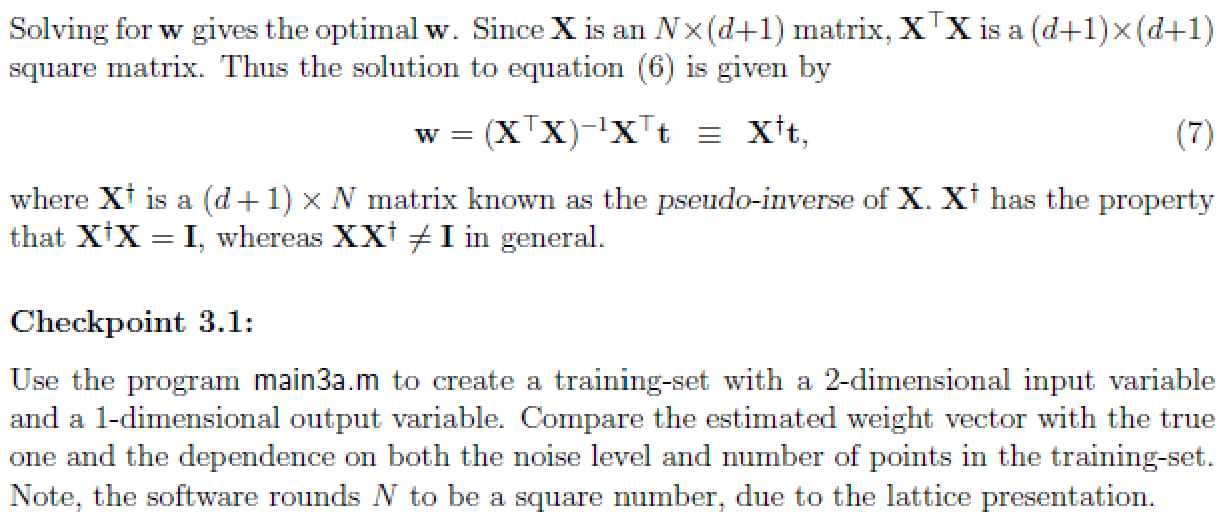
This exercise is focused on linear models for prediction and Fisher's linear discriminant.



Solving for $\mathbf{w}$ gives the optimal $\mathbf{w}$. Since $\mathbf{X}$ is an $N\times(d+1)$ matrix, $\mathbf{X}^\top\mathbf{X}$ is a $(d+1)\times(d+1)$ square matrix. Thus the solution to equation (6) is given by

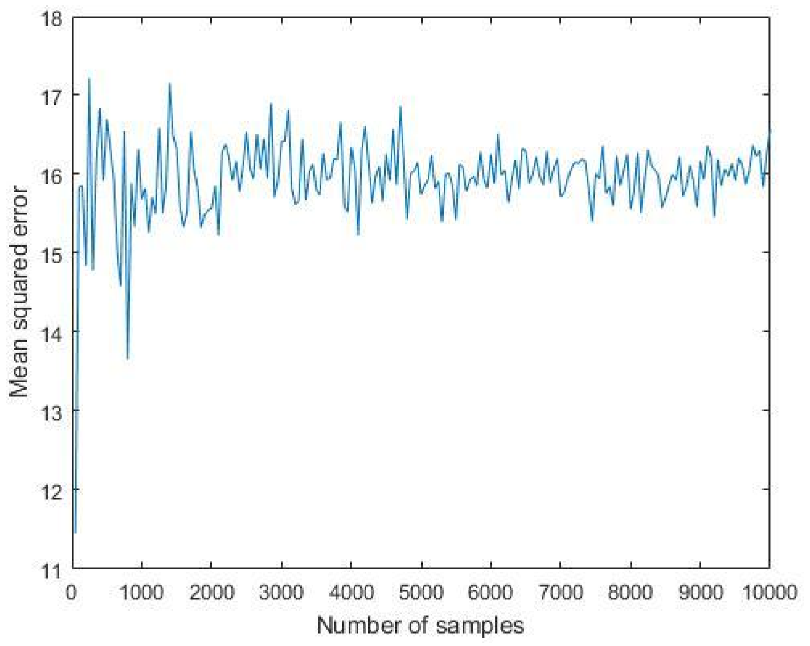
$$\mathbf{w} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{t} \;\equiv\; \mathbf{X}^\dagger\mathbf{t}, \tag{7}$$

where $\mathbf{X}^\dagger$ is a $(d+1)\times N$ matrix known as the *pseudo-inverse* of $\mathbf{X}$. $\mathbf{X}^\dagger$ has the property that $\mathbf{X}^\dagger\mathbf{X} = \mathbf{I}$, whereas $\mathbf{X}\mathbf{X}^\dagger \neq \mathbf{I}$ in general.

