Solution for ML F16 assignment 3

Jonathan Hansen fdg890

December 11, 2016

1 Summarization by the mean

We begin by expanding the summation of the *argmin* expression:

$$\frac{1}{N}(\parallel x_1 - b \parallel^2 + \parallel x_2 - b \parallel^2 + \cdots + \parallel x_N - b \parallel^2)$$

We know that the norm of a vector can be found with the expression $\sqrt{u^T u}$. Using this we can rewrite the function that we have to minimize in the following manner:

$$\frac{1}{N}\Big((x_1 - b)^T(x_1 - b) + (x_2 - b)^T(x_2 - b) + \cdots + (x_N - b)^T(x_N - b)\Big)$$

Furhter expanding each dot product we can write it out as a sum of terms depending on the d'th element of the mean vector:

$$\frac{1}{N}\Bigg(\underbrace{\big((x_{11} - b_1)^2 + \cdots + (x_{1d} - b_d)^2\big)}_{(x_1-b)^T(x_1-b)} + \cdots + \underbrace{\big((x_{N1} - b_1)^2 + \cdots + (x_{Nd} - b_d)^2\big)}_{(x_N-b)^T(x_N-b)}\Bigg)$$

More expanding, this time of the sqaured terms, yields the following result. Notice that we will only write out the very first term to keep things readable:

$$\frac{1}{N}\left((x_{11}^2 + b_1^2 - 2x_{11}b_1 + \cdots + x_{1d}^2 + b_d^2 - 2x_{1d}b_d) + \cdots\right)$$

We can now find the minimum of this function of vector b by paritally deriving it with respect to b at setting it equal to zero. Notice that each partial derivative of the summation is solved independently. Thus the part that we expanded above can be solved as:

$$\left(\frac{2}{N}b_1 - \frac{2}{N}x_{11} + \cdots + \right)\left(\frac{2}{N}b_1 - \frac{2}{N}x_{N1} + \cdots + \right) = 0 \Rightarrow \frac{1}{N}\sum_{i=1}^{N} b_1 = \frac{1}{N}\sum_{i=1}^{N} x_{i1}$$

Proceding like this with all the elements of the initial summation we get a vector that can be used for optimizing and that is equal to b.

2 PCA for high dimensional data and small samples

To answer this question I draw on the lecture notes by Christian Igel handed out in the course "Databases and Data Mining" that I took on DIKU a few years back.

To see that if v is an eigenvector of the $N \times N$ matrix $\mathbf{X}_0\mathbf{X}_0^T$ with eigenvalue λ, then $\mathbf{X}_0^T v$ is a (not normalized) eigenvector of $\mathbf{S} = \mathbf{X}_0^T\mathbf{X}_0$ with the same eigenvalue, we start by multiplying

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

with $\mathbf{X}_0^T$ from the left. This yields

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

Thus we can consider the smaller eigenvalue problem of decomposing $\mathbf{X}_0\mathbf{X}_0^T$ if $N < d$.

3 The Traffic Sign Recognition Data

I have implemented the PCA and the clustering model in `Python3`. I have made substantial use of the libraries `Pandas`, `Numpy` and `MatPlotLib`. All the code is gathered in a single script called `pca.py`. Running it using "python3 pca.py" from the terminal produces the requested plots and reports required values.

Data understanding and preprocessing

Below in Figure 1 is a histogram showing how the distribution of the different sign types in the data. To make the plot I used the `hist` function that is part of the `MatPlotLib.PyPlot` library.

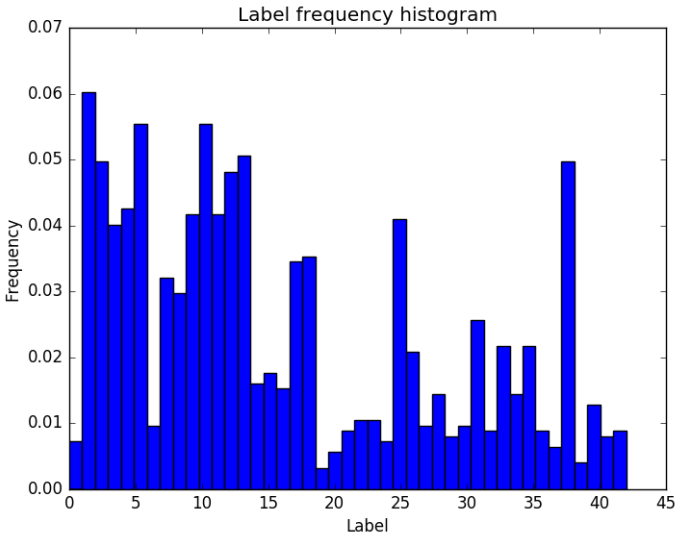

Figure 1: Histogram showing the distribution of class frequencies in traffic sign data.

Principal component analysis

Figure 2 shows the eigenspectrum of of the traffic sign data set. To compute this I used the `cov`, `transpose` and `eigh` functions of the `NumPy` library.

