Rapport sur le projet d'Optimisation
Support-Vector Machines

Kawisorn Kamtue & Clémence Réda

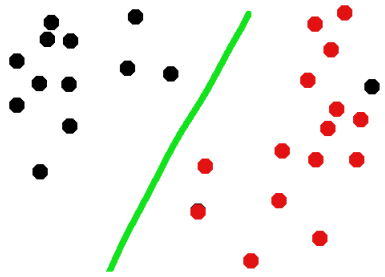
December 19, 2016

1 Support Vector Machine

Les *Support Vector Machine solvers* (SVM) sont une catégorie d'algorithmes d'apprentissage statistique supervisé. Ils permettent de résoudre le problème de classification binaire suivant :

Etant donnés $(x_i)_{i\leq m}$ des points dans $\mathbb{R}^n$, et $(y_i)_{i\leq m}$ les étiquettes des points tels que l'étiquette de x_i soit $y_i \in \{-1, 1\}$, on cherche la droite qui sépare "le mieux possible" les points dans différentes classes, autrement dit, la frontière de Voronoi entre les deux classes.

Figure 1: Exemple de frontière pour deux classes : celles des points noirs et celle des points rouges



La frontière que l'on recherche est une fonction linéaire, donc de la forme (avec deux paramètres de dimension 1 ω et b) :

$$f : X \to \omega^T X + b = \begin{bmatrix} \omega \\ \mathrm{b} \end{bmatrix} \begin{bmatrix} \mathrm{x} \\ 1 \end{bmatrix}$$

telle que :

$$\begin{gathered} \forall i, y_i = -1 \Rightarrow f(x_i) \leq -1 \\ \forall i, y_i = 1 \Rightarrow f(x_i) \geq 1 \\ \Leftrightarrow \forall i, y_i \times f(x_i) \geq 1 \quad (1) \end{gathered}$$

Pour simplifier le problème, on peut prendre $\omega' = \begin{bmatrix}\omega \\ \text{b}\end{bmatrix}$ et $x' = \begin{bmatrix}\text{x} \\ 1\end{bmatrix}$ (que l'on notera par souci de simplicité ω et x).

Pour obtenir un résultat robuste, on souhaite que les deux droites $f(X) = 1$ et $f(X) = -1$ soient les plus distantes possibles. En effet, si ces deux droites sont trop proches, cela signifie que la probabilité d'erreur quant à la prédiction de la classe d'un point proche de ces droites sera importante.

La distance γ entre ces deux droites se calcule de la façon suivante : soient u, v deux points tels que $f(v) = 1$ et $f(u) = -1$. Alors :

$$\|f(v) - f(u)\| = \|\omega \times (v - u)\| = \|\omega\| \times \|(v - u)\| = \|\omega\| \times \|\gamma\| = \|1 - (-1)\| = 2$$

Finalement, le problème d'optimisation à résoudre pourrait être :

$max_w\ \gamma = \frac{2}{\|w\|}$ avec (1)
$\Leftrightarrow min_w\ \|w\|$ avec (1)
$\Leftrightarrow min_w\ \frac{1}{2} \times \|w\|^2$ avec (1) pour faciliter les calculs

Un autre problème se pose si on s'arrête ici : par exemple, dans l'exemple de la frontière de Voronoi que l'on a vu ci-dessus, il n'existe pas de droite telle qu'il n'y ait que de points noirs d'un côté et que des points rouges de l'autre côté, ce qui contredit la condition (1). Le problème est alors infaisable. Pourtant, la droite dessinée en vert peut sembler acceptable comme frontière pour cet ensemble de points.

