Machine Learning and Computational Statistics, Spring 2014

Problem Set 5: PCA and Clustering

Due: Tuesday, April 15 at 3pm (sent to jj1192@nyu.edu)

Important: *See problem set policy on the course web site.* You must show **all** of your work and be rigorous in your writeups to obtain full credit. Your answers to the below questions, including plots and all code that you write for this assignment, should be e-mailed as a PDF document to Mick Jermsurawong.

Food Clustering – US Department of Agriculture Nutrient Database

In this problem set we will explore some properties of Principal Components Analysis (PCA). In addition, we will explore unsupervised learning with K-means and Hierarchical clustering. You may use the programming language of your choice, but we strongly encourage you to use `Matlab/Octave` or `Python`. For these languages, we provide a function to evaluate the quality of the learned clusters and a code sample for plotting dendograms. For `Python` users, please refer to `ps5SkeletonCode.py`. For `Matlab` users, please refer to `ps5SkeletonCode.m` with `randIndex.m`.

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. Let's begin PCA analysis by first subtracting the mean from the food items to get the centralized data matrix X, which is of dimension $m = 1496$ (number of food items) by $p = 150$ (number of nutrients).

   Now compute the singular value decomposition (SVD) of the centralized food data such that $X = USV^T$. The columns of V are the principal components that define the new basis. Letting $W = US$, we notice that $W = USV^TV = XV$, where we used the fact that $V^TV = I$ since V is an orthonormal matrix. Thus, each row of W consists of the dimensionality-reduced data, i.e. the coefficients expressing the data using the principal component coordinate system.

3. Visualize the data by reconstructing using only the first 2 principal components. Relating this to the lecture slide p.9, the entry $W_{i,j}$ corresponds to z_j^i, and the jth column of V corresponds to u_j. Use different colors to plot the data for each food group, and be sure to include a legend in your plot. The projected data now seem to be enclosed in a pair of perpendicular lines. Particularly, fat and oil food items form one straight line, and fish and vegetable items almost form the other. Suppose these perpendicular lines are in the direction of the first two principal components. Examine the first and second principal components to find their five highest absolute weights and their corresponding nutrients. How do the findings help to explain the observation on the perpendicular formation?

4. Apply K-means clustering on the original data, with size of cluster $k = 4$. Visualize the result on the reduced dimensions (2-D). Use a different color for each cluster.

5. 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` and `randIndex.m` in `Matlab` are 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.

6. 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.

Both Python's `sklearn` and `Matlab`'s implementation of K-Means provide various schemes of initializing the centroids. In both implementations, one can specify the number of times to repeat the K-Means computation to pick the minimum from. In `sklearn` this is specified by parameter `init_t`, and in `Matlab` the corresponding parameter is `'replicates'`.

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: Both the `Matlab` and `sklearn` implementations return the value of the objective function: see `inertia_` in `sklearn`, or `sumd` in `Matlab`. 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.

(d) Plot/visualize the clusters in 2D, as done in previous sections – do this for the 2 clusterings corresponding to the minimum and maximum values of the objective function.

7. 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, `Matlab`'s `dendogram` function takes a hierarchical cluster tree as input. We can generate our agglomerative hierarchical cluster tree using the `linkage` function with the `'complete'` metric. Similarly, `SciPy` has a `dendogram` function that takes as input the linkage matrix of a hierarchical clustering. We can also generate this using the `linkage` function and specifying that it use the `'complete'` metric. We use default Euclidean distance for both. Please refer to the sample code for dendogram plotting.

8. 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. `Matlab` and `Scipy` have functions `cluster` and `fcluster`, respectively, which take as input the linkage (dendogram) and the `'distance'` threshold on this Y-axis and uses it to cut the tree.

   For `SciPy/Matlab` implementation, the cut threshold of 3.8 will result in 4 clusters. Visualize the result on the dimensionality reduced (2D) space, and 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).

9. 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.

