Statistical Methods for Machine Learning
Assignment 1

Tudor Dragan (xlq880)
Nicolae Mariuta (rqt629)
Gabriel Carp (slp670)

February 25, 2015

I.2 Probability and Parameter Estimation

I.2.1 Univariate Gaussian distributions

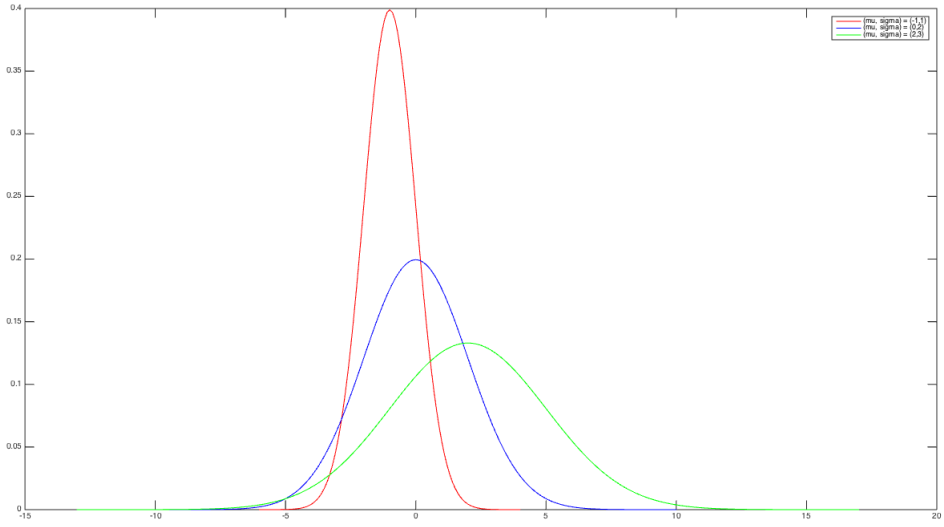

Figure 1: Gaussian distribution for

$$(\mu, \sigma) = (-1, 1), (0, 2), and (2, 3)$$

Ox in the gaussian1d function represents the input values and Oy is the output that is calculated by applying the univariate Gaussian Distribution function with means m and standard

deviation d.

I.2.2 Sampling from a multivariate Gaussian distribution

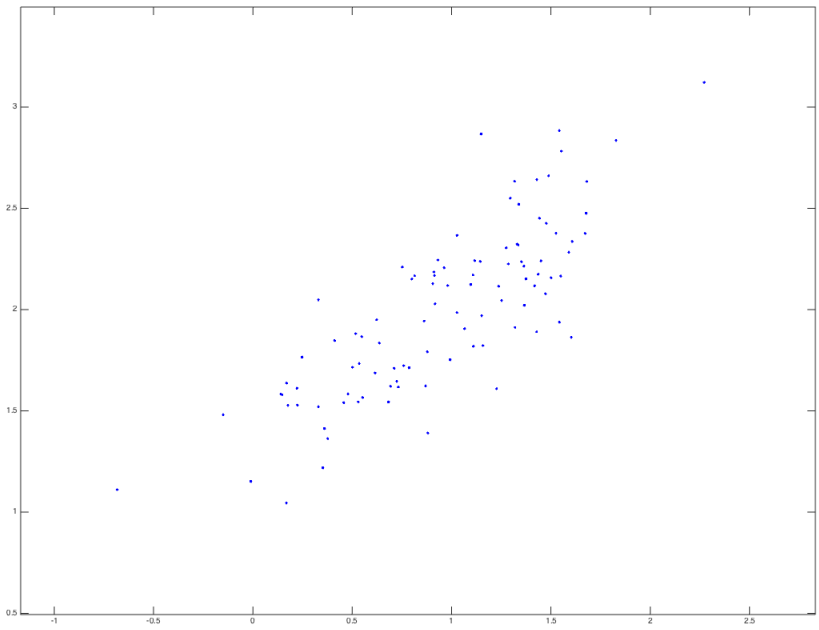

Figure 2: The plot for the input data set

I.2.3 Means

In Figure 3 we have represented the two means: mu which is the given mean ([1, 2]) and m is the quantitive mean that has been calculated by calculating the average of all the points in the data set. Because we generate the data set randomly, the points are chosen independently from each other. Because of this we have this deviation between the sample mean and the quantitive one. If we would have had a sample that was based on a periodical function, then the mean would have been identical because it would comply to the same rule. Furthermore, the smaller the number of our data set the bigger the deviation gets because we would have larger granularity between the set of values. So if we would have an infinite number of data points we would have "almost" the same mean.

