Neural Networks
Assignment 3 and 4

Group 2
M. Boersma(2566199)
E. Weinans (2566963)

September 22, 2015

Exercise 3a: MLP on a regression task

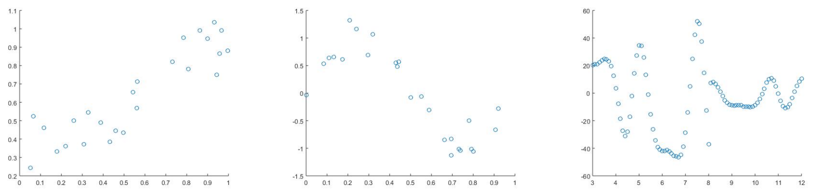
In this exercise, a multilayer perceptron is used for a regression task of three datasets: 'line', 'sinus' and 'irregular' (see figure 1).

To find a good network, different numbers of hidden layers are examined and each of them is evaluated with 5-fold cross-validation. The accuracy of the network is determined with the mean square error.

The values of number of hidden layers that we investigated are 2,3,4,5,7,10,15 and 25. For every value, we randomly assigned indices for the cross-validation. A multiple layer perceptron was trained on 4 of the folds, and the fifth fold was used as a test set. We averaged the erorr over several runs of the K-fold to get a more accurate result. Furtermore, we experimented with different types of Backpropagation, i.e. Gradient Descient, Stochastic Conjugate Gradient and Quasi-Newton. For each Backpropagation method we show the train and test RMSE score (see Figure2) to determine the optimal parameter settings for our neural network. Observe that for the irregular data the RMSE score is way higher than for the others, also observe that the gradient descent algorithm quickly converges to infinity and doesn't produce any usable results.

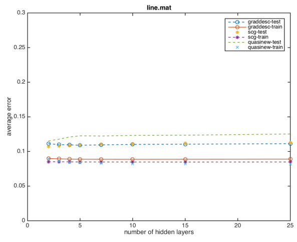
In general we can conclude that given our dataset the Stochastic Gradient Descent performs the best on the testset. Another noticable effect is that the Quasi-Newton method scores quite well on the training set nevertheless it shows the worst performance on the test set. Also, the performance of the sinus and linear data are usable with pretty low RMSE scores. The irregulare data-points need further investigation to be usable.

The same data is analysed by using a polynomial fit with different degrees. Again, 5-fold cross-validation is done and MSE is used as the error-measurement. Figure 3 shows that the polynomial fit works quite well on the line data, showing a low MSE. The polynomial fit works pretty well for the line- and sinus data as long as the degree is lower than 7, after that the error for the test set increases, which could be an indication of overfitting. The irrgular dataset performs better with a high degree polynomial than with a low degree polynomial. However,

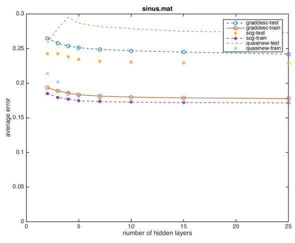


(a) Data for 'line' (b) Data for 'sinus' (c) Data for 'irregular'

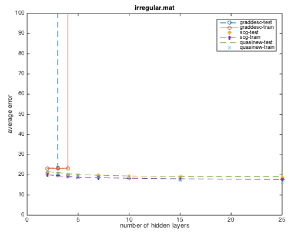
Figure 1



(a) MSE for 'linear' data, 100 cycles

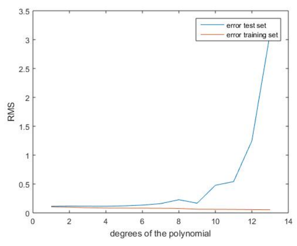


(b) MSE for 'sinus' data, 100 cycles

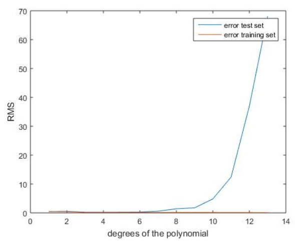


(c) MSE for 'irregular' data, 100 cycles

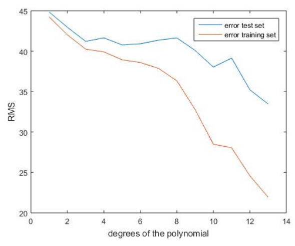
Figure 2: Mean square error for different numbers of hidden layers for training set and test set, after using a multiple layer perceptron for 100 training cycles.



