

CS 294 Homework 2 Solution

adapted from Zeyu Li's solution

Problem 1: K-means

(a) Two dimensional data from four clusters are simulated. The 4 clusters have the same covariance matrix: $C = [0.5, 0; 0, 0.5]$ and different means, see Figure 1. We try different K s and plot K (ranging from 2 to 6) against within cluster sum of squares, see Figure 2. We can see that $K = 4$ is a good choice. The reason is based on "gap" statistics: the cost function falls substantially when $K \leq 4$ and afterwards not a lot more.

Note: the gap statistic is different from using the knee of the plot of the sum-of-squares though the ideas are similar.

We run K-means function with $K = 4$, the clustering result is shown in Figure 3. Comparing with Figure 1, only one point is clustered incorrectly. The sample covariance matrix of the whole data set is:

$$C = \begin{pmatrix} 20.8054 & 0.053 \\ 0.0523 & 0.744 \end{pmatrix}$$

(b) Given a data matrix $X \in \mathbb{R}^{d \times n}$, we standardize the data set to make sure its means =0 and covariance matrix is identity matrix. The second step can be implemented by SVD or QR decomposition of the sample covariance matrix C . We plot K against within cluster sum of squares on standardized data, see Figure 4. The optimal K we choose is 3, and the K-means clustering result is shown in Figure 5.

(c) Comparing the clustering results for un-standardized and standardized data, see Figure 3 and 5, the standardization in this setting does not work. The reason is the following: the four simulated clusters mainly differ in the locations of the centers along x-axis. However the standardization procedure will force the whole data set to distribute around a unit circle, which suppress the difference among the four clusters along x-axis.

Problem 2: GMM

(a) We use two data sets from problem 1 as our testing data. The covariance matrices are: $C_1 = [0.5, 0; 0, 0.5]$ and $C_2 = [5, 0; 0, 5]$. The clustering results using "Mclust" are shown in Figure 6 and 7, respectively. For the second data set, Mclust function chose the wrong cluster number (K).

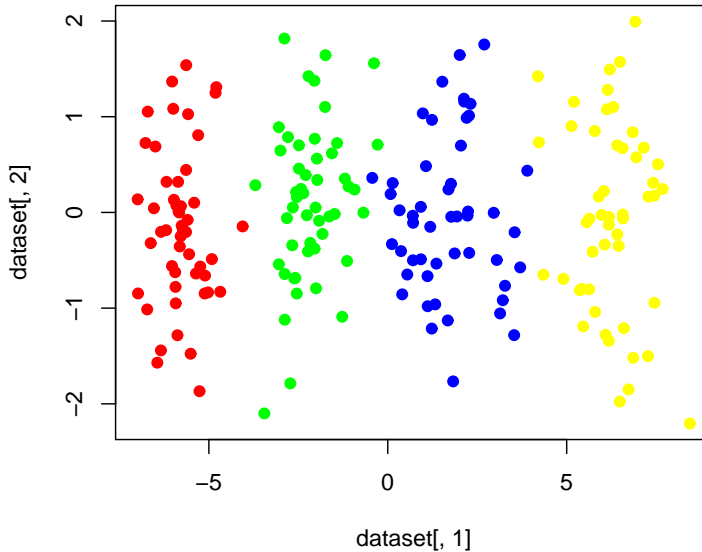


Figure 1: Simulated Data

(b) Intuitively, GMM is a soft assignment clustering method. The responsibility r_{ik} for each data point i represents the probability that x_i belongs to the k -th cluster. When $\sigma \rightarrow 0$, GMM becomes a hard assignment method. That is, given x_i , r_{ik} will be equal to one for the closest cluster, and will be zeros for the other clusters.

Problem 3: Hierarchical Clustering

(a) The hierarchical clustering algorithms with single linkage and complete linkage differ in the way how they measure the dissimilarity between two groups. The single linkage method uses:

$$s_{i,j} = \min\{\|x - y\|^2 \mid x \in \text{cluster } i, y \in \text{cluster } j\}$$

and the complete linkage method uses

$$s_{i,j} = \max\{\|x - y\|^2 \mid x \in \text{cluster } i, y \in \text{cluster } j\}$$

Therefore, the single linkage is suitable to detect chains or elongated structures and the complete linkage tends to yields round cluster and more compact results.

A simple example showing the difference: suppose we have 4 points placed on a line at coordinates: 0,1,3,5.5. Single linkage clusters: ((0,1),3),5.5). Complete linkage clusters: ((0,1),(3,5.5)). See Figure 8 and 9 for another example.