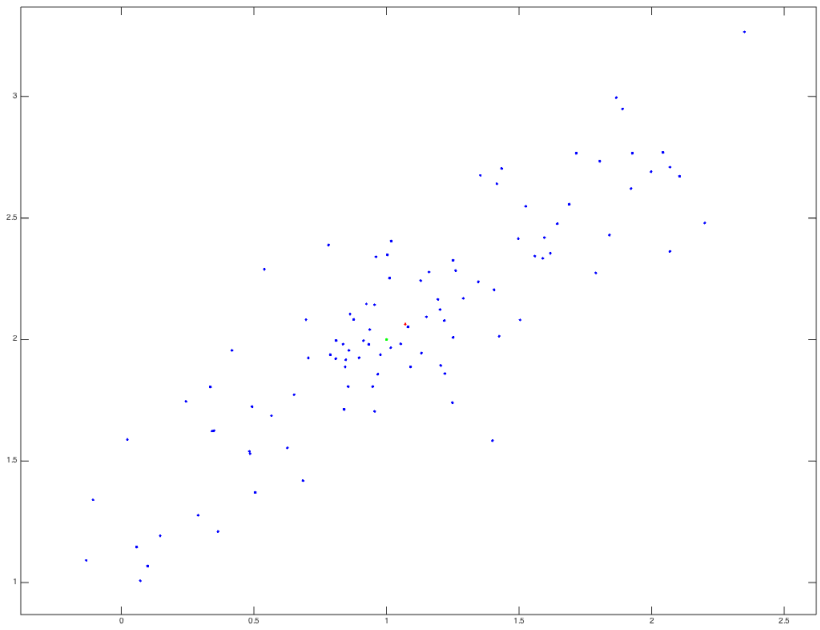

Figure 3: The two means: sample mean (green) and quantitive mean (red)

I.2.4 Covariance: The geometry of multivariate Gaussian distributions

The covariance is the property of a function of retaining its form when the variables are linearly transformed. Therefore if we rotate the data set by 30, 60 or 90 degrees we should get the elliptic shape as the standard data set. In our case, the covariance is a matrix of size (2,2) represented in the following:

$$Cov\ \Sigma_{ML} = \begin{bmatrix} 0.4012 & 0.2798 \\ 0.2798 & 0.2614 \end{bmatrix}$$

The major axes of the ellipse are defined by the eigenvectors of the covariance matrix, with corresponding eigenvalues. The main direction of the data sampled from the Gaussian distribution points along the x axis for

$$\alpha = -2.478$$

as seen in Figure 5.

