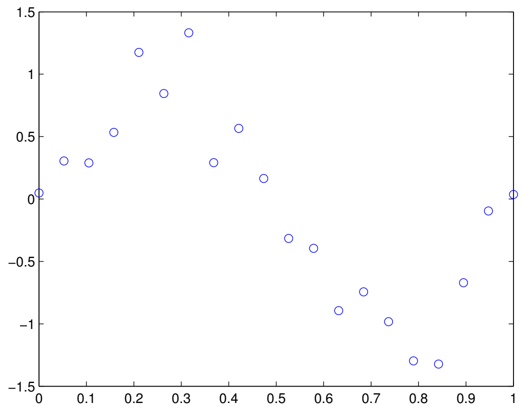
Exercise 4

Philipp Hanslovsky, Robert Walecki

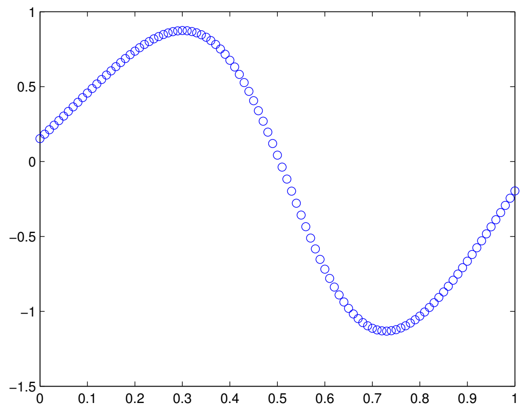
May 21, 2012

4.1.1

The example trains a multi layer perceptron (MLP) using the input array x containg 20 values ranging from 0 to 1 and the target array t, which holds one value for each corresponding value of x. x and t form a sine function distorted by random values. The MLP has one input, three hidden units (perceptrons) and produces a single output. It is created by the command net = mlp(nin, nhidden, nout, outfunction). netopt optimizes the MLP based on input x and target t. Finally a prediction is made (mlpfwd) for an input array plotvals containing 101 values ranging from 0 to 1. The resulting array y is then shown in a plot (see fig 1). Even with the highly distorted sine used for training and using only 3 hidden units the MLP applied to plotvals produces a function close to a sine. The result of increasing the number of hidden units is shown in fig 2 for 50 hidden units and in fig 3 for 100 hidden units.

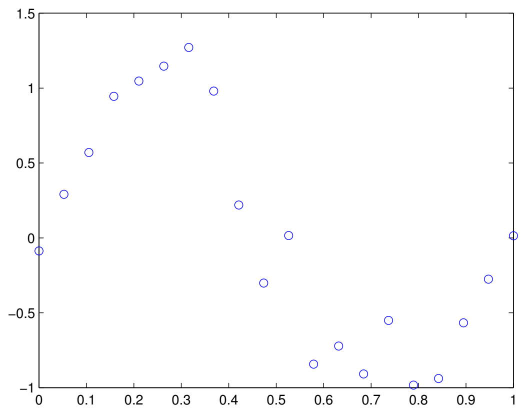


(a) target array t used for MLP training

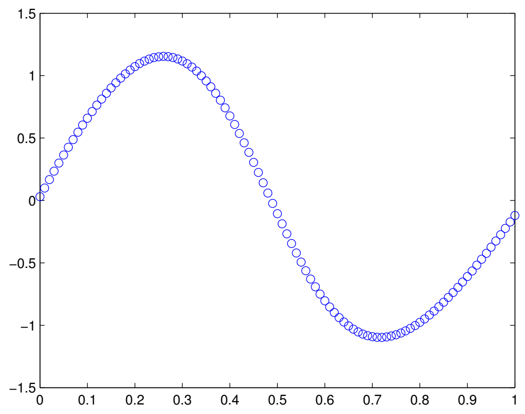


(b) output of applying MLP on input array plotvals

Figure 1: Plots of Training Dataset and output (3 hidden units)