Checkpoint 3.1:

Use the program main3a.m to create a training-set with a 2-dimensional input variable and a 1-dimensional output variable. Compare the estimated weight vector with the true one and the dependence on both the noise level and number of points in the training-set. Note, the software rounds N to be a square number, due to the lattice presentation.

First, we will start by studying the dependence of the error and the number of points in the training-set. For this purpose we are going to plot the error for different values of N, as presented below for a noise standart deviation of 4.



We can see that once we have a number of samples for which the noise introduced by the random number generator function is not relevant, the mean squared error tends to 16. This value is expected, as we have a noise $\sigma = 4 \rightarrow \sigma^2 = 16$. But, why is the main squared error equal to the variance of the noise? To answer this question we should take a look at the previously presented function:

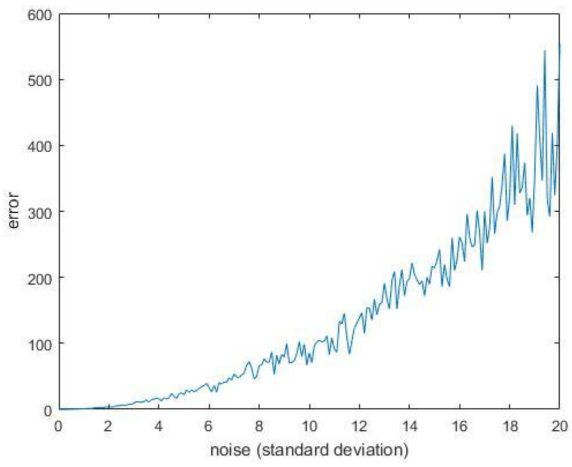
$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} \{y(\mathbf{x}_n;\mathbf{w}) - t_n\}^2$$

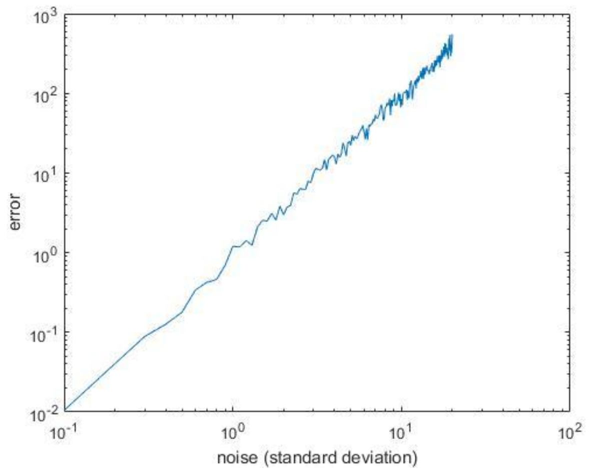
If we take into count that in the case of the matlab script the mean of the sum is computed as well, this equation looks really similar to the one of the variance.

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

Knowing that the mean of the noise is zero all is left in the squared part is the noise for each of the signals.

