TP

SVM linéaires et non linéaires

Introduction

Dans le cadre des méthodes de classifications, on peut utiliser – au lieu d'une fonction log-loss – la fonction de coût de hinge pour ainsi effectuer une méthode SVM (support vector machine). Cela consiste à ajuster les paramètres de l'hyperplans séparateurs pour le mettre à égal distance des vecteurs supports (vecteurs de chaque classe le plus proche de cet hyperplan).

L'avantage de cette méthode est de pouvoir générer, grâce à l'astuce des noyaux, des classifications non-linéaires. L'objectif de ce TP est de tester différents classifieurs de type SVM. On travaille avec un DataSet de 3 fleurs que l'on restreint selon les cas.

PARTIE 1 : Cas linéairement séparable, résolution du problème primal

La première formulation de SVM linéaire est la forme primale, il s'agit simplement du problème d'optimisation qui permet d'obtenir les hyperparamètres de l'hyperplan séparateur (linéaire). Ce problème est formulé à partir d'une fonction de coût qui a deux objectifs : maximiser les marges de l'hyperplan entre les classes et réduire les erreurs de classification. Ce problème d'optimisation est un problème d'optimisation sous contrainte qui s'écrit :

$$(\hat{w}_0, \hat{w}) = \underset{w_0, w}{\operatorname{argmin}} \frac{1}{2}\|w\|^2$$

$$\text{sous contrainte : } \forall m,\ y_m \left(w_0 + w^\top x_m\right) \geq 1$$

Pour utiliser ce problème d'optimisation dans Matlab, on utilise quadprog(H, f, A, b), pour cela on réécrit le problème sous la forme :

$$\min_x \frac{1}{2} x^\top H x + f^\top x$$

sous la contrainte :

$$Ax \leq b.$$

D'où :

$$y_m \left(w_0 + w^\top x_m\right) \geq 1 \implies y_m w_0 + y_m w^\top x_m \geq 1 \implies -y_m w_0 - y_m w^\top x_m \leq -1$$

Soit matriciellement en posant $x = [w\ w_0]^T$:

$$Ax \leq b$$

$$A = \begin{bmatrix} -y_1 & -y_1 x_1^\top \\ -y_2 & -y_2 x_2^\top \\ \vdots & \vdots \\ -y_N & -y_N x_N^\top \end{bmatrix}, \quad b = \begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix}$$

Par ailleurs, la fonction à minimiser peut se réécrire :

