Master SID
Année universitaire 2021-2022

TP4 Apprentissage Automatique 2

Support Vector Machine

simon.bernard@univ-rouen.fr

1 Problème jouet

Pour cet exercice, vous allez reprendre le problème synthétique en 2D que vous avez utilisé au TP2 : $(\mathbf{x}, y_i) \in \mathbb{R}^2 \times \{\lambda_1, \lambda_2\}$, avec les distributions des classes suivant une loi normal ($\lambda_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ et $\lambda_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$).

- ▷ Générez cette base de données synthétiques telle que :
  - chaque classe contient 300 points
  - $\lambda_1 \sim \mathcal{N}((1,1), 0.3)$ et $\lambda_2 \sim \mathcal{N}((-1,-1), 0.3)$
- ▷ Etiquetez les points pour pouvoir entrainer un SVM par la suite
- ▷ Affichez le nuage de points.

On souhaite vérifier les propriétés des SVMs à partir de la solution du problème d'optimisation obtenue sur ces données synthétiques :

- ▷ Entrainez un SVM linéaire avec l'implémentation `SVC` de *Scikit-learn*. Proposez pour cela une paramétrisation pertinente pour vous assurer que
  - le modèle est bien linéaire
  - qu'il n'y a pas de données d'apprentissage mal classées

  Justifiez brièvement cette paramétrisation.
- ▷ Du modèle appris, récupérez :
  - les indices des vecteurs supports
  - le vecteur $\boldsymbol{\alpha}$ solution du problème dual
  - le biais b
- ▷ À partir de ces valeurs, calculez le vecteur $\mathbf{w}$
- ▷ Verifier que les points supports sont telles que $h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b = \pm 1$
- ▷ En appliquant la frontière de décision sur les données d'une grille 2D, tracer la frontière de décision $h(\mathbf{x}) = 0$, ainsi que les frontières de marge $h(\mathbf{x}) = \pm 1$
- ▷ Sur la même figure, faites ressortir les points supports
- ▷ Que devient cette figure lorsque la variance des classes Gaussiennes devient plus grande ? Analysez et commentez ce qu'il se passe.

2 Reconnaissance de chiffres manuscrits

On souhaite mettre en œuvre un SVM pour la classification de chiffres manuscrits. Tout d'abord :

- ▷ Chargez la base *mnist_784* du site *Open ML* [1] en utilisant la fonction `fetch_openml` de *scikit-learn* [2]. Prenez le temps de bien lire la description de la base pour comprendre de quoi il s'agit.
- ▷ Séparez les 60000 premières données pour l'apprentissage des 10000 dernières pour le test
- ▷ Isolez les classes correspondant aux chiffres '1' (qui sera la classe positive) et aux chiffres '8' (qui sera la classe négative)
- ▷ Sur un même graphique, affichez 10 imagettes de la base d'apprentissage, 5 de la classe positive et 5 de la classe négative. Utilisez pour cela la fonction `imshow` de *PyPlot* [3].

1. `https://www.openml.org/d/554`
2. `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_openml.html`
3. `https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.imshow.html`

Maintenant, pour apprendre un classifieur linéaire SVM sur ce problème de classification binaire :

- ▷ Utilisez une procédure de validation croisée sur l'ensemble d'apprentissage pour déterminer la meilleure valeur du paramètre C à utiliser pour cette base de données.
- ▷ Une fois la valeur de C retenue, relancez l'apprentissage d'un SVM sur la base d'apprentissage complète.
- ▷ Testez le classifieur obtenu :
  - ○ Calculez et affichez le taux de bonne classification sur la base de test
  - ○ Affichez 25 exemples de données mal classées
  - ○ Calculez et affichez la matrice de confusion

  Commentez les résultats obtenus
- ▷ Relancez toute la procédure avec une autre paire de chiffres.

Pour finir, vous pouvez tester l'apprentissage d'un SVM pour la classification multiclasse sur le même problème :

- ▷ Pour limiter les temps de calcul, réduisez le problème à la classification de 3 chiffres seulement, par exemple, 0, 2 et 8
- ▷ Proposez un protocole expérimental pour évaluer les performances d'un SVM linéaire sur ce problème, avec l'implémentation de *Scikit-learn*.
- ▷ Quelle stratégie multiclasse est utilisée par cette implémentation ?