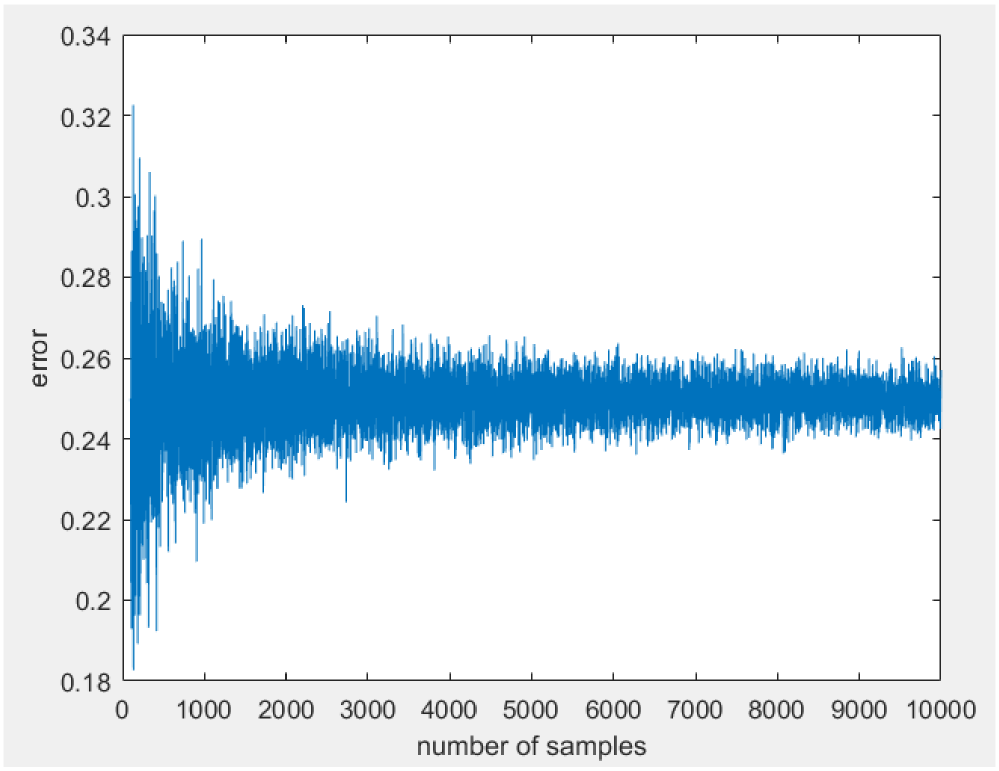
After studying the dependence on the number of samples, we will study the dependence on the noise level. We will proceed similarly as in the previous case, by computing the error for different values of noise. This is presented in the following graph:





In the left, we have plotted the error over a linear scale. From this graph we can see that the error is the square of the standard deviation, due to the reasons explained before.

In the right this relationship is plotted over a logarithmic scale, which shows how the relationship between both of the arguments is quadratic, because the relationship we are plotting is a noisy $\sigma|\sigma^2$



Finally, when we take a look at the weights we can note how they behave in a similar way of the error, as it is expected.

Time Series Prediction

An example where the linear model can be used is in time series prediction. To illustrate this, consider the example of the sunspot measurements. The number of sunspots oscillates almost periodically over a period of some years. The average number of sunspots has been measured yearly since 1700. Imagine we want to predict the average number of sunspots next year. The linear model can be used for this.

Let the number of sunspots in year n be x_n. Let's assume that the number of sunspots in year n only depends on the number of sunspots in the previous d years. This is reasonable since there must be a limit as to how far back one can expect a correlation. This can be expressed as

$$x_n = f(x_{n-1}, x_{n-2}, \dots x_{n-d}). \tag{8}$$

Approximating the function f with a linear model gives

$$x_n = w_0 + \sum_{j=1}^{d} w_j x_{n-j}. \tag{9}$$

This corresponds to equation (1), and hence is the same problem given by equations (2) to (7), where the training set is given by

$$\left.\begin{array}{rcl} \mathbf{x}_n & = & (1, x_{n-d}, \dots, x_{n-1})^\top \\ t_n & = & x_n \end{array}\right\} \quad n = 1, \dots, N-d-1. \tag{10}$$

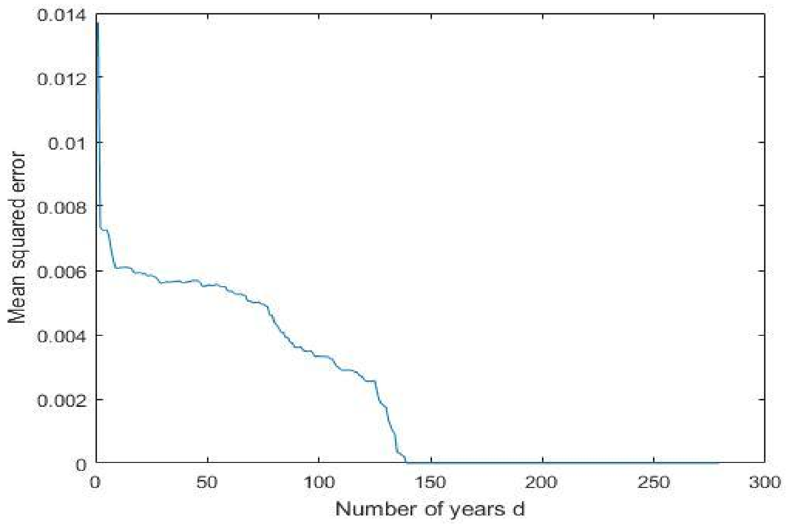
Note the important difference in the notations $\mathbf{x}_n$ and x_n. The weights can be found using equation (7), and the predicted value, x_{n+1}, can be found from

