Introduction à l'apprentissage machine (GIF-4101 / GIF-7005)
Département de génie électrique et de génie informatique
Automne 2018

UNIVERSITÉ LAVAL

Devoir 2

Instructions :

- Formation des équipes :
  - GIF-4101 : le devoir est réalisé en équipe de deux à trois étudiants
  - GIF-7005 : le devoir est réalisé individuellement
- Programmation :
  - Utilisez Python et scikit-learn autant que possible
  - Produisez vos solutions dans le fichiers fournis, en respectant les instructions
  - Les temps de calcul sont vérifiés dans le code. Toute mention d'une durée trop importante entraînera *la note de zéro (0)* pour la sous-question correspondante
- Remise :
  - La remise du rapport et du code source se fait dans monPortail
  - Le remise doit être effectuée au plus tard le mercredi 17 octobre à 9h30
- Pondération :
  - Ce devoir compte pour 5% de la note finale

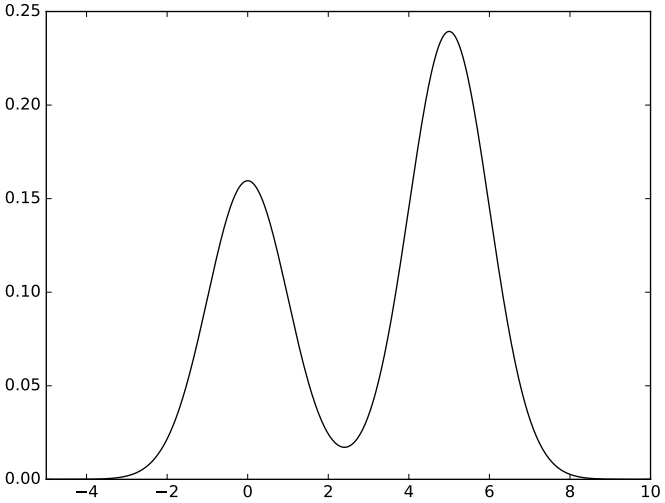
---

1. **Estimation de densité par noyau (5pt)**

   Soit une densité-mélange combinant deux lois normales en une dimension,

   $$p(x) = \sum_i P(\mathcal{G}_i)\, p(x|\mathcal{G}_i) = \pi_1 \mathcal{N}(\mu_1,\sigma_1^2) + \pi_2 \mathcal{N}(\mu_2,\sigma_2^2),$$

   où $\pi_1 = 0{,}4$, $\mu_1 = 0$, $\sigma_1^2 = 1$, $\pi_2 = 0{,}6$, $\mu_2 = 5$ et $\sigma_2^2 = 1$. Le fichier `d2q1.py` [1] contient une fonction nommée `pdf` qui vous permet d'obtenir la densité de probabilité correspondante.
   La figure suivante illustre cette densité-mélange dans le domaine $[-5, 10]$.



1. `http://vision.gel.ulaval.ca/~cgagne/enseignement/apprentissage/A2018/travaux/d2q1.py`

Faites les manipulations suivantes avec scikit-learn en utilisant le fichier `d2q1.py`, joignez les résultats et figures obtenus à votre rapport et fournissez le fichier source modifié de votre solution.

(a) Écrivez une fonction permettant d'échantillonner des données de cette densité. La fonction doit avoir l'entête `sample(n)`, où `n` correspond au nombre de données à échantillonner. Présentez deux figures, montrant respectivement 50 et 10 000 données échantillonnées de cette distribution, sous la forme d'histogrammes de 25 bins dans le domaine $[-5, 10]$. Comparez également la forme des histogrammes obtenus à la vraie fonction de densité.

(b) Montrez le résultat d'une estimation avec noyau *boxcar* [2] (aussi connu sous le nom de *tophat*) appliqué respectivement à 50 et 10 000 données, pour les tailles de noyau (`bandwidth`) de $\{0{,}3, 1, 2, 5\}$. Présentez vos résultats en deux figures, une pour les résultats sur 50 données et une pour les résultats sur 10 000 données. Utilisez une couleur de courbe différente pour chaque taille de noyau et ajoutez une légende à votre figure. Discutez les résultats obtenus (taille du noyau relativement au nombre d'échantillons).

(c) Expliquez les principales différences et similarités entre une estimation par noyau et une approche basée sur les k-PPV. Quel lien peut-on faire entre eux ?

(d) Expliquez pourquoi les k-PPV sont intéressants pour le classement et la régression, mais peu adaptés à l'estimation de densité ?

