Homework 5 - PCA, SVM & Clustering

Harvard CS109B, Spring 2017

Mar 2017

Problem 1: Face recoginition

In this problem, the task is to build a facial recognition system using Principal Components Analysis (PCA) and a Support Vector Machine (SVM). We provide you with a collection of grayscale face images of three political personalities "George W. Bush", "Hugo Chavez" and "Ariel Sharon", divided into training and test data sets. Each face image is of size 250×250, and is flattened into a vector of length 62500. All the data for this problem is located in the file `CS109b-hw5-dataset_1.Rdata`. You can read this file using the `load()` function, which will load four new variables into your environment. The vectorized images are available as rows in the arrays `imgs_train` and `imgs_test`. The identity of the person in each image is provided in the vectors `labels_train` and `labels_test`. The goal is to fit a face detection model to the training set, and evaluate its classification accuracy (i.e. fraction of face images which were recognized correctly) on the test set.

One way to perform face recognition is to treat each pixel in an image as a predictor, and fit a classifier to predict the identity of the person in the image. Do you foresee a problem with this approach?

Instead we recommend working with low-dimensional representations of the face images computed using PCA. This can be done by calculating the top K principal components (PCs) for the vectorized face images in the training set, projecting each training and test image onto the space spanned by the PC vectors, and represent each image using the K projected scores. The PC scores then serve as predictors for fitting a classification model. Why might this approach of fitting a classification model to lower dimensional representations of the images be more beneficial?

The following function takes a vectorized version of an image and plots the image in its original form:

```r
rot90 <- function(x, n = 1){
  #Rotates 90 degrees (counterclockwise)
  r90 <- function(x){
    y <- matrix(rep(NA, prod(dim(x))), nrow = nrow(x))
    for(i in seq_len(nrow(x))) y[, i] <- rev(x[i, ])
    y
  }
  for(i in seq_len(n)) x <- r90(x)
  return(x)
}
plot.face = function(x,zlim=c(-1,1)) {
  #Plots Face given image vector x
  x = pmin(pmax(x,zlim[1]),zlim[2])
  cols = gray.colors(100)[100:1]
  image(rot90(matrix(x,nrow=250)[,250:1],3),col=cols,
        zlim=zlim,axes=FALSE)
}
```

- Apply PCA to the face images in `imgs_train`, and identify the top 5 principal components. Each PC has the same dimensions as a vectorized face image in the training set, and can be reshaped into a 250 x 250 image, referred to as an *Eigenface*. Use the code above to visualize the Eigenfaces, and comment on what they convey. (*Hint*: for better visualization, we recommend that you re-scale the PC vectors before applying the above code; e.g. multiplying the PC vectors by 500 results in good visualization)

- Retain the top PCs that contribute to 90% of the variation in the training data. How does the number of identified PCs compare with the total number of pixels in an image? Compute the PC scores for each image in the training and test set, by projecting it onto the space spanned by the PC vectors.
- Treating the PC scores as predictors, fit a SVM model to the the training set, and report the classification accuracy of the model on the test set. How does the accuracy of the fitted model compare to a naïve classifier that predicts a random label for each image?

Hint: You may use the function `prcomp` to perform PCA and `pr$rotation` attribute to obtain the loading vectors. The variance captured by each principal component can be computed using the `pr$sdev` attribute.

Problem 2: Analyzing Voting Patterns of US States

In this problem, we shall use unsupervised learning techniques to analyze voting patterns of US states in six presidential elections. The data set for the problem is provided in the file `CS109b-hw5-dataset_2.txt`. Each row represents a state in the US, and contains the logit of the relative fraction of votes cast by the states for Democratic presidential candidates (against the Republican candidates) in elections from 1960 to 1980. The logit transformation was used to expand the scale of proportions (which stay between 0 and 1) to an unrestricted scale which has more reliable behavior when finding pairwise Euclidean distances. Each state is therefore described by 6 features (years). The goal is to find subgroups of states with similar voting patterns.

You will need the `cluster`, `factoextra`, `mclust`, `corrplot`, `dbscan`, `MASS`, `ggplot2`, `ggfortify` and `NbClust` libraries for this problem.

Part 2a: Visualize the data

Generate the following visualizations to analyze important characteristics of the data set:

- Rescale the data, and compute the Euclidean distance between each pair of states. Generate a heat map of the pair-wise distances (*Hint:* use the `daisy` and `fviz_dist` functions).
- Apply multi-dimensional scaling to the pair-wise distances, and generate a scatter plot of the states in two dimension (*Hint*: use the `cmdscale` function).
- Apply PCA to the data, and generate a scatter plot of the states using the first two principal components (*Hint:* use the `prcomp` function). Add a 2d-density estimation overlay to the plot via the `geom_density2d` function.