$$x_{n+1} = y(\mathbf{x}_n) = \mathbf{w}^\top \mathbf{x}_n. \tag{11}$$

Checkpoint 3.2:

Use the program main3b.m to perform a time series prediction of the number of sunspots. Compare the actual measurements with the predicted values as a function of the number of weights, d, (hence years) included in the model. Explain the value of the error for very large d.

In this exercise, we are going to study the relation between the number of weights (d) and the mean square error. In order to show this, we represented the error for the different possible values of d.



We can see that the error decreases until the point when d reaches half of the number of data. This can be explained by taking a look at the equations and dimensions used to calculate the weights of the model.

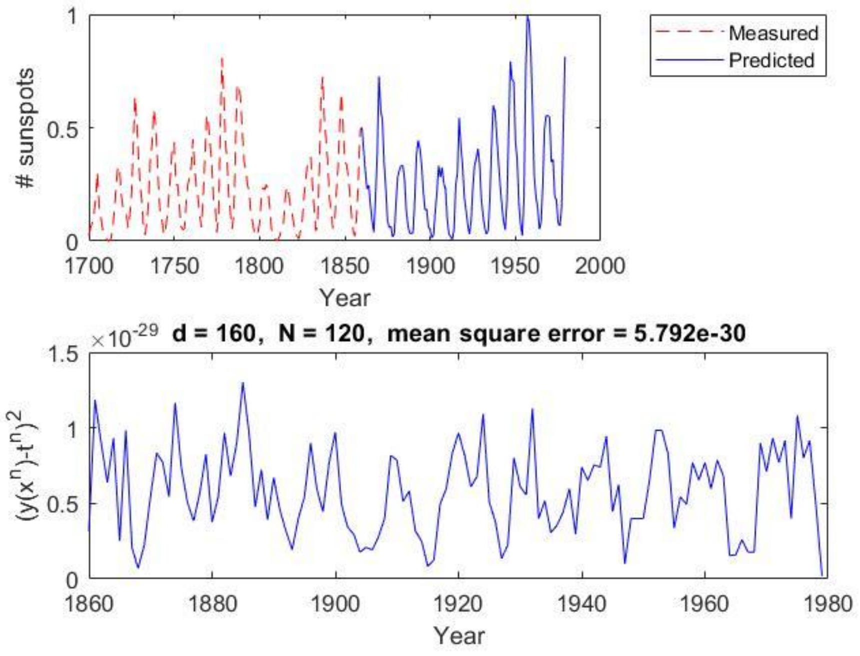
During the model training we take (N-d) training examples (target values), where N is the total ammount of training examples and d is the dimensionality of the system (number of preceeding years that we use to make the prediction). We start our predictions at (d+1) as this is the moment when there is enough input data to make the prediction.

We run the simulation and then plotted the dependency of error on the number of years back that we go. As expected, increasing the number of years we look back (number of inputs) makes the prediction error smaller. This is due to the fact that having more inputs allows us to have more complex estimation function. What is very interesting is that when the number of weights is equal to the number of training samples we will get the exect fitting so that the error goes basically to zero.

