Rapport sur le projet d'Optimisation
Support-Vector Machines

Kawisorn Kamtue & Clémence Réda

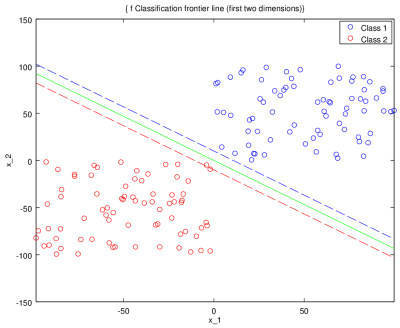
January 4, 2017

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 : celle des points bleus et celle des points rouges



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

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

telle que :

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

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).

L'hyperplan auquel ω est normal sépare l'espace en deux parties qui représentent deux classes. Pour obtenir un résultat robuste, on souhaite que la marge (*i.e.* la somme des distances entre l'hyperplan et les points les plus proches) soit la plus grande possible. En effet, plus la marge est petite, plus la probabilité d'erreur de prédiction de la classe d'un point proche de ces droites sera importante.

La marge γ se calcule de la façon suivante : soient u, v deux points (parmi les plus proches à l'hyperplan) 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 :

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

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, si on a un point bleu au milieu des points rouges, alors il n'existe pas de droite telle qu'il n'y ait que de points bleus 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 l'ensemble de points auquel on a ajouté un point bleu au milieu du nuage de points rouge.

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 :

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

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 $1 \times m$:

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

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 :

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

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 :

$$\begin{gathered}max_{\lambda \in \mathbb{R}^{+m}} -\frac{1}{2}\|\sum_i \lambda_i y_i x_i\|_2^2 + \mathbf{1}^T\lambda \text{ par (1)} \\ \text{avec } \forall i, 0 \leq \lambda_i \leq C \text{ si } z_i > 0 \\ \text{(vient des conditions de KKT -}\textit{complementary slackness}\text{,} \\ \text{vérifiées car le problème est convexe)}\end{gathered}$$

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 :

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

ce qui équivalent à :

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

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 :

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

Cette fontion vaut $+\infty$ si $a < 0$ ou $a > C$.Le problème à optimiser devient alors (en changeant le signe pour obtenir une fonction à minimiser) :

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

5 Résultats

5.1 Comparaison entre les différentes générations de points

5.1.1 Tableau récapitulatif

d est la dimension des points et n le nombre d'échantillons dans la génération. On utilise $\frac{2}{3}$ des points (choisis au hasard uniformément) de l'ensemble de départ pour l'ensemble d'entraînement, et les points restants pour l'ensemble de validation.

Table 1: Comparaison entre les générations de points

Données	C	d	n	N it.	Temps (s)	Meilleur C	Echec (%)
1	1	40	10	11	25,414	1	0 (*)
1	5	40	10	11	0,177	1	0 (*)
1	10	40	10	11	0,168	1	0 (*)
1	5	40000	10	11	0,315	X	0
1	5	40	100	12	0,715	X	0
1	5	40	1000	?	¿ 10	?	?
2	5	200	150	12	0,689	?	0
3	5	200	150	12	0,660	?	0
4	5	200	150	12	0,655	?	0
5	5	200	150	12	0,709	?	4

Quelques remarques :

- De manière générale, utiliser une valeur de C plus grande accélère considérablement la recherche de la solution duale : pas au niveau du nombre d'itérations de la méthode de Newton, mais au niveau du coût de l'appel à la méthode de Newton (voir les trois premiers tests).
- La complexité temporelle de la résolution du problème dual est bien indépendante de la dimension et dépendante de la taille de l'échantillon (voir les tests 4, 5 et 6).
- La valeur "?" signifie que l'algorithme a tourné trop longtemps pour la valeur soit mesurée.
- La notation "(*)" signifie que les tests marqués ont utilisé le même ensemble de données.

5.1.2 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 $n-1$ éléments restants. La valeur de C qui permet d'obtenir une erreur globale (sur l'ensemble d'itérations) minimale est considérée la meilleure.

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

On génère les points selon la procédure $m = 2$ 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 $(x > 0, y > 0)$. Voir le fichier *test1.mat* dans le dossier *test*. La meilleure valeur, au niveau du nombre d'erreurs, de C, choisie par validation croisée dans l'intervalle [1, 10], est 1 (les autres valeurs de C de 2 à 10 donnent le même nombre d'erreurs pour cet ensemble de données).

5.2.1 Pour $C = 1, n = 10, d = 40$

Table 2: Matrice de confusion pour l'ensemble de données 1

Réalité/Prédiction	Classe 1	Classe 2
Classe 1	2	0
Classe 2	0	1

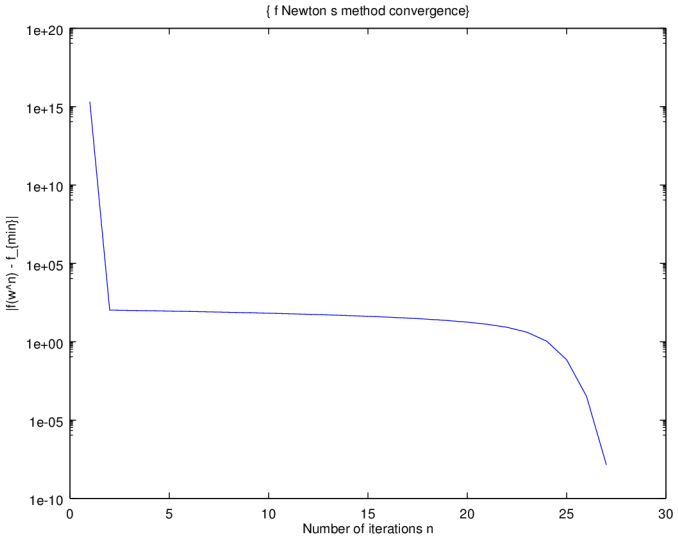


Figure 2: Ensemble de données 1 (échelle semi-log)

