M2 MAS DS,
Aix-Marseille Université, 2024–2025.
A. Marmin.

TP 1: Optimisation sous contraintes

Séparateurs à vaste marge

1 Séparateur à vaste marge simple

Problème: Nous considérons un problème de classification à deux classes. Le jeu de données d'entraînement est composé de N entrées $(\mathbf{x}_n)_{n\in[\![1,N]\!]}$, qui sont des vecteurs de $\mathbb{R}^D$, et des N sorties correspondantes $(y_n)_{n\in[\![1,N]\!]}$, qui prennent leur valeur dans $\{-1,1\}$. Nous supposons que le jeu de données est linéairement séparable et nous considérons le modèle linéaire suivant

$$\psi(\mathbf{x}) = \langle \mathbf{w} \mid \mathbf{x} \rangle + b \tag{1}$$

où $\mathbf{w}$ est le vecteur de poids et b le biais. Nous cherchons donc une frontière de décision affine (i.e. sous forme d'une droite) pour séparer nos données.

Objectif: Trouver un choix de paramètres pour $\mathbf{w}$ et b tel que $\psi(\mathbf{x}_n)$ est strictement positif si y_n vaut 1 et $\psi(\mathbf{x}_n)$ est strictement négatif si y_n vaut -1, c'est-à-dire

$$(\forall n \in [\![1,N]\!]) \quad y_n \psi(\mathbf{x}_n) > 0\,.$$

Méthode: Il y a plusieurs façons de résoudre ce problème. Ici, nous nous intéressons aux **séparateurs à vaste marge (SVM)** dont le but est de définir une frontière de décision en **maximisant la marge**. Cette dernière est définie comme la plus petite distance entre les échantillons et la frontière de décision. Nous cherchons donc des valeurs de $\mathbf{w}$ et b telles que la marge soit maximale.

Nous illustrons le concept des SVM sur la Figure 1: les entrées $\mathbf{x}_n$ correspondant à une sortie $y_n = 1$ sont représentées par des cercles pleins alors que celles correspondant à une sortie $y_n = -1$ sont représentées par des cercles creux. La frontière de décision est représentée par une ligne pleine alors que les deux lignes en pointillé délimitent la marge. Notons que, par le choix du modèle donné par l'équation (1), le vecteur $\mathbf{w}$ est un vecteur normal à la frontière de décision. Les données en rouge sont les vecteurs qui définissent la marge et sont appelés les vecteurs de support. Ainsi, les séparateurs à vaste marge sont également appelés machine à vecteurs de support (Support Vector Machine en anglais).

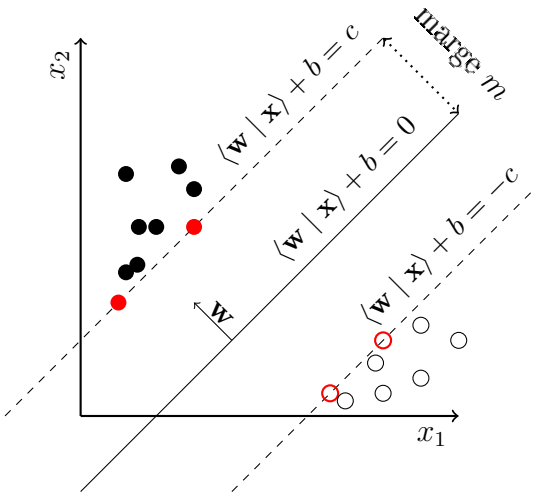

Figure 1: Illustration d'un SVM

Questions:

