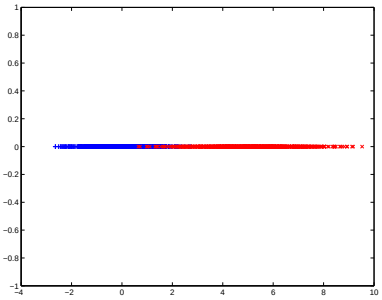
Solutions to exercises of Day 1

Exercise 1.10

The `gendatk` and `gendatp` routines assume local distributions of the data. Based on the small number of samples given, they cannot generalise the underlying distribution well.

Exercise 1.11



Exercise 1.12

(a) 0.5.
(b) A Gaussian probability density function.
(c) $\mu = 0.5$ and $\sigma^2 = \frac{1}{12n} \approx 0.0833/n$ (remember the definition of the standard error of the mean).
(d) The mean will be approximately the same, the variance will be lower.
(e) Nothing changes, only the histogram may become slightly more smooth. We have just sampled a bit more from the Gaussian; the individual estimates haven't improved.

Exercise 1.14

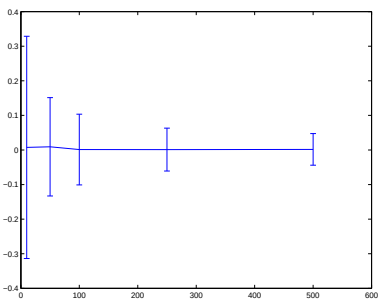
They are random. Because the dataset is spherical, any set of eigenvectors will do.

Exercise 1.15

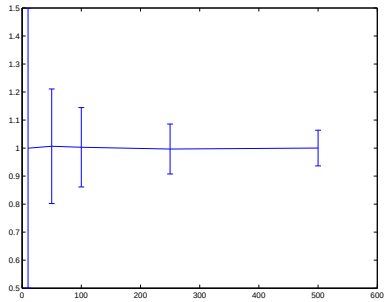
(a) Now you should find unit eigenvectors aligned with the axes, and eigenvalues approximately 3 and 1.

Exercise 1.17

(a) The resulting graphs should look like this (mean in figure (a), standard deviation in (b):)



(a)



(b)

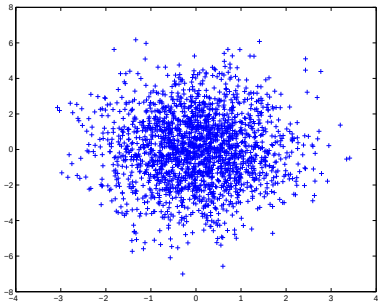
(b) For these settings, nothing much seems to improve for more than 250 samples.

Exercise 1.18

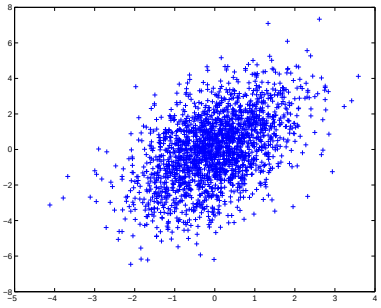
(a) The condition number should increase rapidly for more than 7 or 8 dimensions.

Exercise 1.19

(b) The diagonal elements are the variances of each of the variables (in this case there are two variables, say, x and y). The off-diagonal elements represent the covariances. For the first case, `Sigma = [1 0; 0 4]`, the covariance of x and y is zero. In figure (a), this results in a Gaussian distribution which is aligned with the axes. For the second case, `Sigma = [1 1; 1 4]`, x and y do co-vary, which is visible in the fact that the distribution is oriented at an angle with respect to the axes. We can also say that x and y are correlated.



(a)

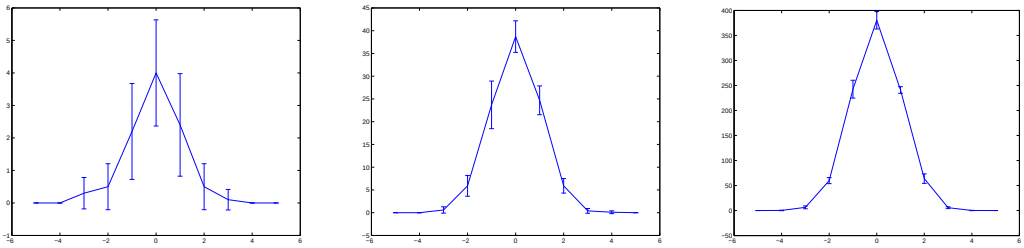


(b)

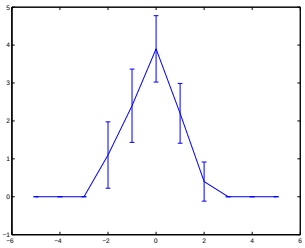
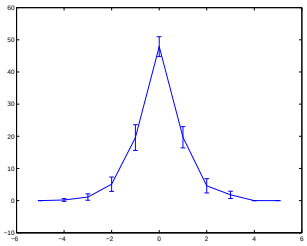
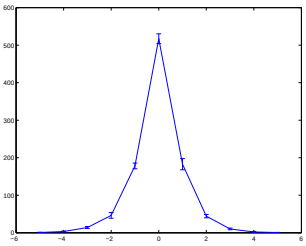
(c) The matrix is not symmetric: covariance of x and y is exactly the same as the covariance of y and x.
(d) The variance of the second variable, y, is much larger than the variance of x. Consequently, the distribution is stretched along the axis corresponding to y.

Exercise 1.20

(a) The output should look like this:



(b) With the script given, at least 1000 samples are needed.
(c) The same holds in general, but the difference is that for the Laplacian the mass is more concentrated around the mean and can therefore be estimated more easily (the histogram's standard deviation is lower w.r.t. its mean than for the Gaussian):

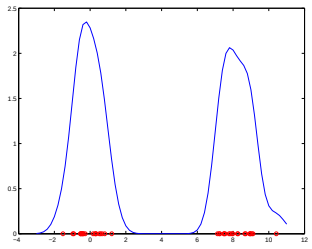
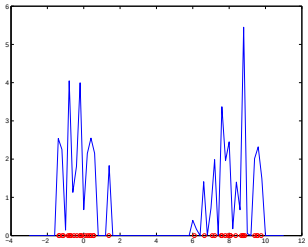
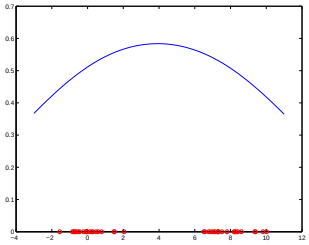
The errors in the tails are significantly larger, though.

Exercise 1.21

(a) Judged visually, $\mathcal{O}(10^4)$.
(b) For smaller bin sizes the variation between different realisations of the histograms is much larger. To get a stable result, for 5×5 bins, $\mathcal{O}(10^3)$ are needed; for 20×20, $\mathcal{O}(10^4 - 10^5)$.

Exercise 1.22

(d) The data consists of two Gaussian clusters with $\sigma = 1$. The width parameter should therefore be around 0.5 to 1. In figure (a) the estimate for $h = 0.5$ is shown while figures (b) and (c) depict the estimates for $h = 0.05$ (too small) and $h = 5$ (too large), respectively.


(a)

(b)

(c)

Exercise 1.23

(a) The log-likelihood increases as h decreases. In the plot in figure (a) below, h varies from 0.01 to 5, and the highest log-likelihood is achieved at $h = 0.01$. This is due to the fact that the log-likelihood is computed on the training set.

Exercise 1.24

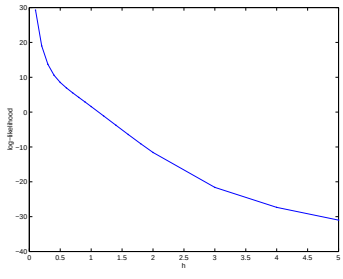
(a) These curves are depicted in figure (b) below, where the test set curve has a clear plateau between $h = 0.5$ and $h = 1$ (and not at $h = 0.01$ as in the training set curve). In the training set curve there is a clear over-training effect.