On tient compte de cette erreur en introduisant les variables $(z_i)_{i \leq m}$. Pour que la condition (1) soit toujours vérifiée, il faut que quand $y_i \times f(x_i) \geq 1$, $z_i = 0$ et lorsque $y_i \times f(x_i) < 1$, $z_i = 1 - y_i \times f(x_i)$. Le but étant de minimiser le nombre de ces erreurs, ie. points mal classés, on utilise un paramètre C constant qui permet d'insister plus ou mins sur la minimisation de ces erreurs :

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

Les fonctions que l'on a introduites sont toutes convexes. Si la dimension des points $(x_i)_i$ est petite, nous allons pouvoir utiliser la méthode de Newton pour résoudre ce problème. On verra par la suite le *kernel trick* qui permettra de ne pas tenir compte de la dimension, mais seulement du nombre d'échantillons $(x_i)_i$.

2 Calcul du dual

Calculons le lagrangien du problème (P). Soit λ le multiplicateur de Lagrange de dimension $1 \times m$:

$$\begin{gathered}\forall w, \lambda \in \mathbb{R}^{2m}, L(\omega, \lambda, z) = \\ = \frac{1}{2}\|w\|^2 + C \times \sum_i z_i - \sum_i \lambda_i \times z_i + \sum_i \lambda_i \times (1 - y_i \omega^T x_i) \\ = \frac{1}{2}\|w\|^2 + C\mathbf{1}^T z - C\lambda^T z + \mathbf{1}^T \lambda - \sum_i \lambda_i \times y_i \omega^T x_i \\ = \frac{1}{2}(\|w - \sum_i \lambda_i y_i x_i\|_2^2 - \|\sum_i \lambda_i y_i x_i\|_2^2) + (C\mathbf{1} - \lambda)^T z + \mathbf{1}^T \lambda\end{gathered}$$

Minimisons L par rapport à ω. Comme le lagrangien est convexe en ω, il faut annuler le gradient :

$$\begin{gathered}\nabla_\omega L(\omega, \lambda, z) = \frac{1}{2}(2\omega - 2\sum_i \lambda_i y_i x_i) + 0 = 0 \\ \Leftrightarrow \omega = \sum_i \lambda_i y_i x_i \ (1)\end{gathered}$$

Minimisons L par rapport à z. Comme le lagrangien est convexe en z, il faut annuler le gradient :

$$\begin{gathered}\nabla_z L(\omega, \lambda, z) = 0 + C\mathbf{1}{1_{z>0}}^T - \lambda {1_{z>0}}^T = 0 \\ \Leftrightarrow mC - \sum_i \lambda_i = 0 \text{ si } z_i > 0 \\ \Leftrightarrow mC = \sum_i \lambda_i \text{ si } z_i > 0 \ (2)\end{gathered}$$

Le minimum en L par rapport à z a une valeur finie ssi $C\mathbf{1} - \lambda = 0$. On obtient le problème dual en injectant les valeurs de ω et de z dans le lagrangien :

$$\begin{gathered}max_{\lambda \in \mathbb{R}^{+m}} -\frac{1}{2}\|\sum_i \lambda_i y_i x_i\|_2^2 + \mathbf{1}^T \lambda \text{ par } (1) \\ \text{avec } \forall i, 0 \leq \lambda_i \leq C \text{ si } z_i > 0 \text{ (vient de (2))}\end{gathered}$$

On obtient la solution optimale du primal (ω^*, z^*) à partir de celle du dual λ^* :

$$(1)\ \omega^* = \sum_i \lambda_i^* y_i x_i$$

3 Utilisation de l'astuce du noyau (*kernel trick*)

Pour pouvoir trouver efficacement la solution au problème avec la méthode de Newton, il faut s'affranchir de la contrainte quadratique sur la dimension des échantillons. On note X la matrice des échantillons, et la matrice du noyau $K = X^T X$, avec $K \geq 0$. On montre alors que le problème dual peut se réécrire de la façon suivante :

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

On remarque que la dimension m des échantillons n'intervient plus, et que donc la complexité de la résolution du problème ne dépend que du nombre d'échantillons.