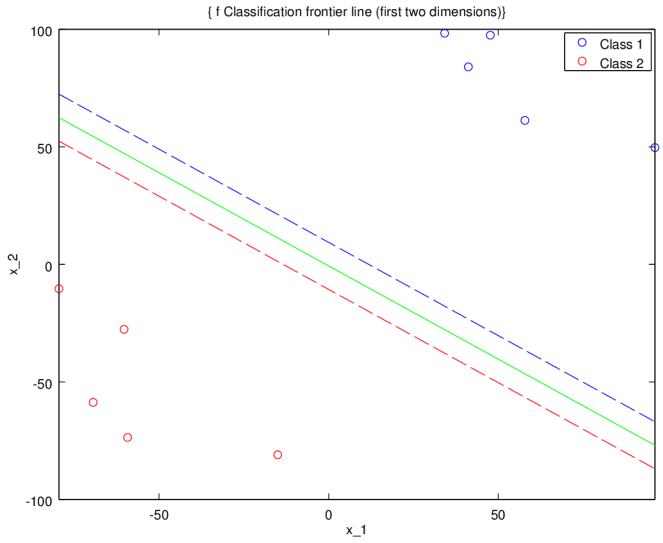


Figure 3: Ensemble de données 1 (Les lignes rouge et bleue en pointillés sont les droites f(x) = 10 et f(x) = -10)

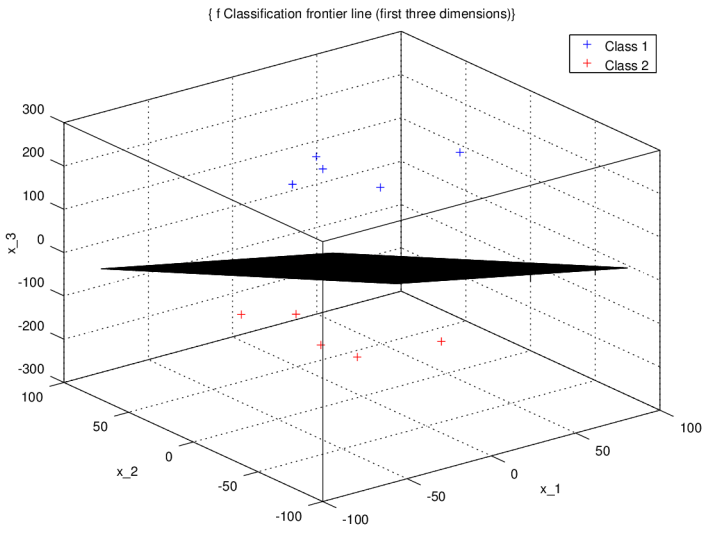


Figure 4: Ensemble de données 1

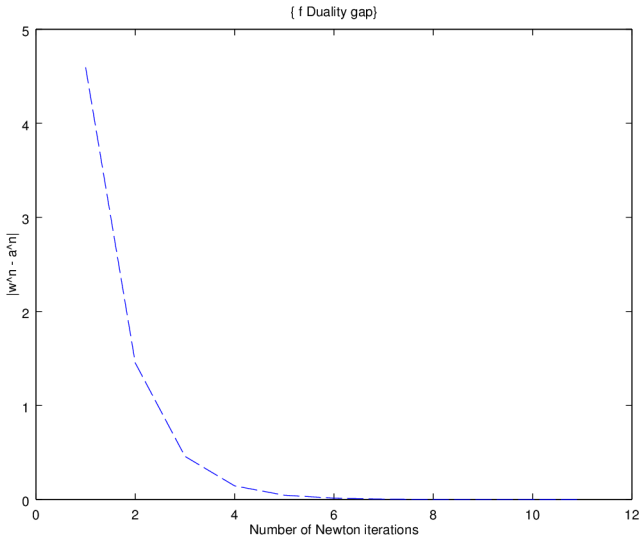


Figure 5: *Duality gap* : ensemble de données 1

5.3 Points centrés réduits générés à partir de deux fonctions gaussiennes

On génère les points selon la procédure $m = 0$ dans *generatedata.m* avec $sep = 10$. 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. Voir le fichier *test2.mat* dans le dossier *test*.

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

Table 3: Matrice de confusion pour l'ensemble de données 2

Réalité/Prédiction	Classe 1	Classe 2
Classe 1	23	0
Classe 2	0	27

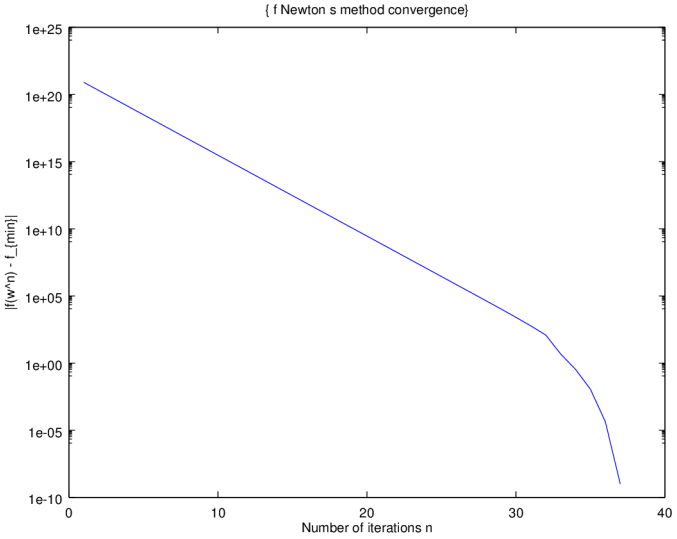


Figure 6: Ensemble de données 2 (échelle semi-log)