(b) To cluster counties such that counties with similar voting behavior should be most likely to be in the same cluster, we apply "hclust" on the data using "ave" linkage.

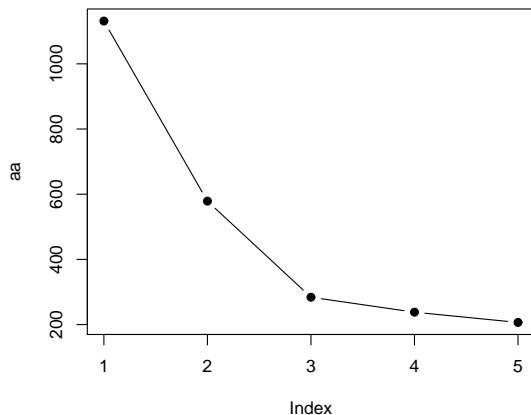


Figure 2: Within-cluster sum of squares versus K

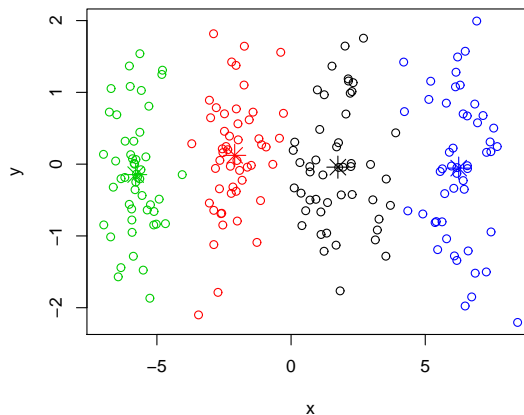


Figure 3: K-means clustering result using $K = 4$

The dendrogram is shown in Figure 10. The Los Angeles county is so different from other counties because the population in Los Angeles is far larger than that in the other counties. At each stage when we join the two most similar clusters with the smallest between-cluster dissimilarity(”average” linkage in this case), Los Angeles is very hard to be merged. Due to the same reason, the dendrogram looks like one using single linkage.

(c) To avoid population influence, we normalize the data by converting the absolute vote counts into relative percentages of a total vote within a country. The normalization code is:

$$\text{data}\$Bush = Bush / (Bush + Kerry), \quad \text{data}\$Kerry = Kerry / (Bush + Kerry)$$

and the clustering result using ”average”, ”single” and ”complete” linkage are shown in Figure 11,12,13,respectively.

Problem 4: PCA for visualization

(a) “house-votes-84.dat” contains 435 instances and each example has 16 features. We first center the data $X \in R^{16 \times 435}$ and then perform eigen-decomposition on the covariance matrix $C = \frac{1}{n}XX^T$ (or directly using SVD on X), Figure 14 shows the distribution of the eigenvalues (variances). We can see that the first eigenvalue accounts for most of the variance (about 47%).

We project the data points onto the first two principal components, and scatter plot the 435 instances in Figure 15, where the Republican is represented as ”green” and the Democrat is coded with ”red”.

These two classes are somewhat separated, which is pretty cool given that the PCA algorithm does not know anything about the labels. An algorithm that explicitly uses the labels would naturally do better, for example, Fisher discriminant analysis.

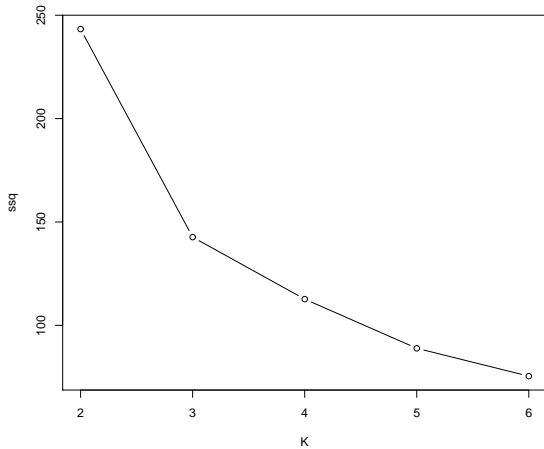


Figure 4: Within-cluster sum of squares versus K (for standardized data)

