Machine Learning: Assignment #4

Winter 2018

Due: Jan 13th, 23:59:59 CST (UTC +8).

1. Spectral Clustering

In this problem, we will try a dimensionality reduction based clustering algorithm – Spectral Clustering. Implement Spectral Clustering algorithm in *spectral.m*/*spectral.py*, then answer the following questions.

(a) We will first experiment Spectral Clustering on synthesis data (in *spectral_exp1.m* or *spectral_exp.ipynb*). Finish *knn_graph.m*/*knn_graph.py* to construct the KNN graph[1] W, then test your algorithm on data *cluster_data.mat*. Visualize the clustering result and compare it with k-means.

Note that we only care about the local connectivity of data points, therefore edges between far away points should be discarded. You should take care with this issue when implementing *knn_graph.m*/*knn_graph.py* and choose a proper threshold.

(b) Now let us try Spectral Clustering on real-world data (in *spectral_exp2.m* or *spectral_exp.ipynb*). The TDT2 corpus consists of data collected during the first half of 1998 and taken from 6 sources, including 2 newspapers (APW, NYT), 2 radio programs (VOA, PRI) and 2 television programs (CNN, ABC).

Try Spectral Clustering on a subset of TDT2 corpus (in *TDT2_data.mat*) and compare the result with k-means. You should evaluate the quality of clustering using accuracy and normalized mutual information[2].

For this part of problem, use *constructW.m*/*constructW.py* to construct the graph W, and choose proper parameters. As usual, you should preprocessing the data and repeat the experiments multiple times then take the average.

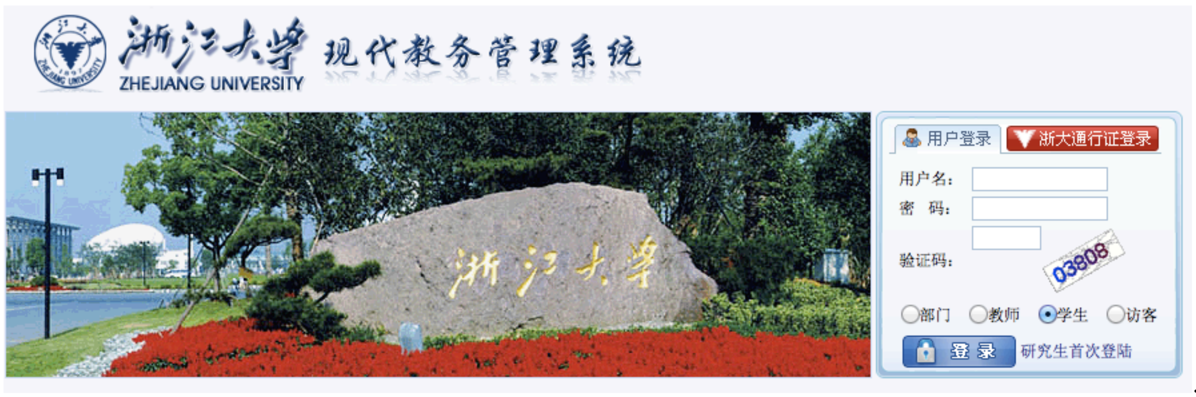
2. Principal Component Analysis

Principal component analysis (PCA) is a mathematical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. Let us deepen our understanding of PCA (you should implement it in *pca.m*/*pca.py*) by the following problems.

(a) Our actions to hack the CAPTCHA system last time have been detected, so they updated the system to add rotation in the CAPTCHA images:

[1]Connect each data point with its K nearest neighbors, you can use binary edge weights or compute edge weights using Heat kernel.

[2]See `http://www.cad.zju.edu.cn/home/dengcai/Data/Clustering.html` and Deng Cai, Xiaofei He, and Jiawei Han, "Document Clustering Using Locality Preserving Indexing", in IEEE TKDE, 2005, for more details.



.

Luckily, we have just learnt PCA. Your task is to implement *hack_pca.m*/*hack_pca.py* to recover the rotated CAPTCHA image using PCA.

Some rotated CAPTCHA samples are provided for you. It's OK for your algorithm to return grayscale image or image with size different form input image.

(b) Now let us apply PCA to a face image dataset in *ORL_data.mat*.

   (i) Run PCA on the dataset, visualize the learnt eigenvectors using *show_face.m* or *show_face.py*. You should see faces in the image, these faces are called Eigenface.

   (ii) Use PCA to do dimensionality reduction[3]. You should experiment with number of reduced dimensionality being 8, 16, 32, 64 and 128 respectively. For each test, run KNN on the dimensionality reduced dataset and report the testing error rate.

   (iii) Will dimensionality reduction cause loss of information? Let us see it visually. Do dimensionality reduction using PCA then recover the original image form the low dimensional feature vectors. Visualize the original and recovered image using *show_face.m*/*show_face.py*. Again, you should experiment with number of reduced dimensionality being 8, 16, 32, 64 and 128 respectively.

Please submit your homework report to at `http://assignment.zjulearning.org:8081` in pdf format, with all your code in a zip archive.

[3]You are strongly encouraged to try LDA on this part of problem. You should be impressed by its outstanding performance.