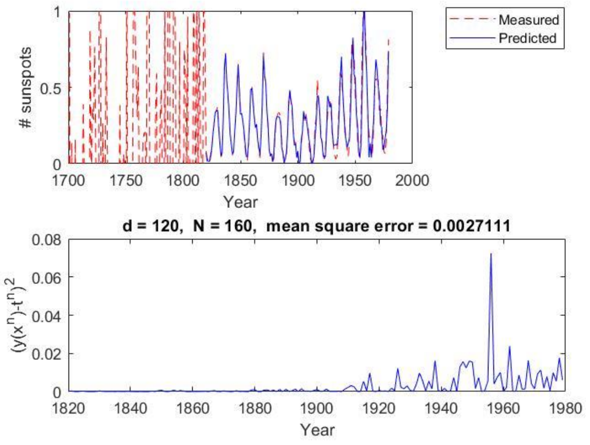
This is caused by the fact that having x weights/variables we are able to precisely reproduce x output values. Whenever there is defficency there will be error on some of the output values. An example with 1 weight we can exactly reprocude 1 input->output combination. With two we can exectly reproduce 2 input->output combinations etc.

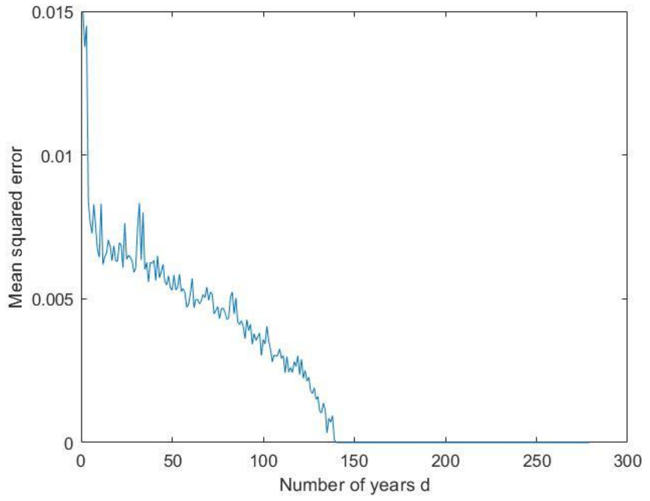
On the other hand a model trained this way will be not flexible at all and will not be able to generalize well on new incoming data, it's overfitting.

We can see the results when d=150 below



These findings drove us to consider how important the previous observations are in order to design a linear model based on them. To probe this we are going to substitute the first d years for a random distribution with mean 0 and check how the model fits that data.





As we can note, the first years in this case don't look a lot like the next ones, but the error didn't vary much from the previous experiment. It is interesting to see how even with really noisy measurements the model is able to predict the first years after the noise correctly.

Fisher's Linear Discriminant

In exercise 2, we saw that a multidimensional variable can be projected onto the directions of largest covariance by a coordinate transformation to the coordinate system spanned by the eigenvectors of the covariance matrix. This may facilitate classification of the data. However, there are also some cases, where the direction that maximizes class separation doesn't correspond to any of the eigenvectors. In such a case, the coordinate transformation does not solve the problem. However, the direction of maximum class separation can be found using Fisher's linear discriminant.

Consider a two-class problem in which there are N_1 points of class C_1 and N_2 points of class C_2. The mean vectors of the two classes are given by

$$\mathbf{m}_1 = \frac{1}{N_1}\sum_{n\in C_1}\mathbf{x}_n \tag{12}$$

$$\mathbf{m}_2 = \frac{1}{N_2}\sum_{n\in C_2}\mathbf{x}_n. \tag{13}$$

Let the projection of a data vector, $\mathbf{x}$, onto a the direction of maximum class separation be

$$y = \mathbf{w}^\top\mathbf{x}. \tag{14}$$

This is the direction along which the probability density functions of the two classes, $p(y|C_1)$ and $p(y|C_2)$, overlap the least. It can be shown by maximizing Fisher's criterion that the direction vector for the projection, $\mathbf{w}$, is given by