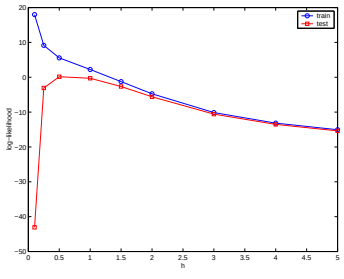
Exercise 1.25

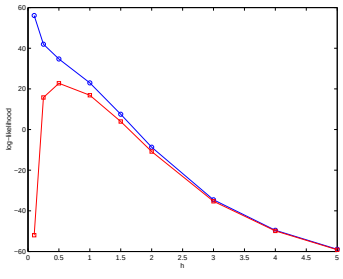
(a) It should be roughly in the same order.

Exercise 1.26

For more training examples, there is a clearer preference for a smaller width parameter. This is clearly visible in figure (c) below.



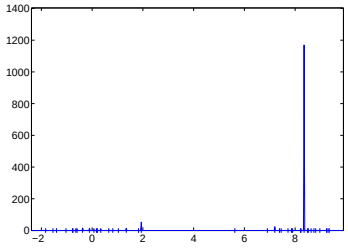
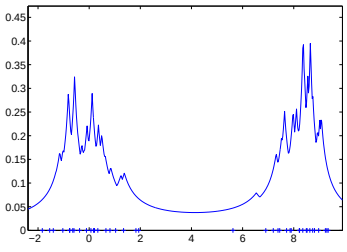
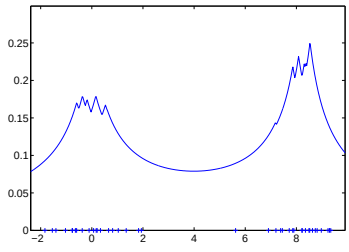
(a)


(b)


(c)

Exercise 1.27

(a) "Infinite peaks" around each sample; as the distance to each training sample goes to zero, the estimated density goes to infinity. You don't see that because we only calculate the density at certain grid points.
(c) Some example are plotted below: (a) $k = 1$, (b) $k = 5$ and (c) $k = 15$. For $k = 5$, the density looks reasonable.


(a)

(b)

(c)

(d) Using cross-validation, just as for h in Parzen density estimation.

Solutions to exercises of Day 2

Exercise 2.1

(c) `w1=gaussm(dataset(a1)); w2=gaussm(dataset(a2)); scatterd(a,'legend');`. Visualize using `plotm(w1,2);plotm(w2,2);`.
(d) Something like: `[getlab(b) +phat1 +phat2]`.
(e) 27/38 and 11/38, respectively.
(f) `P = [(27/38)*phat1 (11/38)*phat2]; [dummy, lab2] = max(P,[],2);` .

Exercise 2.2

Simply replace `gaussm` by `parzenm`.

Exercise 2.4

(a) The classification error is slightly higher on the training set than on the test set.
(b) Also the cross-validation error is higher than the classification error on the test set.

Exercise 2.5

One could select differentially expressed genes with `anova1`.

Exercise 2.6

(a) $p(\omega_2|\boldsymbol{x}) = 1 - p(\omega_1|\boldsymbol{x})$
(b)

$$\begin{aligned}
\log\left(\frac{p(\omega_1|\boldsymbol{x})}{p(\omega_2|\boldsymbol{x})}\right) &= \beta_0 + \beta^T\boldsymbol{x} && (1)\\
\frac{p(\omega_1|\boldsymbol{x})}{p(\omega_2|\boldsymbol{x})} &= \exp\left(\beta_0 + \beta^T\boldsymbol{x}\right) && (2)\\
\frac{1 - p(\omega_2|\boldsymbol{x})}{p(\omega_2|\boldsymbol{x})} &= \exp\left(\beta_0 + \beta^T\boldsymbol{x}\right) && (3)\\
\frac{1}{p(\omega_2|\boldsymbol{x})} - 1 &= \exp\left(\beta_0 + \beta^T\boldsymbol{x}\right) && (4)\\
\frac{1}{p(\omega_2|\boldsymbol{x})} &= \exp\left(\beta_0 + \beta^T\boldsymbol{x}\right) + 1 && (5)\\
p(\omega_2|\boldsymbol{x}) &= \frac{1}{1 + \exp\left(\beta_0 + \beta^T\boldsymbol{x}\right)} && (6)
\end{aligned}$$

Again $p(\omega_2|\boldsymbol{x})$ should not be zero.

Exercise 2.7

(a) Two overlapping circular-shaped distributions with a distance of 1 between their means.
(b) Each of the classes has a Gaussian class-conditional probability distribution with equal covariance matrices. The decision boundary will therefore be linear. The logistic classifier also gives a linear decision boundary, so it is a good candidate model. However, due to the overlap between the classes the classification error can be substantial.
(c) The second dataset is practically linearly separable. The result is that the logistic classifier will give a very steep transition from $p(\omega_1|\boldsymbol{x}) = 0$ to $p(\omega_1|\boldsymbol{x}) = 1$. In the case the data is

overlapping, as in the first dataset, the slope is much less steep.

Exercise 2.8

(a) The decision boundary of the Gaussian classifier is quadratic which does not fit the banana data. So, many classification errors will be made. The Parzen or the kNN classifier should work much better. The difference between the latter two classifiers will not be large.
(b) First fit the models: `w1 = qdc(a);w2 = parzenc(a);w3 = knnc(a);`. Then plot the decision boundaries: `scatterd(a);w = {w1,w2,w3}; plotc(w1); plotc(w2,'red'); plotc(w3,'blue');`and estimate the classification error: `testc(a*w);`.

Exercise 2.9

Increasing these parameters gives smoother decision boundaries. This leads to increased classification error (on the training data, at least)

Exercise 2.10

(a) The second parameter indicates the dimensionality of the dataset, the third indicates the distance between the classes in the first dimension.
(b) The `knnc` should have zero error on the training set, but a very large error on the test set. This indicates that the classifier is overtrained.
(c) In the help is stated that all features, except for feature 1, are not informative. The class overlap is therefore constant for varying dimensionality and the Bayes error does not change.
(d) The performance on the test set in general deteriorates: the `knnc` is actually hampered by the extra, superfluous dimensions.
(e) The test error goes up for increasing dimensionality. A dimensionality of 2 or 3 is optimal.
Example code:

```
err = [];
dims = [2 3 5 10 25 100]; % array of dims to try
nrepeats = 25; % number of repeats
for i=dims
   total = 0;
   for j=1:nrepeats
     a = gendats([10 10],i,2); % generate training data
     b = gendats([1000 1000],i,2); % generate test data
     w1 = knnc(a,1); % train a one nearest-neighbor classifier
     total = total + +testc(b*w1); % calculate error on test data
   end
   err= [err total/nrepeats];
end
plot(dims,err) % plot test error as function of dimensionality
```

Exercise 2.11

(a) We have to recall the two equalities: $A\boldsymbol{x} = (\boldsymbol{x}^T A^T)^T$ and $(\Sigma_i^{-1})^T = \Sigma_i^{-1}$ (because Σ is a symmetric matrix). Using this we can already simplify $(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) = \boldsymbol{x}^T\Sigma_i^{-1}\boldsymbol{x} - \boldsymbol{x}^T\Sigma_i^{-1}\boldsymbol{\mu} - \boldsymbol{\mu}^T\Sigma_i^{-1}\boldsymbol{x} + \boldsymbol{\mu}^T\Sigma_i^{-1}\boldsymbol{\mu}$. Rewrite: $\boldsymbol{x}^T\Sigma_i^{-1}\boldsymbol{\mu} = (\boldsymbol{x}^T\Sigma_i^{-1})\boldsymbol{\mu} = (\Sigma_i^{-1^T}\boldsymbol{x})^T\boldsymbol{\mu} = \boldsymbol{\mu}^T\Sigma_i^{-1}\boldsymbol{x}$ and it becomes $= \boldsymbol{x}^T\Sigma_i^{-1}\boldsymbol{x} - 2\boldsymbol{\mu}^T\Sigma_i^{-1}\boldsymbol{x} + \boldsymbol{\mu}^T\Sigma_i^{-1}\boldsymbol{\mu}$.
Now if we look at $g_1(\boldsymbol{x}) - g_2(\boldsymbol{x})$: $g_1(\boldsymbol{x}) - g_2(\boldsymbol{x}) = \log(p(\omega_1)) - \log(p(\omega_2)) - \frac{1}{2}\log\det(\Sigma_1) +$

