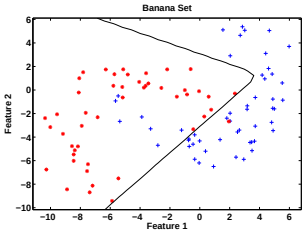
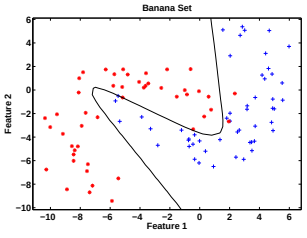
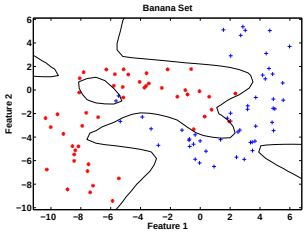
Solutions to exercises of Day 5

Exercise 5.1

(a) The network is a linear classifier, which cannot separate the data well. The decision boundary slowly shifts during the first few iterations, then converges.
(b) With 2 hidden units, the classifier is quadratic (figure (a) below). With 3 (figure (b)) and more hidden units the data is separated well. For 5 hidden units the solution becomes too complicated, and for 10 hidden units the solution is highly overtrained (figure (c)).



(a)


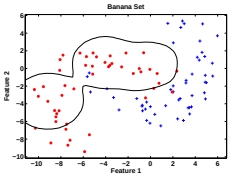
(b)


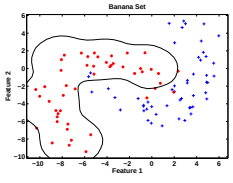
(c)

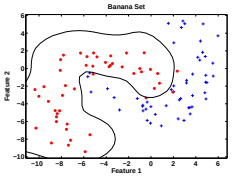
(c) The solution is overtrained, similar as before.
(d) 1 hidden layer of 3 units should suffice.

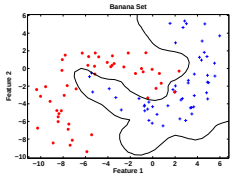
Exercise 5.2

(a) Similar to the previous exercise: there is an optimal number of Gaussians, around 10:



(a)


(b)


(c)


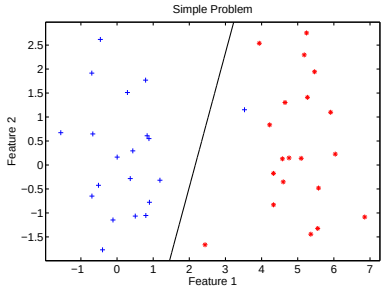
(d)

Exercise 5.3

(a) Results are highly variable due to the small sample size, but it seems an architecture with two hidden layers works well for the Golub dataset. For the Khan dataset a single hidden layer with a reasonable number of units (10-20) may give good results.
(b) Again, results are variable. In general, the most simple classifiers should work best on data such as these.
(c) With more features, the risk of overtraining increases, so it is likely that simpler architectures (less hidden layers, less hidden units) generalise better.

Exercise 5.4

(a) It should look like:



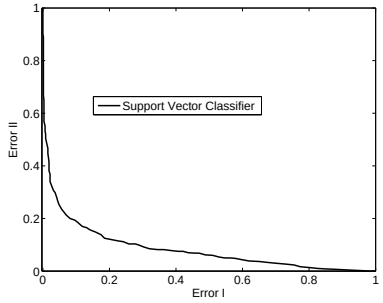
(b) The most simple way is probably to print out the coordinates of the suppport vectors: `+a(I,:)`. If you want to plot them into the figure, you have to enter `hold on; scatterd(a(I,:),'go')`.

Exercise 5.5

(a) No.
(b) Degree 3 and higher is often ok. At degree 7 the solution becomes unstable again.
(c) For σ around 4 solutions look reasonable. Note that this is only visual; actually you should check this, by splitting the data in a training and test set, training on the training set and evaluating on the test set.

