Machine Learning - Assignment 3

Christoffer Thrysøe - dfv107

December 18, 2016

1. Summarization by the mean

If we consider a set $S = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$, we want to prove that:

$$argmin_{b \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{b}||^2 \tag{1}$$

is given by the empirical mean:

$$\mathbf{b} = \bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x_i} \tag{2}$$

First we write out the inner term:

$$||\mathbf{x_i} - \mathbf{b}||^2 = \left(\sqrt{\mathbf{x}_i^2 - \mathbf{b}^2}\right)^2 = (\mathbf{x}_i - \mathbf{b})^2$$

Since we want to find the value of $\mathbf{b}$ which minimizes equation 1, we differentiate with respect to $\mathbf{b}$ and set the equation equal to zero:

$$\begin{aligned}
0 &= \frac{\partial}{\partial \mathbf{b}} \left[\frac{1}{N} \sum_{i=1}^{N} (\mathbf{x_i} - \mathbf{b})^2 \right] \Rightarrow \\
0 &= \frac{1}{N} \sum_{i=1}^{N} -2 (\mathbf{x_i} - \mathbf{b}) \Rightarrow \\
0 &= \frac{1}{N} \sum_{i=1}^{N} -2\mathbf{x_i} + 2\mathbf{b} \Rightarrow \\
\frac{2\mathbf{b}}{2} &= \frac{\frac{1}{N} \sum_{i=1}^{N} 2\mathbf{x_i}}{2} \Rightarrow \\
\mathbf{b} &= \frac{1}{N} \sum_{i=1}^{N} \mathbf{x_i}
\end{aligned}$$

Thus completing the proof that equation 1 is given by the empirical mean, showed in equation 2.

2. PCA for high dimensional data and small samples

The covariance matrix is defined as followed:

$$\mathbf{S} = \mathbf{X_0^T X_0} \tag{3}$$

we want to show that if $\mathbf{v}$ is an eigenvector of $\mathbf{X}_0\mathbf{X}_0^T$ with eigenvalue λ then $\mathbf{X}_0^T\mathbf{v}$ is an eigenvector of $\mathbf{S}$ with eigenvalue λ.
Given the above we can write the relationship between the matrix, eigenvalue and eigenvector as

$$\mathbf{X}_0\mathbf{X}_0^T\mathbf{v} = \lambda\mathbf{v} \tag{4}$$

If we multiply X_0^T from the left, and noting the definition of $\mathbf{S}$ we get the following:

$$\mathbf{X}_0^T\mathbf{X}_0\mathbf{X}_0^T\mathbf{v} = \mathbf{X}_0^T\lambda\mathbf{v} \Rightarrow \tag{5}$$

$$\mathbf{S}\mathbf{X}_0^T\mathbf{v} = \lambda\mathbf{X}_0^T\mathbf{v} \tag{6}$$

comparing equation 4 and 6 $\mathbf{X}_0^T\mathbf{v}$ is the eigenvector of $\mathbf{S}$ with eigenvalues λ, thus completing the proof.

3. The Traffic Sign Recognition Data

The implementations of the following excercises have been wrapped in a file called `main.py`. Calling the function will give the component to which 90 percent of the variance is explained, and save plots for figures 1 2 & 4. test

3.1 Data understanding and preprocessing

The plotting function is located in the file `understanding.py`.The program works by splitting the dataset into features, i.e. pictures consisting of 1568 pixel values, and target values, i.e. traffic sign class. The number of each traffic sign class is then accumulated and divided by the total number of signs, resulting in the frequency of each traffic sign. Figure 1 is a histogram showing the frequency of each class, i.e. traffic sign.