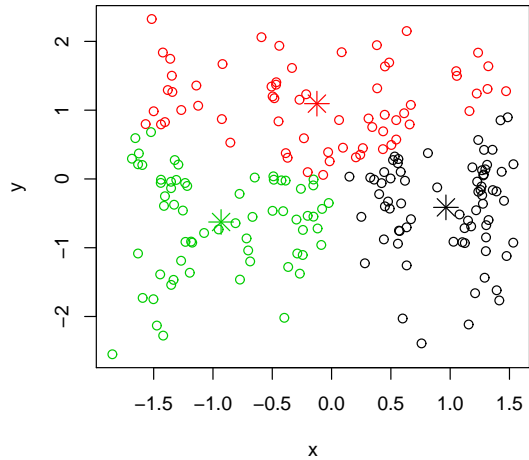


Figure 5: K-means clustering result using $K = 3$ (for standardized data)

(b) Given a data matrix $X \in \mathbb{R}^{d \times n}$, suppose its SVD can be written as: $X = USV^T$, then $X^T = VSU^T$. We denote

$$A = XX^T, \quad B = X^T X$$

therefore $A = US^2U^T \in \mathbb{R}^{d \times d}$ and $B = VS^2V^T \in \mathbb{R}^{n \times n}$. The matrix A is called covariance matrix and matrix B is usually called gram matrix, they share the same eigenvalues. Also note $\text{rank}(B) = d$, its eigenvalues $\{s_{d+1}, \dots, s_n\}$ will be all zeros. We run PCA on X^T and project X^T onto the linear space spanned by the first two principal components, see Figure 16. This figure shows how the variables are correlated with others.

(c) We use the "iris" dataset from UCI machine learning repository. "iris" contains 3 classes, 150 instances and 4 features. Figure 17 shows the distribution of the eigenvalues, and Figure 18 shows the data distribution after projecting all examples onto the 2D eigen-space. Using the simple PCA, we can roughly know the distribution of our high-dimensional data, which will guide us to design more complicated machine learning algorithms.

Problem 5: PCA for classification

Given a data matrix $X \in \mathbb{R}^{d \times n}$, we perform PCA on X . For an vector $x \in \mathbb{R}^d$, the reconstructed error can be measured as $e = \|x - \tilde{U}\tilde{U}^T x\|_2^2$, where \tilde{U} is the first k eigenvectors.

Different strategies can be used to apply PCA for a classification problem. In problem 5, we construct the eigen-space using all the available data, including training and testing examples. After projecting all of the examples onto this eigen-space, we use nearest-neighbor classifier to do the classification.

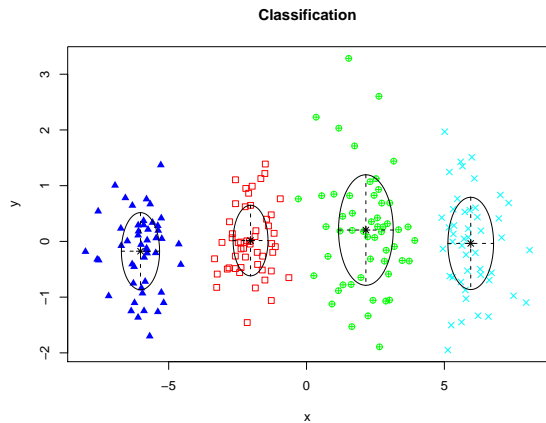


Figure 6: clustering results for data set 1 using Mclust

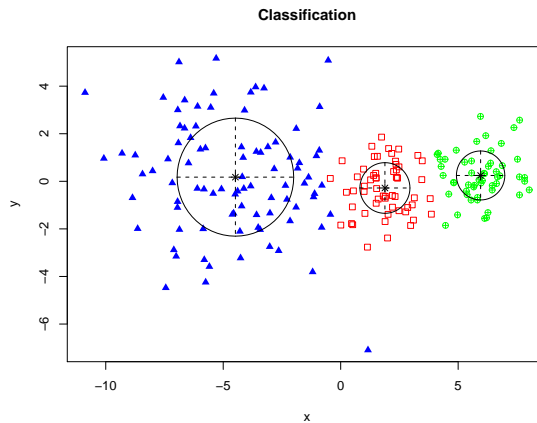


Figure 7: clustering results for data set 2 using Mclust

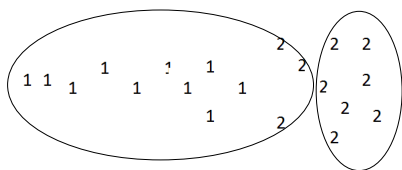


Figure 8: Hierarchical Clustering using single linkage

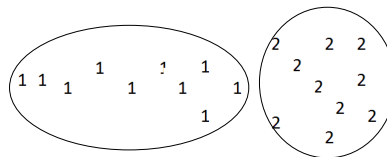


Figure 9: Hierarchical Clustering using complete linkage

(a) We run PCA on the positive training examples (trainPos), the distribution of the eigenvalues is shown in Figure 19. We can see that the first 3 eigenvalues account for 72.5% of the total variances.