(a) MSE for 'line' data



(b) MSE for 'sinus' data



(c) MSE for 'irregular' data

Figure 3: Mean square error for different numbers of degrees for training set and test set, after using a polynomial regression on a 5-fold cross-validation averaged over 100 runs.

as plotted in figure 4 here we also see that for polynomials with higher degrees, at some point the test error start to increase again (due to overfitting). Figure 4 also shows that for degrees higher than 20, the error of the training set starts to increase, indicating that the polyfit function does not work well in this range.

When comparing the polyfit to the MLP, we can conclude that for this data, MLP works better. For the line data the error of the MLP and a low degree of the polynomial is similar. For the sinus data it is also quite similar, but the polyfit never gets an error as low as the MLP. For the irregular dataset MLP very clearly outperforms polyfit.

Exercise 3b: MLP on a classification task

In the next exercise, a multiple layer perceptron is used again, but this time for a classification task. The dataset provided consists of vectors wit a length of 2576, which can be transformed to a 56x46 matrix. Every matrix can now be visualized, showing the face of a person that is either wearing glasses or not wearing glasses. We will use the original data and we will apply

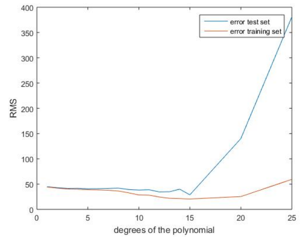
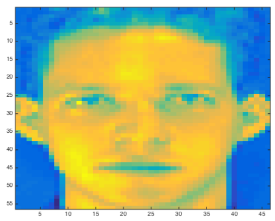
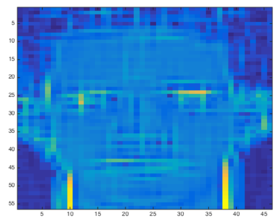


Figure 4: Mean square error for different numbers of degrees for training set and test set, after using a polynomial regression on a 5-fold cross-validation averaged over 100 runs for the irregular data.

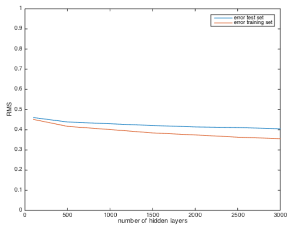


(a) Original face data

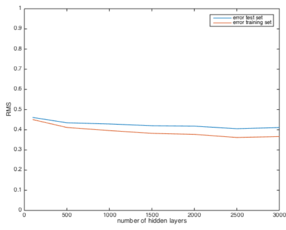


(b) Face data after applying Gabor filter.

Figure 5: Faca data



(a) MSE for glass data, after 10-fold cross-validation



(b) MSE for glass data, after 10-fold cross-validation, with Gabor filter applied

Figure 6

a Gabor filter see Figure 5 left and right respectively to the data and use that as input for the MLP-network.

A network for this dataset consists of 2576 input values (x=0,...,2576) and one output value (y=1 for glasses, y=0 for no glasses). The network was created for 100, 500, 1000, 1500, 2000, 2500 and 3000 hidden layers. In the previous exercise we used a linear function however for this classification task we will use a logistic function because it is better suitable for classification.

This time, 10-fold cross-validation was used. The indices were not assigned randomly, as done in the previous assignment, because there were more images of people without classes than people with classes. Therefore the ratio of people with and without glasses was kept constant for all 10 different folds (this is called stratified sampling).

Figure 6 shows that the error decreases as the number of hidden layers increases. However, if we read the line really carefully we observe that the test error decreases slower than the error of the training set. This might be an indication for overfitting. If we want to be sure of overfitting we would need to increase the hidden-layers even more and then observe that while the training error decreases the error for the test set increases. Nevertheless, for due to limited computing power we decided not to train MLP-networks with more than 3000 hidden layers.

With a confusion matrix we get more insight into the type of mistakes the network makes. We use the optimal number of hidden-layers discovered, which would be in our case the maximum of 3000, and then train the network on the whole data set and plot the confusion matrix. Table 1 shows the results for the non-filtered picture data. We observe that the network predicts 'no glasses' in 289 of the instances and 'glasses' in 111 of the instances. It makes 8 mistakes, all by misclassifying glasses for no-glasses.