$\frac{1}{2}\log\det(\Sigma_2) - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\Sigma_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^T\Sigma_2^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2) = \log(p(\omega_1)) - \log(p(\omega_2)) - \frac{1}{2}\log\det(\Sigma_1) + \frac{1}{2}\log\det(\Sigma_2) - \frac{1}{2}\boldsymbol{x}^T\Sigma_1^{-1}\boldsymbol{x} + \boldsymbol{\mu}_1^T\Sigma_1^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_1^T\Sigma_1^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{x}^T\Sigma_2^{-1}\boldsymbol{x} - \boldsymbol{\mu}_2^T\Sigma_2^{-1}\boldsymbol{x} + \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma_2^{-1}\boldsymbol{\mu}_2$. Collecting the terms from the equation, we obtain for the quadratic term: $W = \frac{1}{2}(\Sigma_2^{-1} - \Sigma_1^{-1})$; for the linear term: $\boldsymbol{w}^T = \boldsymbol{\mu}_1^T\Sigma_1^{-1} - \boldsymbol{\mu}_2^T\Sigma_2^{-1}$; and for the bias: $w_0 = \log\frac{p(\omega_1)}{p(\omega_2)} + \frac{1}{2}\log\frac{\det\Sigma_2}{\det\Sigma_1} - \frac{1}{2}\boldsymbol{\mu}_1^T\Sigma_1^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma_2^{-1}\boldsymbol{\mu}_2$.

Exercise 2.12

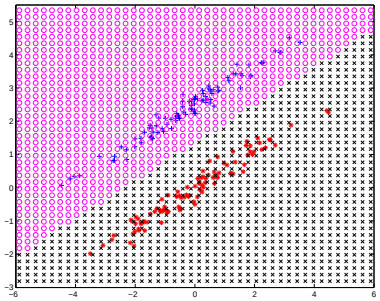
(a) We are using the results from the previous exercise, and substitute $\Sigma^{-1} = \Sigma_1^{-1} = \Sigma_2^{-1}$. The quadratic term disappears $W = 0$, and we only have $\boldsymbol{w}^T = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\Sigma^{-1}$ and $w_0 = \log\frac{p(\omega_1)}{p(\omega_2)} - \frac{1}{2}\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2$.

Exercise 2.13

(a) Define the function $h_i(\boldsymbol{x}) = (\boldsymbol{x}-\boldsymbol{\mu}_i)^2 = \boldsymbol{x}^T\boldsymbol{x} - 2\boldsymbol{\mu}_i^T\boldsymbol{x} + \boldsymbol{\mu}_i^T\boldsymbol{\mu}_i$, similar to the Gaussian-based classifiers. The discriminant function becomes $h_1(\boldsymbol{x}) - h_2(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{x} - 2\boldsymbol{\mu}_1^T\boldsymbol{x} + \boldsymbol{\mu}_1^T\boldsymbol{\mu}_1 - \boldsymbol{x}^T\boldsymbol{x} + 2\boldsymbol{\mu}_2^T\boldsymbol{x} - \boldsymbol{\mu}_2^T\boldsymbol{\mu}_2 = 2(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T\boldsymbol{x} + \boldsymbol{\mu}_1^T\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^T\boldsymbol{\mu}_2$.
(b) We simplify the covariance matrix even further by $\Sigma = \sigma^2\boldsymbol{I}$, and we get that $\Sigma^{-1} = \sigma^{-2}\boldsymbol{I}$. This results in $\boldsymbol{w}^T = \sigma^{-2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T$ and $w_0 = \log\frac{p(\omega_1)}{p(\omega_2)} - \frac{1}{2\sigma^2}(\boldsymbol{\mu}_1^T\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^T\boldsymbol{\mu}_2)$. This is identical to the nearest mean classifier, up to a scaling factor of $-\frac{1}{2\sigma^2}$.

Exercise 2.14

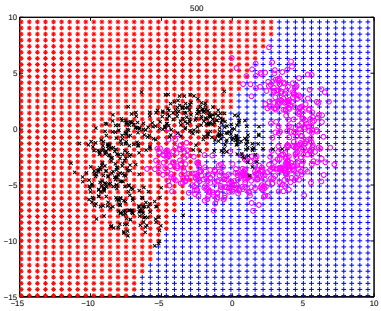
(a) Linear. The `ldc` will therefore fit best with a straight line separating the clusters:



With sufficient data a quadratic classifier also gives good results, but with a smaller training set the performance might become worse.

Exercise 2.15

(a) The class-conditional densities are clearly not Gaussian. See the figure below. Since this boundary is not the optimal boundary, the resulting error rate is relatively high, approximately 15%.



(b) The optimal decision boundary is non-linear and non-quadratic and can not be approximated by assuming two Gaussian classes.

Exercise 2.17

As explained in the lecture, for a two-class problem these classifiers are identical.

Exercise 2.18

(a) The inverse sigmoidal transforms the posterior probabilities `a*w` into the original values of the discriminant function. For *any* linear discriminant function this is proportional to the distance from x_i to the decision boundary:

$$d(x_i, \text{decision boundary}) = \frac{\mathbf{w}^T \mathbf{x}_i + w_0}{||\mathbf{w}||}.$$

(b) Code for computing the Fisher criterion given the distances to the decision boundary `d`:

```
% calculate the mapped mean for class 1
mu1 = mean(seldat(d,1));
% calculate the mapped mean for class 2
mu2 = mean(seldat(d,2));
% calculate the Fisher criterion J as described in the lecture
crit = (abs(mu1 - mu2)^2)/(sum((seldat(d,1)-mu1).^2)+sum((seldat(d,2)-mu2).^2));
```

(c) Using the code given under (b), you can also calculate the Fisher criterion for the `nmc`. Given the same data, the Fisher criterion for the Fisher classifier is systematically higher than for the nearest mean classifier.

Exercise 2.19

(c) The decision boundary clearly shows the axis-parallel splits of the feature space.
(d) For the default purity-based splitting criterion any degree of early pruning, that is `prune`>0, is equally good and gives a test error of 0. The optimal model is a decision tree consisting of only two splits and sacrifices some performance on the training data for a simpler model with better performance on the test data. Other splitting criteria show similar results.
(e) The influence of the splitting criteria on the test error is substantial. The default purity-based splitting criterion gives a test error of approximately 30%, while the Fisher criterion gives a test error of about 3%.

Exercise 2.20

(c) The quadratic, linear and Fisher discriminant are not affected, the nearest mean and the Parzen density classifier are. This is because in the first three classifiers a covariance matrix

$\hat{\Sigma}$ is estimated. Therefore, the scale of the feature values is automatically estimated in these classifiers. Decision trees are also scale-insensitive by design, since the splitting criteria used are scale-insensitive. In principle, the 1-nearest neighbor classifier is not scale-insensitive but for this particular dataset scaling does not seem to influence performance.
(d) In many cases it is an advantage, because it means that you don't need to optimize the scaling of the features. The disadvantage is that the training set should be large enough to make the estimation of the scale reliable.