Given an image $x = \text{trainPos}[4] \in \mathbb{R}^{361}$, we plot the reconstruction error VS k (number of eigenvectors we are using), see Figure 20. The reconstruction error is 3.51899 at $K = 5$, and 0.3919702 at $K = 50$. That is, the error drops dramatically from $K = 1$ to $K = 50$.

The reconstruction error can be represented as: $e = \|x - \tilde{U}\tilde{U}^T x\|_2^2$. When we use all of the eigenvectors, we have $UU^T = I$ and $e = 0$. That is, the reconstruction error for any example will be guaranteed to reach zero.

(b) Similarly, given an testing example $x = \text{testPos}[4]$, we plot the reconstruction error in Figure 21. Because we use trainPos to construct our PCA projection, we need more principal components to reconstruct the image in testPos in order to reach the same fixed error threshold.

Unlike the training error, the test error is not guaranteed to reach zero even if we keep all the principal components. In particular, if a test point lies outside the subspace spanned by the training examples (in other words, the test point cannot be expressed as a

CA 2004 Election Results by County

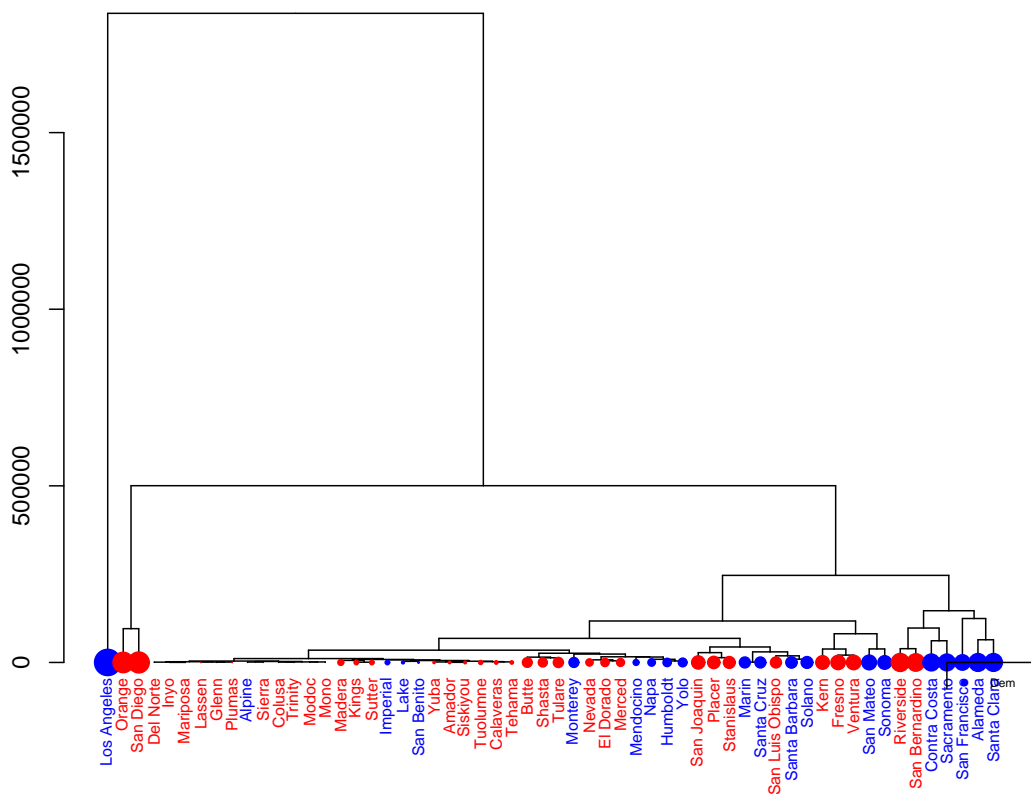


Figure 10: clustering on election 2004 data, un-normalized data, average linkage

linear combination of training examples), then that test point will always have a non-zero error. But in general, if the number of training examples is pretty large, then the test error will be decently small. In our case, the test error drops pretty close to zero.

