gaussiennes (2D) :



Tracé de la frontière de classification

Pour $C = 5, n = 150, d = 200$

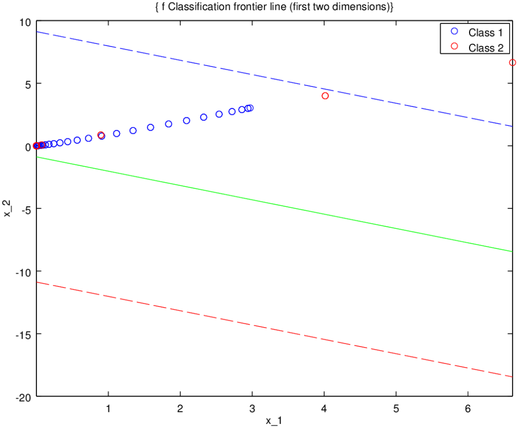
Points centrés réduits avec des fonctions gaussiennes (3D) :



Tracé de la frontière de classification

Pour $C = 5, n = 150, d = 200$

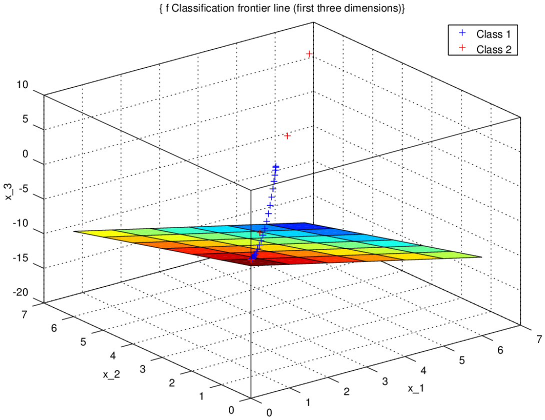
Génération avec des fonctions gaussiennes (2D) :



Tracé de la frontière de classification

Pour $C = 5, n = 150, d = 200$

Génération avec des fonctions gaussiennes (3D) :



Extensions

Adding to the project

- **Cross validation** (choice of the best value for C);

Extensions

Adding to the project

- **Cross validation** (choice of the best value for C);
- Implementation of **Coordinate Descent**;

Extensions

Adding to the project

- **Cross validation** (choice of the best value for C);
- Implementation of **Coordinate Descent**;
- Implementation of **ACCPM**;

Demo of the SVM