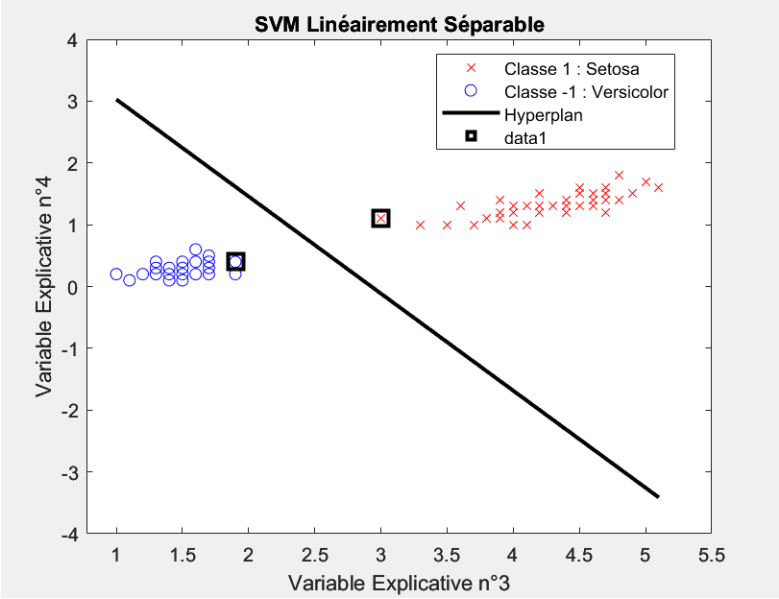
$$\frac{1}{2}w^\top w$$

Par identification,

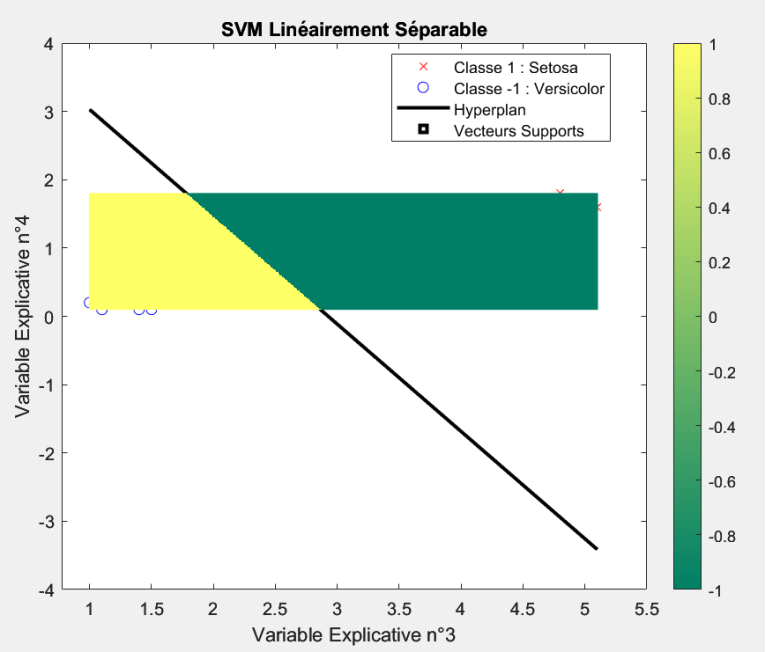
$$H = \begin{bmatrix} 0 & 0^\top \\ 0 & I_d \end{bmatrix}$$

Pour le vecteur f de quadprog, on a f = [0 0]$^\top$.

Quadprog nous permet donc d'obtenir tous les (w0,w) paramètres de l'hyperplan. Ce qui nous permet de repérer les vecteurs supports, tracer l'hyperplan, et attribuer à chaque point sa classe :



En ajoutant l'image binaire fourni, on obtient :



L'utilisation d'une SVM linéaire (forme primale) suffit dans ce cas, à obtenir un hyperplan séparateur efficace.

PARTIE 2 : cas linéairement séparable, résolution du problème dual

Un des problèmes de la forme primal est qu'il s'agit d'un problème d'optimisation sous contraintes explicites, ce qui peut être difficile à résoudre directement, surtout dans des espaces de grande dimension. Pour reformuler ce problème, on utilise des variables auxiliaires appelées multiplicateurs de Lagrange. Cette reformulation, appelée forme duale, conduit à un nouveau problème d'optimisation. Bien que la forme duale soit aussi un problème sous contraintes, elle présente plusieurs avantages : elle simplifie certains calculs, permet d'exploiter le kernel trick pour résoudre des problèmes non linéaires et réduit la charge de calcul. Cette forme s'écrit :

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmax}} \left\{ \sum_{m=1}^{M} \alpha_m - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j y_i y_j \left(x_i^\top x_j \right) \right\}$$

Sous les contraintes :

$$\sum_{m=1}^{M} \alpha_m y_m = 0$$

$$\alpha_m \geq 0, \quad \forall m \in \{1, \ldots, M\}.$$

Cette forme est équivalente à la forme primale sous condition d'optimalité de Karush-Kuhn-Trucker (admis). Les $alpha_m$ sont les multiplicateurs de Lagrange.

De même que précédemment, on adapte la formue duale à l'utilisation de quadprog(H, f, A, b, Aeq, beq, lb, ub, x0, options). Dans ce cas il faut récrire le problème sous cette forme :

$$\min_x \frac{1}{2} x^T H x + f^T x$$

Sous les contraintes :

$$Ax \leq b, \quad Aeqx = beq, \quad lb \leq x \leq ub.$$

Tout d'abord, l'inversion du signe à l'intérieur de argmax permet de se ramener à un problème de minimisation. En remarquant que la double somme peut être exprimée sous la forme d'un produit matriciel $\alpha^T H \alpha$, on peut alors déduire :

$$H_{ij} = y_i y_j (x_i^\top x_j), \quad H \in \mathbb{R}^{M \times M}$$

On a également :

$$f = -\mathbf{1}_M, \quad f \in \mathbb{R}^M.$$

$$A = -I_M, \quad b = \mathbf{0}_M.$$

En regardant les bornes sur alpha :

$$lb = \mathbf{0}_M, \quad ub = \infty.$$

Par utilisation de la première contrainte :

$$Aeq = [y_1, y_2, \ldots, y_M], \quad beq = 0.$$

Quadprog permet alors de résoudre le problème d'optimisation de la forme duale. On déduit w de α en se référant à l'annulation du gradient du lagrangien par rapport à w tel que :