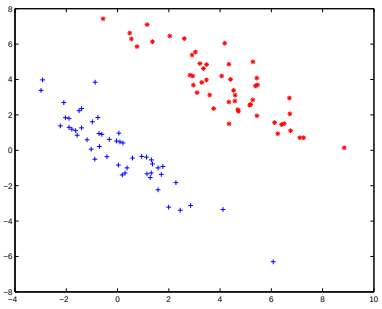
Solutions to exercises of Day 3

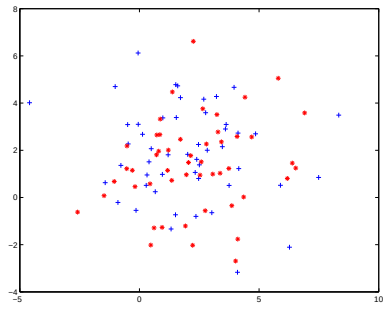
Exercise 3.1

(a) Yes, features 5 and 1.
(b) Different algorithms give different subsets. Feature 3 is always selected, though.

Exercise 3.2

(a) The scatterplot should look like the figure (a) below, not like (b):


(a)


(b)

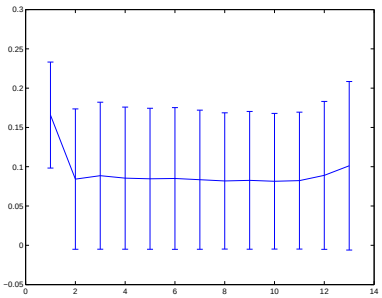
(b) Yes.

Exercise 3.3

(a) Either 2, or a range between 2 and 3-5. In either case, for more than (say) 6 features the error should increase. In figure (a) below, average and standard deviation over 5 experiments are shown.

Exercise 3.4

(a) Anything, as long as it is not feature 12, since it's quite politically incorrect. Features 2, 3, 6, 7, 10 and 13 look good.
(b) Somewhere around 3, most probably a range between 3 and 9 or so. It depends on the particular split, see the figure below.
(c) Best: 13, 10, as predicted. Worst: 12. The results can change quite a bit when repeated.
(d) They stay globally the same, taking into account that the results change quite a bit during repeated running.



(e) Suddenly feature 12 is ranked higher. Also, the optimal number of features is much higher, sometimes even 13. This is caused by using the non-optimal `'maha-s'` criterion in

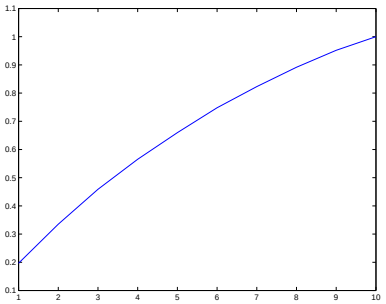
conjunction with the 1-nearest neighbour classifier.

Exercise 3.5

(a) Over 2000, depending on the random splits of the cross-validation.
(b) If you pick $\Delta = 2.5$, you will end up with around 25 genes. The test set error may actually decrease.
(c) $\Delta = 3$ may give roughly 7 genes, with a still reasonable test error.
(d) In general, for high-dimensional data with small sample size such as microarray data, the simplest classifiers perform best.

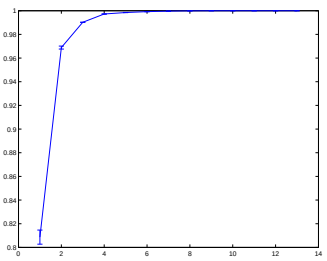
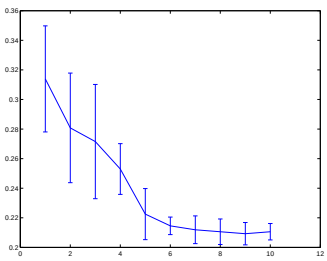
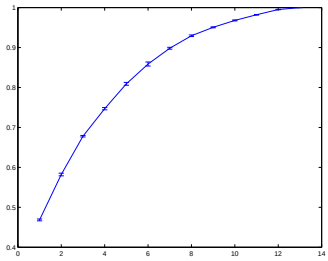
Exercise 3.6

(a) This plot tells us that PCA is not very informative in this case, as the explained variance is nearly the same for all principal components:



Exercise 3.7

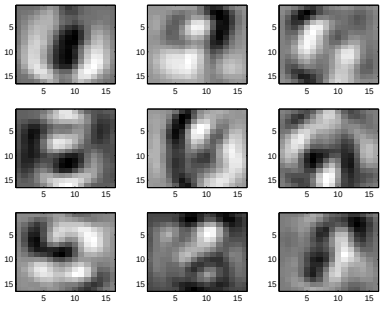
(b) They are higher, because they are not optimised with respect to classification (figure (b) below).
(c) The answer can vary quite a bit, but it will be more than 2. In figure (b) below, roughly 6 dimensions suffice.
(d) The intrinsic dimensionality seems to lie between 2 and 4 (figure (a) below, note the scale of the retained variance-axis).
(e) The first basis vector is (nearly) feature 10. `var(a)`shows that the variance of this feature is huge compared to that of the other features, so PCA will find this dimension as the first component.
(f) No, now the intrinsic dimensionality should be around 8 (figure (c)):

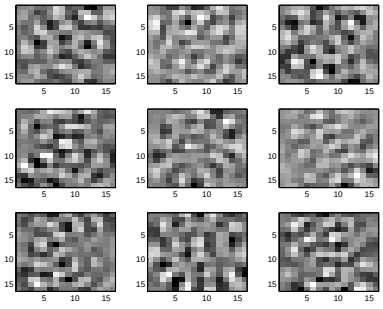

(a)

(b)

(c)

Exercise 3.8

(a) Around 50.
(b) First global blobs (figure (a) shows basis vectors 1-9), then later (figure (b), 51-59) higher

frequencies:


(a)

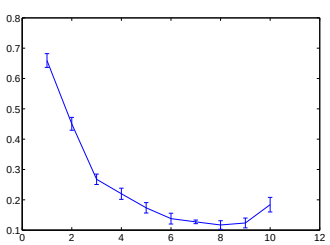

(b)

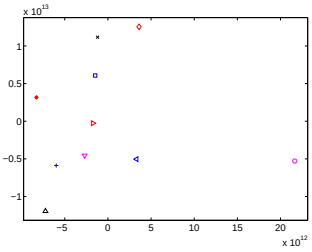
Exercise 3.9

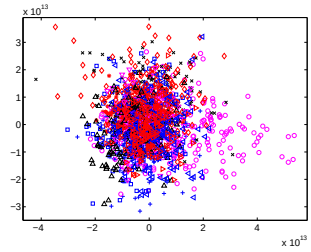
(a) Because the data has 3 classes, `fisherm` can map to at most 3-1 = 2 dimensions.

Exercise 3.10

(a) There are 10 classes, so at most 10-1 = 9 dimensions.
(b) Between 6 and 8, see figure (a) below.
(c) Performance becomes much worse, from less than 20% error to 50%.
(d) All samples in a single class are mapped onto the exact same point (the within-scatter is zero), so there are only 10 points, as in figure (b) below. This is possible since there are only a small number of points (200) in 256 dimensions.
(e) The test data is highly overlapping in the mapped space (figure (c) below). The solution found on the training data is overtrained: it fits the training data perfectly, but does not generalise.

