Exercise work: pt.1

Data Analysis and Knowledge Discovery

Jarno Vuorenmaa

503618

jkivuo@utu.fi

Marco Willgren

502606

mahewi@utu.fi

First of all we need to read the file *wine.data*. We store the data into two variables *x* and *y*. *X* consists of a list of labels (1, 2 and 3) and *y* consists of the feature data. *InitialFeatureMatrix*() method is used to create a matrix where all the same feature's values are parsed to an array. All the features are added to the matrix in the right order (1, 2, ... , 13). We point to a feature matrix in this document and this points to this presentation of data.

1. Plotting a histogram of each feature

We created a method *defaultHistogram*() to plot a histogram of each feature separately. The method gets a matrix as a parameter. The matrix is consists of the feature data. The method adds one feature of every sample into a plot and draws the histogram accordingly. This is done for all of the 13 features.

Most of histograms have a common factor and most of the values are pretty evenly distributed.

2. Plotting histograms with visualizing each classes

To be able to plot the histograms visualizing each different cultivation, we need to separate the samples by their cultivation. Variable *x* has the information about the class. We need three subplots, because we have three different cultivations. We proceeded in the same manner as in the first assignment as we have a method to add one feature of every sample of a cultivation into one subplot. The method draws a histogram including three subplots for every 13 features.

By visualizing each cultivation separately, it can be seen that the cultivations differ from one another.

3. Making a parallel plot of the features

In this assignment (parallel plot), we wanted to create a plot for every cultivation with different colors. *Class1* is red, *Class2* is green and *Class3* is blue. The method *parallelFeatures*() creates three subplots. Each of them shows every sample of each cultivation separately.

The parallel plot doesn’t deliver much additional information because one of the features has significantly larger values than others. Therefore it dominates the plot over other features. By zooming and scaling the plot we could get better information.

4. Calculating the correlation coefficients

In this assignment we first calculate the correlation coefficients between each pair of features. After calculating the corcoeff of two features we check if they have an interesting correlation (either < -0,7 or > 0,7). If found, the specific details are printed in the console and the feature indexes are saved to the *interestingCorrelations* array:

FeatureA: Total phenols(5)
FeatureB: Flavanoids(6)
Correlation factor: 0.864563500095

FeatureA: Flavanoids(6)
FeatureB: OD280/OD315 of diluted wines(11)
Correlation factor: 0.787193901867

There are two interesting correlations which have a correlation factor higher than 0.7. First interesting correlation of value 0,86 is with feature 5 (total phenols) and feature 6 (flavanoids). The second interesting correlation is with features 6 (flavanoids) and 11 (OD280/OD315 of diluted wines). Their correlation factor is 0,79. There are also two more features that have a correlation factor of 0,79.

5. Visualizing interesting correlations with a scatter plot

As was noted previously, we have two interesting correlations (feature5/feature6 and feature6/feature11). The method *visualizeAsScatterPlot*() gets these strong correlations from the previous method which calculates correlation coefficients and returns interesting correlations. Interesting correlations are those, which have the factor higher than 0,75.

After this method creates a scatter plots of these interesting pairs. The features are colored in different colors (blue and red) to show the scatter more clearly and the colors are labeled.

6. Calculating the eigenvalues and eigenvectors of the covariance matrix

In this assignment we first calculate the covariance matrix of the feature matrix using *numpy’s* method *cov*(<matrix>). After this, we calculate the eigenvalues and eigenvectors from the covariance matrix. This is done by using *numpy’s linalg* library’s method *eig*(<covariance matrix>)

7. PCA

In this assignment we calculate the Principal Component Analysis of the feature matrix. First there is a call to the 6. assignment’s method to get the eigenvalues and eigenvectors from the feature matrix. Then we create a list of eigenvalue, eigenvector tuples (Eigen pairs) and the list is sorted in the way that the tuples are in decreasing order. Then we insert these tuples in a two-dimensional matrix by stacking them (*dimMatrix = np.hstack(eigPairs[0][1])*). After this we transpose the *dimMatrix* and multiply it with the feature matrix. The resulting matrix is the PCA result. After this we create a scatter plot of the result matrix.