(c) We perform PCA using all of the data, and project the training/testing examples onto the same eigen-space. The NN classifier is used to classify face versus non-face in this space. Figure 22 shows the test error as K increases. When K is bigger than 50, the decreasing of the testing error is very small. That is: it is unnecessary to include more than 50 principle components to do the classification in our experimental setting (bigger K causes more computational overhead).

CA 2004 Election Results by County

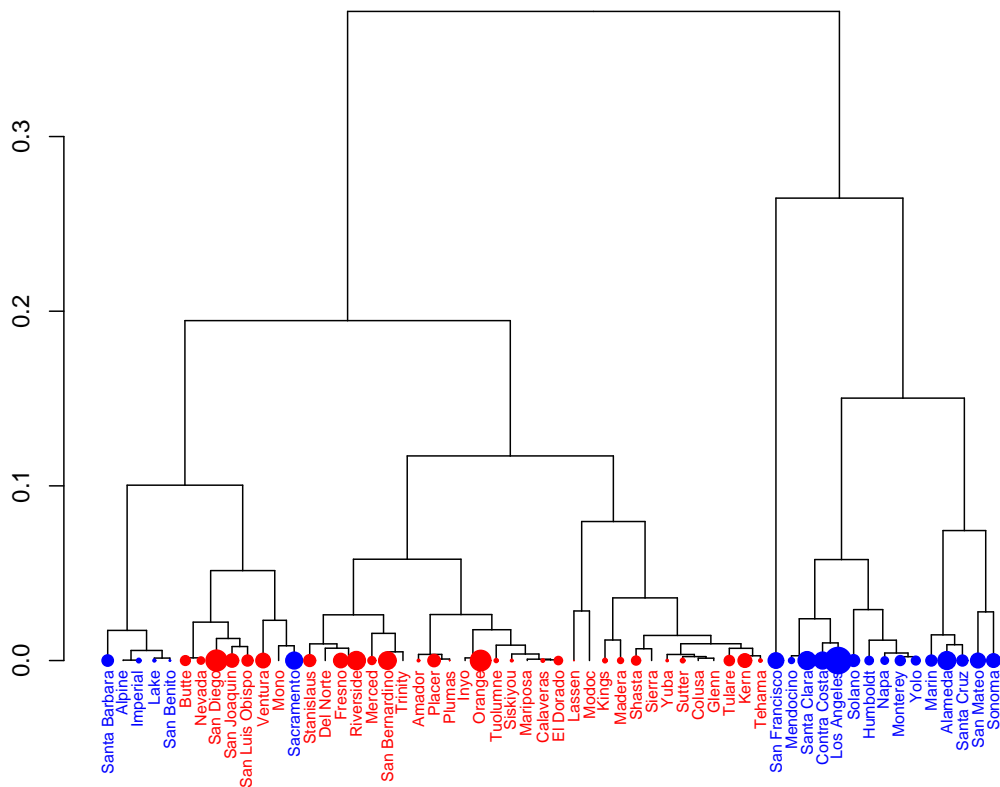


Figure 11: clustering on election 2004 data, normalized data, average linkage

CA 2004 Election Results by County

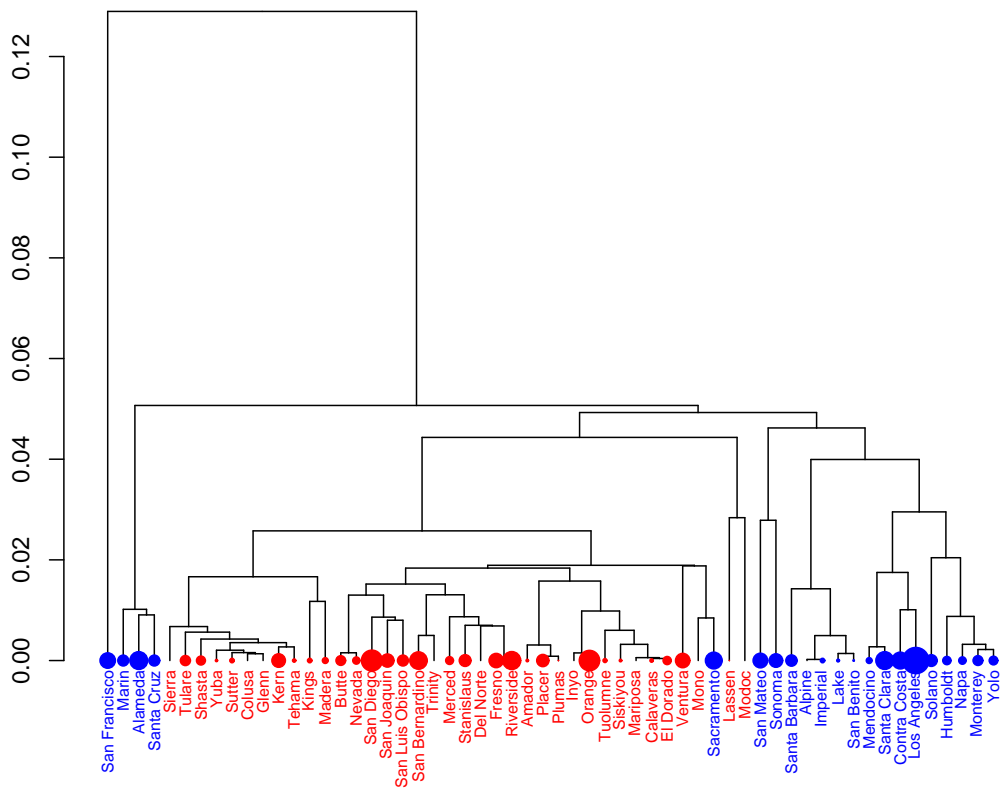


Figure 12: clustering on election 2004 data, normalized data, single linkage

CA 2004 Election Results by County

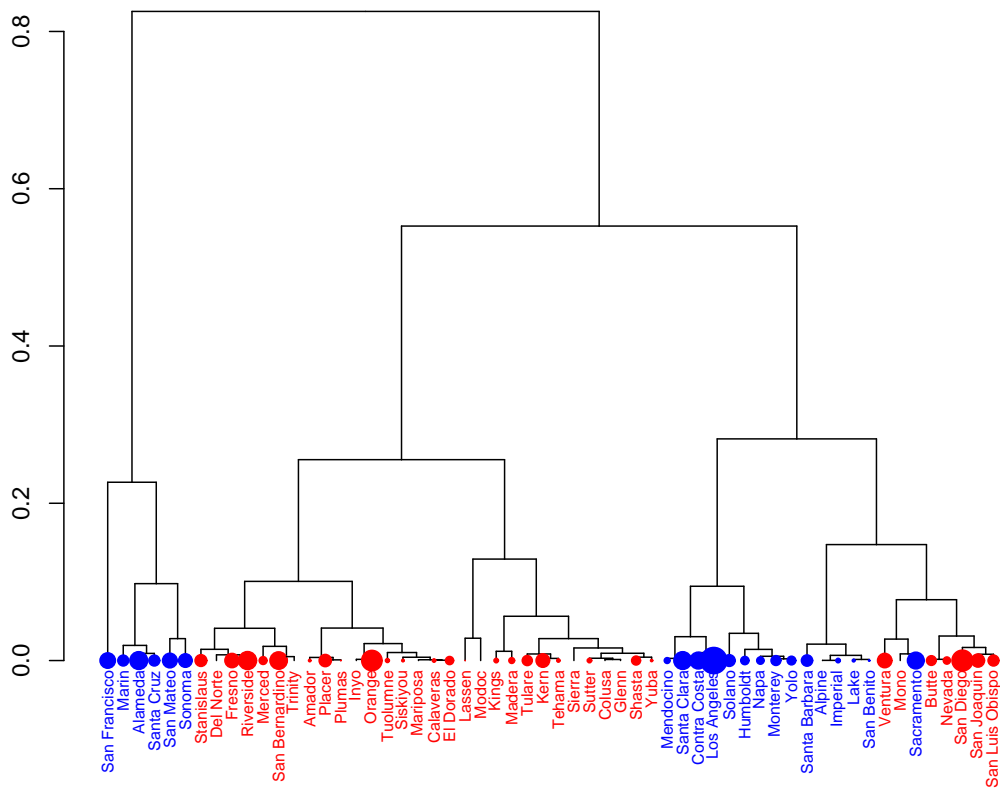