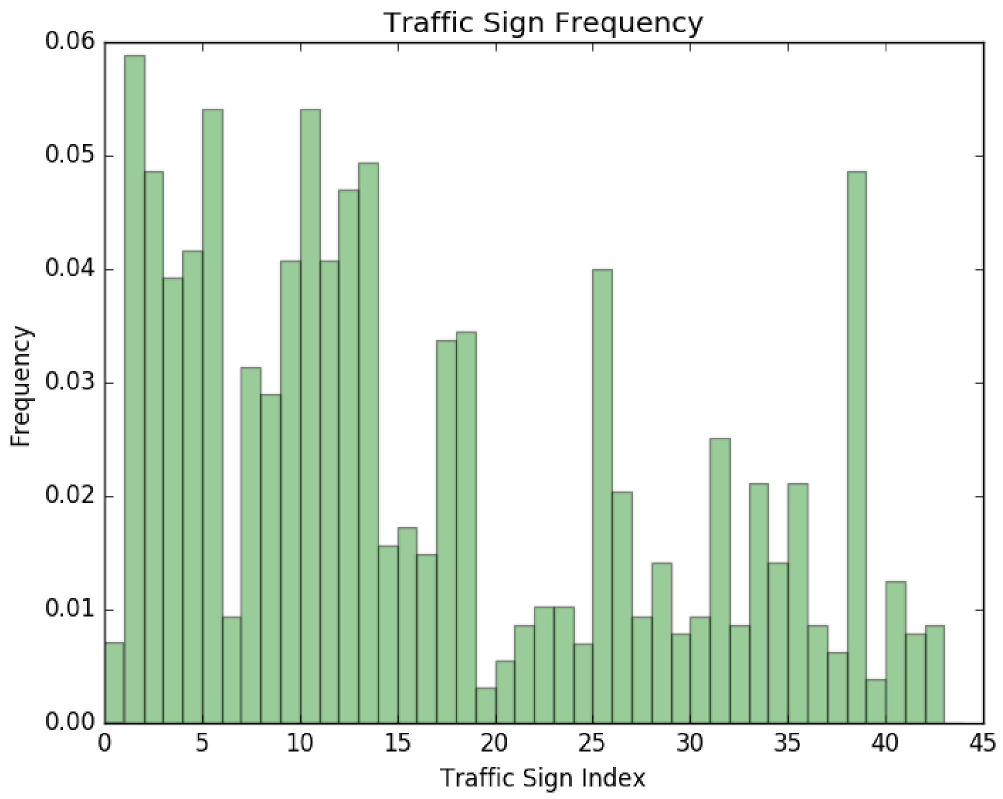

Figure 1: Histogram showing the frequency of each traffic sign from the dataset `ML2016TrafficSignsTrain.csv`

3.2. Principal component analysis

The implementation of the PCA algorithm is located in the file `PCA.py`. The algorithm performs the following steps:

1. Separate features and target values of each traffic sign
2. For each feature, calculate the mean and withdraw it from the data points
3. Compute the covariance matrix of the mean-less data
4. Compute eigenvalues and eigenvectors of the covariance matrix
5. Sort the eigenvectors based on the corresponding eigenvalue from largest to smallest
6. Take the top k (dimension to reduce to) eigenvectors. These correspond to the k principle components
7. Project the data into k dimensions by multiplying the principle components and the mean-less data

Eigenspectrum

To compute the eigenspectrum the eigenvalues have been listed in descending order, and plotted on a logarithmic scale as shown in figure 2.