- Exprimer la marge m en fonction de $\mathbf{w}$.
  (Indication: il est facile d'exprimer $2m$ en utilisant deux vecteurs de support; un de chaque côté de la frontière de décision.)
- Exprimer la contrainte que doit vérifier chaque point $\mathbf{x}_n$ en fonction des données y_n.
- Exprimer la distance d'un point $\mathbf{x}_n$ à la frontière de décision.
  Montrer que cette distance est invariante par mise à l'échelle de $\mathbf{w}$ et b.
  Conclure que nous pouvons donc choisir que ψ prend les valeurs ± 1 pour les vecteurs de support.
- Formuler la recherche des paramètres $\mathbf{w}$ et b comme un problème d'optimisation.
- Transformer ce problème en la minimisation d'une fonction quadratique sous contraintes.
  La condition de Slater est-elle vérifiée pour ce problème?
- Ecrire le Lagrangien du problème.
- Appliquer le thèorème de Karush-Kuhn-Tucker et donner une expression de $\mathbf{w}$.
- A l'aide de la question précédente, écrire le problème dual comme un problème d'optimisation quadratique sous contraintes linéaires.
- Une fois $\mathbf{w}$ et b calculés, nous pouvons classer une nouvelle donnée $\mathbf{x}$ en calculant simplement $\psi(\mathbf{x})$ puis en regardant son signe.
  A l'aide des conditions KKT, montrer que parmi les vecteurs de données $(\mathbf{x}_n)_{n \in [\![1,N]\!]}$, seuls les vecteurs de support interviennent dans le calcul de $\psi(\mathbf{x})$.

  Remarque 1. *Nous voyons ici l'intérêt de résoudre le problème dual plutôt que le primal: à l'aide des variables duales, il est possible de classer une nouvelle donnée* $\mathbf{x}$ *en calculant seulment le produit scalaire entre cette donnée et les vecteurs de support. A l'inverse, si nous résolvons seulement le problème primal, il nous faut calculer explicitement le produit scalaire* $\langle \mathbf{w} \mid \mathbf{x} \rangle$*, ce qui peut-être coûteux si la dimension des données* D *est grande.*

Remarque 2. *Si le jeu de données n'est pas linéairement séparable, on peut le transformer dans un domaine où il le devient. Nous utilisons alors une fonction* $\phi : \mathbb{R}^D \to \mathbb{R}^M$ *(feature map) où les vecteurs* $(\phi(\mathbf{x}_n))_{n\in[\![1,N]\!]}$ *sont appelés les vecteurs de caractéristiques (feature vectors). Nous pouvons alors faire le même raisonnement que précédemment et obtenir un problème dual quadratique qui ne dépend plus de* $\langle \mathbf{x}_{n_1} \mid \mathbf{x}_{n_2} \rangle$ *mais de* $\langle \phi(\mathbf{x}_{n_1}) \mid \phi(\mathbf{x}_{n_2}) \rangle$*. Nous définissons alors la fonction noyau* k *tel que*

$$(\forall (\mathbf{x}, \mathbf{y}) \in \mathbb{R}^D \times \mathbb{R}^D) \quad k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}) \mid \phi(\mathbf{y}) \rangle \ .$$

Il n'y a alors pas besoin d'exprimer explicitement ϕ*, seul l'expression de* k *est nécessaire: c'est l'astuce du noyau. Parmi les noyaux couramment utilisés, nous pouvons citer les noyaux gaussiens* $k(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{\|\mathbf{x}-\mathbf{y}\|}{\sigma}\right)$ *et les noyaux polynomiaux* $k(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x} \mid \mathbf{y} \rangle + K)^\alpha$ *où* K *est une constante et* α *est le degré du polynôme. Le théorème de Mercer permet de caractériser les fonctions noyaux: ce sont les fonctions* k *qui sont continues, symétriques et semi-définie positives.*

A noter que l'astuce du noyau n'est applicable que si nous passons par le problème dual sinon il nous exprimer explicitement la fonction ϕ*. Nous verrons comment utiliser l'astuce du noyau en pratique dans l'application de la Section 3.*

2 Superposition de classes et marge souple

Nous avons vu dans la partie précédente que les SVM peuvent produire une séparation stricte des données en deux classes distinctes à condition que ces dernières soient linéairement séparables dans un espèce de représentation. Toutefois, en pratique, les données de deux classes différentes peuvent être présentes dans une même région et une séparation exacte cause un appauvrissement du pouvoir de généralisation du classificateur. Nous allons donc maintenant relâcher le modèle précédent afin d'autoriser certains points à être classifiés dans la mauvaise classe.