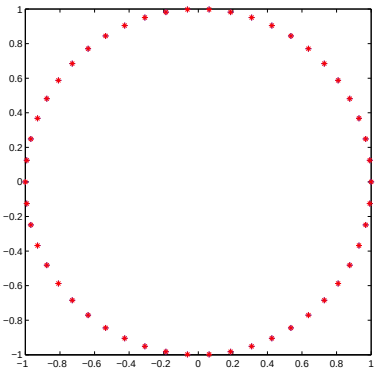
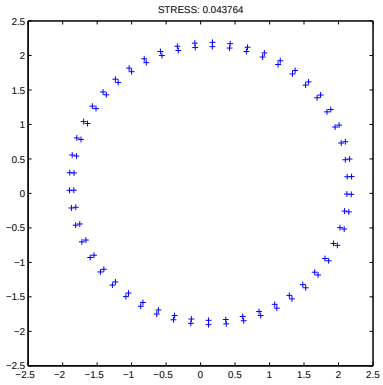

(a)

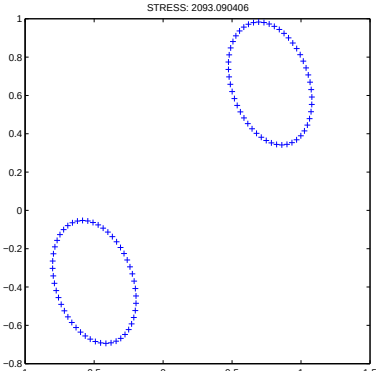

(b)

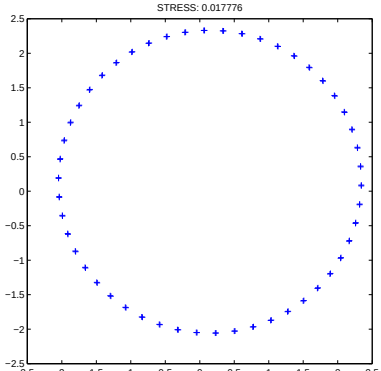

(c)

Exercise 3.11

(a) You will see one circle: the two circles have been plotted exactly on top of each other (figure (a) below).
(b) Two circles are visible. Initially, they are twisted a bit; they end up being of different size (figure (b)).
(c) For `opt.q = -2`, the circles are more distorted and further apart (figure (c)); for `opt.q = 2`, the circles are perfectly round but on top of each other (figure (d)). This follows from the definition of the stress measure. The resulting stress values are very different, because they are incomparable.

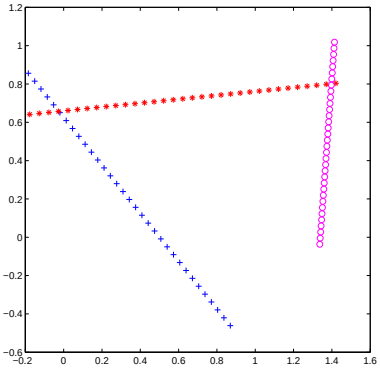
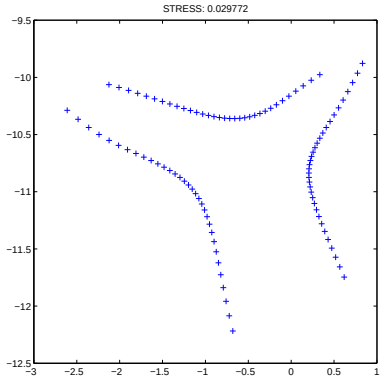

(a)


(b)


(c)


(d)

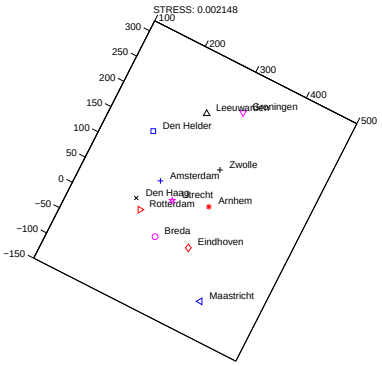
(d) The algorithm starts very wild, but should end up in the same configuration. It may hit a local optimum, and will take (much) longer to converge.
(e) PCA (figure (a)) lets the lines overlap. Obviously, MDS (figure (b)) distorts the fact that they are lines:


(a)


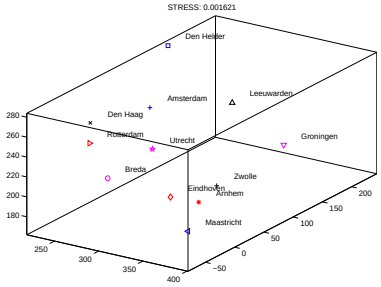
(b)

Exercise 3.12

(a) `opt.q = 0` should give reasonable results (figure (a) below).
(b) The cities do not lie in a plane (figure (b)), which the Netherlands should approximately be. This means there are residual stresses due to the distance measure used (over roads) which cannot be removed in 2D.



(a)



(b)

(c) There will be more emphasis on the applicable distances, sometimes resulting in suboptimal solutions. The algorithm may take a long time to converge.

Exercise 3.13

For $d = 1$, the stress is very large; for $d = 2$ and $d = 3$, it is getting smaller; for $d > 3$, it is the same as for $d = 3$. In other words, the distance matrix can be embedded reasonably well in 2D and perfectly in 3D and higher, which is to be expected.

Exercise 3.14

MDS should show better clusters than PCA.

Solutions to exercises of Day 4

Exercise 4.1

(a) At the finest level, there are 14 clusters. These small clusters are organised in three larger clusters.
(b) This is an artificial problem. However, the desired number of clusters will usually depend on the application. If the small clusters have a physical meaning, such as e.g. individual species while the three larger clusters represent animal kingdoms, then the specific problem which the biologist is studying will determine the appropriate number of clusters, i.e. the level of detail considered.

Exercise 4.2

(b) There are no vertical stems that are distinctly longer than the other. The tree grows gradually, not in leaps and bounds.
(c) In terms of the lengths of the stems there is no difference.

Exercise 4.3

(a) These lengths correspond to the distances (complete linkage) between the two clusters being joined by the bridge consisting of two stems and a horizontal bar.
(b) The closer together the samples in a cluster and the further apart the clusters, the better the clustering.
(c) Long stems correspond to large distances between clusters and therefore a potential point to cut the dendrogram.

Exercise 4.4

(a) Yes, in the single linkage dendrogram, there are basically two lengths of vertical stems: those indicating the joining of the smallest clusters, and those indicating the joining of the larger three clusters. This is less pronounced with complete linkage.
(b) The average linkage dendrogram is roughly similar to the complete linkage dendrogram.

Exercise 4.5

(a) Two is a good solution, since there are two clear elongated Gaussians.
(b) Yes, the two cigars are separated, although some outliers make the interpretation of the clustering solution more complex.
(c) No, complete linkage does not succeed in finding the two cigar-shaped clusters, since its linkage scheme favors circular clusters. Half of the samples from the other cigar are closer (in the complete link sense) than the rest of the samples in the same cluster.

Exercise 4.6

(a) Two clusters appear OK, but a case could be made for some other configurations: the dendrogram is not absolutely clear.
(b) Absolutely not; there are two outliers in one cluster and the rest are in the other cluster.

(c) This is unclear, since there is little structure in the dendrogram.

Exercise 4.8

(a) When samples become prototypes, the prototype is always amongst the samples. This obviously reduces the number of possible initial conditions and has the disadvantage of being more susceptible to suboptimal local minima. An advantage is that it reduces the possibility of a situation where one prototype ends up without any samples.
(b) K-means is better suited for datasets with spherical clusters, and should therefore work better on the `messy` dataset.

Exercise 4.9

(b) The algorithm has converged, i.e. no alteration of the prototype positions results in a decrease of the within-scatter. However, the within-scatter of this solution is significantly higher than the within-scatter associated with solutions where there is a single prototype per cluster. In order to reach the global minimum, where each prototype is in a single cluster, one of the prototypes now sharing a cluster with another prototype needs to move towards the cluster without a prototype. In order to achieve this, the distances between the objects already assigned to the prototype that needs to move will increase, which will result in an *initial* increase in the within-scatter. Since the K-means algorithm only decreases the within-scatter, it remains stuck in this local minimum.
(c) It does not suffer from this particular problem, since it is a deterministic algorithm, independent of the random initial conditions. However, one might argue that most variants of hierarchical clustering avoid this problem by simply not minimizing an objective function at all. Note that also for hierarchical clustering randomization can be useful, for example to assess cluster stability.
(d) Run the algorithm several times with different initial settings, and choose the best result.