Exercise 5.6

(a) The scales of the various features differ quite a bit. Feature 34 has zero standard deviation, i.e. it is useless for classification.
(b) Train and test error should be around 15%. The ROC looks like this:



Errors of Type I here are false positives, so we will likely be interested only in the leftmost part of the ROC.
(c) This classifier seems to be overtrained: no error on the training set, 50% error on the test set.
(d) Already initially, results on the test set improve. Changes to the weights or addition of kernels do influence training error, but do not seem to yield a significantly better test error.

Exercise 5.7

Increasing C means errors are weighted more, so the classifier will make less errors on the training set at the expense of a larger weight vector, and hence a more complex decision boundary.

Exercise 5.8

(a) Two classifiers each assign posterior probabilities to two classes.

Exercise 5.9

Your mileage may vary – the combination is not guaranteed to yield the best result.

Exercise 5.10

The combined classifier generally gives lower error than any of the three individual classifiers,

but close to the one trained on the KL features (`mfeat_kar`).

Exercise 5.11

(a) Most should give errors around 20%; `treec` performs significantly worse.
(b) Your mileage may vary, but a voting combiner on the first four classifiers may yield better results.

Exercise 5.12

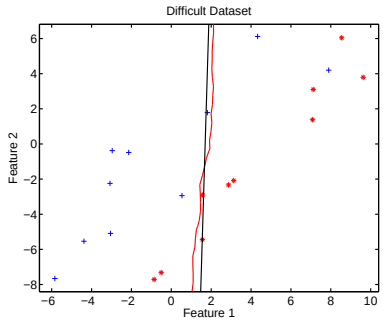
(a) Performance is best on the KL features (roughly 10% error), worst on the morphological features (over 50% error).
(b) This should be possible, but it is not easy to find better performance than using just the KL features.

Exercise 5.13

For small training sets, the bagged classifier can do a little better.

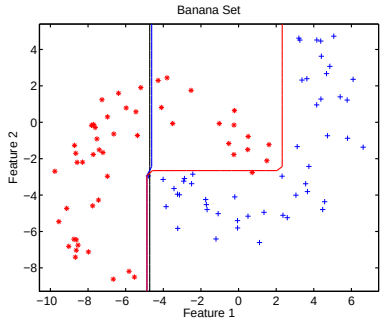
Exercise 5.14

Note that the resulting classifier can be nonlinear. For example (black line is `nmc`, red line the bagged version using `votec`):



Exercise 5.15

Results are illustrated below. For depth 3 (red line), the basic shape of the classifier fits the banana data; for lower depth (black, blue lines), the decision boundary is too simple. The full tree classifier is much more complex and likely to overtrain (dashed line).



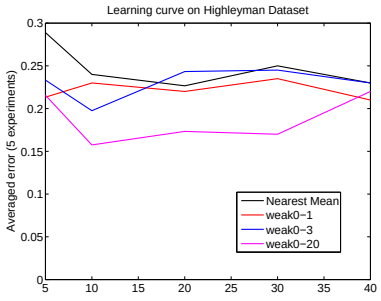
Exercise 5.16

The result is still a linear classifier, but its performance can be better than that of the original

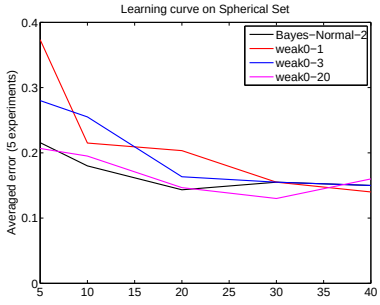
classifier.

Exercise 5.17

(a) For `nmc`, adding data does not help much (see (a) below).
(b) For `qdc`, adding training data does help (b).