4 Méthode de la barrière logarithmique

Enfin, on peut s'affranchir des contraintes d'inégalité sur le multiplicateur de Lagrange λ en posant la fonction barière suivante :

$$\Phi(\lambda) = \sum_i(-log(C-\lambda_i) - log(\lambda_i)) = \sum_i log(\frac{1}{(C-\lambda_i)\lambda_i}) = -\sum_i log((C-\lambda_i)\lambda_i)$$

Le problème à optimiser devient alors :

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

5 Résultats

5.1 Comparaison entre les différentes générations de points

d est la dimension des points et n le nombre d'échantillons dans la génération.

Table 1: Comparaison entre les générations de points

Génération	C	d	n	N itérations	Temps Newton (s)	Taux de succès (%)
1	1	2	10	11	81,017	100
1	5	2	10	11	0,614381	100
1	10	2	10	11	0,3478	100
2	10	2	40	12	0,379235	100
2	100	2	40	12	0,512416	100
3	20	2	40	12	0,387925	78,7
4	10	2	40	12	0,32249	75,0
5	20	2	50	12	0,465928	57,3

5.2 Points dans les quadrans $(x, y > 0)$ et $(x > 0, y < 0)$

On génère les points selon la procédure 1 dans *generatedata.m*. Les points de la première classe sont dans le quadran $(x, y > 0)$ et ceux de la seconde classe sont dans le quadran $(x > 0, y > 0)$.

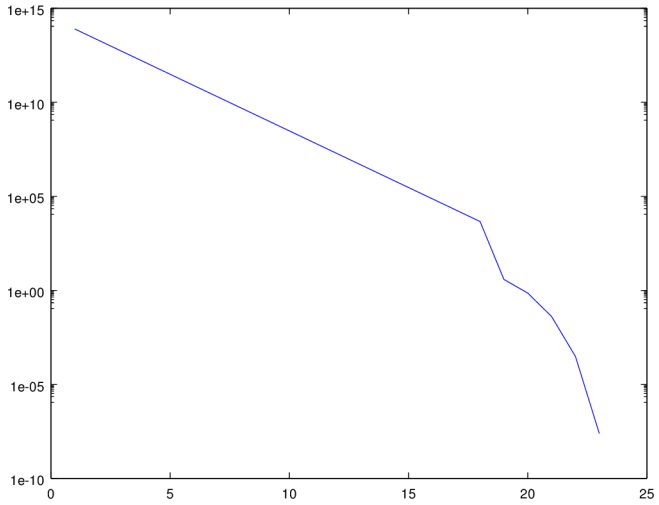


Figure 2: Convergence vers le minimum de la fonction objectif par la méthode de Newton (échelle semi-log)

$$x = \begin{bmatrix} 79,4566 & 6,3054 & 10,2126 & 58,7432 & 96,0460 & ... \\ 56,1444 & 46,6231 & 20,4822 & 60,5679 & 12,0744 & ... \\ 1,0000 & 1,0000 & 1,0000 & 1,0000 & 1,0000 & ... \end{bmatrix}$$
$$\begin{bmatrix} ... & -89,3088 & -5,7061 & -15,7588 & -1,9675 & -59,3514 \\ ... & -42,7449 & -17,0933 & -58,5002 & -21,2904 & -48,6966 \\ ... & 1,0000 & 1,0000 & 1,0000 & 1,0000 & 1,0000 \end{bmatrix}$$
$$y = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

5.2.1 Pour $C = 1$

$$\lambda = \begin{bmatrix} 8,6314.10^{-7} \\ 1,2616.10^{-6} \\ 1,0513.10^{-6} \\ 9,8250.10^{-7} \\ 6,7446.10^{-7} \\ 1,3976.10^{-6} \\ 9,4634.10^{-7} \\ 8,2267.10^{-7} \\ 9,0923.10^{-7} \\ 1,0908.10^{-6} \end{bmatrix}$$