Figure 13: clustering on election 2004 data, normalized data, complete linkage

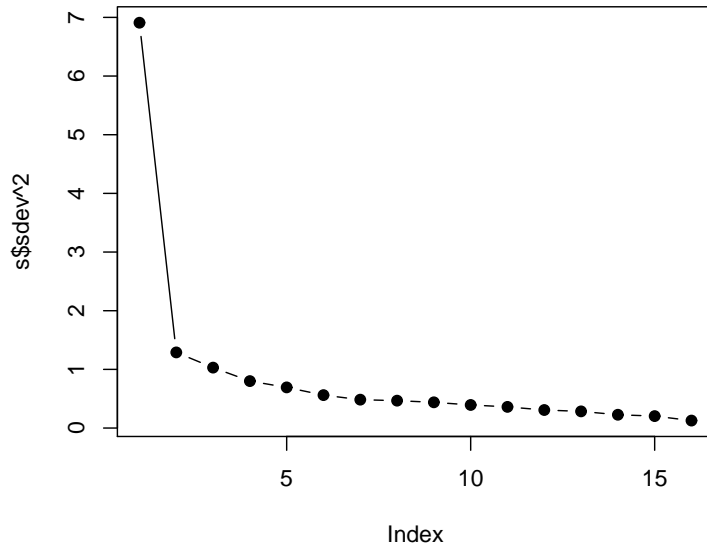


Figure 14: eigenvalues

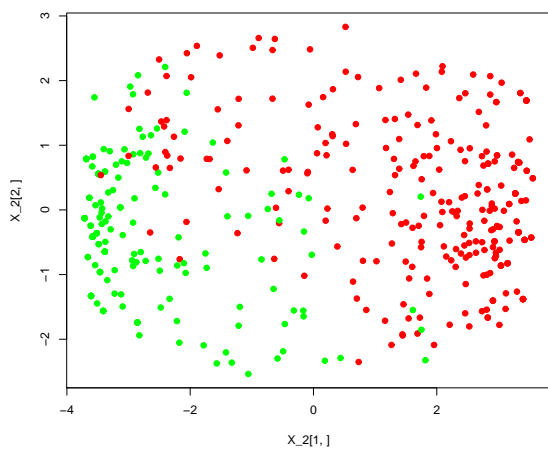


Figure 15: Projecting data X onto 2-dim eigen-space

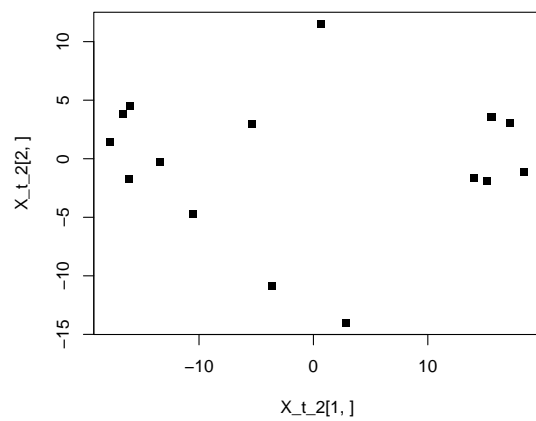


Figure 16: Projecting X^T onto 2-dim eigen-space

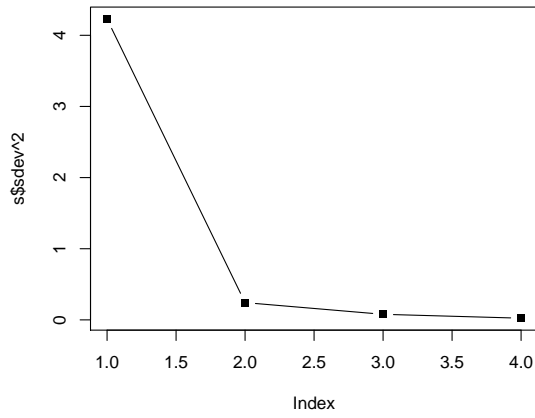


Figure 17: distribution of the eigenvalues for "iris" data

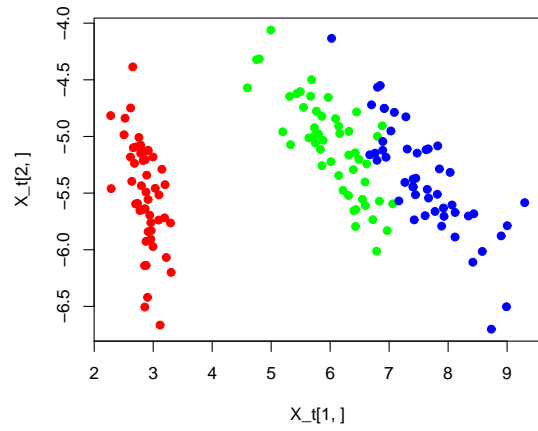