We repeated the experiment for Gabor filtered data. Table 2 shows the confusion matrix which shows similar results as for the confusion matrix for the original data. Therefore, we conclude that applying a filter does not give a significant improvement in our case. What would probably help better is a dimension reduction of the image, for example, taking the first 10 principle components instead of the whole image.

	no glasses real	glasses real
no glasses predicted	281	8
glasses predicted	0	111

Table 1: Confusion matrix of a MLP-network with 3000 hidden layers for non-filtered training data

	no glasses real	glasses real
no glasses predicted	281	7
glasses predicted	0	112

Table 2: Confusion matrix of a MLP-network with 3000 hidden layers for Gabor filtered training data

Exercise 4a: RBF on a classification task

In this exercise, a classification of the glasses vs no-glasses dataset is made again, but this time a Radial Basis Function (RBF) network is used instead of MLP. In a RBF-network, there is only one hidden layer. However, the number of nodes in this hidden layer can be adjusted. The number of nodes can also be understood as the number of centres of different functions. The cluster centres can be found using K-means clustering. To check the accuracy of the network, 10-fold cross-validation is used again in a similar way as with the MLP-network.

The RBF model has three kernels: gaussian, thin plate spine (tps) and $r^4log(4)$. All kernels are investigated, and confusion matrixes are created (see tables 3, 4 and 5.) These tables are all made with a cut-off value of 0.5. The effect of this cut-off value of the missclassifications is visible in figure 7, which is plotted for the $r^4log(4)$ kernel. when the cut off value decreases, the network favours saying 'glasses' more often than 'no glasses'. This gets clearer when looking at table 6.

We can say, that for this dataset, MLP seems to work better than RBF. However, more options exist for the RBF network, that are not implemented in the netlab package used here and thus could not be investigated. It is possible that with different kernels or a different setting for the cut-off value, the RBF network improves.

Exercise 4b: SVM for a simple dataset

In this exercise, a support vector machine is used on the linearly separable dataset that we created for the first assigment. A support vector machine tries to maximize the space between two classes of datapoints. The support vector machines has some characteristics that we will investigate here. First, there are three different kernels: the RBF-kernel, the polynomial kernel and the linear kernel. Second, the data can be preprocessed or not (this is explained in more depth later). Last, there is a regularization parameter that tells the model how bad it is to missclassify datapoints in comparison to finding the maximum space between the two classes. This can influence the decision boundaries that you find, since sometimes there is a trade-off between maximizing the space between the classes and classifying the datapoints correctly.

First, the RBF-kernel in the SVM is used on the normal data and the preprocessed data. The preprocessing consists of two steps: mean reduction and covariance equalization.

The mean reduction is done to ensure that the updates of the weights can adjust in a

	no glasses real	glasses real
no glasses predicted	281	119
glasses predicted	0	0

Table 3: Confusion matrix for an RBF network with gaussian kernel.

	no glasses real	glasses real
no glasses predicted	270	110
glasses predicted	11	9

Table 4: Confusion matrix for an RBF network with tps kernel

	no glasses real	glasses real
no glasses predicted	264	105
glasses predicted	17	14

Table 5: Confusion matrix for an RBF network with $r^4 log(r)$ kernel

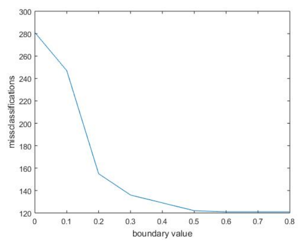
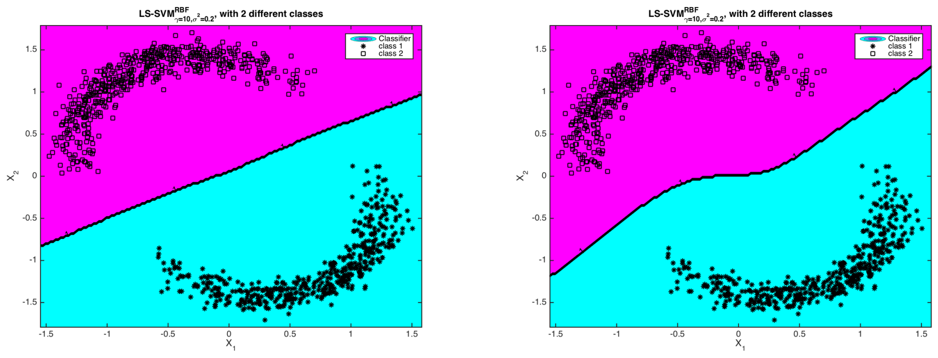