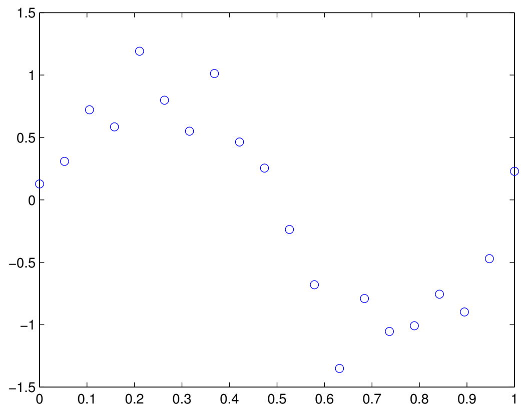


(a) target array t used for MLP training

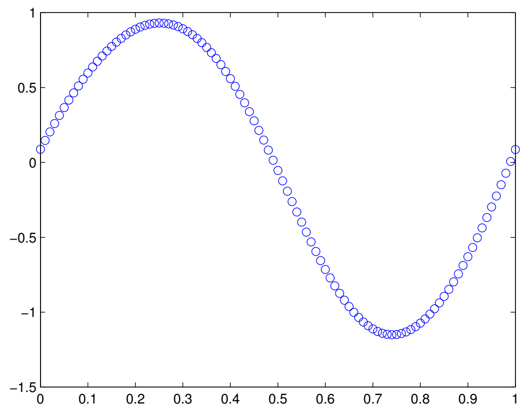


(b) output of applying MLP on input array plotvals

Figure 2: Plots of Training Dataset and output (50 hidden units)

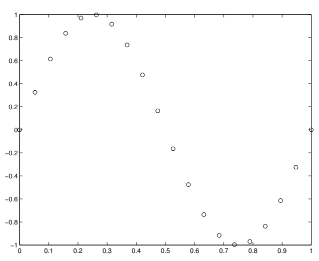


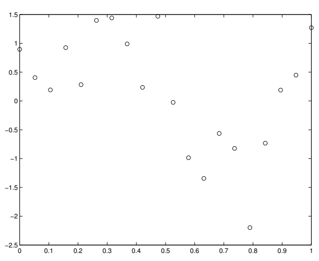
(a) target array t used for MLP training

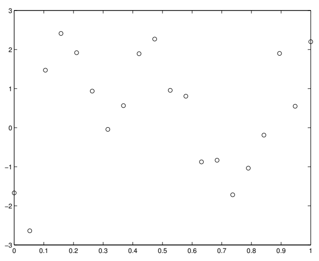


(b) output of applying MLP on input array plotvals

Figure 3: Plots of Training Dataset and output (100 hidden units)


(a) 0


(b) 0.6


(c) 1.4

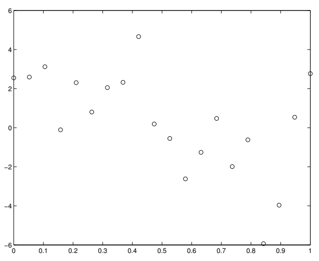
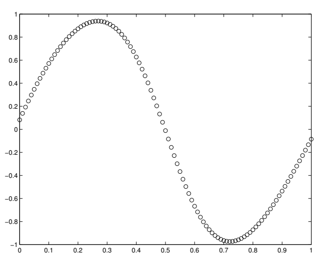
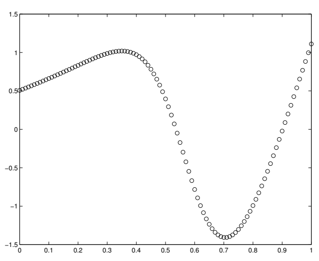
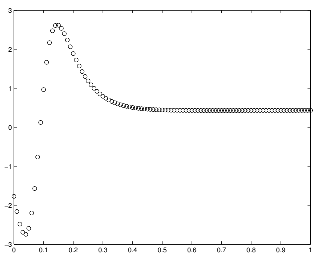

(d) 2

Figure 4: training data for various noise factors


(a) 0


(b) 0.6


(c) 1.4

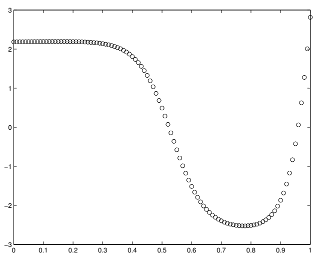

(d) 2

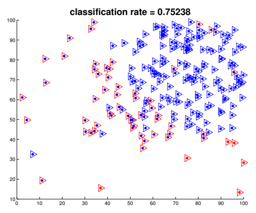
Figure 5: prediction for various noise factors

4.1.2

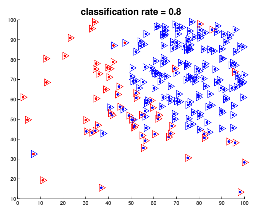
Increasing the weight of the noise results in badly trained MLPs. The MLPs cannot predict a sine function anymore. Fig 4 and 5 show the target data used for training the MLPs and the corresponding predictions. With a noise factor bigger than or equal to 0.6 the prediction is totally off. Therefore data with only little noise should be chosen for training.

