University of Ioannina,
Computer Science Department

Data Mining – CS 059
Fall 2012

Assignment 3

The deadline for the assignment is Friday, December 7. For late submissions the late policy on the page of the course will be applied. The details for the turn-in will be announced on the page of the course.

Question 1

In the course textbook (Introduction to Data Mining by Tan, Steinbach, Kumar) except for the clustering algorithms that we described in class, there are also the algorithms CLARANS, BIRCH, ROCK, CHAMELEON, DENCLUE, and CURE (also described in the free online textbook Mining Massive Datasets by Rajaraman and Ullman). Select one of these algorithms, and describe the main idea in your own words in 2-3 paragraphs. You can read the corresponding paper if you need additional details.

Question 2

This is Exercise 8.27 from the textbook Introduction to Data Mining by Tan, Steinbach, Kumar. Prove that the Sum of Square Errors (SSE) for a cluster C_i is proportional to the sum of distances between all points in the cluster C_i. More specifically, prove that

$$\sum_{x\in C_i} \|\boldsymbol{x}-\boldsymbol{c_i}\|^2 = \frac{1}{2m_i}\sum_{x\in C_i}\sum_{y\in C_i} \|\boldsymbol{x}-\boldsymbol{y}\|^2$$

where: m_i is the number of points in the cluster C_i ; $\boldsymbol{c_i}$ is the centroid of the cluster C_i; the clustered points are d-dimensional real vectors; and $\|\cdot\|$ is the Euclidean distance. (**Hint**: Start with the case that $d = 1$).

Question 3

We have a collection D of N documents, where each document is "bag of words" drawn from a vocabulary W of m words. Document d_i can be represented as an m-dimensional vector where d_{ij} is the number of times that word w_j appears in document d_i. Consider a clustering $C = \{c_1 \dots, c_K\}$ of the documents in D where $1 \le K \le N$. When $K = N$ each document is in a cluster by itself (singleton clusters), while when $K = 1$, all documents are together in a single cluster. A cluster c_i contains the concatenation of all words of all the documents in the cluster. Therefore, it can also be represented as an m-dimensional vector over the set W of words: the sum of the vectors of all the documents in the cluster. For cluster c_i we use n_{ij} to denote the number of times that word w_j appears in the cluster. We use n_i to denote the number of words in cluster c_i, and n to denote the number of all words in all documents. If we normalize the vector for cluster c_i we obtain a probability distribution $P(W|c_i)$, where $p_{ij} = p(w_j|c_i) = \frac{n_{ij}}{n_i}$ is the conditional probability of word w_j in cluster c_i. This is the probability that if we randomly select a word from the clusterc_i this word will be w_j. We use P_i to denote the distribution $P(W|c_i)$. We also define $p(c_i) = \frac{n_i}{n}$ to denote the probability of cluster c_i. This is the probability that if we pick a randomly a word among all the words in all documents, this will be from cluster c_i.

We can now define the conditional entropy of the words W given the clustering C. We have

$$H(W|C) = \sum_{c_i \in C} p(c_i) H(W|c_i) = -\sum_{c_i \in C} p(c_i) \sum_{w_j \in W} p(w_j|c_i) \log p(w_j|c_i)$$

The entropy of the words in cluster c_i is given by $H(W|c_i) = H(P_i)$, the entropy of the word distribution within the cluster. We call $H(P_i)$ the *entropy of the cluster* c_i, and $H(W|C)$, *the entropy of the clustering* C.

Consider now two clusters (e.g., cluster c_1 and cluster c_2), and suppose we merge them to create a new cluster c_*. We have that $p(c_*) = p(c_1) + p(c_2)$

1. Show that

$$P_* = P(W|c_*) = \frac{p(c_1)}{p(c_*)} P_1 + \frac{p(c_2)}{p(c_*)} P_2$$

We define a generalized version of Jenshen-Shannon divergence between distributions P_1 and P_2 as

$$D_{JS}(P_1, P_2) = \frac{p(c_1)}{p(c_*)} D_{KL}(P_1||P_*) + \frac{p(c_2)}{p(c_*)} D_{KL}(P_2||P_*)$$

$D_{KL}(P_1||P_*)$ is the KL-divergence between distributions P_1 and P_*. Now, let *C* be the clustering that has clusters c_1 and c_2, and *C** the clustering after merging the clusters c_1 and c_2 into cluster c_*.