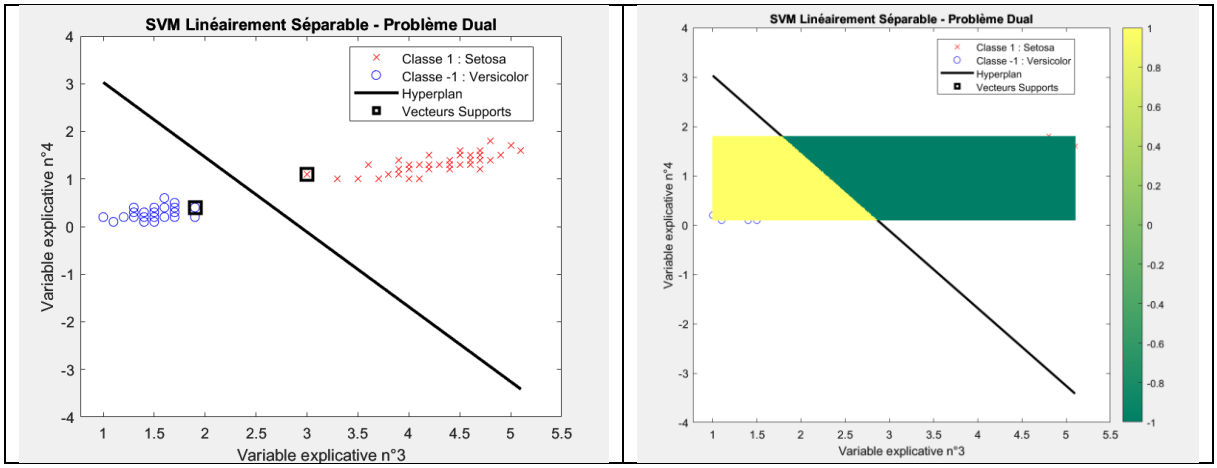
$$\mathbf{w} = \sum_{m=1}^{M} \alpha_m y_m \mathbf{x}_m$$

Pour w0, on peut utiliser la contrainte de la forme primale qui s'écrit sous forme d'égalité dans le cas d'un vecteur support*. Cela permet donc d'écrire :

$$w_0 = y_s - \mathbf{w}^\top \mathbf{x}_s$$

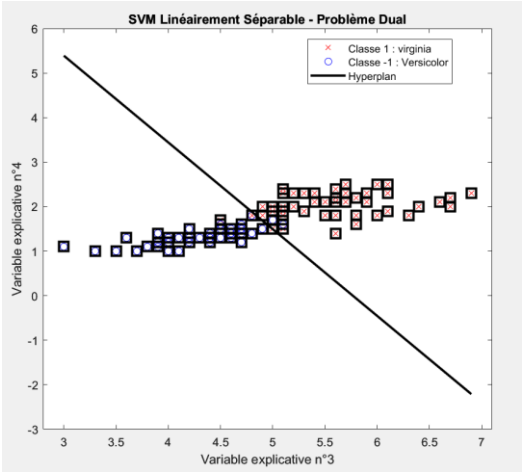
***Remarque :** La contrainte de la forme primale permet de dire que pour tout individu m, soit alpha_m = 0 soit xm est un vecteur support. C'est pour cela que si alpha_m > 0 alors on a un vecteur support. Il suffit donc de trouver une vecteur tel que alpha_m > 0 pour obtenir une contrainte d'égalité.

Ces étapes nous on donc permis de résoudre le problème dual et d'obtenir les paramètres de l'hyperplan séparateur. Ainsi :



PARTIE 3 : Cas non linéairement séparable, résolution du problème dual

On commence par reprendre le script précédent en modifiant le jeu de données importé. L'exécution du script avec suppression de Setosa fait retourner à la fonction quadprog « No feaseble solution found ». Cela signifie que le problème d'optimisation posé à quadprog ne possède pas de solution – ou plutôt – que les données ne sont pas séparables dans le cadre de l'hyperplan linéaire. Comme on peut le voir ça ne fonctionne plus du tout :



Les classes versicolor et virginica nesont pas strictement linéairement séparables. Il est donc nécessaire d'introduire une marge souple (avec un paramètre C) pour gérer ces données. Cette méthode sur le fait de supposer que les classes sont approximativement séparables : cela signifie que l'on autorise de légers dépassements de la marge optimale, en trouvent un compromis entre la maximisation de la marge et la minimisation des erreurs sur les données d'entrainement. Dans les cas où les classes sont totalement non linéaire, on a recours à l'astuce de noyau (section suivante).

D'après le cours un problème dual à marge souple s'écrit :

$$\hat{\alpha} = \arg\max_{\alpha} \left\{ \sum_{m=1}^{M} \alpha_m - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j y_i y_j (x_i^\top x_j) \right\}$$

$$\text{s.c.} \quad \sum_{m=1}^{M} \alpha_m y_m = 0, \quad 0 \leq \alpha_m \leq C, \forall m = 1, \ldots, M$$

Adapter cela à quadprog, c'est écrire le problème sous la forme :

$$\min_{\alpha} \frac{1}{2} \alpha^\top Q \alpha + f^\top \alpha$$

$$\text{s.c.} \quad A\alpha \leq b, \quad A_{eq}\alpha = b_{eq}, \quad \text{LB} \leq \alpha \leq \text{UB}$$