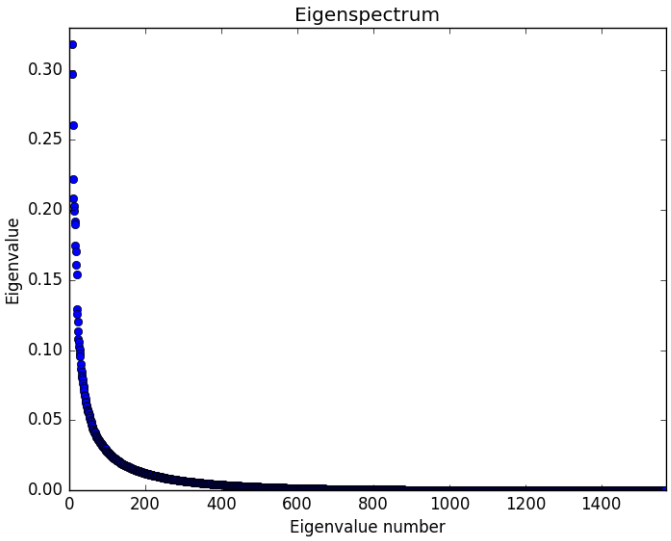

Figure 2: Plot of the eigenspectrum resulting in PCA on traffic sign data.

Running my PCA algorithm reveals that 90 % of the variance of the data can be explained using 228 components. These results can be seen in the table below. This is a decent results since the dimensions of the data can be reduced to one quarter and still give a chance to classify most of the traffic signs correctly.

Number of components of PCA necessary to explain 90 % of variance	
Number of components needed	228
Percentage of variance explained	90.02

Shifting the data using just the two principal components results in the data that is plotted in 3. In the scatterplot round signs are blue, upwards pointing triangles are red, diamond-shaped signs are green, downwards pointing triangles are yellow and octagons are magenta. The first four cluster up pretty nicely, but not suprisingly octagons and rounds are mixed. The difference between these two types of signs is not well described with just the two principal components.

Clustering

In Figure 4 is the same scatter plot as in the above figure, but added to it is the cluster centers acquired by running an iterative 4-means clustering algorithm on the traffic sign data. However the original cluster centers are of course points with the full set of dimensions that the traffic sign data has. In order plot them on the same scatter plot as the principal component shifted data they had to be shifted as well. I am not sure I did this the right way. What I did was to group the cluster centers together two on two to get matrices with dimensions that allowed to compute the dot product with the two principal components. The resulting points that are

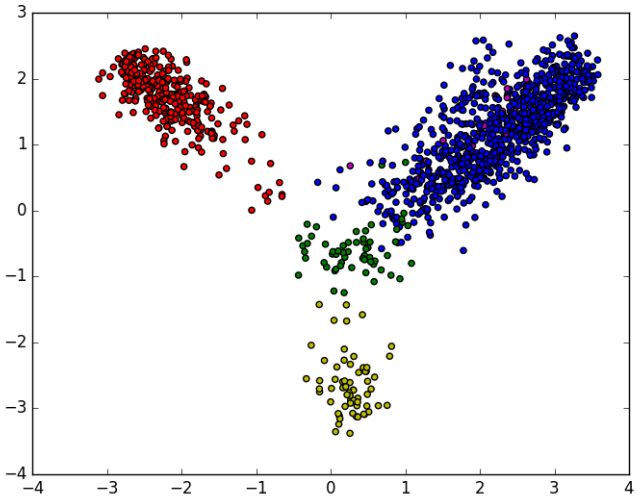

Figure 3: Scatter plot of traffic sign data using the first two principal components.

cyan on Figure 4 are no really centered in the clusters that you see. Assuming that the way I shifted makes sense, this shows us, that the cluster centers are not close to any actual points in the shifted data. One way to interprete this result is that shape is dominating the variance in the principal components, but that some (combination of) other features are better suited for clustering in the full dataset.

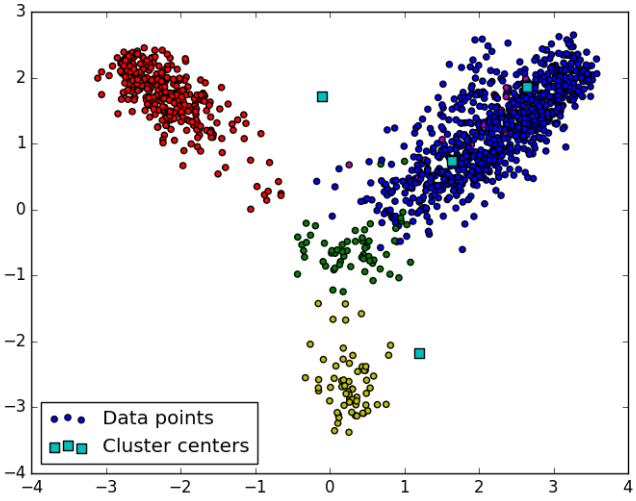

Figure 4: Scatter plot from Figure 3 with the resulting cluster centers of a 4-means clustering added.