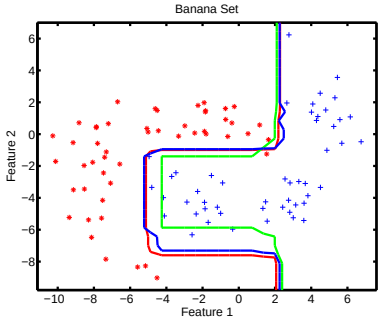
(a)



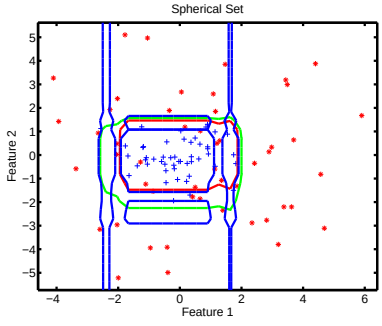
(b)

Exercise 5.18

Results for `stumpc` look like this:



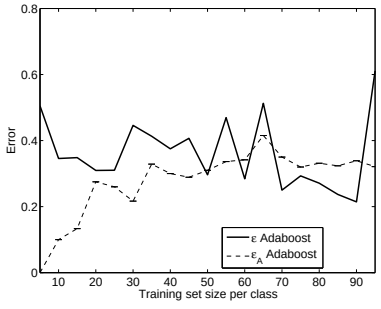
(a)



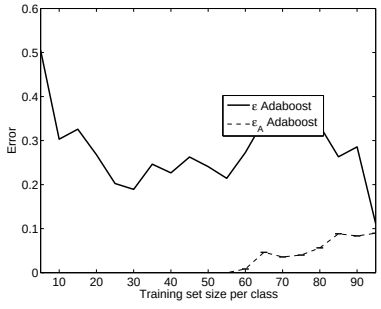
(b)

Exercise 5.19

The results for 5 and 100 boosting steps may look like this:



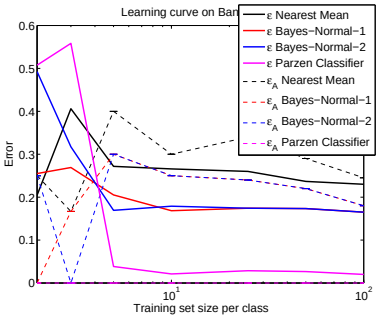
(a)



(b)

Exercise 5.20

(a) The result may look like:



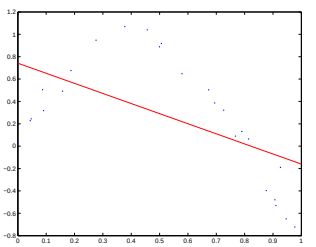
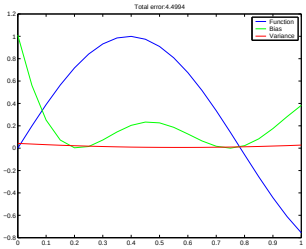
(b) In the graph above, `nmc` and `ldc`/`qdc`, respectively.
(c) The Parzen classifier improves most with increasing sample size.
(d) `nmc` shows the smallest difference between apparent and true error, `parzenc` the largest.

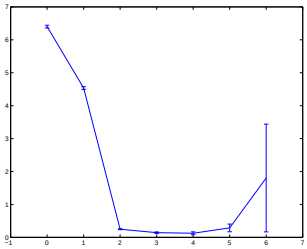
Exercise 5.21

(a) We can only generate a short learning curve, due to the small sample size. The curves are very noisy. Some true errors seem to go to 0; this could be due to selection bias (if you selected the interesting features on the training set only).
(b) The learning curves become more clear, the true errors much higher. It seems selection bias did play a role. This means the feature selection should ideally be incorporated in the call to `cleval` if no test set is used.

Exercise 5.22

(a) The plot should look like figure (a) below. The line is not a good fit (high bias), but does not change much for different datasets (low variance).


(a)


(b)

(c)

(b) The output should look like figure (b) above.
(c) Variance is measured as the squared deviation between the output and the average output. It measures how much the output varies when the input data varies, i.e. the noise sensitivity.
(d) Squared bias is measured as the squared deviation between the average output and the true output. It measures how much, on average, the output differs from the truth, i.e. the model's inflexibility.
(e) The line gives a reasonable fit in the middle, but a poor one at the ends of the data. The variance is low because the line model is too simple to fit completely to the data. Therefore, when the data set changes, the line does not change too much.
(f) The errorbar should look like figure (c) above. Here, the optimum is found for `deg = 3`, but it may be different.