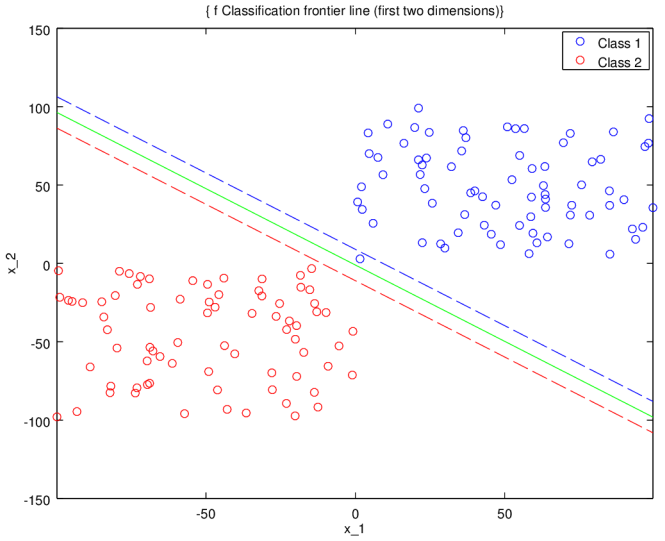


Figure 7: Ensemble de données 2 (Les lignes rouge et bleue en pointillés sont les droites f(x) = 10 et f(x) = -10)

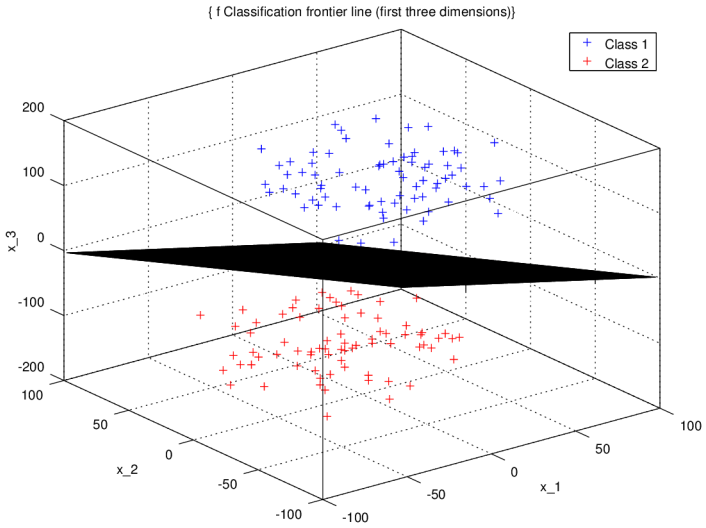


Figure 8: Ensemble de données 2

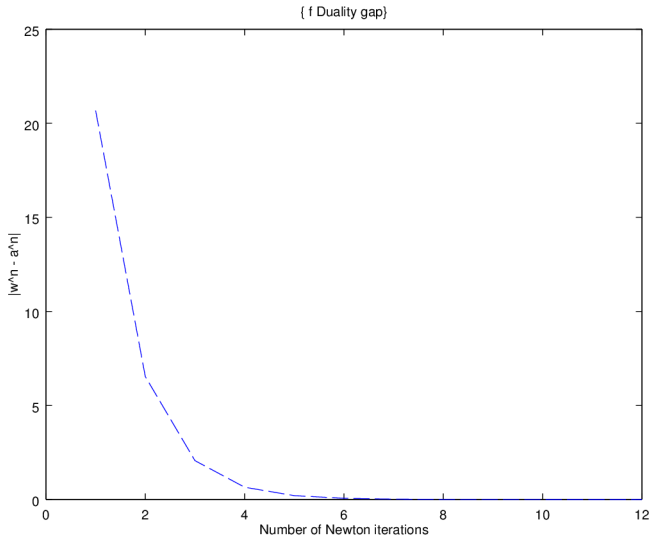


Figure 9: *Duality gap* : ensemble de données 2

5.4 Points centrés réduits générés avec des fonctions gaussiennes

On utilise la procédure $m = 0$ dans *generatedata.m* avec $sep = 100$. Voir le fichier *test3.mat* dans le dossier *test*.

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

Table 4: Matrice de confusion pour l'ensemble de données 3

Réalité/Prédiction	Classe 1	Classe 2
Classe 1	28	0
Classe 2	0	22

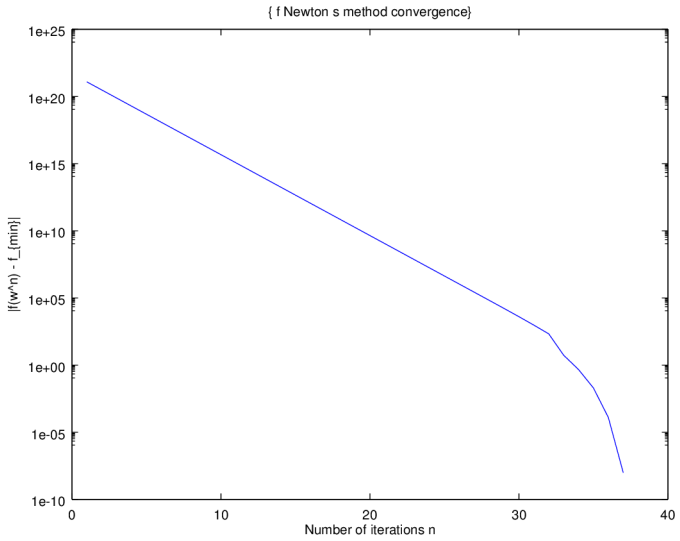


Figure 10: Ensemble de données 3 (échelle semi-log)