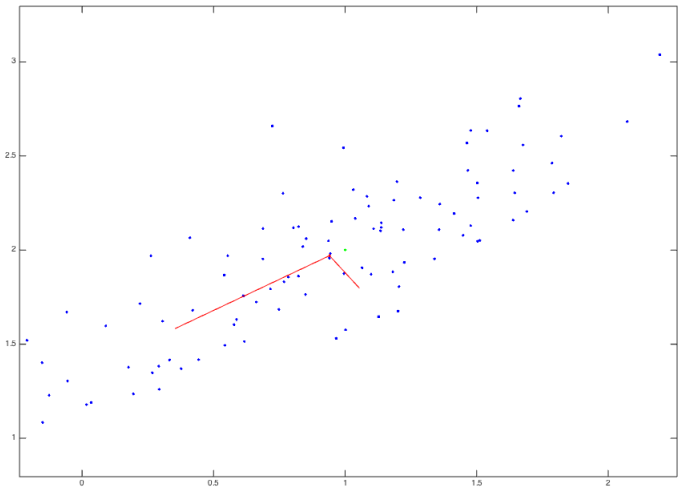

Figure 4: Covariance and eigenvectors

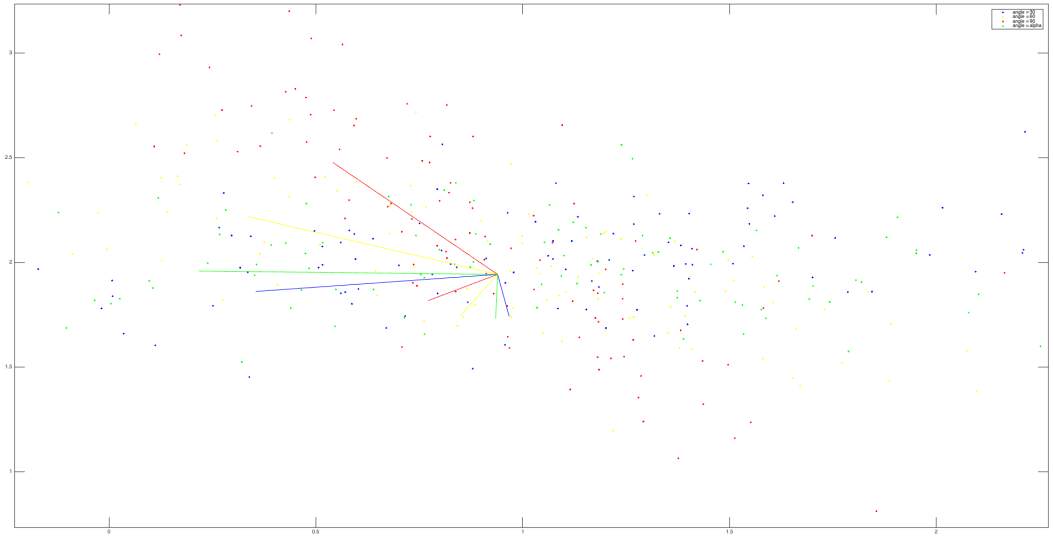

Figure 5: Combined plot of rotated datasets

I.3 Classification

I.3.1 Nearest neighbor

The kNN function is applied for K = {1, 3, 5}. The kNN function starts out by calculating the distances for each point in the test data relative to the train data points. After that, we sort the rows by distance and create a counting vector that holds the K number of points that are closest to the current analyzed point. We check then the highest count in the vector and calculate the accuracy (1 - empirical risk) by checking the values from the train and test data sets.

K	1	3	5
Training Data	0.770	0.7700	0.7400
Test Data	0.8158	0.8158	0.6842

Table 1: K-NN accuracy values

I.3.2 Hyper-parameter selection using cross-validation

For the cross validation we split the train data matrix into 5 parts. We took a k from 1 to 25 with step set to 2 and applied the cross validation algorithm like so:

- We exclude one part at a time from the train matrix and apply the kNN function on the part left out as the test data.
- We calculate the average k and add it to the kRiskMatrix.
- The last step searches for the minimum risk for a given k (k_{best}).
- In our case, the best value of k should be 3.

The entire kRiskMatrix can be seen in Table 2.

K	1	3	5	7	9	11	13	15	17	19	21	23	25
Risk	0.22	0.20	0.25	0.22	0.22	0.22	0.22	0.21	0.21	0.22	0.21	0.22	0.21

Table 2: Number of neighbors as suggested by hyperparameter selection

I.3.3 Data normalization

After the normalization the two parameters reside in almost the same interval and that's why both parameters have the same weight in the classification algorithm. Before normalization, one of the parameters had values between 4 and 7 and the second one had values between 0.20 and 0.40. Because of this the second parameter had a smaller effect in the classification's performance. We applied the normalization by calculating the mean for the train data

set and the standard deviation for each parameter by applying the function in the lecture slides.

The means and variances for the untransformed data are shown here:

$$mean_{train} = \begin{bmatrix} 5.766 & 0.301 \end{bmatrix}$$
$$stdDev_{train} = \begin{bmatrix} 0.830 & 0.041 \end{bmatrix}$$
$$mean_{test} = \begin{bmatrix} 5.928 & 0.319 \end{bmatrix}$$
$$stdDev_{test} = \begin{bmatrix} 0.859 & 0.046 \end{bmatrix}$$

The means and variances for the normalized data are shown here:

$$mean_{train} = \begin{bmatrix} 0 & 0 \end{bmatrix}$$
$$stdDev_{train} = \begin{bmatrix} 1 & 1 \end{bmatrix}$$
$$mean_{test} = \begin{bmatrix} 0 & 0 \end{bmatrix}$$
$$stdDev_{test} = \begin{bmatrix} 1.036 & 1.119 \end{bmatrix}$$

The error for the training and the test data is as follows:

K_{Best}	3
Training Data	0.070
Test Data	0.1842

Table 3: Error for $k_{Best} = 3$

The f_{norm} function is represented in our code as $f_{norm} = (X - mean(X))/std(X)$. After obtaining this function we also normalized the test set and applied the methods from the previous exercise for finding k_{best}. After that we calculated the empirical error. Our results show that $k = 1$ is the argmin for the test set and the standard deviation is a little higher (0.23). We also have noticed that for $k = 3$ the empirical error is lower. That is probably because our data set is not large enough and the result should be better if we would have a much larger data test set.