Pour se faire, nous introduisons une pénalité sur les point mal classifiés. Dans le modèle précédent, nous n'autorisions pas les points à être mal classifieé; implicitement nous appliquions une erreur infinie à un tel point. Nous relâchons cette contrainte en donnant désormais une erreur finie aux points mal classifiés. Cette erreur sera proportionnelle à la distance à la frontière de décision. Nous introduisons donc, pour chaque donnée $\mathbf{x}_n$, une erreur e_n qui vaut 0 pour les points qui sont du bon côté de la marge mais qui vaut $|y_n - \psi(\mathbf{x}_n)|$ pour les autres. Ainsi, quatre cas sont possibles:

1. $e_n = 0$: $\mathbf{x}_n$ est bien classifié et est sur la marge ou du bon côté de celle-ci.
2. $e_n \in]0, 1]$: $\mathbf{x}_n$ est bien classifié mais se situe entre la marge et la frontière de décision.
3. $e_n = 1$: $\mathbf{x}_n$ est sur la frontière de décision.
4. $e_n > 1$: $\mathbf{x}_n$ esl mal classifié.

On remplace donc les contraintes $y_n(\langle \mathbf{w} \mid \mathbf{x}_n \rangle + b) \geq 1$ par la contrainte $y_n(\langle \mathbf{w} \mid \mathbf{x}_n \rangle + b) \geq 1 - e_n$ et nous ajoutons à la fonction objectif le terme $C \sum_{n=1}^{N} e_n$, avec $C > 0$ un coefficient qui équilibre le terme d'erreur avec le terme de marge.

Remarque 3. *En faisant tendre* C *vers l'infini, nous retrouvons le classificateur précédent.*

Question:

- Ecrire le problème d'optimisation associé à la relaxation décrite ci-dessus.
- En divisant la fonction objectif par C, faire le lien avec un problème de régression logistique régularisé. Quelle forme a la fonction d'erreur? Quelle forme a la régularisation? Tracer la fonction d'erreur obtenue et la fonction d'erreur de la régression logistique et comparer les.
- Exprimer le Lagrangien du problème et les conditions KKT et écrire le problème dual. (Ne pas oublier de vérifier la condition de Slater).
- Discuter de la nature du point $\mathbf{x}_n$ en fonction de la valeur des variables duales associées.

3 Applications numériques

3.1 Jeu de données "moons"

Nous allons commencer par tester les SVM implémentés dans Scikit-learn sur un jeu de données simple qui n'est pas linéairement séparable.

- Charger le jeu de données "moons" de Scikit-Learn à l'aide de la fonction `make_moons()` du module `sklearn.datasets`.
- Afficher le jeu de données en utilisant la fonction `scatter()` de `matplotlib`.
- Lire la documentation des classes `LinearSVC` et `SVC` du module `sklearn.svm` (en particulier, les méthodes `fit()`, `predict()` et `decision_function()`).
- Utiliser ces classes pour appliquer un SVM avec un noyau linéaire, un noyau gaussien, et un noyau polynomial au jeu de données.
- Afficher le résultat de chaque SVM et commenter.
  Aide: En cas de difficulté pour l'affichage des zones de décision, un exemple de fonction d'affichage est disponible dans le fichier *myplot.py*.
- Jouer avec les paramètres des noyaux pour observer leurs effets.
- Ajouter un point dans le jeu de données pour superposer les deux classes et faire varier le paramètre C.

3.2 Jeu de données "iris"

Même exercice que pour le jeu de donnée précédent en utilisant le jeu de données iris disponible dans le module `sklearn.datasets`. Ce dernier peut être chargé avec la fonction `load_iris()`. A noter que ce jeu de données contient trois classes et les données d'entrées sont de dimension 4. Pour des raisons de visualisation, nous pouvons nous limiter au choix de deux ou trois de ces dimensions en écartant les autres.

4 Implémentation d'un SVM (exercice optionnel)

Réimplémenter un SVM à partir de l'étude théorique fait ci-dessus et comparer le à l'implémentation de SVM sur un des jeux de données précédents.

Aide: La résolution numérique d'un problème d'optimisation quadratique sous contraintes linéaires peut être réalisée efficacement avec un algorithme de points intérieurs. Plusieurs implémentations sont disponibles en Python, notamment celle de la bibliothèque CVXOPT. Une étude des algorithmes de points intérieurs ne fait pas partie du spectre de ce cours mais les plus curieux pourront se pencher sur le dernier chapitre du livre "Convex Optimization" de Stephen Boyd et Lieven Vandenberghe disponible gratuitement sur la page web du premier auteur.