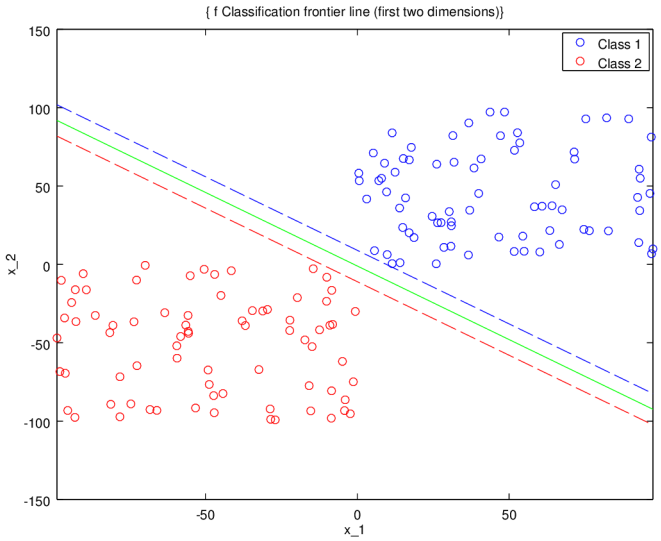


Figure 11: Ensemble de données 3 (Les lignes rouge et bleue en pointillés sont les droites f(x) = 10 et f(x) = -10)

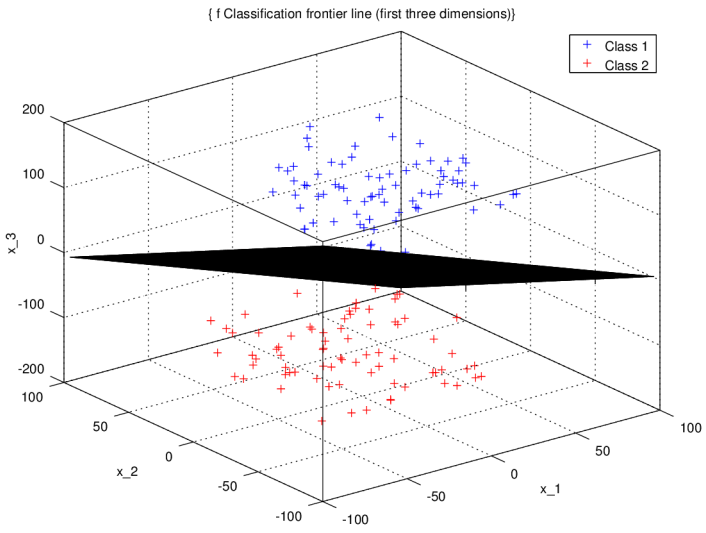


Figure 12: Ensemble de données 3

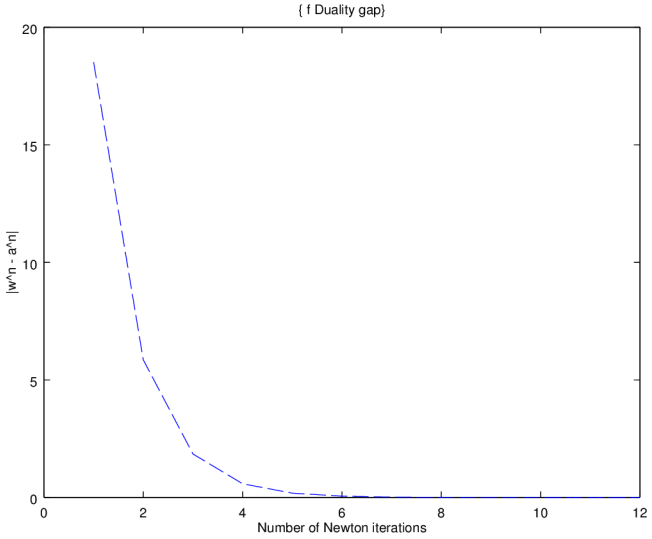


Figure 13: *Duality gap* : ensemble de données 3

5.5 Points centrés réduits générés avec des fonctions gaussiennes

On utilise la procédure $m = 0$ dans *generatedata.m* avec $sep = 0$. Voir le fichier *test4.mat* dans le dossier *test*.

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

Table 5: Matrice de confusion pour l'ensemble de données 4

Réalité/Prédiction	Classe 1	Classe 2
Classe 1	28	0
Classe 2	0	22

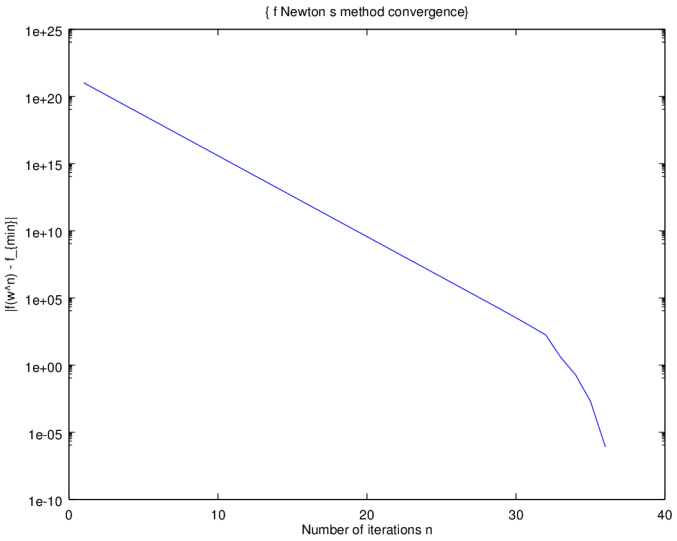


Figure 14: Ensemble de données 4 (échelle semi-log)

