A2DI - TP n°8

Ce 8ième TP abordent les thématiques de la **régularisation** et de la **réduction de dimension**.

Exercice n°1 : La vérité n'est pas la meilleure solution

Dans cet exercice, on montre que le meilleur choix d'un paramètre (au sens de l'erreur de généralisation) n'est pas forcément sa valeur réelle !
Nous reprenons le cas de la régression polynomiale abordée dans le TP n°3.

Questions :

1. Générez un polynôme de degré 8 en exécutant le code suivant :
```
p=8
theta=np.asarray([-1.34e-1,1.5,5.75e-2,-5.9e-2,-3.75e-3,1.5e-3,7.6e-5,-9.9e-6,-4.7e-7])
x=np.linspace(-10.0,10.0,200)
mypoly = 0.0
for i in range(p+1):
      mypoly = mypoly + np.power(x,i)*theta[i]
```
2. Tracez ce polynôme sur l'intervalle $[-10; 10]$.
3. Tirez 20 points au hasard (uniformément) sur l'intervalle $[-10; 10]$ et sauvegardez les dans un textttnumpy array noté `X`.
4. Générez un vecteur `y` qui correspond à l'image des points contenus dans `X` par le polynôme `mypoly`.
5. Ajoutez un bruit centré Gaussien d'écart type 3 à `y`.
6. Superposez les points ainsi générés à votre figure précédente.
7. Utilisez le code de la régression polynomiale produit lors du TP n°3 pour apprendre un polynôme à l'aide des 20 points bruités. On entraînera un polynôme de degré 1 à 8.
8. Sur une nouvelle figure, tracez l'erreur de test et de train en fonction du degré du polynôme. On constate que la meilleure généralisation n'est pas obtenue pour un degré 8.

Exercice n°2 : Impact d'une réduction dimension sur les performances

Dans cet exercice, on s'intéresse à la reconaissance de caractères manuscrits. Nous travaillerons sur un sous-ensemble du dataset `MNIST` qui contiendra exclusivement des "8" et des "9".
On comparera le taux de bonne classification obtenu avec une régression logistique, puis une régression logistique avec ACP et enfin une régression logistique avec pénalité *sparse*.

Questions :

1. Chargez le dataset en exécutant le code suivant :
```
import cPickle
import gzip
f = gzip.open('mnist.pkl.gz', 'rb')
train_set, valid_set, test_set = cPickle.load(f)
f.close()
X=train_set[0]
c=train_set[1]
```

2. Sélectionnez les exemples correspondant uniquement aux caractères “8” et “9” ainsi que leur classes. Vous convertirez les classes en 0 et 1.

3. Découpez en 2 parts égales les données pour le train et le test.

4. Instanciez une régression logistique à l'aide de la classe définie dans `sklearn` :
```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(C=1e12,solver='liblinear')
```

5. Appelez la méthode `fit` de l'objet `clf` pour entraîner l'algorithme.

6. Appelez la méthode `predict` de l'objet `clf` pour tester l'algorithme sur l'ensemble de test et calculez le taux de bonne classification.

7. Instanciez une ACP à l'aide de la classe définie dans `sklearn` :
```
from sklearn import decomposition
pca = decomposition.PCA(n_components=110)
```
L'instruction ci-dessus permet de conserver 110 dimensions correspondant aux 110 plus grosses valeurs propres.

8. Appelez les méthodes `fit` et `transform` de l'objet `pca` pour obtenir une représentation intermédiaire des données.

9. Ré-entraînez la régression logistique à partir des données dans cette représentation et re-calculez le taux de bonne classification correspondant.

10. Ré-instanciez une régression logistique à l'aide de la classe définie dans `sklearn` :
```
clf = LogisticRegression(penalty='l1',C=100,solver='liblinear')
```
Cette instruction va incorporer une pénalité L_1 *sparse* dans la fonction de coût à minimiser.

11. Ré-entraînez la régression logistique avec cet objet et re-calculez le taux de bonne classification correspondant.