Summarize the results of these visualizations. What can you say about the similarities and differences among the states with regard to voting patterns? By visual inspection, into how many groups do the states cluster?

Part 2b: Partitioning clustering

Apply the following partitioning clustering algorithms to the data:

- **K-means clustering** (*Hint:* use the `kmeans` function)
- **Partitioning around medoids (PAM)** (*Hint:* use the `pam` function)

In each case, determine the optimal number of clusters based on the Gap statistic, considering 2 to 10 clusters (*Hint:* use the `clusGap` function). Also determine the choice of the optimal number of clusters by producing elbow plots (*Hint:* use `fviz_nbclust`). Finally, determine the optimal number of clusters using the method of average silhouette widths (*Hint:* use `fviz_nbclust` with argument `method="silhouette"`). Do the choices of these three methods agree? If not, why do you think you are obtaining different suggested numbers of clusters?

With your choice of the number of clusters, construct a principal components plot the clusters for *K-means* and *PAM* using the `fviz_cluster` function. Are the clusterings the same? Summarize the results of the clustering including any striking features of the clusterings.

Generate silhouette plots for the *K-means* and *PAM* clusterings with the optimal number of clusters. Identify states that may have been placed in the wrong cluster (*Hint:* use the `fviz_silhouette` function).

Part 2c: Hierarchical clustering

Apply the following hierarchical clustering algorithms to the data:

- **Agglomerative clustering** with Ward's method (*Hint*: use the `agnes` function)
- **Divisive clustering** (*Hint*: use the `diana` function)

In each case, summarize the results using a dendogram. (*Hint:* use the `pltree` function in the `cluster` library to plot the dendograms, and the `cutree` function to derive cluster groups from hierarchical clustering model). Determine the optimal number of clusters using Gap statistic, and add rectangles to the dendrograms sectioning off clusters (*Hint:* use `rect.hclust`). Do you find that states that predominantly vote for Republicans (e.g., Wyoming, Idaho, Alaska, Utah, Alabama) are closer together in the hierarchy? What can you say about states that usually lean towards Democrats (e.g. Maryland, New York, Vermont, California, Massachusetts)? Comment on the quality of clustering using Silhouette diagnostic plots.

Hint: The following code will help you reformat the output of the `agnes` and `diana` functions in order to apply the presented methods to find the optimal number of clusters:

```r
agnes.reformat<-function(x, k){
# x: Data matrix or frame, k: Number of clusters
  x.agnes = agnes(x,method="ward",stand=T)
  x.cluster = list(cluster=cutree(x.agnes,k=k))
  return(x.cluster)
}

diana.reformat<-function(x, k){
# x: Data matrix or frame, k: Number of clusters
  x.diana = diana(x,stand=T)
  x.cluster = list(cluster=cutree(x.diana,k=k))
  return(x.cluster)
}
```

Based on your choice of the optimal number of clusters in each case, visualize the clusters using a principal components plot, and compare them with the clustering results in Part 2b.

Part 2d: Soft clustering

We now explore if soft clustering techniques can produce intuitive grouping. Apply the following methods to the data:

- **Fuzzy clustering** (*Hint:* use the `fanny` function)
- **Gaussian mixture model** (*Hint:* use the `Mclust` function)

For the fuzzy clustering, use the Gap statistic to choose the optimal number of clusters. For the Gaussian mixture model, use the internal tuning feature in `Mclust` to choose the optimal number of mixture components.

Summarize both sets of results using both a principal components plot, and a correlation plot of the cluster membership probabilities. Compare the results of the clusterings. Comment on the membership probabilities of the states. Do any states have membership probabilities approximately equal between clusters? For the fuzzy clustering, generate a silhouette diagnostic plot, and comment on the quality of clustering.

Hint: use the `membership` attribute to obtain the cluster membership probabilties from the cluster model, and the `corrplot` function to generate a correlation plot.

Part 2e: Density-based clustering

Apply DBSCAN to the data with `minPts = 5` (*Hint:* use the `dbscan` function). Create a knee plot (*Hint:* use the `kNNdistplot` function) to estimate `eps`. Summarize the results using a principal components plot, and comment on the clusters and outliers identified. How does the clustering produced by DBSCAN compare to the previous methods?