4.2.1

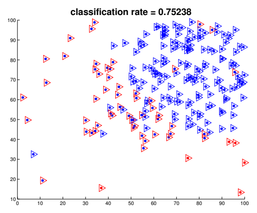
Fig 6 - 8 show the scatter plots of the predictions as well as of the original labels and the decision boundaries. As you can see in the decision boundary plots training the MLP seems to be some kind of lottery (four different results in five tries!). The classification rates are to be found on the corresponding plots.



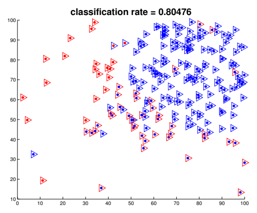
(a)



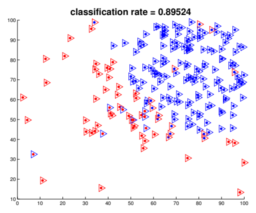
(b)



(c)

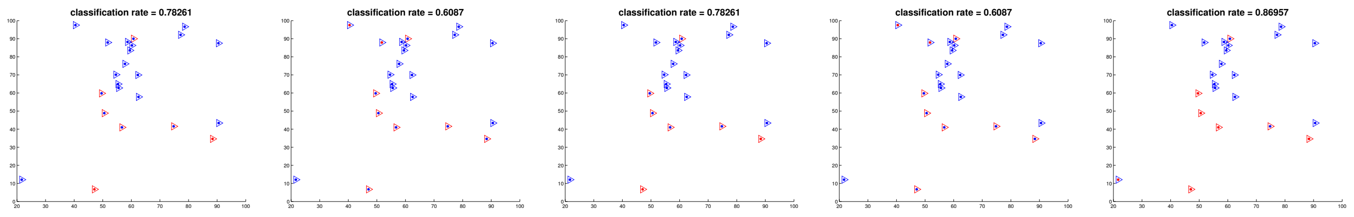


(d)



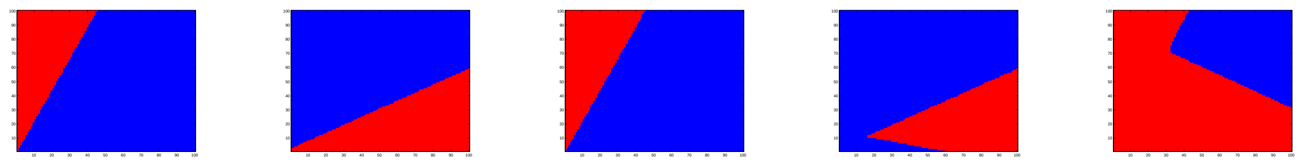
(e)

Figure 6: Predictions for the training dataset: Blue means pass, red means fail. The circles are the predictions, the triangles are the actual labels. The classification rate is shown on top of each plot.



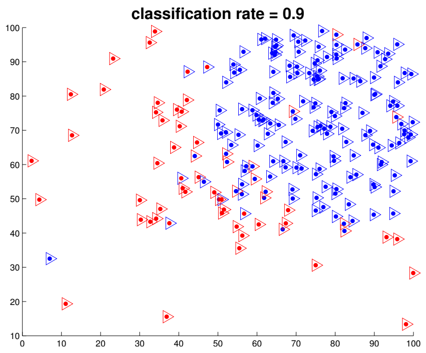
(a) **(b)** **(c)** **(d)** **(e)**

Figure 7: Predictions for the test dataset: Blue means pass, red means fail. The circles are the predictions, the triangles are the actual labels. The classification rate is shown on top of each plot.

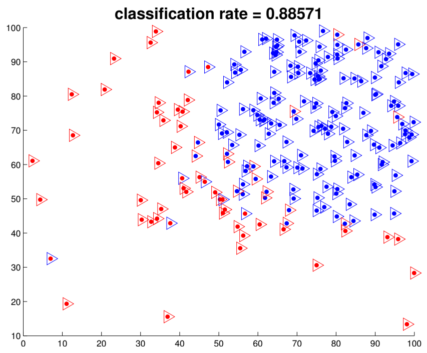


(a) **(b)** **(c)** **(d)** **(e)**

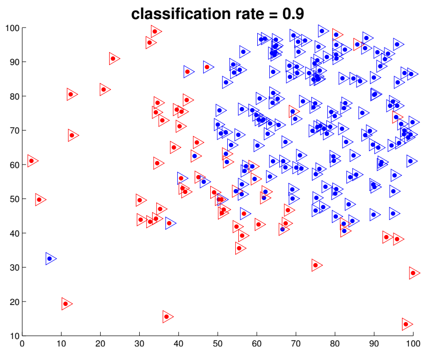
Figure 8: Decision boundaries: Blue is pass. Even though using the same dataset and parameters for the training of the MLPs they significantly vary in their predictions.



