tp2

Unknown Author

April 9, 2015

1 K-plus proche voisin, noyaux

1.1 Données USPS

Télécharger les deux fichiers : http://webia.lip6.fr/mapsi/uploads/Cours/2014_tme3_usps_train.txt http://webia.lip6.fr/mapsi/uploads/Cours/2014_tme3_usps_test.txt

C'est une base de données de chiffre manuscrit (de 0 à 9), sous format d'un tableau de pixels de 16*16. Les données sont à plat, chaque ligne correspond à un exemple, un vecteur 1d de 16*16=256 dimensions. Les valeurs sont entre 0 et 2, en fonction de l'intensité du pixel.

- Utilisez la fonction suivante pour charger les données.
- Etudiez la lligne qui permet d'afficher un tableau. Vous pouvez en faire une fonction pour qu'elle soit plus pratique
- Affichez quelques images.

In [3]:
```
def load_usps(filename):
    with open(filename,"r") as f:
        f.readline()
        data =[ [float(x) for x in l.split()] for l in f if len(l.split())>2]
    tmp = np.array(data)
    print tmp.shape
    return tmp[:,1:],tmp[:,0].astype(int)

def plot(mat):
    plt.imshow(mat.reshape((16,16)),interpolation="nearest",cmap=cm.binary)
```

1.2 K-plus proche voisin

Les k-nn (nearest neighbor) est un algorithme de classification relativement simple. Il considère comme classe pour un point à classer la classe majoritaire parmi ses k plus proches voisins. Pour cela, l'algo doit calculer la distance à tous les points d'apprentissage, puis les trier et récupérer les k plus proches.

Compléter l'algo des K-nn suivant :

In [6]:
```
from tools import *
class Knn(Classifier):
    def __init__(self,k=3):
        # k indique le nombre de voisin
        self.k=k
    def fit(self,data,y):
        # Enregistre les donnees
        self.data=data
```

```
        self.y=y
    def closest(self,data):
        #Renvoie les indices des k plus proches
        y=[]
        # A completer
        return y

    def predict(self,data):
        #Renvoie la classe majoritaire
        y=np.zeros(data.shape[0]).astype(int)
        res=self.closest(data)
        for i,x in enumerate(res):
            y[i]=Counter(self.y[res]).most_common()[0][0]
        return y
```

Utiliser les k-nns pour classifier :

- les données artificielles de la semaine derniere
- les données réelles USPS

Etudier en particulier :

- l'erreur en apprentissage
- l'erreur en test (sur le fichier test)
- l'évolution en fonction du nombre de voisins

Etudier quelques images qui sont mal classifiées

Proposer une méthode pour obtenir les cellules de Voronoi en 2-d (tous les points d'une cellule appartiennent au vosin le plus proche. Les frontières sont en fonction des equidistances entre point).

1.3 Noyaux

Rappel : un noyau est une fonction de $K : X \times X \to \mathbb{R}$ qui est dit admissible s'il existe une fonction $\phi : X \to X'$ et $K(x, x') =< \phi(x), \phi(x') >$.

En particulier (vous montrerez) :

- cK est un noyau pour $c \in \mathbb{R}^+$
- $K + K'$ est un noyau
- KK' également
- $(1+ < x, x' >)^d$ aussi
- Que pensez-vous de la projection quadratique du TP précédent ? de la gaussienne ?
- Voyez-vous un intérêt à cette fonction k plutot qu'au produit scalaire explicite ? Pensez au dernier exemple.
- Que devez vous changer dans le perceptron pour pouvoir le rendre "kernelisable" ?

Programmez le noyau gaussien : $k(x, x') = Cexp(-\|x - x'\|^2/\sigma^2)$

1.4 Données vélib

Télécharger l'archive velibdata.pkl. Vous pouvez importer les données avec les lignes de codes ci-dessous. Ces données sont les relevés des stations velib parisiennes pendant 7 jours. La matrice *velib* contient le nombre de velib disponbile à la borne, chaque ligne étant une borne, et chaque colonne 10min depuis le temps de référence (données aggrégées par 10min). Le dictionnaire *infoId* contient les informations sur les stations, sout le format : id -> (adresse, altitude, id velib, nom, total velib disponible, x, y).

- Visualiser les données d'une station sur toute la période
- Visualiser les données d'une station, les jours aggrégées
- Proposer une méthode pour lisser ce signal (en pensant au noyaux).
- Faites le lien avec l'estimateur de Nadaray-Watson : $y(x) = \frac{\sum_{i=1}^{N} K(x,x_i) y_i}{\sum_{i=1}^{N} K(x,x_i)}$
- Construiser une nouvelle représentation lissée des stations. Faites tourner les k-nn dessus. Observez les plus proches voisins

In []:
```
import pickle
velib,infoId=pickle.load(file("datavelib.pkl"))
```

1.5 Retour aux images

Proposez un noyau pour lisser les images. Testez.