On a :

$$(Q)_{ij} = y_i y_j (x_i^\top x_j)$$

$$f = -\mathbf{1}_M$$

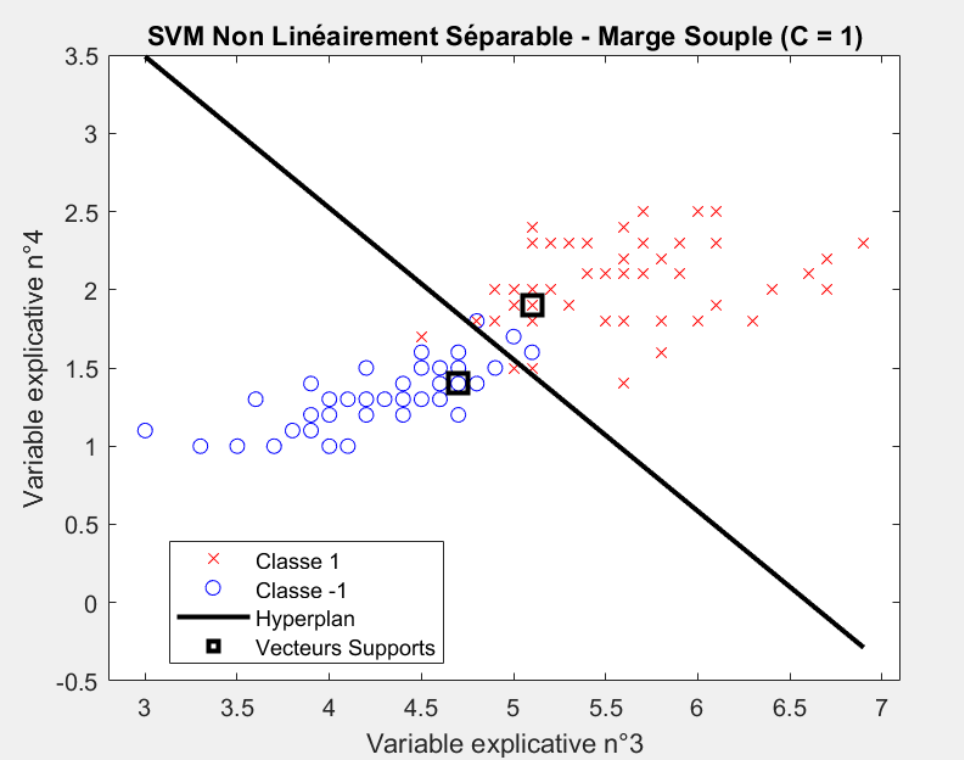
$$A_{eq} = y^\top, \quad b_{eq} = 0$$

(Première contrainte)

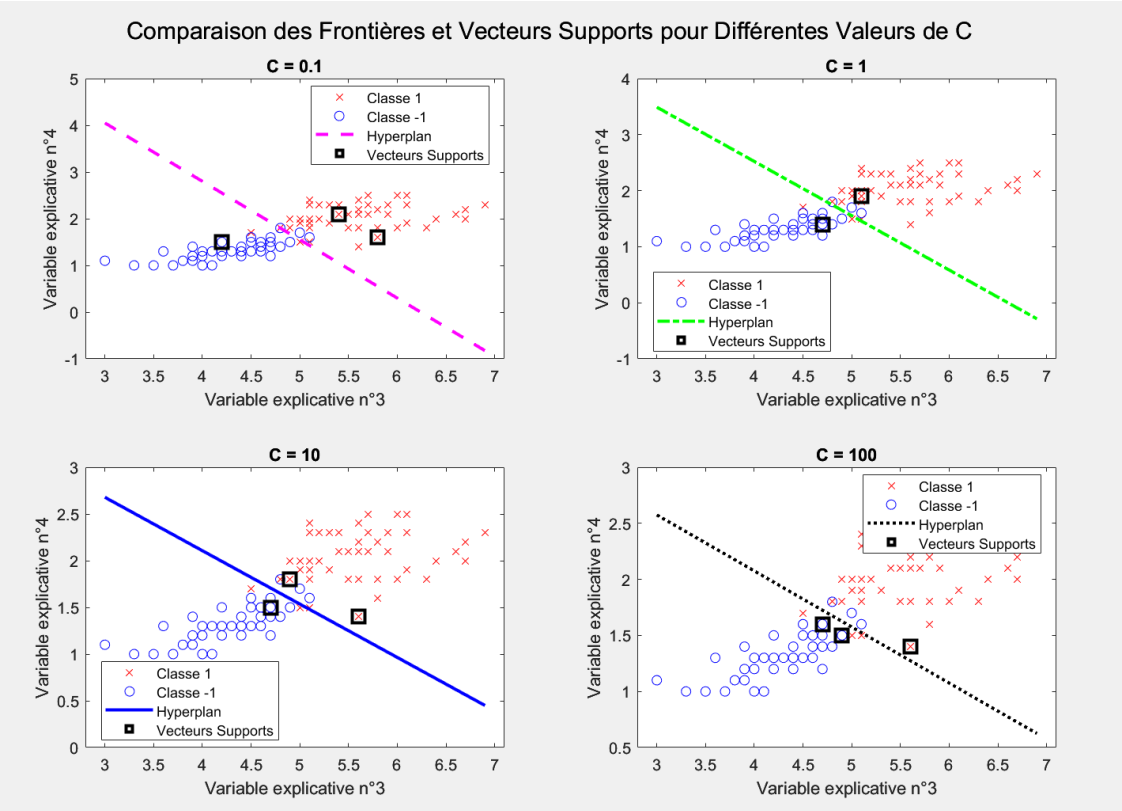
A et b ne sont pas nécessaires, car les bornes sont gérés par LB et UB, bornes de alpha tel que :

$$LB = 0, \quad UB = C \cdot \mathbf{1}_M$$

De même que dans le cas précédent, on applique quadprog, on récupère les paramètres de l'hyperplan et on obtient :