(a) hidden units: 100, iterations: 1000, function: linear

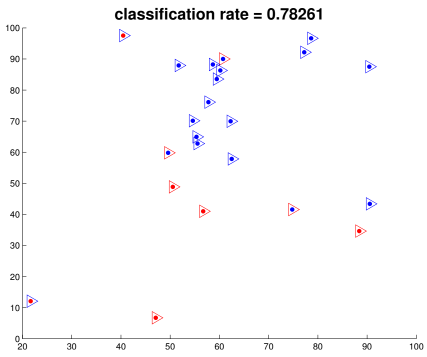


(b) hidden units: 100, iterations: 1000, function: logistic

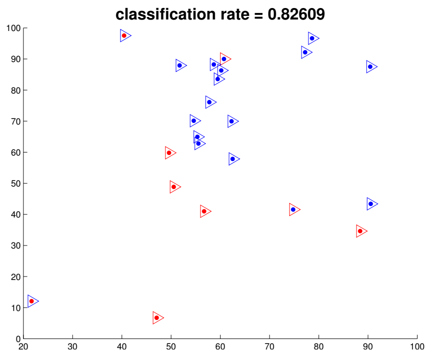


(c) hidden units: 100, iterations: 5000, function: logistic

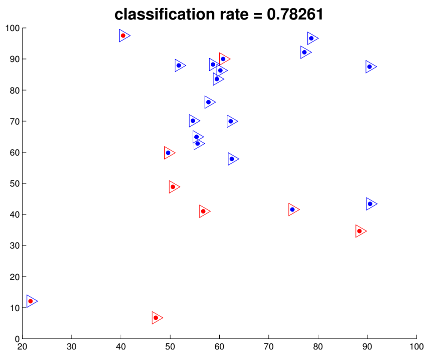
Figure 9: Predictions for the training dataset: Blue means pass, red means fail. The circles are the predictions, the triangles are the actual labels. The classification rate is shown on top of each plot.



(a) hidden units: 100, iterations: 1000, function: linear



(b) hidden units: 100, iterations: 1000, function: logistic



(c) hidden units: 100, iterations: 5000, function: logistic

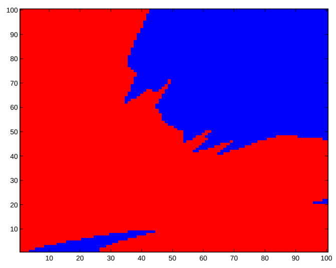
Figure 10: Predictions for the test dataset: Blue means pass, red means fail. The circles are the predictions, the triangles are the actual labels. The classification rate is shown on top of each plot.

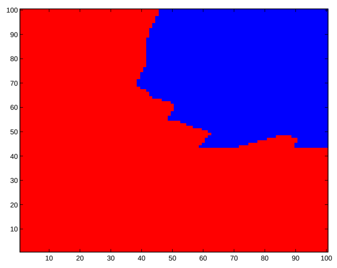
4.2.2

Increasing both the number of hidden units and the number of iterations improve the classification result. However, the number of hidden units should not be chosen too high to avoid overfitting of the data. The choice of the number of iterations is a questions of precision as well as time. A larger number of iterations always yields in better results, but will naturally need more time to run. Using a logistic function instead of a linear one did not improve classification (see fig 9 - 11). On account of the observations in section 4.2.1 only vague predictions on the effects of changing the parameters are possible.

4.2.3

A number close to 100 seems to be a good choice for the number of hidden units to both avoid overfitting and get a high enough dimensionality at the same time. We chose it to be 95. A high number of iterations yields in a better result and time was not a factor so we chose the number of iterations to be 5000 rather than 1000. As stated above the choice of function does not seem to matter, thus we arbitrarily chose a logistic function.