Figure 7: Number of missclassifications, depending on the cut-off value for the $r^4 log(r)$ kernel

	no glasses real	glasses real
no glasses predicted	277	117
glasses predicted	4	2

Table 6: Confusion matrix for an RBF network with $r^4 log(r)$ kernel, where the cut-off value is placed at 0.



(a) Decision boundary RBF-kernel with original data (b) Decision boundary RBF-kernel with preprocessed data

Figure 8: Decision boundaries RBF-kernel

correct way. This is easiest understood by thinking of the extreme example where all inputs are positive, which causes the sign of the weight updates to be either all positive or all negative. Therefore the weights can only all increase or all decrease for a particular input pattern. This slows down the learning rate. It can be solved by centering the data around zero, by subtracting the mean of the data for every datapoint.

The covariance equalisation is done because linearly dependent data (high covariance) slows down the learning of the network. This can be understood is you assume that one input is always twice some other input. In that case the datapoints do not add information to the network, which makes it impossible to get good results.

Figure 8 shows the decision boundary for the RBF-kernel for both the preprocessed (b) and the original data (a). We observe that the decision boundary is slightly different in both versions. Figure 8 (b) shows that the decision boundary is bend which is due to the mean removal and the covariance equalization.

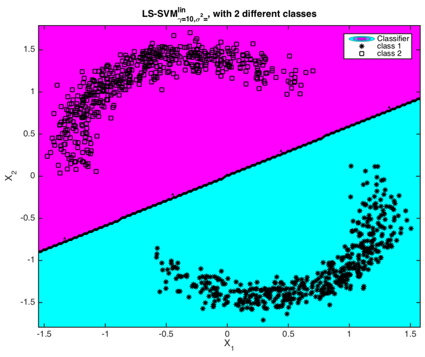
All in all, for this linearly separable dataset, all options of the SVM toolkit give good results. The plots show that the decision boundary changes depending on the kernel, the preprocessing and the regularization. However, all decision boundaries divide the two classes correctly.

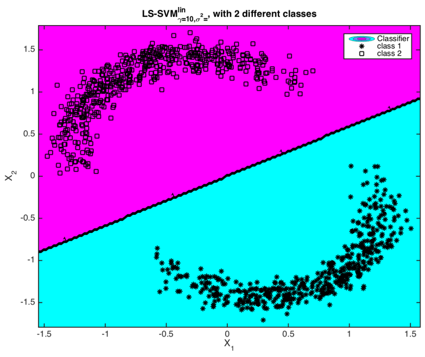
Furthermore, we plotted the decision boundary for the preprocessed data and original data for different kernels. Figure 10 shows the decision boundary for a polynomial kernel with degree 10 and Figure 9 shows the linear kernel which looks similar to the decision boundary learned with the Perceptron Convergence Algorithm which we investigated in previous exercises. We also investigated the effect of the regularization parameter, Figure 11 shows the different decision boundaries for different settings of regularization. We observe that the regularization makes the curve slightly smoother, Figure 11(a) shows that the curve has a less steep bend than for the regularization parameter of 1. This is due to the fact that fitting higher values is penalized more when the regularization parameter is set high. This prevents overfitting of which the effect is demonstrated in Figure 11.

Exercise 4c: SVM on a classification task

In this last exercise, we will investigate the glass-data again, this time using SVM. The data has multiple dimensions, and is linearly non-separable which makes it a more interesting dataset than the one we used before. In comparison with MLP, SVM is expected to be faster and more accurate.

10-fold crossvalidation is used again, and the average error is plotted for each one of the kernels in figure 12. This figure shows that for these parametersettings, there is no indication of overfitting. Furthermore, it shows that the linear kernel yields the lowest error for this dataset.