On observe que la frontière ne sépare par strictement toutes les données, ce à quoi on s'attendais puisque l'on s'est placé dans le cas d'une SVM à marge souple. Avec C=1, la SVM trouve le bon compromis entre maximiser la marge et minimiser les erreurs sur les données d'entrainement. Dans le cas où C est différent de 1 :



Pour C faible (0.1), la marge est large et la frontière plus lisse. Pour C grand (100), la marge est étroite mais une priorité est accordée à la précision des données d'entrainement.

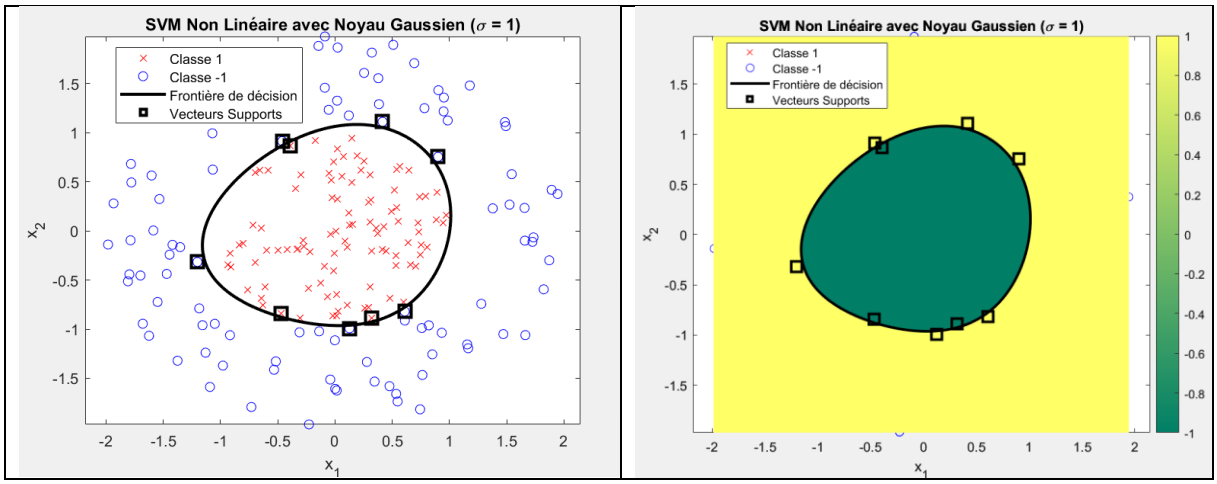
Note : Dans la suite, j'ai parfois pu directement interpréter les marges sans nécessairement mettre les vecteurs supports qui m'obligent à tracer des figures supplémentaires.

PARTIE 4 : Exemple de classifieur non linéaire

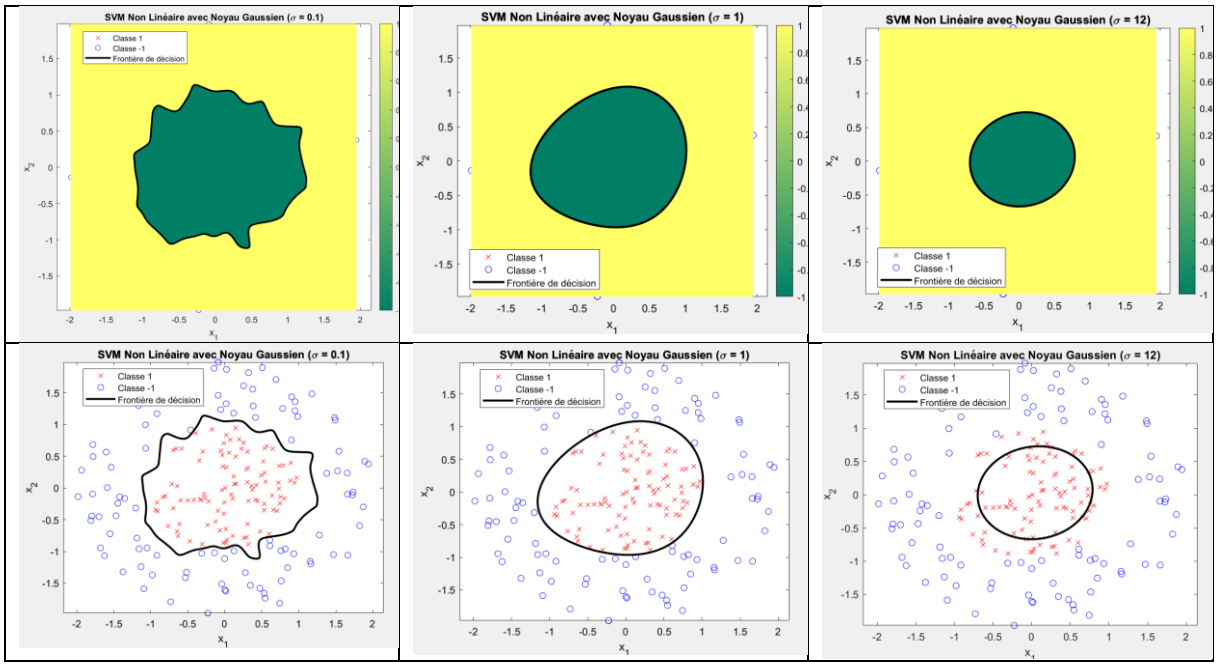
Lorsque l'on regarde la formulation du problème dual, on sait que l'on peut étendre celui-ci à l'aide de l'astuce du noyau, c'est-à-dire en remplacement le produit scalaire des x par un produit scalaire K qui est un noyau de Mercer. Ici on se conforme à l'exercice qui souhaite un noyau gaussien :