2. Show that the increase in entropy is
$$H(W|C^*) - H(W|C) = p(c_*)\, D_{JS}(P_1, P_2)$$
There are clustering approaches that aim at minimizing the entropy of a clustering and use the increase in entropy as a distance metric between two clusters. We can then apply an agglomerative algorithm, or a *k*-means like algorithm

Consider a toy example were the vocabulary consists of just two words W = {“sports”, “politics”}, and we have three documents d_1, d_2, d_3. Document $d_1$contains two occurrences of word “sports”, document d_2 contains three occurrences of word “sports”, and document $d_3$contains five occurrences of word “politics”.

3. What is the entropy of the documents, and what does this mean? What is the distance of the two closest documents and what does this mean? What is the entropy of a cluster that contains all three documents?

Question 4

The goal of this question is to experiment with K-means clustering. You can do your own implementation of the K-means algorithm that we described in class, or use a suitable existing implementation. (K-means is implemented in MATLAB, WEKA, and there are several available implementations online but it is a simple algorithm and it may be easier to implement it yourselves).

You will apply the algorithm to the Twitter data that you used for Assignment 1. For this question you will not use all of the data. Using the location field (the 7th field in the file) you will select the users that live in following cities: London, Los Angeles, New York, San Francisco, and Hollywood. Make sure that the method you use for selecting the users finds most of the different ways in which the city is described (e.g., ”NYC” for “New York”, “Los Angeles” v.s. “Los Angeles, CA”). Inspect the data manually, and throw away users that specify more than one city.

After obtaining the appropriate users, take the profile description and do the same preprocessing (removing stop-words, etc) as in Assignment 1. The result will be a real vector for each with one entry for each term, and the count of times that the term appears in the profile description. The implementation of the vectors will depend on the implementation of K-means that you use (e.g., you will probably not need to create the zero entries of the vector). Apply K-means clustering on these vectors, for K = 5.

Evaluate the results of the K-means clustering by comparing against the five classes defined by the five cities. These will serve as the external validation of your algorithm. Produce the cluster-city confusion matrix, and compute the Precision and Recall for each cluster.

Submit your code, the file with the users that you selected (two columns, one with the city and one with the profile description), and the same file with the users grouped by cluster. Submit also a report with the details of your experiment: the implementation that you used for K-means; the pre-processing that you did on the data and the way you selected the input users; the confusion matrix and the Precision-Recall values for your clustering.

Similar to Assignment 1 if there is another dataset that you are interested in analyzing and clustering, you can contact me with your suggestion.

Question 5

In class we described the Minimum Description Length principle and its application to the problem of co-clustering. In this question you will apply MDL to the problem of segmentation of a one-dimensional sequence. The input is a sequence of 0/1 numbers, and the goal is to segment the sequence into segments such that the total cost of encoding the sequence is minimized. Note that in this variation of the problem you are **not** given the number K of segments as input.

1. Define the encoding cost of the sequence. Clearly state what is the model cost and what is the data cost given the model. **Hint**: A segmentation into K segments is defined by K-1 boundary points.
2. Give a heuristic algorithm for the problem of finding a segmentation that minimizes the encoding cost. Describe the algorithm in English and/or using pseudocode.
3. The segmentation problem can be solved optimally in polynomial time using dynamic programming. Define the dynamic programming recurrence, and the dynamic programming array.
4. **Bonus**: Implement an algorithm that solves the problem. Test your algorithm on a sequence of 1000 values that you will create as follows:
    a. Select two boundary points randomly.
    b. Create random 0/1 for each segment with different probability: in the first segment the probability of value 1 is 0.7, in the second 0.3, and in the third 0.9
    c. Submit a file with the true breakpoints and one with the ones that your algorithm finds on 3 different random inputs.