TD n^o 6 : Réseaux de neurones / perceptrons multicouches

Exercice 1 : (**Mise en route**)

1. Pour utiliser le modèle de perceptron multicouche en classification supervisée, il faut déclarer le tableau des données des caractéristiques ainsi que les classes. L'exemple suivant génère le tableau des 100 premiers entiers naturels (pour chaque point d'entrainement, il n'y a qu'une seule caractéristique) ainsi qu'un tableau des classes observées : il s'agit de la classe 1 si l'entier est compris dans $[26, 75]$, et de la classe 2 dans le cas contraire.
```
Ntrain=100
X = numpy.asarray(range(0,Ntrain)).reshape(Ntrain,1)
y=[1]*Ntrain
for i in range(Ntrain):
    if X[i]<25 or X[i]>75:
        y[i]=2
```
2. On définit alors le classifieur et on calcule les "bons" poids :
```
from sklearn.neural_network import MLPClassifier
mlpc=MLPClassifier(hidden_layer_sizes=(100), learning_rate='adaptive')
mlpc.fit(X, y)
```
3. On génère un ensemble test :
```
Ntest=50
Xt = numpy.random.randint(0,Ntrain,(Ntest)).reshape((Ntest,1))
yt=[1]*Ntest
for i in range(Ntest):
    if Xt[i]<25 or Xt[i]>75:
        yt[i]=2
```
4. La prédiction sur l'ensemble test se fait alors par :
```
predictions =mlpc.predict(Xt)
```
5. On peut avoir accès à l'« accuracy » :
```
from sklearn.metrics import accuracy_score
print("accuracy:",accuracy_score(predictions, yt))
```

Exercice 2 : (**Utilisation d'un perceptron multicouche sur des données médicales**)

Nous allons utiliser des données mises à disposition sur le site :

`https://www.kaggle.com/ronitf/heart-disease-uci`.

Ces données contiennent 14 attributs :

1. `age` : age in years
2. `sex` : sex (1 = male ; 0 = female)
3. `cp` : chest pain type (1 : typical angina – 2 : atypical angina – 3 : non-anginal pain – 4 : asymptomatic)
4. `trestbps` : resting blood pressure (in mm Hg on admission to the hospital)
5. `chol` : serum cholestoral in mg/dl
6. `fbs` : (fasting blood sugar > 120 mg/dl) (1 = true ; 0 = false)
7. `restecg` : resting electrocardiographic results (0 : normal – 1 : having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV) – 2 : showing probable or definite left ventricular hypertrophy by Estes' criteria)
8. `thalach` : maximum heart rate achieved
9. `exang` : exercise induced angina (1 = yes ; 0 = no)
10. `oldpeak` : ST depression induced by exercise relative to rest
11. `slope` : the slope of the peak exercise ST segment (1 : upsloping – 2 : flat – 3 : downsloping)
12. `ca` : number of major vessels (0-3) colored by flourosopy
13. `thal` : 3 = normal ; 6 = fixed defect ; 7 = reversable defect
14. `target` : have disease or not (1=yes, 0=no)

1. Récupérez le fichier `heart.csv`. En python, pour charger les données on utilise les commandes suivantes :
```
import os
import pandas as pd
def load_heart_data(heart_path="./"):
```

```
    csv_path = os.path.join(heart_path, "heart.csv")
    return pd.read_csv(csv_path)
heart = load_heart_data()
heart.info()
```

2. Écrire une fonction qui à partir des données récupérées (données et labels), construit deux ensembles de données : un ensemble d'apprentissage et un ensemble de test. Pour cela on utilisera `shuffled_indices = np.random.permutation(n)` qui permute la liste des entiers de 0 à $n-1$.
3. Construisez le perceptron multicouche. Pour cela, on copiera d'abord les données (`X=train_set.copy()`) ; on supprimera l'attribut que l'on veut prédire (`del X["target"]`) ; puis on construira le vecteur des classes (`labels=train_set["target"]`).
4. Prédisez les classes des individus de l'ensemble de tests. Calculez le taux de bonnes réponses.
5. Faites varier le nombre d'unités cachées, ainsi que la structure du réseau. On pourra par exemple étudier l'influence du nombre de neurones cachés sur les performances, lorsqu'on fixe le nombre de couches cachées, et l'influence du nombre de couches cachées lorsqu'on fixe le nombre de neurones.

Exercice 3 : (**Perceptrons multicouches pour la regression**)

Nous allons construire un estimateur de la fonction $f(x,y) = \sqrt{(x^2+y^2)}$ à partir d'un échantillonnage bruitée de cette fonction

1. Construire un ensemble de 1000 points équirépartis dans $[-4,4] \times [-4,4]$. Calculez ensuite les valeurs de la fonction $f(x,y)$ auxquelles on ajoutera un bruit uniforme dans $[-0,1;0,1]$.
2. Visualisez les données générées :
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
surf0=ax.scatter(X[:,0], X[:,1], y)
ax.zaxis.set_major_locator(LinearLocator(10))
ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))
plt.show()
```
3. Construire un perceptron multicouche qui estimate la fonction $f(x,y)$ grâce à `MLPRegressor()`, puis entrainez-le avec `fit()`.
4. Construisez ensuite un nouveau jeu de données de test, et prédisez la valeur de la fonction $f(x,y)$.
5. Visualisez les données d'entrainement et d'estimation :
```
fig1 = plt.figure()
ax1 = fig1.add_subplot(111, projection='3d')
surf1=ax1.scatter(Xt[:,0], Xt[:,1], predictions)
surf2=ax1.scatter(X[:,0],  X[:,1],  y, c='r')
ax1.zaxis.set_major_locator(LinearLocator(10))
ax1.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))
plt.show()
```
6. Calculez la somme des erreurs d'estimation, ainsi que l'erreur moyenne.
7. Changez l'amplitude de l'erreur, ainsi que la structure du perceptron multicouche.

Exercice 4 : (**Perceptrons multicouches pour le XOR**)

1. Créer un jeu de points dans le plan : les points qui ont les deux coordonnées de même signe seront d'une certaine classe, et ceux dont les coordonnées sont de signe contraire seront de l'autre classe.
   — On pourra utiliser `numpy.random.uniform(-4,4,(5000,2))` pour générer une matrice de 5000 lignes et 2 colonnes de nombres aléatoires suivant une distribution uniforme sur l'intervalle [-4,4]
   — Pour chacun de ces points, il faudra calculer la classe : il sera de la classe 1 si et seulement si les 2 coordonnées sont de même signe.
2. Visualisez les données générées (`Scatter plots` de matplotlib).
3. Construisez un prédicteur en utilisant un perceptron multicouche.
4. Pour évaluer la performance du prédicteur, on construira une centaine de nouveaux points, on utilisera le prédicteur pour affecter le point à une classe, puis on comparera la prédiction avec la "vraie" classe.
5. Refaire la même chose en mettant dans les données d'apprentissage un peu de bruit : pour x% des 5000 points d'apprentissage, changer la classe. Testez un pourcentage d'erreur allant de 0% à 100% et tracez la courbe du pourcentage d'erreurs de prédiction en fonction du pourcentage d'erreur sur l'ensemble d'apprentissage.