Introduction to Machine Learning, Spring 2016

Problem Set 5: Clustering

Due: Monday, March 21, 2016 at 10pm (upload to NYU Classes.)

Important: *See problem set policy on the course web site.*

Instructions. You must show **all** of your work and be rigorous in your writeups to obtain full credit. Your answers to the below, plots, and all code that you write for this assignment should be uploaded to NYU Classes.

Food Clustering – US Department of Agriculture Nutrient Database

In this problem set we will explore unsupervised learning with K-means and Hierarchical clustering. In `ps5SkeletonCode.py`, we provide a `Python` function to evaluate the quality of the learned clusters and a code sample for plotting dendograms.

Dataset:

Looking at the USDA nutrient database, one would find various types of foods with their corresponding nutritional contents: `http://ndb.nal.usda.gov/ndb/search/list`. In this problem, we will experiment with 4 food groups: `Cereal-Grain-Pasta, Finfish-Shellfish, Vegetables, Fats-Oils`.

The data contains detailed categorizations of each food item. For example, there are 9 categories of `Kale`. They range from `raw, frozen and unprepared, cooked and boiled, cooked and drained without salt`, etc. In addition, common knowledge suggests that major food groups can be further categorized. Vegetables can be leaves/stems, roots, or buds. Based on their nutritional contents, one might expect to see these items clustered in hierarchies, from the major food groups, sub-groups, and finally to variants of the same food items.

The following data files are provided with the assignment and can be downloaded from the course website:

1. `dataDescriptions.txt` – gives the names of all the attributes (nutrients).

2. `dataCereal-grains-pasta.txt, dataFinfish-shellfish.txt, dataVegetables.txt, dataFats-oils.txt` – the data files, one for each food group.

Please note that the attributes in the data files are delimited by the `caret (^)` character.

1. Numerical values vary widely across different type of nutrients. However, small numerical values in some micro-nutrients such as minerals and vitamins may characterize the food items as well as larger numerical values in macro-nutrients like protein and carbohydrates. Therefore it becomes important to normalize the nutrient values. For this problem, transform the features to be in the range $[0, 1]$. E.g., for the j^{th} dimension, the value of the i^{th} data point will become

$$Normalized(X_{ij}) = \frac{X_{ij} - min(X_{\cdot j})}{max(X_{\cdot j}) - min(X_{\cdot j})},$$

   where `min` and `max` for $X_{\cdot j}$ are calculated over the j^{th} attribute/dimension on the complete dataset.

2. Apply K-means clustering on the original data, with size of cluster $k = 4$.

3. To quantify the performance of this clustering with respect to known class labels (in this case, the original food groups), a simple metric, Rand Index is used. A function `randIndex` in `Python` is provided in the skeleton code. It calculates the similarity value between two class label assignments, the ground truth labels and the predicted labels.

   If C is a ground truth class assignment and K the clustering class assignment, let us define a and b as:

   - a, the number of pairs of elements that are in the same class in C and in the same class in K

- b, the number of pairs of elements that are in different classes in C and in different classes in K

The raw (unadjusted) Rand index is then given by:

$$RI = \frac{a+b}{\binom{n_{\text{samples}}}{2}},$$

where $\binom{n_{\text{samples}}}{2}$ is the total number of possible pairs in the dataset (without ordering). One of the important properties of this metric is that it is invariant to re-labeling of the clusters.

Let's look at an example using 4 data points. The ground truth labeling C is a row vector $< 1, 2, 2, 3 >$, where each element C_i represents the actual class label that data point x_i belongs to. Suppose that our clustering algorithm returns $K =< 1, 1, 1, 3 >$, where K_i is the cluster assigned to data point x_i. We use the notation (C_i, C_j) and (K_i, K_j) to refer to a pair of data points in C and K, respectively. For example, when $i = 0$ and $j = 1$, we have $(C_0, C_1) = (1, 2)$ and $(K_0, K_1) = (1, 1)$. In this case, C and K disagree on the class assignment for data point x_0 and x_1. The data points belong to different classes according to C, but they belong to the same class according to K. By contrast, they both agree that data points x_1 and x_2 belong to the same class. The total number of such instances is a. Similarly, they both agree that data points x_2 and x_3 belong to different classes. The total number of such instances is b. The ratio between the sum of these two situations, and the total possible ways of pairing is the RI we will be using.