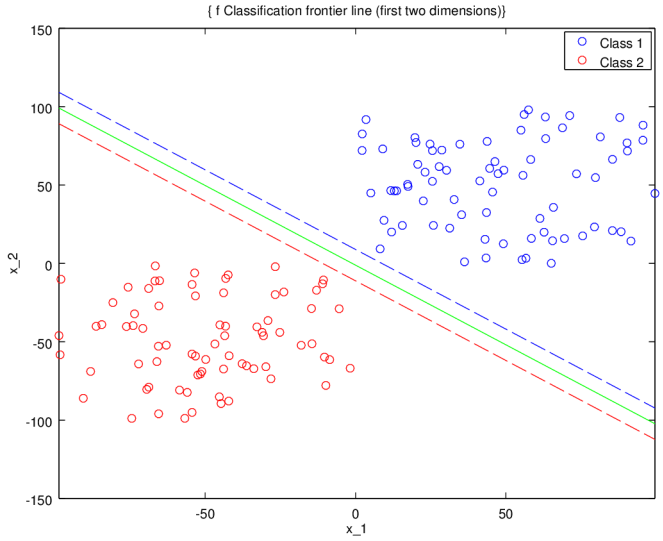


Figure 15: Ensemble de données 4 (Les lignes rouge et bleue en pointillés sont les droites f(x) = 10 et f(x) = -10)

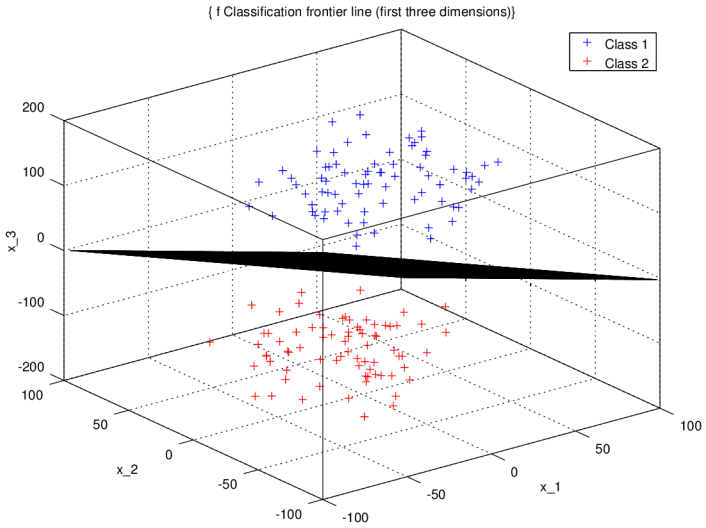


Figure 16: Ensemble de données 4

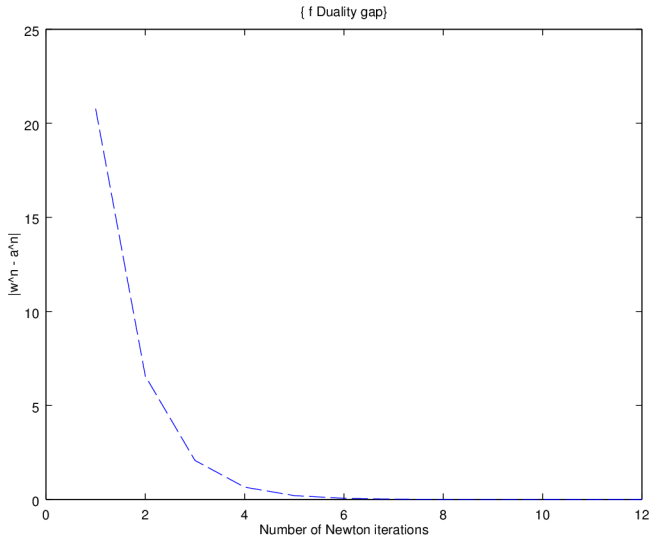


Figure 17: *Duality gap* : ensemble de données 4

5.6 Points générés avec des fonctions gaussiennes

On utilise la procédure $m = 1$ dans *generatedata.m* avec les paramètres par défaut. Voir le fichier *test5.mat* dans le dossier *test*.

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

Table 6: Matrice de confusion pour l'ensemble de données 5

Réalité/Prédiction	Classe 1	Classe 2
Classe 1	9	1
Classe 2	17	23

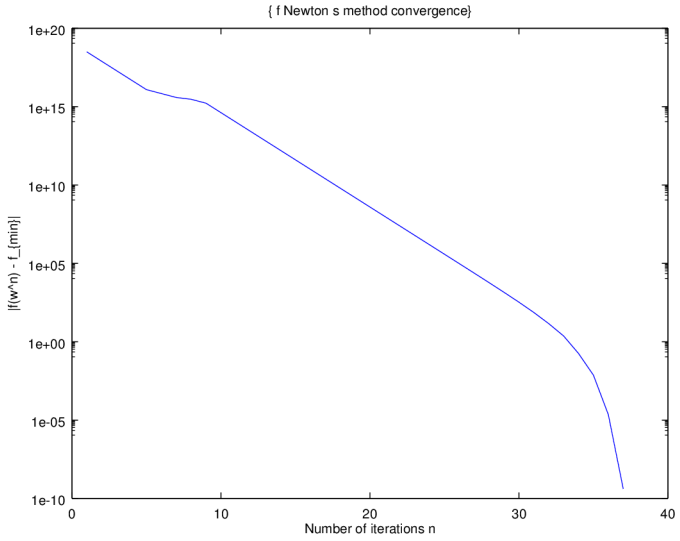


Figure 18: Ensemble de données 5 (échelle semi-log)

