Introduction to Data Mining: Assignment #3

Summer 2014

Due: June 18th, 23:59:59 CST (UTC +8).

1. K-Nearest Neighbor

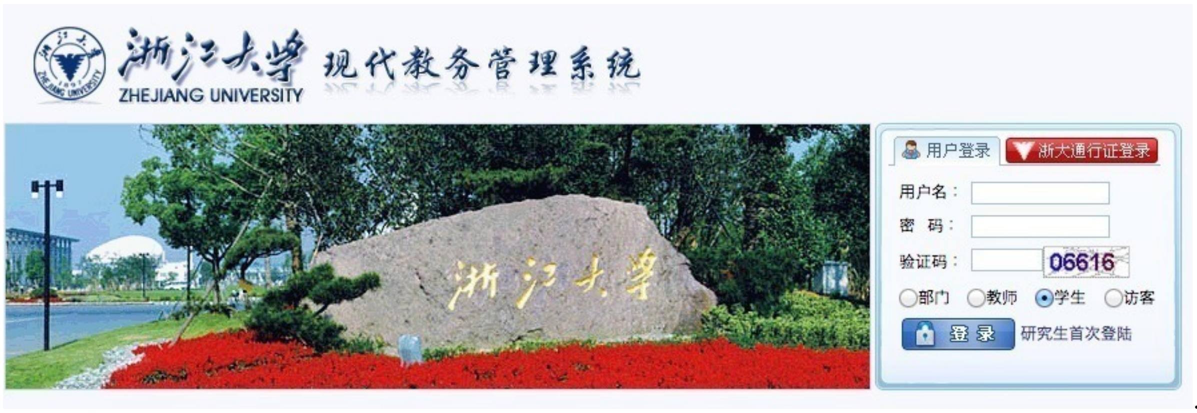
In this problem, we will play with K-Nearest Neighbor (KNN) algorithm and try it on real-world data. Implement KNN algorithm (in *knn.m*), then answer the following questions.

(a) In *knn_exp.m*, try KNN with different K (you should at least experiment K = 1, 10 and 100) and plot the decision boundary.

You are encouraged to vectorize[1] your code, otherwise the experiment time might be extremely long. You may find the MATLAB build-in functions *pdist2, sort, max* and *hist* useful. Also, you can use the function *eudist2*[2] written by Prof. Deng Cai[3].

(b) We have seen the effects of different choices of K. How can you choose a proper K when dealing with real-world data ?

(c) Now let us use KNN algorithm to hack the CAPTCHA of a website that we are all familiar with:



.

Finish *hack.m* to recognize the CAPTCHA image using KNN algorithm.

You should label some training data yourself, and store the training data in *hack_data.mat*. Helper functions *extract_image* and *show_image* are give for your convenience.

Remember to submit *hack_data.mat* along with your code and report.

[1] `http://www.mathworks.cn/cn/help/matlab/matlab_prog/vectorization.html`

[2] `http://www.cad.zju.edu.cn/home/dengcai/Data/code/EuDist2.m`

[3] Prof. Deng Cai is an expert on MATLAB, you can find all his code at `http://www.cad.zju.edu.cn/home/dengcai/Data/data.html`. You can learn how to write fast MATLAB code by reading his code.

2. Decision Tree[4] and ID3

ID3 is a classic method to construct decision tree by branching the tree to maximize information gain. With this intuition, answer the following questions.

(a) There are 27 balls, where one of them is lighter than the others. Please design a decision tree to find out which one is lighter with 3 trials. Prove that your decision tree is optimal in terms of number of trials.

(b) Consider the scholarship evaluation problem: selecting scholarship recipients based on gender and GPA. Given the following training data:

Gender	GPA	Scholarship	Count
F	Low	+	10
F	High	+	95
M	Low	+	5
M	High	+	90
F	Low	-	80
F	High	-	20
M	Low	-	120
M	High	-	30

Draw the decision tree that would be learned by ID3 algorithm and annotate each non-leaf node in the tree with the information gain attained by the respective split.

3. K-Means Clustering

Finally, we will run our first unsupervised algorithm – k-means clustering. Implement k-means algorithm[5] (in *kmeans.m*), then answer the following questions.

Note that there are different kind of methods to setup initial cluster centers for k-means algorithm, we will use a simple one – randomly choose K samples from dataset as initial cluster centers.

(a) Run your k-means algorithm on *kmeans_data.mat* with the number of clusters K set to 2. Repeat the experiment 1000 times. Use *kmeans_plot.m* to visualize the process of k-means algorithm for the two trials with largest and smallest SD (sum of distances from each point to its respective centroid).

(b) You should observe the issue that the outcome of k-means algorithm is very sensitive to cluster centroids initialization form the above experiment. How can we get a stable result using k-means?

[4]Decision Tree are mostly used as building blocks for Random Forest nowadays. Microsoft use Random Forests for real time pose estimation in Kinect. You can play with Random Forest on the digit dataset we used last time, however it is beyond the scope of our course.

[5]You can reuse the *knn.m* to simplify your implementation.

(c) Run your k-means algorithm on the digit dataset *digit_data.mat* with the number of clusters K set to 10, 20 and 50. Visualize the centroids using *show_digit.m*. You should be able to observe that k-means algorithm can discover the patterns in dataset without any label information.

(d) Another important application of k-means is Vector quantization[6]. Vector quantization is a classical quantization technique from signal processing. It works by dividing a large set of points (vectors) into groups, then representing the data points by their group centroid points, as in k-means and some other clustering algorithms.

Here we will use vector quantization to do image compression. By clustering image pixel value into K groups, we can represent each pixel with $\log(K)$ bits, instead of 24 bits (RGB, each channel has 8bit depth).

Finish *vq.m*. Compress images with K set to 8, 16, 32 and 64. I have provided you some sample images, however use your own photos is encouraged.

What is the compress ratio if we set K to 64 (Optionally, you can compute the compress ratio using Huffman encoding) ?

4. Spectral Clustering

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

(a) We will first experiment Spectral Clustering on synthesis data (in *spectral_exp1.m*). Finish *knn_graph.m* to construct the KNN graph[7] 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* and choose a proper threshold.

(b) Now we let us try Spectral Clustering on real-world data. 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 the normalized mutual information metric[8].

[6] `https://en.wikipedia.org/wiki/Vector_quantization`

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

[8] 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.

For this part of problem, use *constructW.m* 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.

(c) Now let us play Spectral Clustering on your own data! Use a crawler[9] to collect your Renren social network information and construct a network graph. Then run Spectral Clustering on the graph (in *spectral_exp3.m*) using different K (number of clusters). Can you see any meaningful clusters from the clustering result?

You need to write some scripts to preprocess the adjacency list form the crawler to get the adjacency matrix W. You can use any programming languages you like (e.g. Python or my favourite – Ruby). You can also try alternative edge weighting methods, say weighting using number of shared friends.

Please submit your homework report in **pdf** format, with all your code in a zip archive.

[9]See `https://github.com/5kg/renren`. You can clone the repository or use the ”Download ZIP” link.