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

En appliquant la même logique que précédemment, on a :



Le paramètres sigma contrôle la « largeur » du noyau gaussien.



On remarque donc que des valeurs trop grandes ou trop petites peuvent entraîner un sur-surajustement (cas sigma faible) ou un sous ajustement (cas sigma grand) de la frontière de décision.

Dans le cas sigma faible, la frontière de décision devient trop complexe et s'adapte aux moindres variations des données (risque de faible marge). Pour sigma élevé, le modèle ignore les variations locales importantes.

Cela illustre donc le nécessité d'un compromis marge-erreur. Il faut une marge maximum pour une séparation large et stables et une erreur minimale pour classifier correctement.

Remarque : Pourquoi la marge est si importante ?

Il faut rappeler que le but des classifieurs est non seulement de classer correctement les données mais aussi de faire en sorte que les prédictions restent les correctes pour un jeu de données complètement différent.

Il faut en effet comprendre que l'entrainement de tout modèle va simplement capturer des dépendances entre variables explicatives - soit sous ou sûr représentés dans sa base de données d'entrainement - et les généraliser. Mais l'on est par sûr que « l'interdépendance » des variables soit la même la fois suivante. C'est pourquoi on utilise la mesure mathématique du risque de généralisation : plus le risque empirique s'en rapproche moins on est confronté à ce problème. Seulement quand on écrit les formules mathématiques, on s'aperçoit que s'approcher du risque de généralisation ne se fait qu'au prix d'une diminution de la précision. D'où un compromis nécessaire (compromis biais-variance), pour que le risque structurel soit minimum (on n'aura jamais une soustraction nulle entre le risque de généralisation et le risque empirique).

Pour les SVM, on peut justement établir une comparaison entre les risques et les marges en s'inspirant de la géométrie et de la théorie.

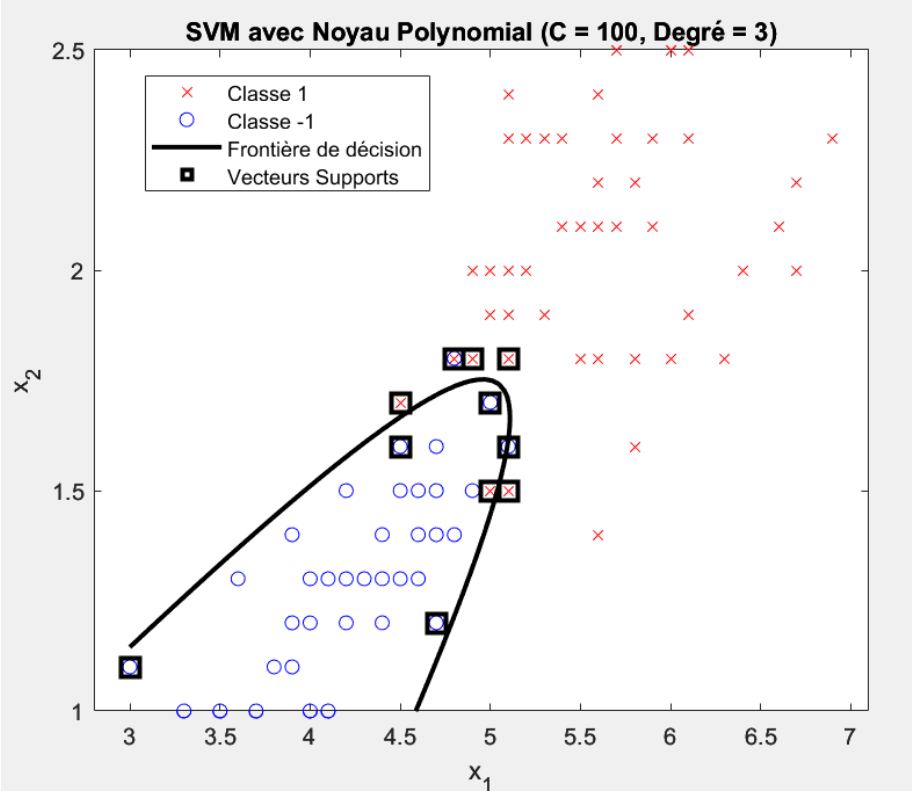
D'abord **géométriquement**, choisir une grande marge revient à maximiser la distance entre l'hyperplan et les points les plus proches, ce qui rend la classification robuste aux variations locales. Mais, cela peut entrainer des erreurs à l'entrainement car la frontière est moins serrée autour des points.

