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

UNIVERSITÉ LAVAL

Devoir 3

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 :
  - Produisez vos solutions dans le fichiers fournis, en respectant les instructions et n'utilisant pas d'autres modules que ceux qui y sont importés
  - 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 14 novembre à 9h30
- Pondération :
  - Ce devoir compte pour 5% de la note finale

---

1. **Algorithme de rétropropagation (5pt)**
   Soit la fonction d'activation de neurone tangente hyperbolique, définie par l'équation suivante.

$$\mathrm{f}(z) = \tanh(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)} = \frac{\exp(2z) - 1}{\exp(2z) + 1}$$

   Développez les équations pour mettre à jour les poids et biais d'un perceptron multicouche avec des neurones utilisant la fonction d'activation tangente hyperbolique. N'oubliez pas de donner les équations pour mettre à jour la couche de sortie ainsi que les couches cachées.

2. **Discriminants non linéaires (5pt)**
   Soit les classifieurs suivants, présentés en classe :
   - Classement par les k-plus proches voisins avec distance euclidienne [1] ;
   - Séparateurs à vastes marges avec noyau gaussien [2], notez que le paramètre du noyau de la fonction est $\gamma = \frac{1}{2\sigma^2}$;
   - Perceptron multicouche pour le classement [3].

   Vous devez tester ces discriminants avec le jeu de données Pendigits [4], pour la reconnaissance de chiffres manuscrits (10 classes) à partir de 8 points des tracés des caractères (16 variables) [5]. Normalisez les données dans $[0, 1]^D$ avant vos traitements [6]. Utilisez 5 000 instances choisies au

1. `neighbors.KNeighborsClassifier`
2. `svm.SVC`
3. `neural_network.MLPClassifier. Pour cette méthode, utilisez le paramètre max_iter=100`
4. `https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits`
5. `Utilisez la fonction fetchPendigits() fournie dans d3q2.py`
6. `preprocessing.minmax_scale`

hasard comme données d'entraînement et les 5 992 données restantes pour les tests. Notez que la dernière colonne du jeu de données correspond aux étiquettes de classe.

Effectuez les opérations suivantes, en utilisant le fichier `d3q2.py`[7] :

(a) Déterminez les hyperparamètres importants des trois algorithmes présentés ci-haut qui peuvent influencer significativement les résultats d'apprentissage (limitez-vous à un maximum de deux hyperparamètres importants distincts par algorithme). Donnez une suite de valeurs que l'on devrait tester pour chacun de ces hyperparamètres, pour faire un bon ajustement de la configuration des classifieurs sur le jeu Pendigits.

(b) Testez ces trois classifieurs sur Pendigits. Pour cet exercice, ne vous fiez pas aux valeurs par défaut des fonctions de scikit-learn pour les hyperparamètres importants que vous avez identifiés en (a). Tentez plutôt de déterminer empiriquement les hyperparamètres utilisés pour chaque d'algorithme sur le jeu Pendigits, en utilisant par exemple une méthode de recherche en grille (lorsqu'applicable).

Faites votre ajustement des hyperparamètres sans utiliser les données de test. Vous êtes cependant libres d'utiliser la méthodologie de votre choix sur le jeu d'entraînement. Rapportez les résultats finaux sur tout le jeu de test, avec des modèles utilisant les hyperparamètres finaux et entraînés sur l'ensemble du jeu d'entraînement.

Pour cette question, limitez-vous à présenter les éléments suivants dans votre rapport :
— Justifier les différents choix que vous avez faits pour l'ajustement des hyperparamètres ;
— Rapporter les paramètres d'entraînement que vous avez déterminés empiriquement pour chacun des algorithmes ;
— Rapporter les performances (entraînement et test) dans un tableau synthèse ;
— Discuter brièvement des résultats (performance en entraînement, performance en généralisation, complexité des classifieurs et temps de calcul).

3. **Discrimination avec noyau et descente du gradient (10pt)**

Soit un discriminant avec noyau gaussien, entraîné par descente du gradient avec la fonction d'erreur suivante :

$$
\begin{aligned}
E(\boldsymbol{\alpha},w_0|\mathcal{X}) &= \sum_{\mathbf{x}^t\in\mathcal{Y}} [1 - r^t\,\mathrm{h}(\mathbf{x}^t|\boldsymbol{\alpha},w_0)] + \lambda \sum_{\alpha^t\in\boldsymbol{\alpha}} \alpha^t, \\
\text{avec :} & \\
\mathrm{h}(\mathbf{x}^t|\boldsymbol{\alpha},w_0) &= \sum_{\mathbf{x}^s\in\mathcal{X}} \alpha^s r^s K(\mathbf{x}^s,\mathbf{x}^t) + w_0, \\
\mathcal{Y} &= \{\mathbf{x}^t \in \mathcal{X} \mid r^t \mathrm{h}(\mathbf{x}^t|\boldsymbol{\alpha},w_0) < 1\}, \\
\alpha^t &\geq 0 \quad \forall t.
\end{aligned}
$$

Les valeurs cibles sont dans $r^t \in \{-1,1\}$ et le classement se fait selon le signe de $\mathrm{h}(\mathbf{x}^t|\boldsymbol{\alpha},w_0)$. Le paramètre λ permet de contrôler le niveau de régularisation effectué.

(a) Donnez les équations des gradients des poids α^t et de la constante w_0.

(b) Implémentez ce classifieur en respectant l'interface scikit-learn, en donnant au minimum les fonctions `fit`, `predict` et `score`. Utilisez à cette fin le fichier `d3q3.py`[8]. Pour l'optimisation des paramètres du classifieur, utilisez la méthode L-BFGS disponible dans SciPy[9] (référez-vous aux commentaires du fichier Python fourni pour plus de détails).

7. http://vision.gel.ulaval.ca/~cgagne/enseignement/apprentissage/A2018/travaux/d3q2.py
8. http://vision.gel.ulaval.ca/~cgagne/enseignement/apprentissage/A2018/travaux/d3q3.py
9. scipy.optimize.fmin_l_bfgs_b

(c) Expérimentez avec ce classifieur en utilisant un jeu de 1000 données synthétiques *moons* selon deux classes et avec du bruit blanc ($\sigma_{\text{bruit}} = 0{,}3$) [10]. Divisez le jeu en deux parties de taille égale, une pour l'entraînement et une pour le test. Affichez les données ainsi que la frontière obtenue avec le classifieur pour une paramétrisation de λ et σ (étalement du noyau gaussien) permettant d'obtenir un taux d'erreur inférieur à 10 % en évaluation sur le jeu de test [11]. Pour cette sous-question, donnez dans votre rapport :
   — Les différents paramètres d'entraînement évalués par la recherche en grille ainsi que les performances associées ;
   — Les valeurs de paramètres finaux retenus ainsi que les performances correspondantes (entraînement et test) ;
   — Le graphique des frontières de décision de la configuration retenue et des données utilisées.

12/10/2018
CG & MAG

10. `datasets.make_moons(n_samples=1000, noise=0.3)`

11. Pour obtenir un taux d'erreur inférieur à 10 % sur le jeu de données *moons*, il vous faudra probablement resserrer la recherche en grille sur le paramètre $\sigma \in [0, 1]$.