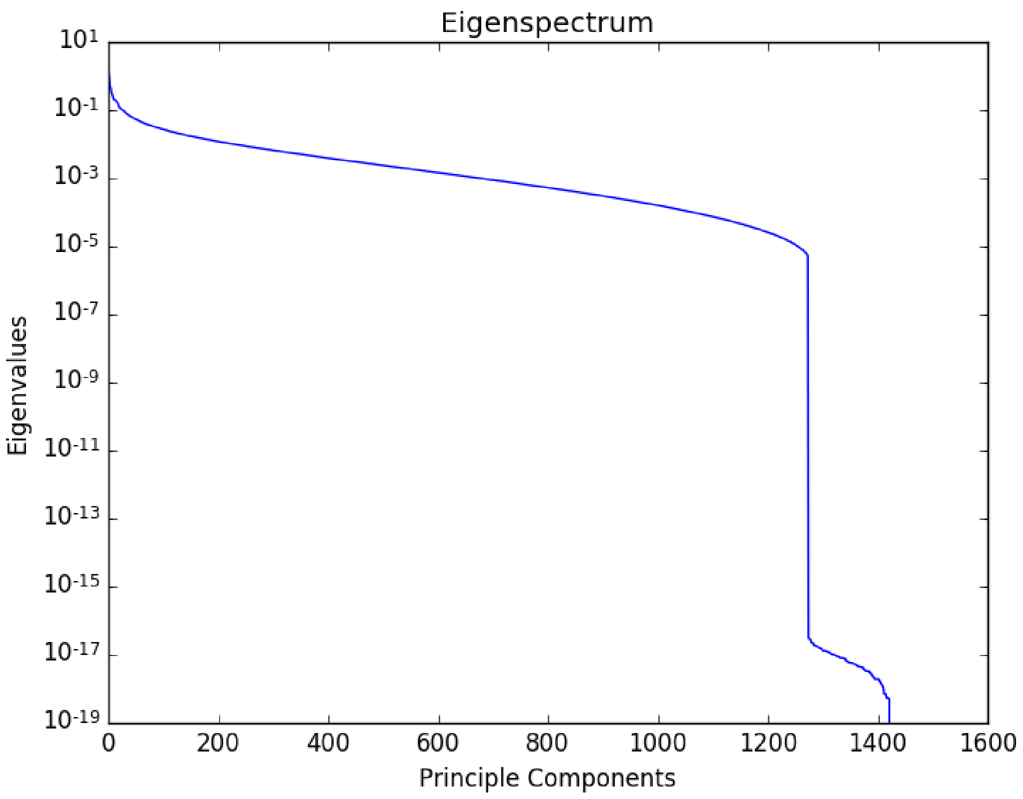

Figure 2: Histogram showing the frequency of each traffic sign from the dataset `ML2016TrafficSignsTrain.csv`

The number of components necessary to explain 90 % of the data was determined to be 228 components. From figure 2 it is evident that the variance drops quickly after a few components, then slowly decreases and for the last 200-300 components the variance plummets.

Projection onto two dimensions

The data has been projected down to two dimensions, by performing PCA with k = 2, that is multiplying the first two principle components onto the mean-less data. Figure 3 shows a scatter plot for the for the five different mappings from classes to shapes of the traffic signs.