Exercise 4.10

A large jump in the fusion graph implies that two clusters were merged that were, relative to the other cluster distances, quite far apart.

Exercise 4.11

(a) From $g = 2$ to $g = 3$.
(b) The fusion graph is ambiguous about the exact number of clusters: either two or three clusters are associated with large fusion jumps of roughly the same magnitude. This is due to the complete linkage distance: the distance between furthest samples of clusters at $(0, 0)$ and $(1, 1)$ (which are merged to result in a total of two clusters) is roughly half the distance between furthest samples in the merged cluster at $(0, 0)/(1, 1)$ and the cluster at $(2, 2)$, which are merged to result in a single cluster. In single linkage this does not occur, since the minimal distances between the clusters at $(1, 1$ and $(2, 2)$ is the same as the minimal distance between the merged cluster at $(1, 1)/(2, 2)$ and the cluster at $(0, 0)$.

Exercise 4.12

(a) There are two pronounced jumps, at 3 and 14 clusters.

Exercise 4.13

(a) At three clusters, since the largest fusion jump occurs between $g = 2$ and $g = 3$.

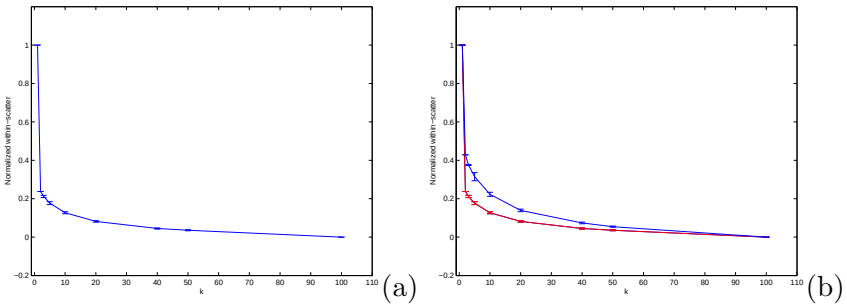
(b) No, three outliers constitute the members of two of the three clusters with all the other samples in the third cluster.
(c) While the size of the fusion jump between $g = 1$ and $g = 2$ is not very convincing (not much larger than the other jumps) the resulting clustering is better. This stems from the fact that complete linkage is less prone to outliers than single linkage.

Exercise 4.14

(b) Example code:

```
g = [1 2 3 5 10 20 40 50 100]; % number of clusters to try
nreps = 10; % number of repeats
for i=1:length(g)
   for j=1:nreps
     [p,labs,ws(i,j)] = kmclust(a,g(i),0,0); % store within-scatter matrix ws
   end
end
WS_mean = mean(ws,2); % calculate average within-scatter
WS_std = std(ws,0,2); % calculate standard deviation
errorbar(g,WS_mean/WS_mean(1),WS_std/WS_mean(1)); % plot normalized within scatter
```

(c) 1) The within-scatter (figure (a) below) decreases monotonically as g increases, and for $g = n$, it is zero. 2) The largest decrease in within-scatter occurs when we go from one to two clusters, revealing that there are probably two clusters in the data.



(a) (b)

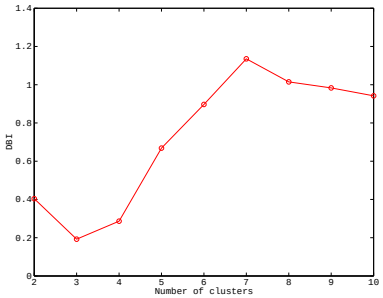
(d) For `d = 10` (red line in figure (b) above) the decrease in within-scatter when going from one cluster to two clusters is much larger than for `d = 5` (blue line). This indicates that for `d = 10` we have a "stronger" clustering than for `d = 5`. This is to be expected since the parameter `d` specifies the distance between class means in the first dimension.

Exercise 4.15

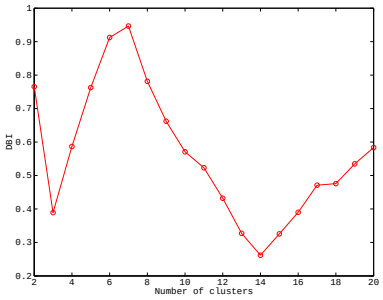
There should be no large jump between $g = 1$ and $g = 2$.

Exercise 4.16

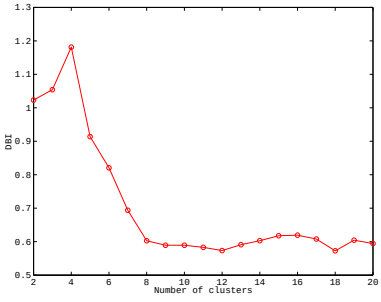
(b) It should have a minimum at three clusters.
(c) The DBI curve is shown in the figure below, with a clear minimum at three clusters.



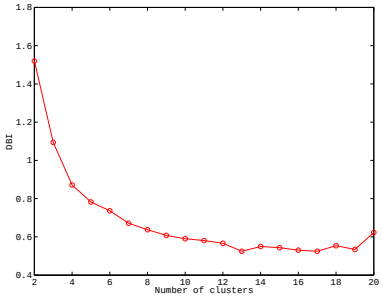
(d) There is a pronounced minimum at 3 clusters and a slightly lower minimum at 14 clusters. The first minimum (3 clusters) corresponds to the situation where the smaller clusters are grouped in three large clusters (four in top-left, four in top-right and six at the bottom) while the minimum at 14 corresponds to the fine cluster structure in the data. The valley at fourteen is more pronounced since the ratio of the maximal within-scatter to the minimal distance between any pair of these clusters is smaller than for the three cluster configuration.



(e) The DBI curve is not minimal at two clusters, and starts flattening at around eight clusters. Roughly the same result is obtained for complete and single linkage (figures (a) and (b) below). The problem is that DBI assumes *circular* clusters, while the cigars dataset consists of strongly elongated clusters.



(a)



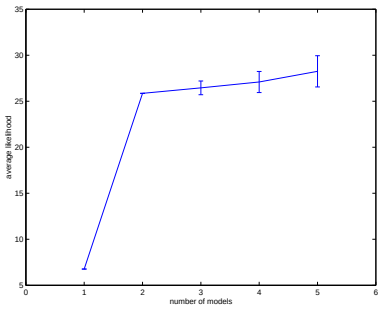
(b)

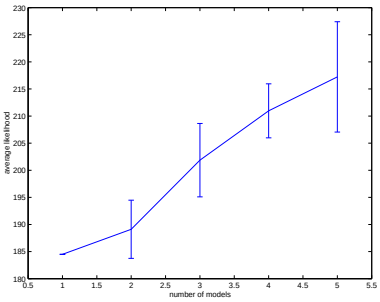
Exercise 4.17

(a) `'circular'`: diagonal covariance matrix with *equal* variances on the diagonal
`'aligned'`: diagonal covariance matrix with *unequal* variances on the diagonal
`'gauss'`: full covariance matrix, i.e. no constraints on the entries.
(b) The optimal g should be 2.
(c) When the likelihood curve flattens, it is an indication of very small increases in likelihood with more clusters. The point where it starts flattening is usually an indication of the desired number of clusters. The curve of likelihood as a function of the number of models is as shown

in figure (a) below. Note that results may vary due to local minima for $g = 2$ (and often the error bar will be larger). Example code:

```
ncomp = [1:5] % array of number of clusters to try
nreps = 20; % number of repeats
for i=ncomp
   for j=1:nreps
     lik(j) = em(a,i,'gauss',0,0,0); % fit unconstrained Gaussian mixture model
   end
   L_mean(i) = mean(lik); % calculate mean likelihood
   L_std(i) = std(lik); % calculate standard deviation
end
errorbar(ncomp,L_mean,L_std); % likelihood as function of the number of components
```