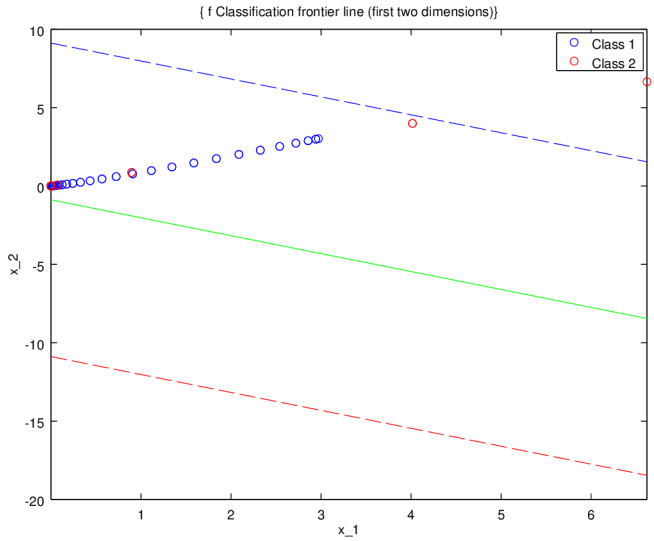


Figure 19: Ensemble de données 5 (Les lignes rouge et bleue en pointillés sont les droites f(x) = 10 et f(x) = -10)

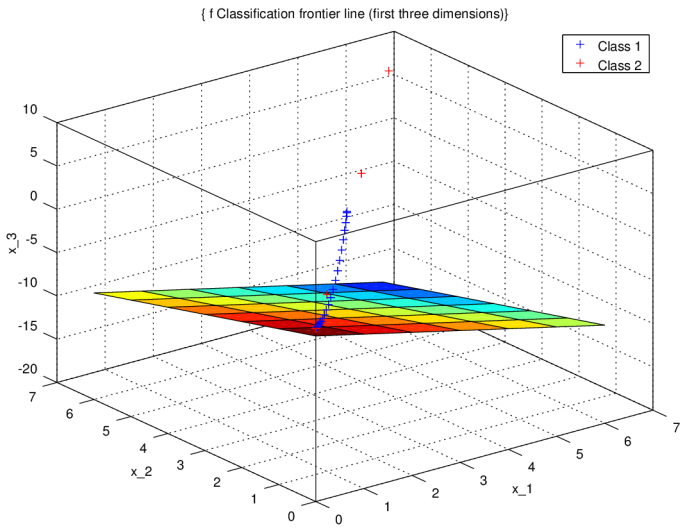


Figure 20: Ensemble de données 5

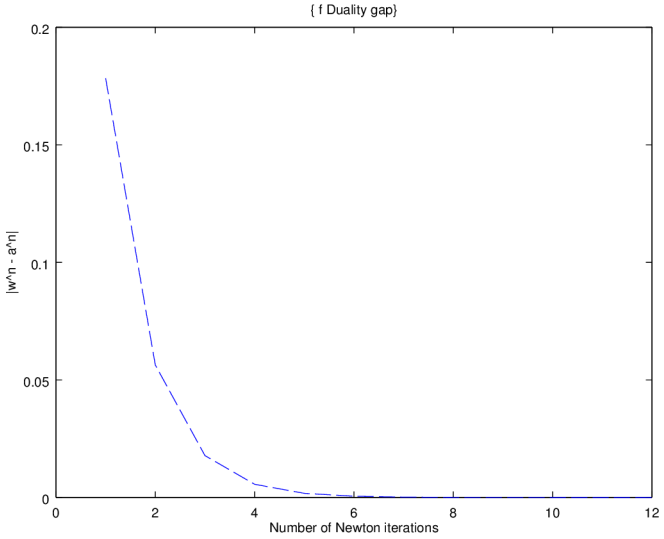


Figure 21: *Duality gap* : ensemble de données 5

6 Comparaison avec *la descente de coordonnées*

6.1 Résultats des ensembles de données précédents avec cette méthode

d est la dimension des points et n le nombre d'échantillons dans la génération. On utilise $\frac{2}{3}$ des points (choisis au hasard uniformément) de l'ensemble de départ pour l'ensemble d'entraînement, et les points restants pour l'ensemble de validation. On considère les mêmes ensembles que précédemment (l'ensemble 1 est celui utilisé dans les trois premiers tests de la section précédente).

Table 7: Comparaison entre les générations de points

Données	C	d	n	Temps (s)	Echec (%)
1	1	40	10	0,021	0
1	5	40	10	0,015	0
1	10	40	10	0,012	0
3	1	200	150	0,322	0
3	1	2000	150	0,396	0
3	1	20000	150	1,501	0
3	1	200	1500	99,407	0
5	1	200	150	0,392	54

On constate que l'algorithme est beaucoup plus rapide que le précédent, mais donne de mauvais résultats dans le cas où les points sont générés à l'aide de deux fonctions gaussiennes.

On constate qu'il y a encore la dépendance en la taille de l'échantillon, mais l'algorithme *Coordinate Descent* tourne toujours plus rapidement que le précédent (voir les tests 4, 5, 6 et 7).

