Deep Learning

TP 2 - Perceptron Multi-Couches

Nistor Grozavu

nistor.grozavu@lipn.univ-paris13.fr

Note. Le TP doit être effectué en Python en utilisant numpy et scikit-learn. Préparez un rapport.

A) Régression avec un Perceptron Multi-Couches

1. Créez un ensemble de points (une 20ène) suivant une fonction sinusoïde avec un bruit gaussien d'écart type 0,2. On va essayer de retrouver la fonction à partir des points avec un Perceptron Multi-Couches (MLP). Analyser la documentation sur MLP dans la librairie scikit-learn et analysez toutes les parametres.

   x = np.linspace(0,8,20).reshape(-1,1)

   y = np.sin(x) + np.random.randn(20)*0.2

2. Initialisez le MLP avec la fonction *MLPRegressor*. On veut 3 neurones cachés, une fonction d'activation linéaire et un pas d'apprentissage (learning rate) de 0,1 avec un nombre d'iterations de 100.

3. Lancez l'apprentissage du MLP. Utilisez la fonction *fit*.

4. La fonction *predict* permet d'utiliser le MLP appris pour faire de la régression.

5. Tracez sur une même figure les points d'apprentissage, la fonction sinusoïde réelle et celle estimée par le MLP.

B) Classification avec un Perceptron Multi-Couches

1. Les données IRIS sont constituées de 4 colonnes qui correspondent à 4 variables de description. La 5ème colonne donne la classe de chaque objet. Transformez ces données en deux matrices contenant les données et le label (target).

2. Pour permettre l'apprentissage par un MLP, il faut « binariser » les numéros de classe.

3. Initialisez le MLP avec la fonction *MLPClassifier*. On veut 2 neurones cachés, une fonction d'activation logistique et un pas d'apprentissage de 0,2.

4. On garde les mêmes options que dans l'exercice precedent. Lancez l'apprentissage du MLP avec la fonction *fit*. On veut utiliser l'algorithme d'optimisation *de descente de gradient stochastique (lbfgs)*.

5. Affichez les résultats désirés et les résultats obtenus dans une même figure. Vous pouvez utiliser un ACP par exemple. Chaque point aura une couleur différente selon sa classe. Les résultats sont-ils bon ?

6. Essayez de normaliser les données (centrées – réduites). Cela améliore-t-il la classification ?

7. Essayez de faire varier les paramètres du MLP (nombre de neurones cachés, etc...). Comment ces paramètres influencent les résultats ?

C) **Réduction de dimensions avec un Perceptron Multi-Couches**

1. Entraînez un MLP à retrouver les vecteurs d'entrée en sortie (la sortie doit être la même que l'entrée) sur la base de données Iris. On utilisera ici deux neurones cachés et une fonction d'activation linéaire.

2. Après apprentissage, calculez les sorties de la couche cachée pour chaque point de données en utilisant la matrice de poids (*coef*) et les données en entrées. Vous obtenez deux valeurs par points, qui donnent de nouvelles coordonnées pour chaque point. Affichez ces valeurs dans une figure (avec les classes réelles représentées en couleur).

3. Que représente cette figure ?

4. Si on utilise une fonction d'activation logistique, qu'est ce que ça change (et pourquoi ?).