(a) hidden units: 100, iterations: 1000, function: linear


(b) hidden units: 100, iterations: 1000, function: logistic

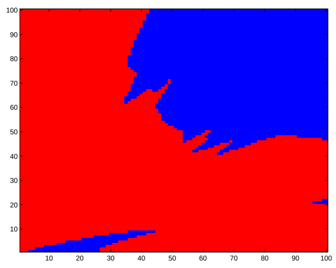
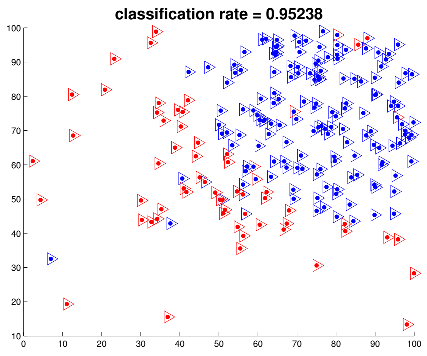
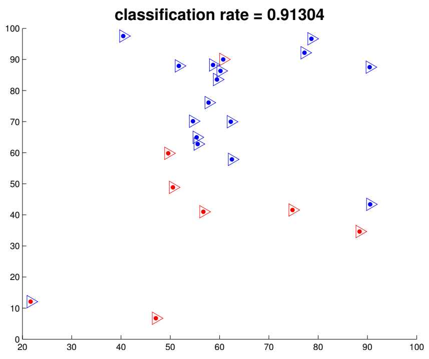

(c) hidden units: 100, iterations: 5000, function: logistic

Figure 11: Decision Boundaries for different MLPs. Blue is pass.



(a) Training



(b) Test

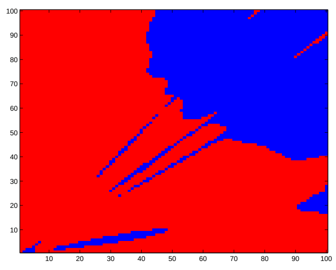
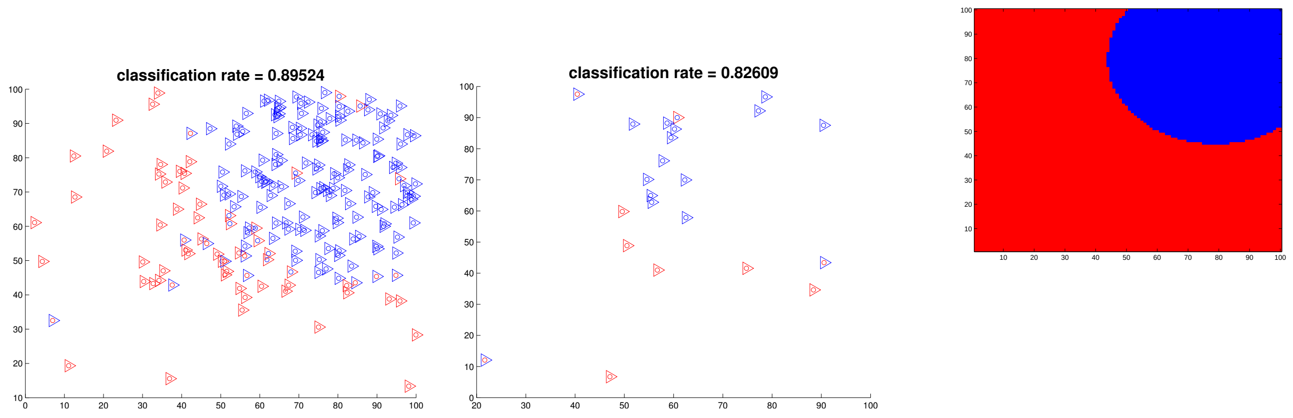

(c) Decision Boundary

Figure 12: Classification of training data as well as test data and decision boundary. The parameters are: hidden units - 95, iterations - 5000, function - logistic



(a) Training **(b)** Test **(c)** Decision Boundary

Figure 13: Classification of training data as well as test data and decision boundary using GRBF.

4.3

Both QDA and Gaussian RBF (GRBF) use normal distributions, therefore they assume the data is normally distributed in feature space. (However, the Gaussian functions are not normalized in GRBF, that is accounted for by the weights). QDA takes into account the prior and maximizes $p(y_i|x)$ with respect to y_i the probability as a product of exponential functions (see eq 1). GRBF does not make use of the prior, however it uses the weight and then maximizes the weighted sum (linear combination) of all functions.

$$p(y_i|x) = p(y_i) const \exp(-1/2(x - \mu_i)^T \Sigma_i^{-1}(x - \mu_i)) \tag{1}$$

The decision boundary is similar to that of QDA, however it is not restricted to quadric surfaces as it is a linear combination of exponential functions. The classification rates are shown in fig 13. They are a bit lower than compared to when using MLP. Naturally one might think that GRBF is the better choice as it is related to QDA and the decision boundary can be expected to be somewhat circular. However, the results speak differently which might be explained by outliers, caused by e.g. students who just needed to pass the exams (and did not go for a good grade) and then prepared well for the oral exams or others who had a lucky shot on one of the exams or lost motivation throughout the semester. Therefore MLP is the better choice be cause it seems to adapt better to that kind of situation.