Ensuite, d'un point de vue **théorique**, on voit que l'on peut utiliser le risque associé au coût de hinge pour obtenir une SVM duale du cas approximativement séparable. Le coût de hinge pénalise les points qui ne respectent pas la marge. La minimisation équivaut donc à trouver un compromis entre le risque empirique et de généralisation.

Ainsi on sent bien que, maximiser la marge c'est « comme » minimiser le risque empirique et inversement, ce qui nous ramène au compromis biais-variance mais en résonnant avec les marges. D'où ce que j'ai dit plus haut en commentaire des figures.

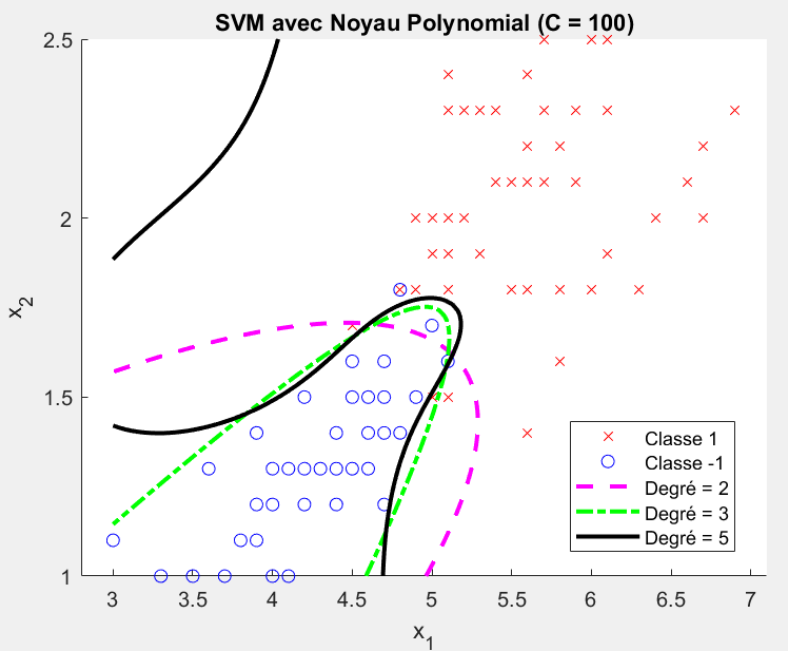
PARTIE 5 : Classifieur polynomial pour le problème des iris

On fait de même que pour la partie 4 en changeant la formule du noyau en noyau polynomial de degré 3. On a alors :



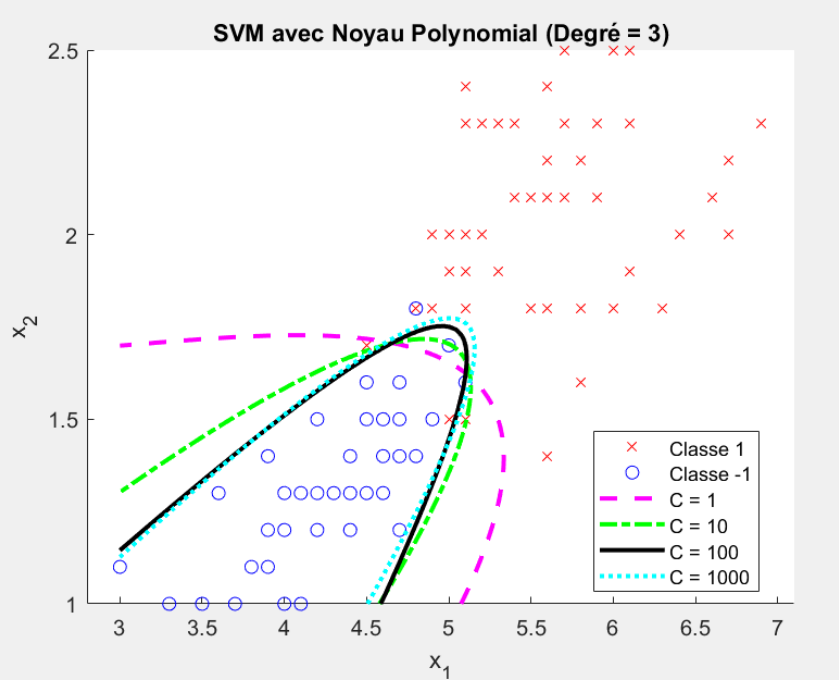
On fait désormais varier C et le degré d.