$$\mathbf{w} \propto \mathbf{S}_w^{-1}(\mathbf{m}_2 - \mathbf{m}_1), \tag{15}$$

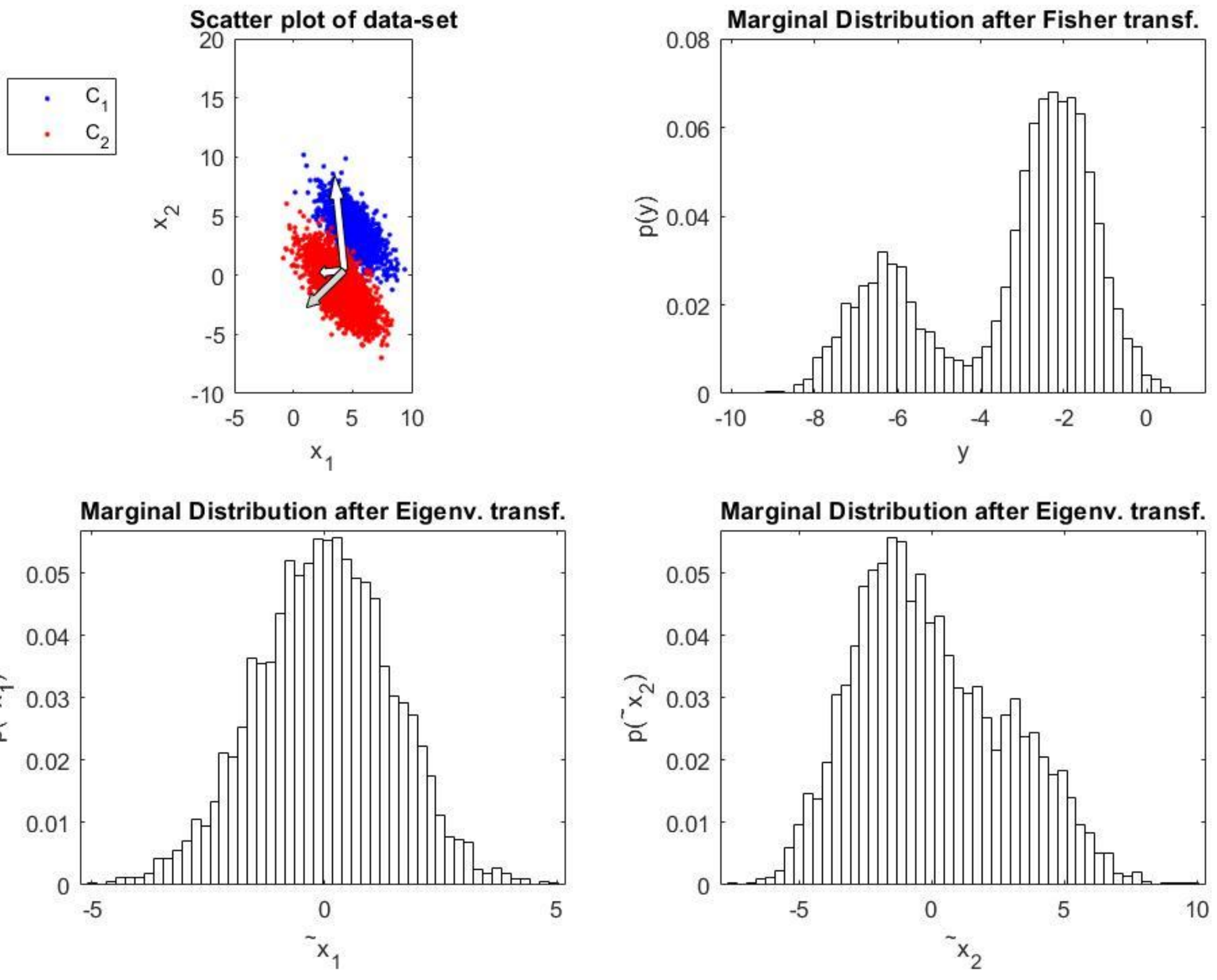
where $\mathbf{S}_w$ is the total within-class variation matrix, given by

$$\mathbf{S}_w = \sum_{n\in C_1}(\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^\top + \sum_{n\in C_2}(\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^\top. \tag{16}$$