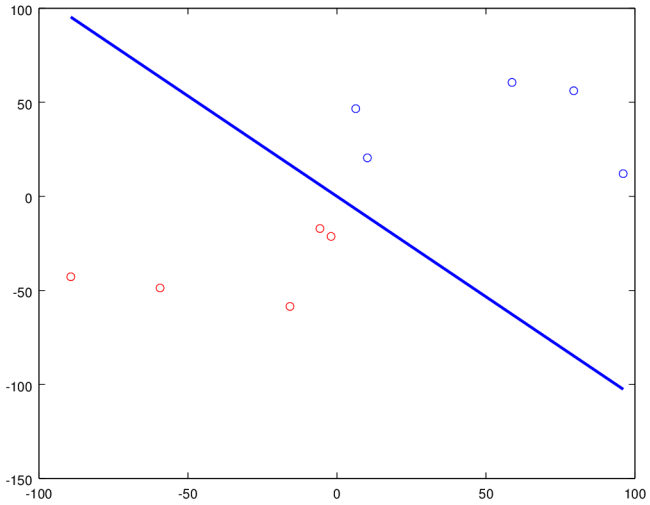


Figure 3: Tracé de la frontière de classification entre les points (bleus pour la première classe, rouges pour la deuxième)

$$\omega = \begin{bmatrix} 4,1948.10^{-4} \\ 3,9299.10^{-4} \\ -3,3357.10^{-7} \end{bmatrix}$$

On retrouve les mêmes courbes, et les mêmes valeurs de λ et de ω, pour les valeurs suivantes de C, ce qui paraît cohérent.

5.3 Points centrés réduits générés à partir de deux fonctions gaussiennes

5.3.1 Pour $C = 10$

On génère les points selon la procédure 2 dans *generatedata.m*. On tire les coordonnées en utilisant la fonction *randn*, qui retourne des éléments centrés réduits générés par une Gaussienne, auxquels on retire ou ajoute 10. Les x et y sont stockés dans le fichier *test2* dans le dossier *test*.

5.3.2 Validation croisée pour le choix de la meilleure valeur de C

Les deux fonctions *choiceC* et *crossvalidation* permettent de sélectionner la meilleure valeur de C pour un échantillon donné, par la méthode de *leave-one-out*, où, pour un échantillon de taille n, à chaque itération on choisit un élément e comme ensemble de test, et l'entraînement du SVM se fait sur les $n - 1$

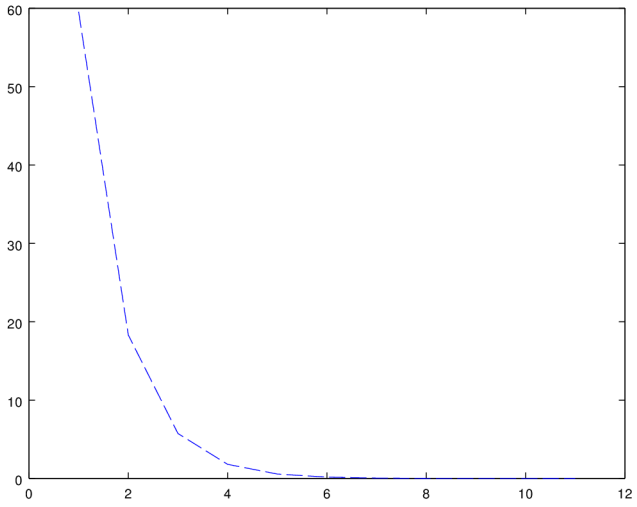


Figure 4: *Duality gap* : tracé de $\|\omega^*\| - \|a^*\|$ en fonction du nombre d'itérations de la méthode de Newton