2. **Discriminants linéaires et k-plus proches voisins pour la classification (15pt)**

Soit un discriminant linéaire, avec lequel le classement est effectué selon :

$$\mathrm{h}(\mathbf{x}|\mathbf{w},w_0) = \mathbf{w}^\top\mathbf{x} + w_0, \quad \mathbf{x}^t \in \begin{cases} C_1 & \mathrm{h}(\mathbf{x}|\mathbf{w},w_0) \geq 0 \\ C_2 & \text{autrement} \end{cases} .$$

On effectue un entraînement avec une descente du gradient basée sur le critère d'erreur suivant :

$$E(\mathbf{w},w_0|\mathcal{X}) = \frac{1}{2}\sum_{\mathbf{x}^t\in\mathcal{Y}} \frac{[r^t - \mathrm{h}(\mathbf{x}^t|\mathbf{w},w_0)]^2}{\|\mathbf{x}^t\|^2},$$

où $r^t \in \{-1{,}1\}$ et $\mathcal{Y}$ est l'ensemble des données de $\mathcal{X}$ mal classées,

$$\mathcal{Y} = \{\mathbf{x}^t \in \mathcal{X} \mid r^t\mathrm{h}(\mathbf{x}^t|\mathbf{w},w_0) \leq 0\}.$$

Si l'ensemble $\mathcal{Y}$ est vide, alors $E(\mathbf{w},w_0|\mathcal{X}) = 0$.

Effectuez les opérations suivantes :

(a) Donnez le développement mathématique complet des équations permettant d'effectuer la mise à jour des poids $\mathbf{w}$ et w_0 par descente du gradient, selon le critère d'erreur proposé.

(b) Implémentez une classe Python correspondant à ce discriminant linéaire, en programmant au minimum les fonctions `fit`, `predict` et `score` de l'interface de scikit-learn. Utilisez à cette fin le fichier `d2q2.py` [3]. Remettez le code source résultant de votre implémentation avec la remise de votre devoir sur monPortail.

2. Une estimation de densité avec un noyau *boxcar* correspond à ce qui a été présenté dans le cours comme étant une estimation naïve d'histogramme.

3. `http://vision.gel.ulaval.ca/~cgagne/enseignement/apprentissage/A2018/travaux/d2q2.py`

(c) Testez la performance de ce discriminant linéaire sur deux jeux de données synthétiques, produits selon deux configurations avec la fonction `datasets.make_classification` :
— Données synthétiques comprenant 100 instances selon deux classes et en deux dimensions, avec un cluster par classe [4] ;
— Données synthétiques comprenant 100 instances selon trois classes et en deux dimensions, avec un cluster par classe [5].
Pour le jeu à trois classes, utilisez une approche un contre tous pour faire du classement multiclasses avec des discriminants linéaires. Tracez des graphiques comparatifs des régions de décision et validez le bon fonctionnement de votre implantation du classifieur. Comme l'exercice se limite à valider le bon fonctionnement du classifieur, rapportez les résultats sur l'ensemble des données, utilisées également pour faire l'entraînement du classifieur.

(d) Comparez les résultats de votre classifieur avec les discriminants linéaires suivants :
— Méthode paramétrique de loi normale multivariée [6] ;
— Descente du gradient avec le critère du perceptron [7] ;
— Régression logistique [8].
Utilisez les deux jeux de données suivants pour faire votre comparaison :
— Iris de Fisher : jeu de 150 données pour l'identification d'iris, avec données en 4 dimensions et 3 classes. Le jeu est disponible avec la commande `datasets.load_iris`.
— Breast cancer wisconsin : jeu pour l'identification du cancer du sein, disponible avec la commande `datasets.load_breast_cancer`. Le jeu comporte 569 instances en 30 dimensions et selon 2 classes.
Normalisez préalablement les données selon les valeurs minimales et maximales du jeu [9]. Faites vos expérimentations selon une validation croisée à trois plis. Pour cette sous-question, limitez-vous à :
— Rapporter les paramètres d'entraînement utilisés pour chaque algorithme, s'il y a lieu ;
— Rapporter les taux d'erreur en entraînement et en test dans un tableau synthèse ;
— Discuter brièvement des résultats obtenus (performance en entraînement, performance en généralisation et temps de calcul).

(e) Utilisez maintenant une méthode aux k plus proches voisins sur les jeux de données mentionnés en (d), en normalisant les données. Vous pouvez directement utiliser l'implémentation de scikit-learn [10]. Utilisez les paramètres par défaut de ce classifieur, sauf pour :
— *Pondération (weights)* : testez avec *uniform* et *distance* pour le paramètre *weights*.
— *Nombre de voisins (k)* : pour les deux choix de pondération, testez avec les valeurs $\{1, 3, 5, 7, 11, 13, 15, 25, 35, 45\}$
Pour l'évaluation, utilisez une évaluation de type *leave-one-out*. Pour chaque jeu, rapportez la performance de chaque paramétrisation dans une figure synthèse (voir les directives dans le fichier `d2q2.py`). Discutez brièvement des résultats obtenus.

3/10/2018
CG & MAG

4. `make_classification(n_features=2, n_redundant=0, n_clusters_per_class=1)`
5. `make_classification(n_features=2, n_redundant=0, n_clusters_per_class=1, n_classes=3)`
6. `discriminant_analysis.LinearDiscriminantAnalysis`
7. `linear_model.Perceptron`
8. `linear_model.LogisticRegression`
9. `preprocessing.minmax_scale`
10. `neighbors.KNeighborsClassifier`