6.2 Ensemble de données 1 ($C = 1$)

Table 8: Matrice de confusion pour l'ensemble de données 1

Réalité/Prédiction	Classe 1	Classe 2
Classe 1	2	0
Classe 2	0	1

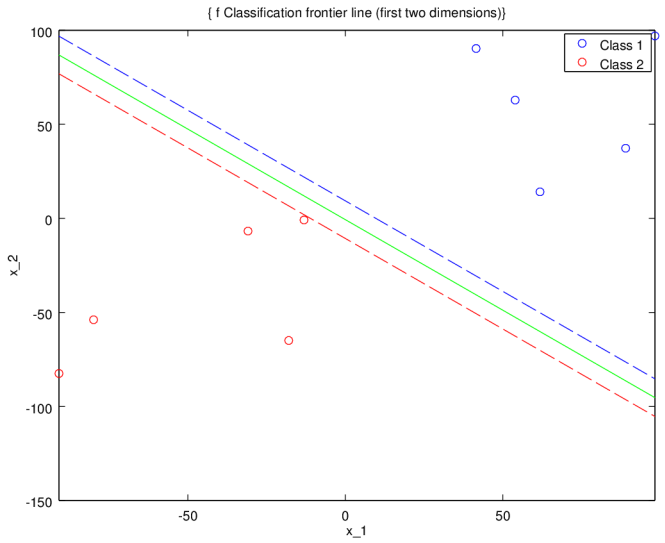


Figure 22: Ensemble de données 1 (Les lignes rouge et bleue en pointillés sont les droites f(x) = 10 et f(x) = -10)

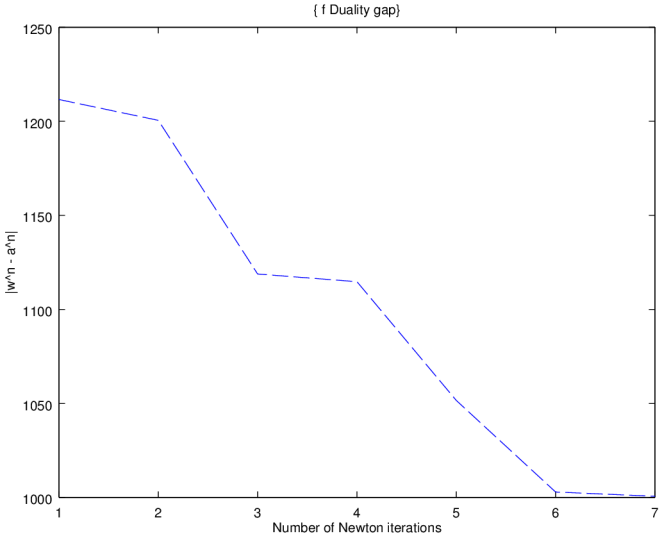


Figure 23: *Duality gap* : ensemble de données 1

6.3 Ensemble de données 3

Table 9: Matrice de confusion pour l'ensemble de données 3

Réalité/Prédiction	Classe 1	Classe 2
Classe 1	26	0
Classe 2	0	24

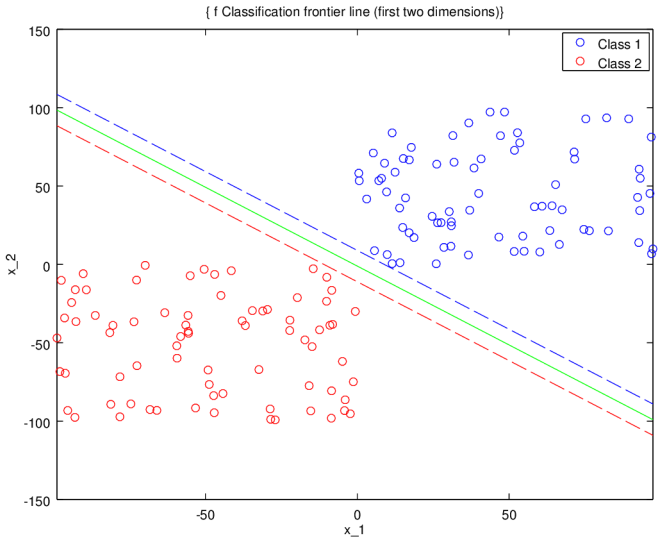


Figure 24: Ensemble de données 3 (Les lignes rouge et bleue en pointillés sont les droites f(x) = 10 et f(x) = -10)

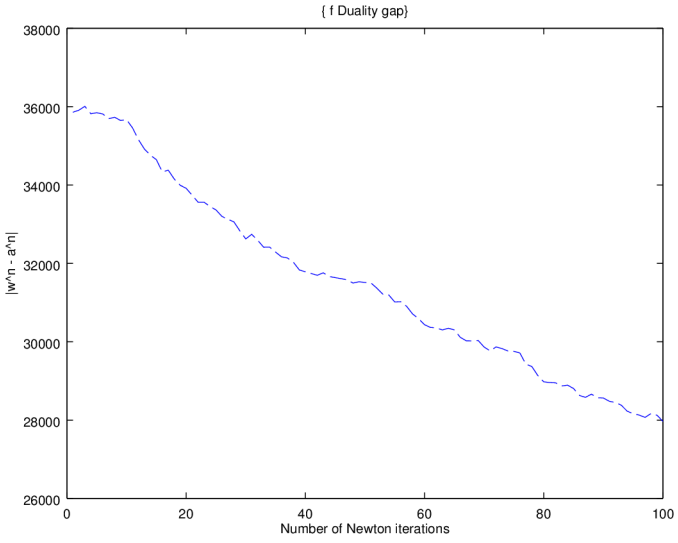


Figure 25: *Duality gap* : ensemble de données 3

6.4 Ensemble de données 5

Table 10: Matrice de confusion pour l'ensemble de données 5

Réalité/Prédiction	Classe 1	Classe 2
Classe 1	21	20
Classe 2	7	2

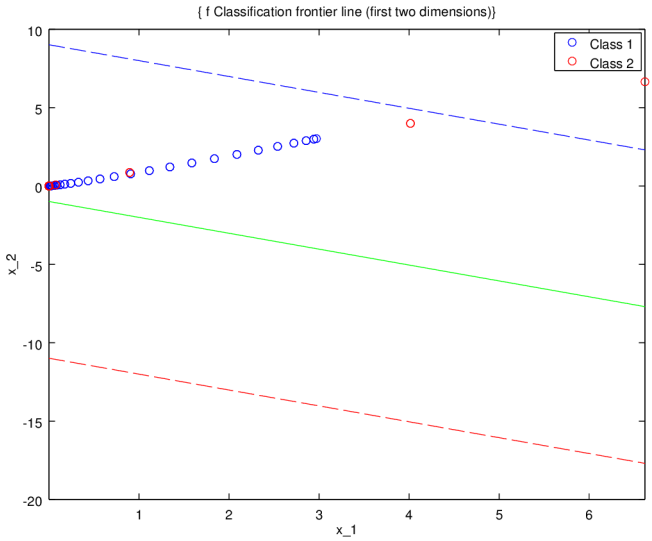


Figure 26: Ensemble de données 5 (Les lignes rouge et bleue en pointillés sont les droites f(x) = 10 et f(x) = -10)