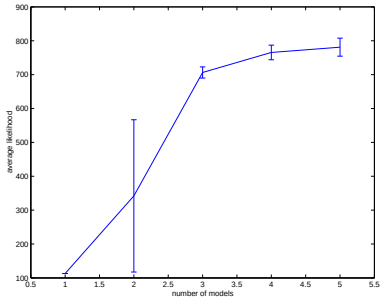


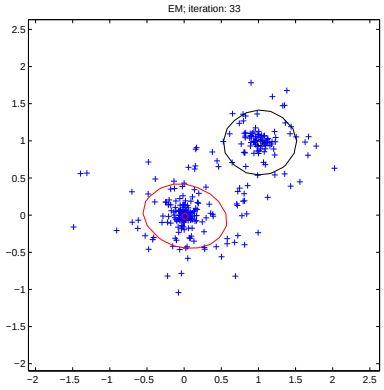
(a)


(b)

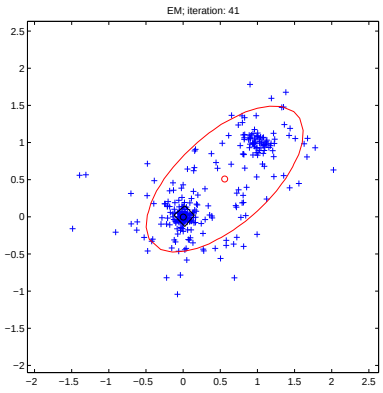
(d) The log-likelihood (figure (b) above) increases roughly linearly with the number of clusters, i.e. there is no clear flattening of the curve indicating a preference for a given number of clusters.

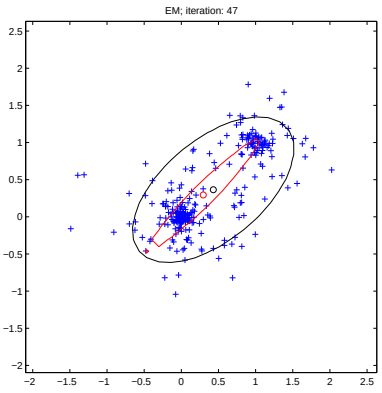
(e) The optimal number of clusters is two, with a covariance matrix of type 'circular' (the other types give similar performance). The log-likelihood vs. the number of clusters is depicted in figure (a), the desired solution with two clusters in (b) and two frequently occurring undesirable solutions with two clusters in figures (c) and (d). These undesirable solutions are the reason why the variance is so large at two clusters and why the curve does not flatten at two, but at three clusters.



(a)


(b)



(c)


(d)

Exercise 4.18

(a) The probabilities of the two sequences given the models are

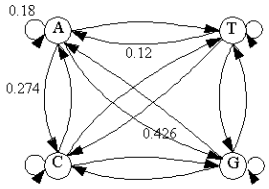
- IID:
  $P(CCGAT) = P(C)P(C)P(G)P(A)P(T) = 0.1 \times 0.1 \times 0.1 \times 0.2 \times 0.6 = 0.00012$
  $P(CATAT) = P(C)P(A)P(T)P(A)P(T) = 0.1 \times 0.2 \times 0.6 \times 0.2 \times 0.6 = 0.0014$

- Weight matrix:
  $P(CCGAT) = P_1(C)P_2(C)P_3(G)P_4(A)P_5(T) = 0.1 \times 0.2 \times 0.6 \times 0.1 \times 0.15 = 0.00018$
  $P(CATAT) = P_1(C)P_2(A)P_3(T)P_4(A)P_5(T) = 0.1 \times 0.3 \times 0.05 \times 0.1 \times 0.15 = 0.0000225$

- First-order Markov chain:
  $P(CCGAT) = P(C)P(C|C)P(G|C)P(A|G)P(T|A) = 0.1 \times 0.1 \times 0.1 \times 0.3 \times 0.05 = 0.000015$
  $P(CATAT) = P(C)P(A|C)P(T|A)P(A|T)P(T|A) = 0.1 \times 0.35 \times 0.05 \times 0.6 \times 0.05 = 0.0000525$

Exercise 4.19

(a) The sum of all terms in a row must be one, since this is the probability that the base corresponding to a specific row is followed by any of the bases.
(b) Element a_{ij} of the transition matrix is the probability $P(q_t = j|q_{t-1} = i)$ of going from state i to state j. In a Markov chain this is indicated by a directed edge from state i to state j (see figure below, only the transition probabilities of going out from state A are indicated).

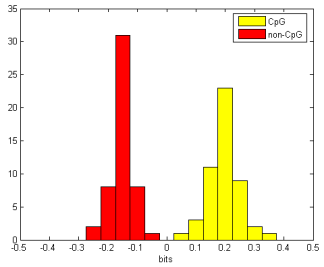


(c) The transition probability matrices were estimated on a modest number (50) of sequences and bases (on average 500 per sequence). Maximum likelihood estimates only converge to the true probabilities with much more training data. You might want to edit the script `cpgislands.m` to see the effect of increasing the number of sequences and the maximum number of bases on the estimated transition matrices.

(d) The log-likelihood ratio matrix of CpG versus non-CpG is

$$LL = \begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} -0.7370 & 0.41865 & 0.5799 & -0.8074 \\ -0.9131 & 0.3044 & 1.8126 & -0.6915 \\ -0.6233 & 0.4626 & 0.3316 & -0.7347 \\ -1.1638 & 0.5708 & 0.3951 & -0.6820 \end{pmatrix} \end{array}$$

Positive values correspond to transitions that are more likely in CpG islands and negative values to transitions that are more likely in non-CpG islands. Indeed, the log-odds for ratio for the transition from C to G is a large positive value. See the figure below for a histogram of the log-odds (in bits) for all sequences in the test set. CpG islands are shown in yellow and non-CpG islands are shown in red.



(e) Consecutive stretches of length at least 100 are: 19011–19123 and 63353–63492. You can use the script `cpgstretches.m` to find these. Similar results are also obtained with a simpler method implemented in `cpgplot` (`https://www.ebi.ac.uk/Tools/seqstats/emboss_cpgplot/`).
(f) The main disadvantage of our approach is the arbitrary fixed window size. One would expect CpG islands to be of arbitrary length and have sharp boundaries. A hidden Markov model is probably better since we can make a single model for the entire sequence (details in Durbin et al., pp. 51–53).

Exercise 4.20

We could use the full-fledged Viterbi algorithm on a trellis for this calculation. However, it is easier to note that only four paths can generate a sequence of length seven such as the consensus sequence `ACACATC`. Using the general equation for the joint probability of an observed sequence x and a state sequence Q in terms of transition probabilities a_{ij} and emission probabilities $b_j(x)$ (see lecture)

$$P(x, Q) = \prod_{t=1}^{N} P(q_t|q_{t-1}) \prod_{t=1}^{N} P(x_t|q_t) = \prod_{t=1}^{N} a_{q_{t-1},q_t} \prod_{t=1}^{N} b_{q_t}(x_t),$$

we have:

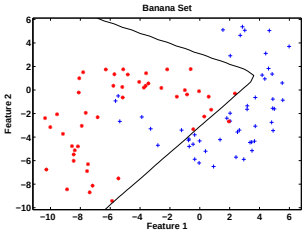
$P(ACACATC \ \& \ 1234567) = 0.8 \times 1 \times 0.8 \times 1 \times 0.8 \times 0.6 \times 0.4 \times 0.6 \times 1 \times 1 \times 0.8 \times 1 \times 0.8 = 0.047$
$P(ACACATC \ \& \ 1234456) = 0.8 \times 1 \times 0.8 \times 1 \times 0.8 \times 0.6 \times 0.4 \times 0.4 \times 0.2 \times 0.6 \times 0 \times 1 \times 0 = 0$
$P(ACACATC \ \& \ 1234445) = 0.8 \times 1 \times 0.8 \times 1 \times 0.8 \times 0.6 \times 0.4 \times 0.4 \times 0.2 \times 0.4 \times 0.2 \times 0.6 \times 0 = 0$
$P(ACACATC \ \& \ 1234444) = 0.8 \times 1 \times 0.8 \times 1 \times 0.8 \times 0.6 \times 0.4 \times 0.4 \times 0.2 \times 0.4 \times 0.2 \times 0.4 \times 0.4 = 0.00013$

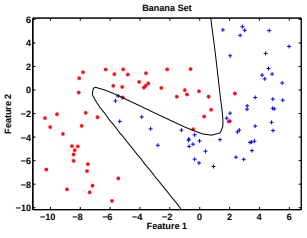
The optimal state sequence is the one with the highest probability: 1-2-3-4-5-6-7.