- Pour d qui varie :



Le degré polynomial permet d'ajuster la linéarité du modèle. Plus on augmente le degré d, plus le modèle est non linéaire. Cela a pour avantage de coller davantage aux données, mais il pourrait y avoir un risque de surajustement (et par extension une augmentation du risque de généralisation).

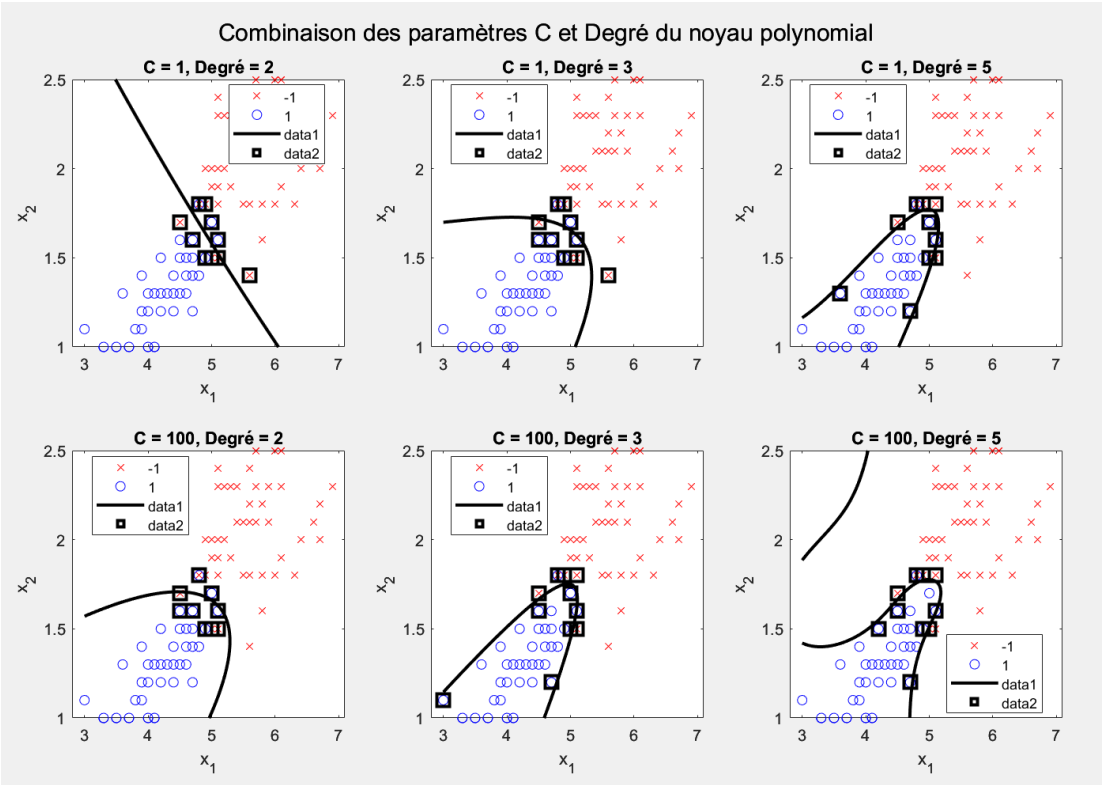
- Pour C qui varie à d fixé :



Pour rappel, C est le compromis entre marge large et erreurs sur les données d'entrainement. Plus C est élevé, plus on cherche à minimiser les erreurs sur les données d'entrainement.

Inversement, un C plus faible introduit une marge plus large (plus de tolérance à l'erreur comme on le voit pour C=1).

En combinant C et d :



Pour C (=1) petit et d (=2) faible : Marge large, frontière simple. Bonne généralisation, mais détails proches de la frontière pas pris en compte.

Pour C (=1) petit et d (=5) grand : Marge large, complexité du noyau élevée. La généralisation pourrait être limitée.

Pour C (=100) grand et d (=2) petit : Frontière précise mais simple. Les données proches de les frontières sont mieux classées mais certaines relations non linéaires ne sont peut être pas pris en compte.

Pour C (=100) grand et d (=5) grand : La frontière s'ajuste parfaitement en raison de sa complexité, caractéristique qui nuira au risque de généralisation.

Cette analyse nous permet donc de comprendre comment mieux ajuster les paramètres de manière optimal. Tout dépend donc du jeu de données auquel on a à faire pour le choix des paramètres C et d.