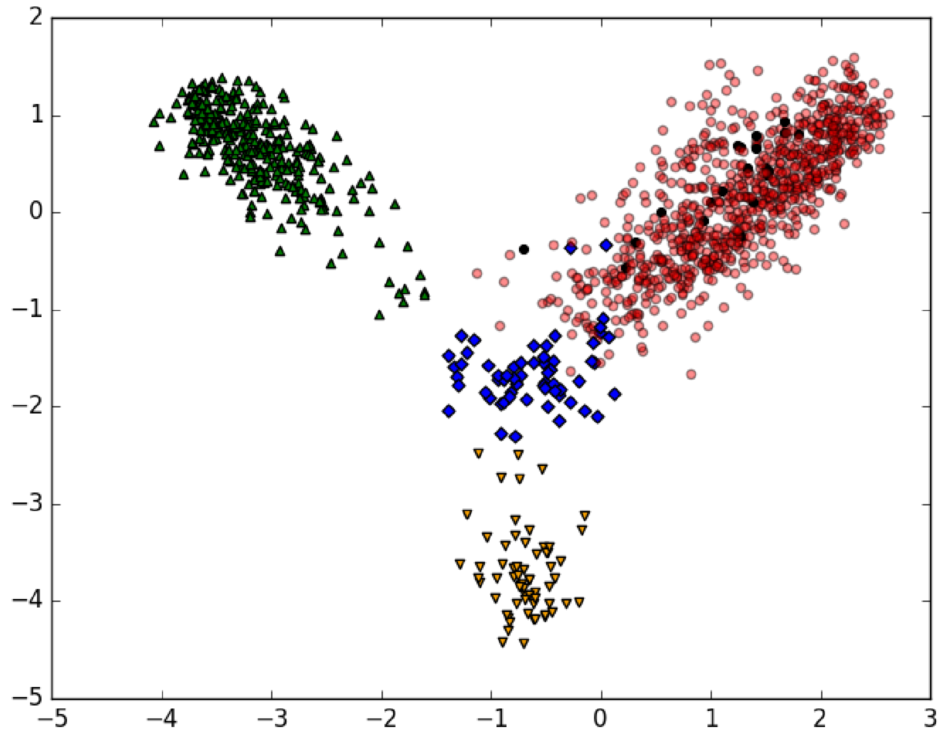

Figure 3: Scatter plot of the data `ML2016TrafficSignsTrain.csv` projected onto the two first principal components. each data marker defines a traffic sign shape

The below table shows the correlation between the shape of the traffic signs and the colour which they were plotted in. The shape of the marker corresponds to the shape of the traffic sign:

Shape	Color
Round	*Red*
Upwards pointing triangle	*Green*
Diamond	*Blue*
Downwards pointing triangle	*Yellow*
Octagon	*Black*

From figure 3 we can see that the projection onto the two first principal components, describes the data well, as the data is somewhat divided into four clusters. Two classes are not well differentiated, using the two first components, namely the round and octagon shaped signs. This is probably as the round and octagon shapes are very similar, and therefore described similar by the two principal components.

3.3. Clustering

The implementation of k-means clustering is included in the file `kmeans.py`. The algorithm works as followed:

1. k cluster points are initialized, with a given position in the data. For this assignment the data is initialized to have the same position as the first four data points.

2. For each data point in the input data, the point is assigned the cluster with the smallest euclidean distance.
3. For each cluster, the mean of the assigned points are calculated and the cluster position is then updated with the mean.
4. The two steps above are repeated until the clusters no longer move. The position of these clusters are then returned. (For this implementation the algorithm stops once the clusters doesn't move, although as the clusters may converge for infinity, this is bad practice, but i didn't experience this.

Figure 4 shows the cluster position projected onto the two first principle components, using the same projection calculation as assignment 3.2. The starting positions of the four clusters correspond to the first four data points. All clusters converged after 20 iterations.