10. Suppose that we are performing clustering for the purpose of constructing features to use later within supervised learning. We want additional features describing membership values, i.e. of whether a data point belongs to each of the emerged clusters. Such an approach is one way to getting non-linear classifiers. For example, suppose we are trying to predict whether a 4-year old child will like a particular food item, e.g. a potato. If we knew the food category (e.g. vegetable, fats-oils), we could describe this potato further by adding 4 more attributes, for whether it belongs to each of the 4 food categories. This could be really helpful as the learning task may now be linearly separable with respect to the new feature vector. When the food categories are not known, performing clustering and using cluster membership for the additional attributes serves a similar role.

    Why might using a 0/1 assignment resulting from K-means be problematic? Could it be that a certain type of potato is more nutritionally similar to some pasta than to roots like carrots? This will motivate probabilistic approaches to clustering, such as Gaussian mixture models.

11. Let's now try to get a better understanding of PCA that we used for visualization.

    Recall that the principal component can also be found by decomposing the sample covariance matrix, that is, $\Lambda = \frac{1}{m}X^TX = \frac{1}{m}VSU^TUSV^T = V\frac{S^2}{m}V^T$. The relation between the singular values σ_i given along the diagonal of S (found using SVD on X) and the eigenvalues λ_i (of the sample covariance matrix Λ) is thus given by $\sigma_i^2/m = \lambda_i$.

    The total variance in a data set is defined as the sum of the variances of the individual components. Assuming that the data has aleady had its mean subtracted, we have that the total variance of the data in the original basis is given by

$$\text{TotalVar} = \sum_{j=1}^{p} \text{Var}(X_{\cdot j}) = \sum_{j=1}^{p} \frac{1}{m}\sum_{i=1}^{m} X_{ij}^2 = \frac{1}{m}\sum_{i,j} X_{ij}^2 = \sum_j \lambda_j,$$

    where in the last step we used the fact that[1]

$$\sum_{i,j} X_{ij}^2 = \text{trace}(X^TX) = \text{trace}(VS^2V^T) = \text{trace}(V^TVS^2) = \text{trace}(S^2) = m\sum_j \lambda_j.$$

    We next notice that the variance of the projected data given by the k'th principal component v_k (i.e. the k'th column of V) is given by the corresponding eigenvalue λ_k:

$$\frac{1}{m}(Xv_k)^T(Xv_k) = \frac{1}{m}v_k^TX^TXv_k = \frac{1}{m}v_k^T(X^TXv_k) = \frac{1}{m}(v_k^Tm\lambda_kv_k) = \lambda_k.$$

    Thus, we can calculate the proportion of variance explained by the i'th principal component as $\lambda_i/\sum_j \lambda_j = \sigma_i^2/\sum_j \sigma_j^2$.

    Plot the proportion of variance explained by the first 10 principal components.

12. ***Extra credit.*** Reconstruct all the food items using $k = 5, 10, 25, 50, 75, 100, 149$ principal components. You can reconstruct all the food items at once using the formula

$$R_k = W_kV_k^T + \text{mean food items},$$

    where k is the number of principal components to use and the general notation M_k refers to the submatrix of M obtained by using the first k columns of M. The reconstruction error using the first k principal components is defined as $\sum_{i,j}(\tilde{X}_{ij} - [R_k]_{ij})^2$, where $\tilde{X}$ is the original, uncentralized, data matrix.

    For $k = 1, 2, \ldots, p$ compute the sum of the last $p - k$ squares of singular values, i.e. $E_k = \sum_{j=k+1}^{p} \sigma_j^2$.

    Plot (a) the reconstruction error as a function of k and (b) E_k as a function of k. Comment on the relationship between the two plots.

[1] $\text{trace}(M) := \sum_i M_{ii}$ is the sum of the diagonal entries of a square matrix, which has the property that $\text{trace}(AB) = \text{trace}(BA)$. $(X^TX)_{jj} = \sum_i (X^T)_{ji}(X)_{ij} = \sum_i X_{ij}^2$, and thus $\text{trace}(X^TX) = \sum_j (X^TX)_{jj} = \sum_{ij} X_{ij}^2$.