Figure 18: Data distribution in 2D space for "iris" data

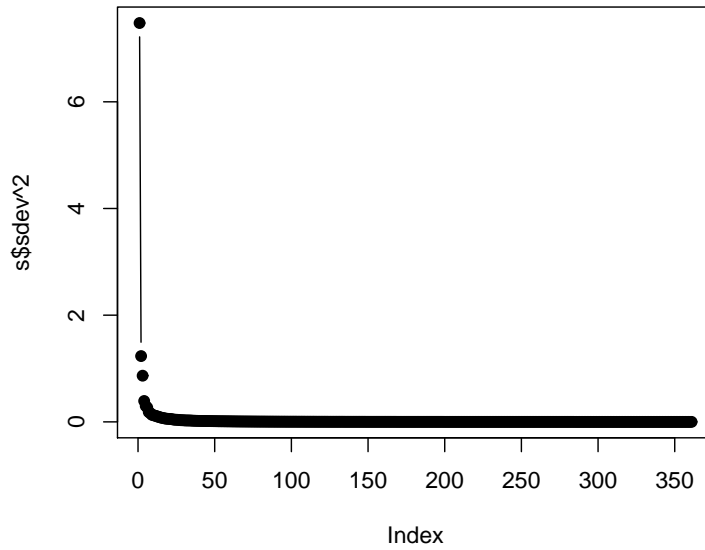


Figure 19: distribution of the eigenvalues (PCA on trainPos data set)

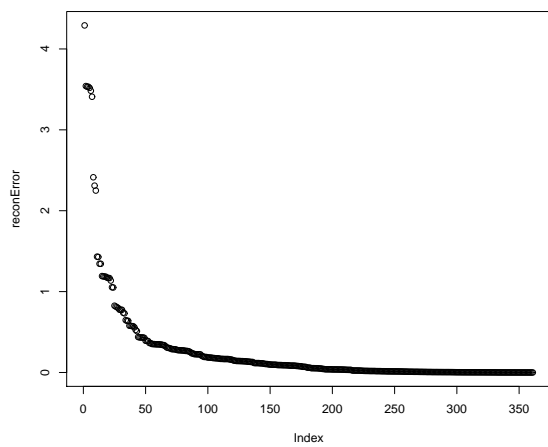


Figure 20: Reconstruction Error versus K (for a training example)

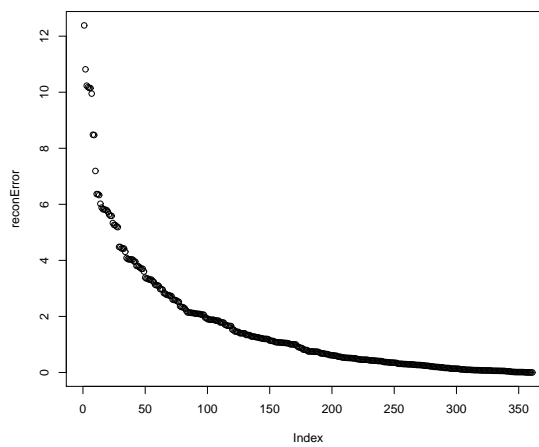


Figure 21: Reconstruction Error versus K (for a testing example)

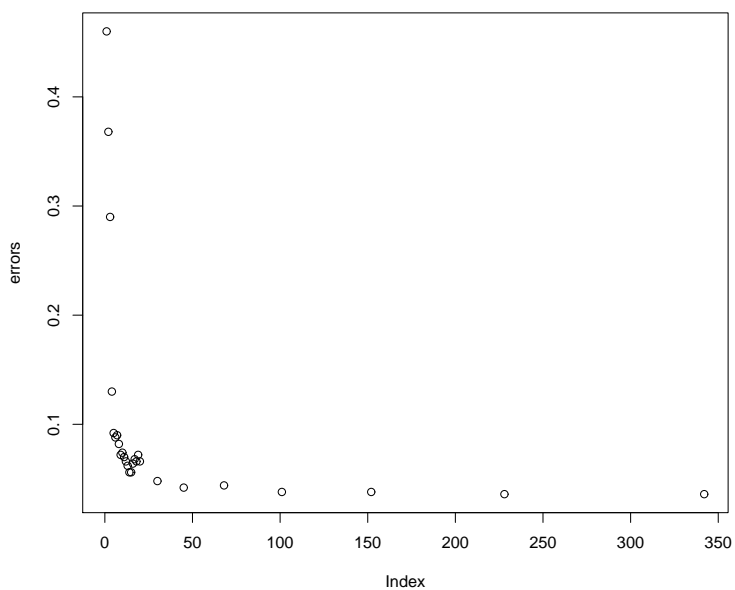


Figure 22: Test error(%) versus K