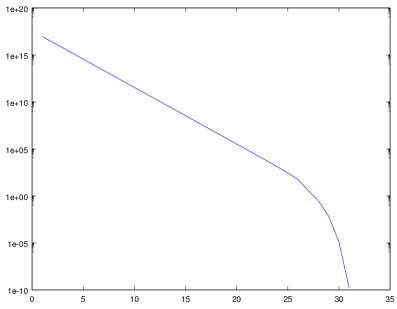


Figure 5: Convergence vers le minimum de la fonction objectif par la méthode de Newton (échelle semi-log)

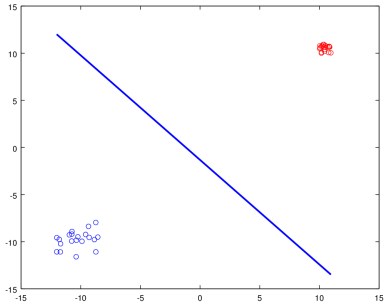


Figure 6: Tracé de la frontière de classification entre les points (bleus pour la première classe, rouges pour la deuxième)

éléments restants. La valeur de C qui permet d'obtenir une erreur globale (sur l'ensemble d'itérations) minimale est considérée la meilleure. On teste ces fonctions sur l'échantillon ci-dessus, en recherchant la meilleure valeur de C entre C minimum et C maximum :

Table 2: Recherche de la meilleure valeur de C

C maximum	C minimum	Meilleure valeur
13	15	13
10	15	10
5	10	5

5.4 Points centrés réduits générés avec des fonctions gaussiennes

On utilise la procédure 2 dans *generatedata.m* avec $sep = 100$. Voir le fichier *test3* pour les valeurs de x et y.

Table 3: Matrice de confusion

Réalité/Prédiction	Classe 1	Classe 2
Classe 1	98	12
Classe 2	52	138

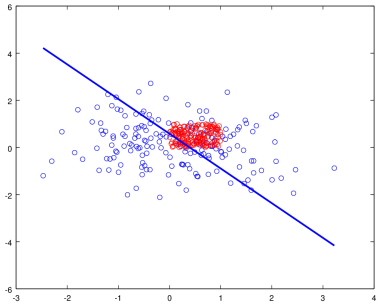


Figure 7: Tracé de la frontière de classification entre les points (bleus pour la première classe, rouges pour la deuxième)

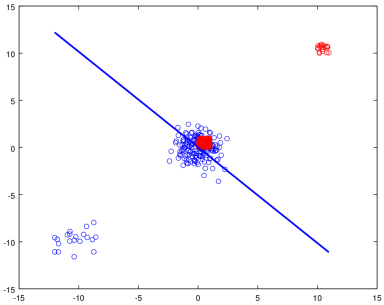


Figure 8: Tracé de la frontière de classification entre les points (bleus pour la première classe, rouges pour la deuxième)

5.5 Points centrés réduits générés avec des fonctions gaussiennes

On utilise la procédure 2 dans *generatedata.m* avec $sep = 0$. Voir le fichier *test4* pour les valeurs de x et y.

Table 4: Matrice de confusion

Réalité/Prédiction	Classe 1	Classe 2
Classe 1	75	0
Classe 2	75	150

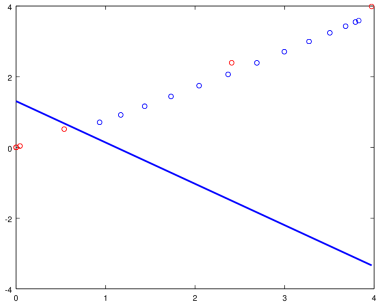


Figure 9: Tracé de la frontière de classification entre les points (bleus pour la première classe, rouges pour la deuxième)

5.6 Points générés avec des fonctions gaussiennes

On utilise la procédure 3 dans *generatedata.m* avec les paramètres par défaut. Voir le fichier *test5* pour les valeurs de x et y.

Table 5: Matrice de confusion

Réalité/Prédiction	Classe 1	Classe 2
Classe 1	26	4
Classe 2	124	146