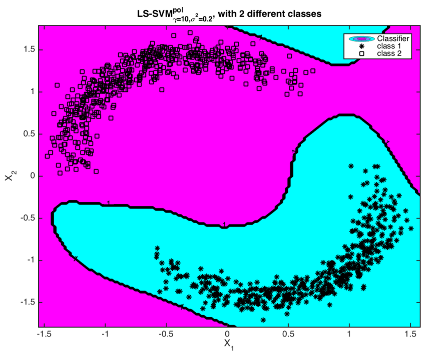


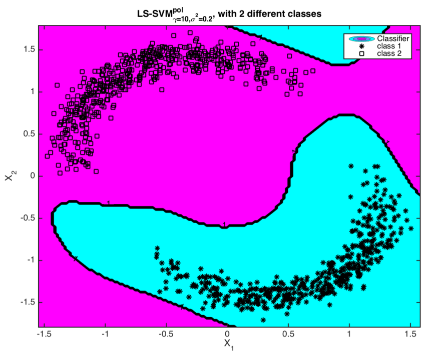


(a) Decision boundary linear-kernel with original data

(b) Decision boundary linear-kernel with preprocessed data

Figure 9: Decision boundaries linear-kernel

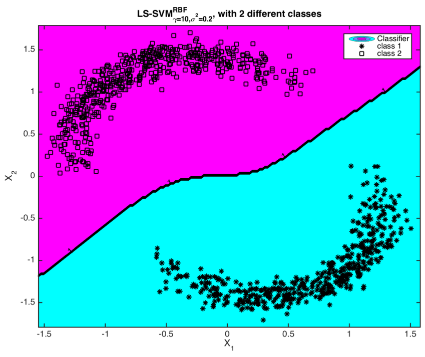


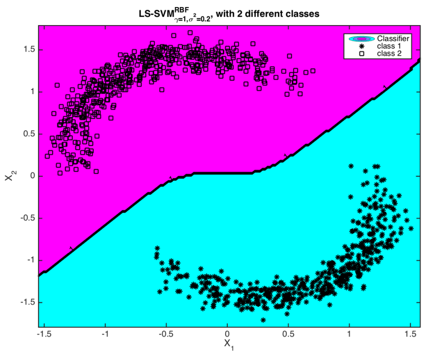


(a) Decision boundary poly-kernel with original data

(b) Decision boundary poly-kernel with preprocessed data

Figure 10: Decision boundaries poly-kernel with 10 degrees

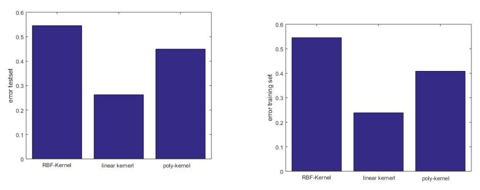




(a) Decision boundary for regularization equal to 10

(b) Decision boundary for regularization equal to 1

Figure 11: Decision boundary for preprocessed data with an RBF kernel for different regularization settings



(a) RMS for test set (b) RMS for training set

Figure 12: The RMS for the three possible kernels (MLP-kernel, linear kernel and poly-kernel) in the SVM network for the test set and training set.

	no glasses real	glasses real
no glasses predicted	278	2
glasses predicted	3	117

Table 7: Confusion matrix of glass-data after using SVM, linear kernel.

The SVM does a better job at classifying the individuals with and without glasses than MLP, especially with the linear kernel. This is visible in the error plots, but also in the confusion tables 7, 8 and 9. It is especially interesting to see that the RBF kernel on SVM performs very badly on this dataset, classifying all individuals in the no-glasses class. SVM also outperforms the RBF network when it uses a linear or polynomial kernel. As described before, there are some options for RBF networks that have not been investigated here, which make it impossible to say that SVM is the best choice for this dataset. However, the results do show that the SVM with the linear kernel classifies almost every datapoint correctly.

	no glasses real	glasses real
no glasses predicted	278	8
glasses predicted	3	111

Table 8: Confusion matrix of glass-data after using SVM, poly-kernel.

	no glasses real	glasses real
no glasses predicted	281	119
glasses predicted	0	0

Table 9: Confusion matrix of glass-data after using SVM, RBF-kernel.