Using the ground truth labels from the 4 food groups, compute the Rand Index for:

(a) A random permutation of ground truth labels. This is used to provide a baseline to compare the other numbers we will obtain from this measurement.

(b) Labels obtained using K-mean clustering in the previous question.

4. The K-means algorithm aims to choose centroids C that minimizes the within cluster sum of squares objective function on a dataset X with n samples:

$$J(X, C) = \sum_{i=0}^{n} \min_{\mu_i \in C} (||x_i - \mu_i||^2)$$

Given enough iterations, K-means will always converge. However it may end up a local minimum of $J(X, C)$, which depends on the initialization of the centroids (cluster means). As a result, this computation is typically performed several times, with different initialization of the centroids. The clustering which achieves the minimum value of the objective function is returned as the final result.

Python's `sklearn` implementation of K-Means provides various schemes of initializing the centroids. One can specify the number of times to repeat the K-Means computation to pick the minimum from. In `sklearn` this is specified by the parameter `init_t`.

Set this value to `1`, so that the respective K-Means execute exactly 1 computation. Also, set the centroid initialization schemes to be uniformly 'random'. With these settings:

(a) Run K-Means 20 times. For each of these, record the value of the objective function $J(X, C)$. For this particular dataset, most of these 20 runs should return the same value for the objective function, but there will be a few with much higher values.

Note: `sklearn` returns the value of the objective function; see `inertia_`. You can alternatively also compute this value using the returned centroids and the data.

(b) Report the distinct values of the objective functions that you observed, highlighting the minimum. In some cases, you may need to run this a few more times to get distinct values.

(c) For each of these distinct clustering solutions, report the corresponding `Rand Index` value.

5. Next, we want to visualize the possible structures of food using hierarchical clustering and dendrograms. Randomly select 30 food items from each of the 4 food groups. Create a dendrogram and review the labels of the food items from the dendrograms. Do the food items cluster into 4 groups as expected? Identify any distinct clusters and the corresponding food items.

   To create a dendogram, `SciPy` has a `dendogram` function that takes as input the linkage matrix of a hierarchical clustering. We can generate this using the `linkage` function, specifying that it use the `'complete'` metric. We use the default Euclidean distance for both. Please refer to the sample code for dendogram plotting.

6. To obtain actual clusters from hierarchical clustering, one way is to cut the dendogram along the Y-axis using some threshold (referring back to the lecture, this corresponds to a horizontal cut). Note that the height of each edge in the dendogram describes the distance between the children in a node. From the code sample with the toy data, you can verify that the distance between two leaves is the height at which they are joined. `Scipy` has the function `fcluster` which take as input the linkage (dendogram) and the `'distance'` threshold on this Y-axis and uses it to cut the tree.

   The cut threshold of 3.8 will result in 4 clusters. Evaluate its performance using the Rand Index. You will find that agglomerative clustering using the complete metric performs much worse than K-means clustering when forced to return 4 clusters. Next, vary the threshold (decreasing its value will result in more clusters, which in this case will help obtain a higher RI) and report the best RI that you find (also report the corresponding threshold and the number of clusters).

7. We now want to examine sub-clusters within a food group. Let's take the `Cereal-Grain-Pasta` food group. Apply K-means using number of clusters $k = 5$, 10, 25, 50, 75. Report the largest cluster size for each k and display 10 randomly sampled food items from each cluster. If the largest cluster size is less than 10, display all the food items in that cluster. As k increases, comment on the size of the largest cluster and the uniformity of food items in it.