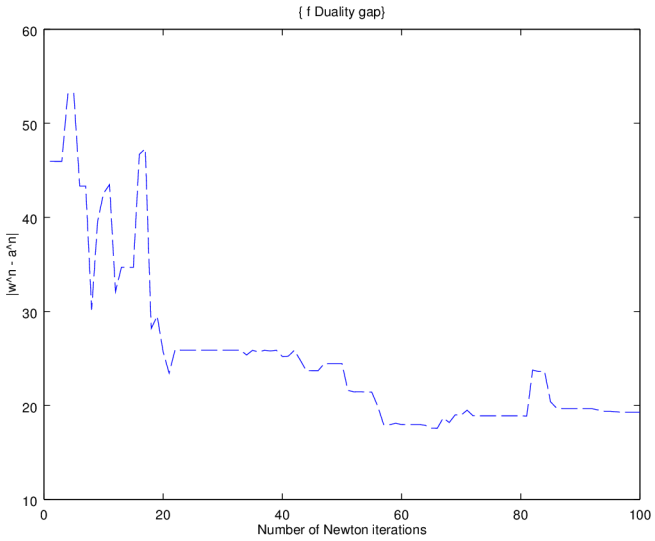


Figure 27: *Duality gap* : ensemble de données 5

7 Implémentation de l'algorithme ACCPM

Nous avons implémenté l'algorithme ACCPM (ou *Analytic center cutting-plane method*), qui recherche le centre analytique du polytope $P = \{x | Ax \leq b\}$ correspondant au problème primal (voir fichier *accpm.m*).

Déterminons les valeurs de A et b. En reprenant le problème primal, on a :

$$\begin{array}{c} \forall i, y_i(\omega^T x_i) \geq 1 - z_i \\ \Leftrightarrow \begin{bmatrix} -\mathrm{y}_1 x_1 & -1 & 0 & 0 & ...0 & 0 \\ -\mathrm{y}_2 x_2 & 0 & -1 & 0 & ...0 & 0 \\ ... & ... & ... & ... & ... & ... \\ -\mathrm{y}_n x_n & 0 & 0 & 0 & ...0 & -1 \end{bmatrix} \begin{bmatrix} \omega^T \\ \mathrm{z} \end{bmatrix} \leq -\mathbf{1} \\ \text{(par calcul)} \end{array}$$

On choisit alors : $A = [diag(y) \times x^T Id_n]$ et $b = -\mathbf{1}$ (voir *main.m*).

7.1 Résultats

Sur certaines données, on obtient de bons résultats (ici avec un ensemble de données de taille 8 et de dimension 2) :

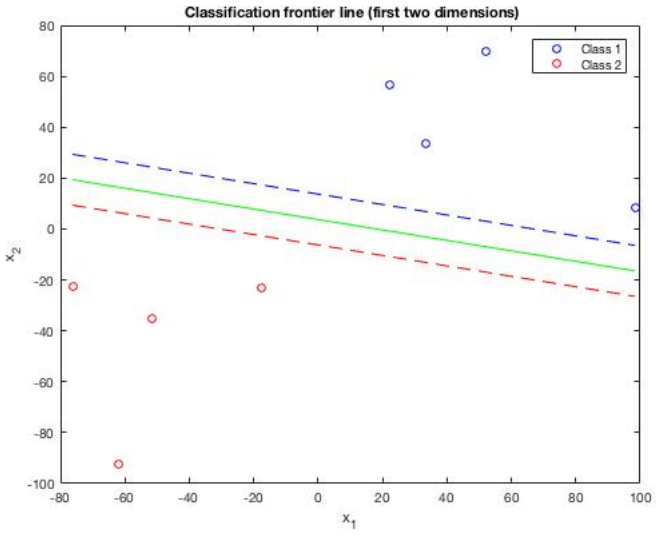


Figure 28: Tracé de la frontière de classification en utilisant l'algorithme AC-CPM

Mais souvent, on a un problème de précision (sur le calcul de la hessienne), qui est peut-être dû au choix initial de x strictement dans le polytope (donc tel que $Ax < b$). La convergence est donc souvent lente.

7.2 *Constraint dropping*

De plus, il semblerait que le *constraint dropping* implémenté fonctionne mal. Soit la mesure de pertinence suivante pour la contrainte i : $a_i^T x \leq b_i$, où a_i est la $i$ème ligne de A, et si on note H^* la hessienne de la fonction objectif :

$$(1)\ \eta_i = \frac{b_i - a_i^T x}{\sqrt{a_i^T \times H^{*-1} \times a_i}}$$
(voir *computeirr.m*)

Nous avons implémenté les stratégies suivantes :

- Pas de *dropping* (paramètre $drop = 2$);
- Supprimer les contraintes i, qui, si la mesure de pertinence (1) est notée η_i pour la contrainte i, sont telles que $\eta_i \geq m$, où m est le nombre de contraintes dans le polytope;

- Supprimer les contraintes dans l'ordre de non-pertinence, de sorte à n'en garder que $3n$, où n est la taille de l'ensemble de données.