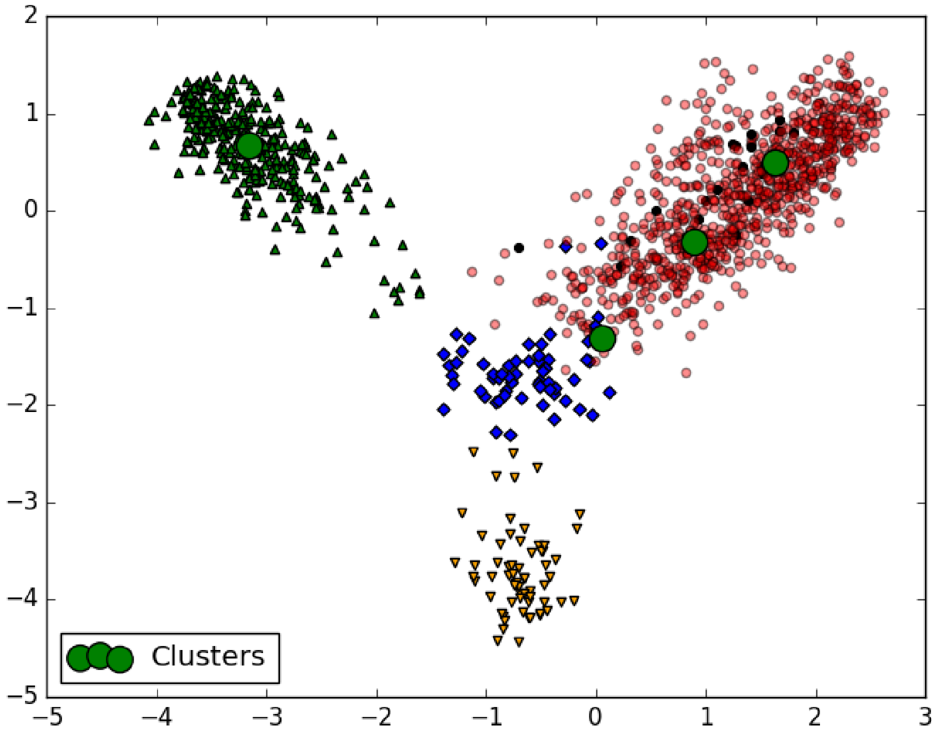

Figure 4: The result of running 4-clustering on the dataset `ML2016TrafficSignsTrain.csv`, where the position of the clusters (green dots) have been projected onto the first two principle components.

I also tried to perform the clustering on random data points. Figure 5, shows the result of doing so.

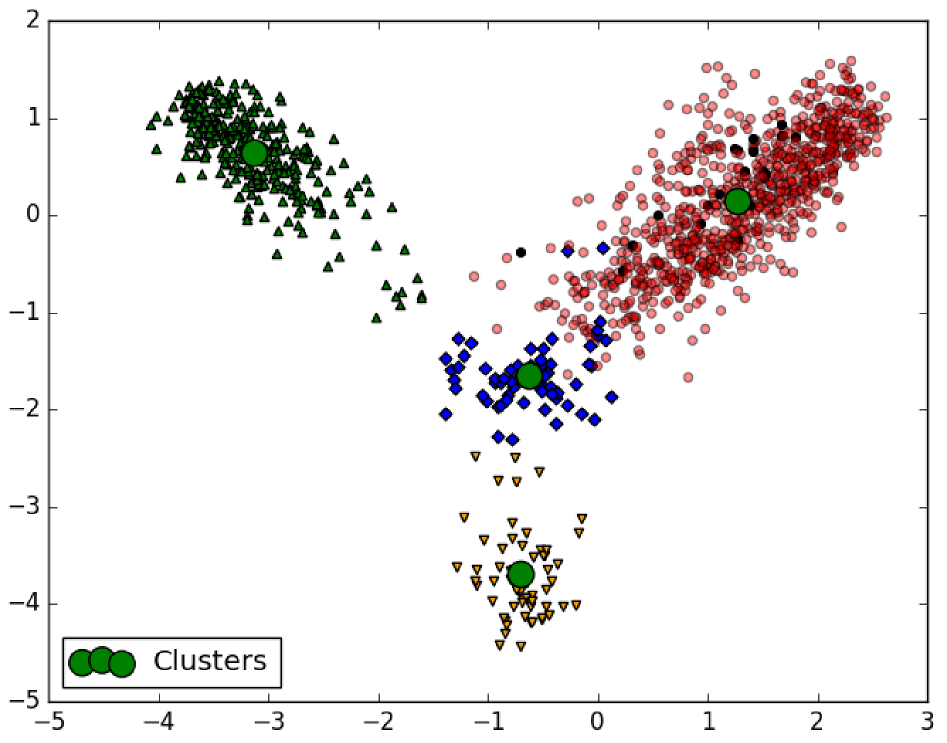

Figure 5: The result of running 4-clustering on the dataset `ML2016TrafficSignsTrain.csv`, with random starting points for the clusters.

In general I believe the clusters favoured the position corresponding to the rounded and upward triangle shaped traffic signs (red & green). Also more often than not, a cluster is positioned on the downward triangle shape (yellow). These observations could be the result that the shapes are represented by a lot of observations, thus influencing the mean greatly and thus dictating the cluster position, but also that these shapes are well divided into clusters and could therefore easily be grouped, therefore I believe i got meaningful clusters.