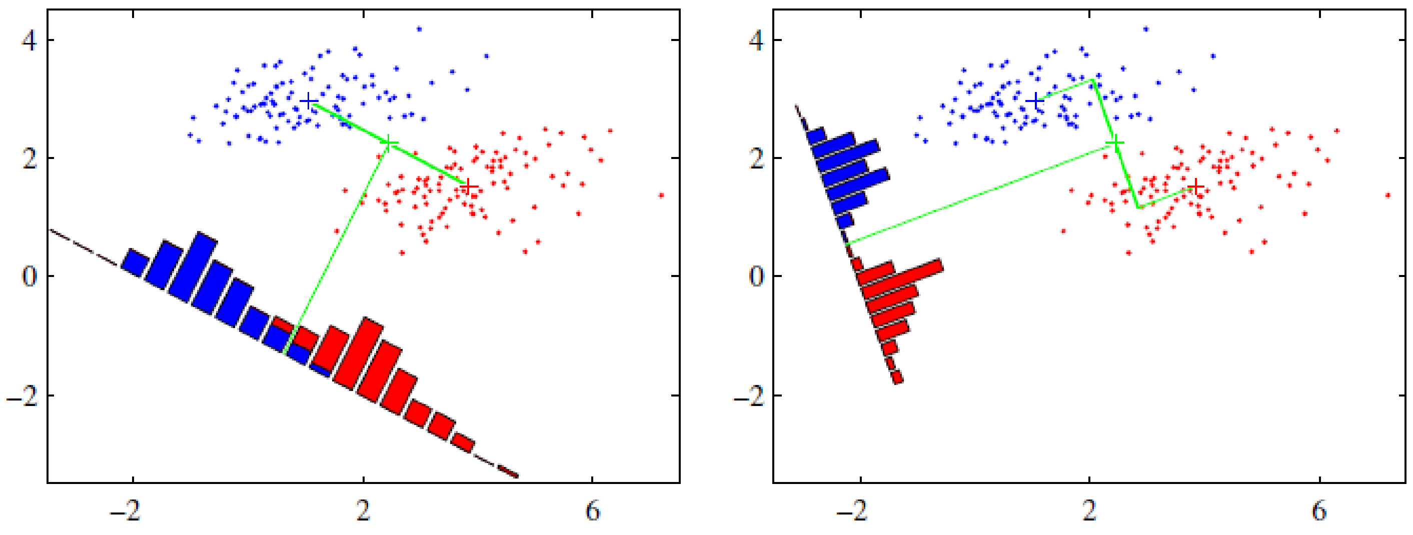
Checkpoint 3.3:

Use the program main3c.m to find the direction maximizing class separation for a two-class problem. In the figure class C_1 is color coded as blue and class C_2 as red. Compare the projection of the data-set onto one-dimension with the projections found using eigenvector transformation as illustrated in exercise 2. Compose different data-sets and compare the performance of the two methods in each case.

When it comes to dimensionality reduction the use of eigenvectors usually gives good results, as we saw in the previous exercise. However, it is not always the case, specially when the eigenvalues are similar. For example, in this case, even though a feature space where acceptable separation between classes is possible, the eigenvectors are not able to capture that dimension, as we can note from the figure:

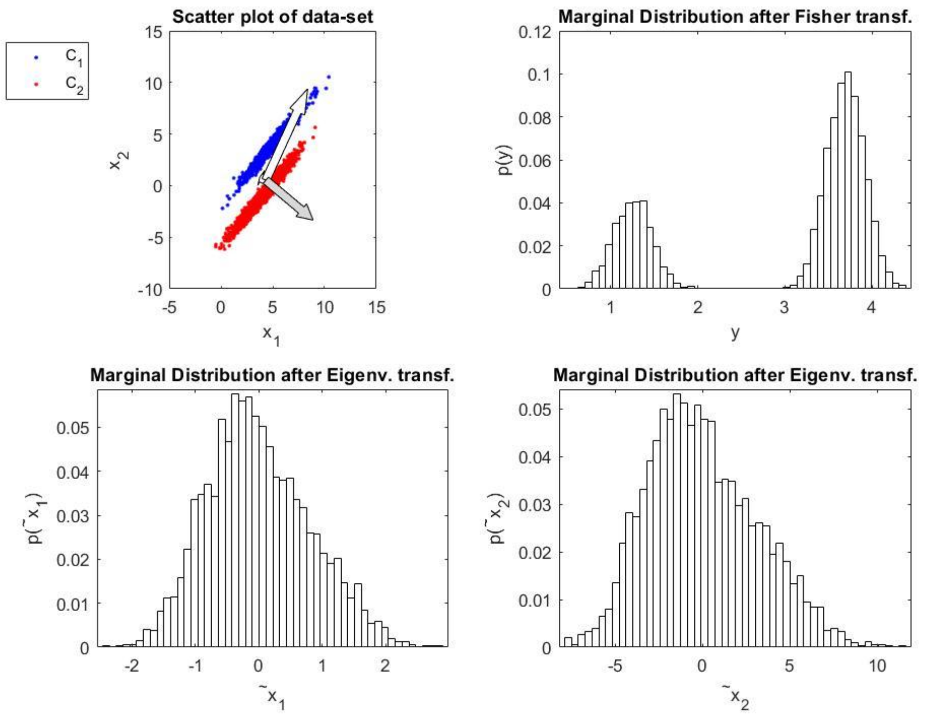


It is then when a different linear discriminant can be introduced, as it is the case of Fisher's Linear Discriminant. In this case, in order to optimize the dimensionality reduction two different parameters are used from the different distributions, the mean and the variance. First of all we want to minimize the distance between the means of the classes. However, this approach alone does not always provide good results as we can see:

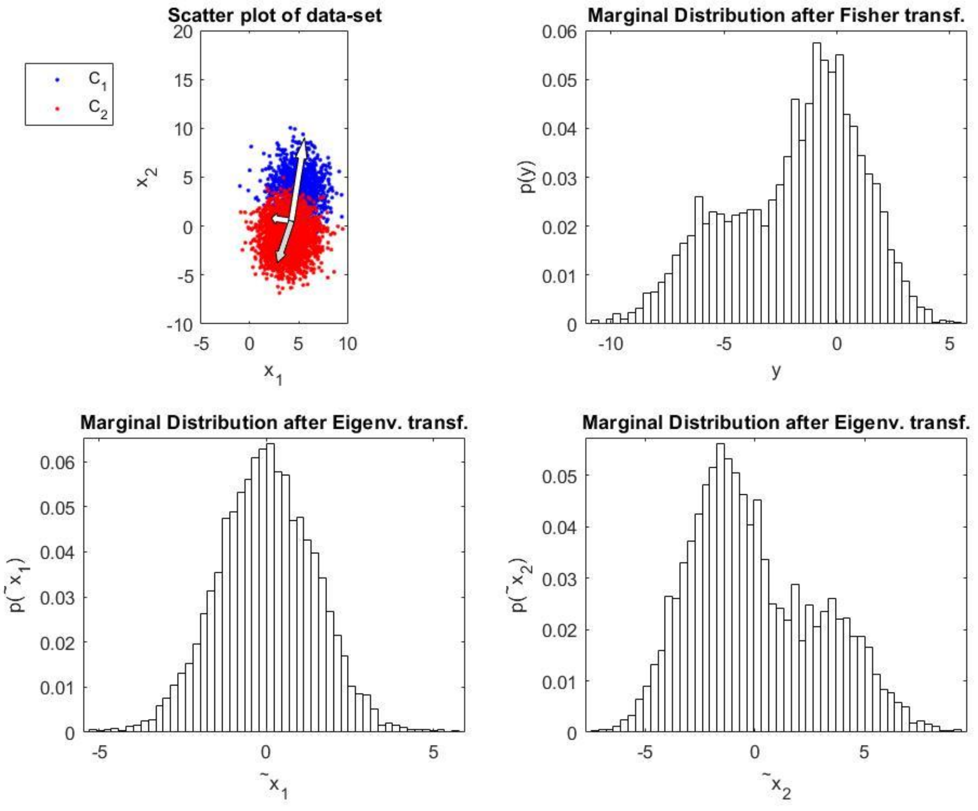


These are the cases when the distributions approaches a linear distribution (i.e. they have a strongly non-diagonal covariance matrix, with a correlation coefficient absolute value close to 1). The solution to this problem is to, at the same time we are trying to minimize the distance between means, minimize the variance within each class, thereby minimizing the class overlap. Taking this approach we can see in the second figure that we are able to separate between clases quite accurately.

In the following case, if we would have performed dimensionality reduction based on the largest eigenvalue our results would be far from optimal, specially when compared with the ones out of the Fisher's discriminant.



On the other hand, when the dimensions of the distributions are not really correlated we obtain comparable results, as presented below.



One thing wroth noting is that to acquire this discriminant we require some train data with points assigned to given classes.The final conclusion would be that depending on the task different method might be useful.