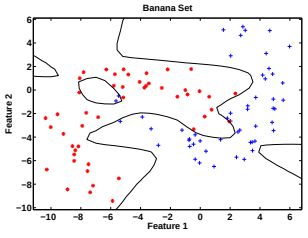
Solutions to exercises of Day 5

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)


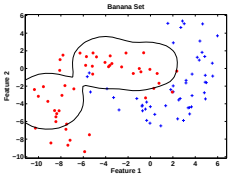
(b)


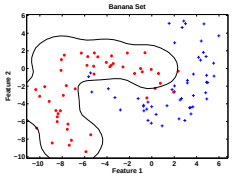
(c)

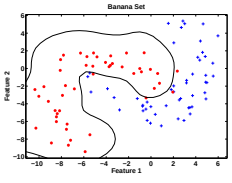
(c) The solution is overtrained, similar as before.
(d) 1 hidden layer of 3 units should suffice.

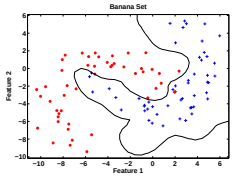
Exercise 5.2

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



(a)


(b)


(c)


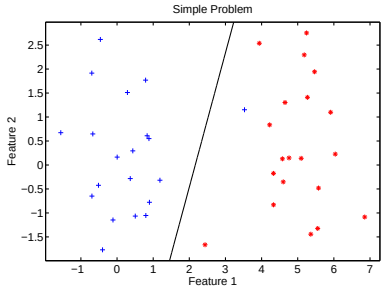
(d)

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:



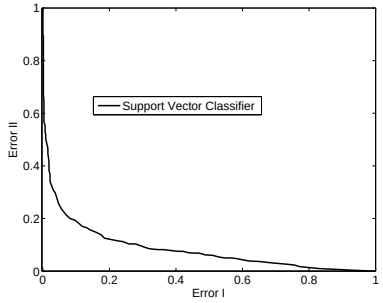
(b) The most simple way is probably to print out the coordinates of the suppport vectors: `+a(I,:)`. If you want to plot them into the figure, you have to enter `hold on; scatterd(a(I,:),'go')`.

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

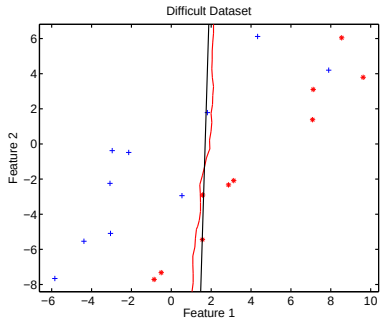
(a) Performance is best on the KL features (roughly 10% error), worst on the morphological features (over 50% error).
(b) This should be possible, but it is not easy to find better performance than using just the KL features.

Exercise 5.13

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

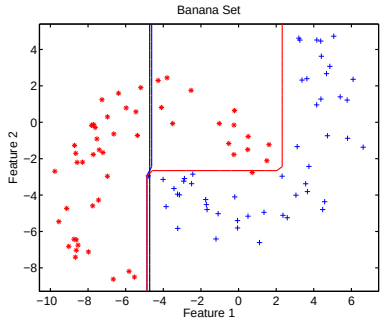
Exercise 5.14

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).



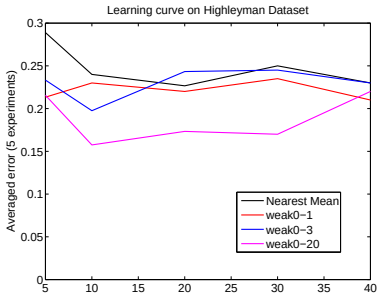
Exercise 5.16

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

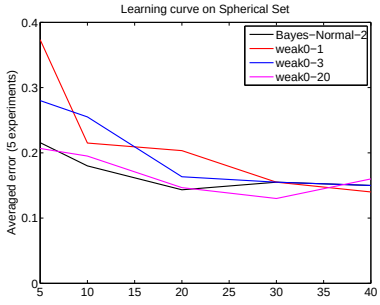
classifier.

Exercise 5.17

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



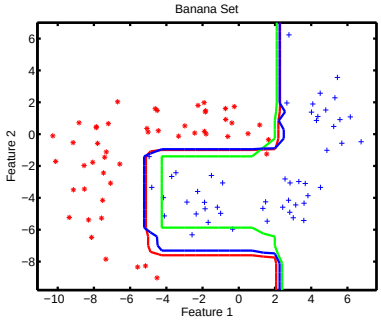
(a)



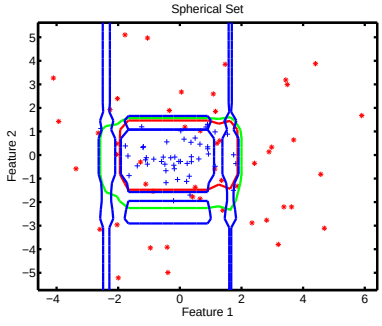
(b)

Exercise 5.18

Results for `stumpc` look like this:



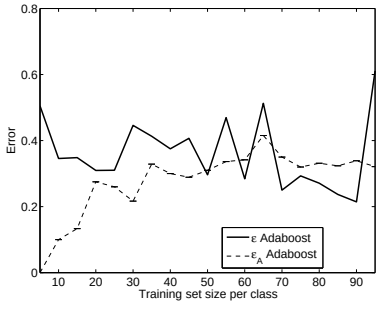
(a)



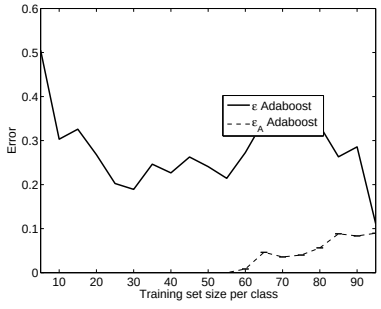
(b)

Exercise 5.19

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



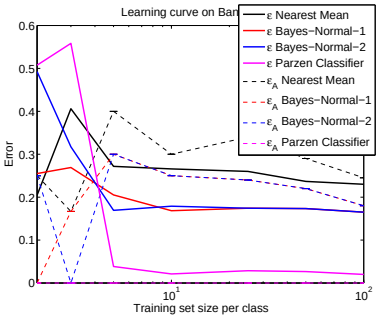
(a)



(b)

Exercise 5.20

(a) The result may look like:



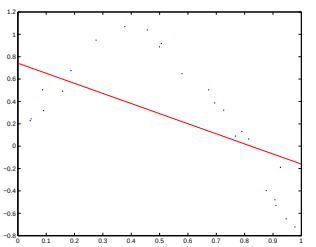
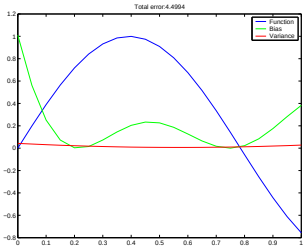
(b) In the graph above, `nmc` and `ldc`/`qdc`, respectively.
(c) The Parzen classifier improves most with increasing sample size.
(d) `nmc` shows the smallest difference between apparent and true error, `parzenc` the largest.

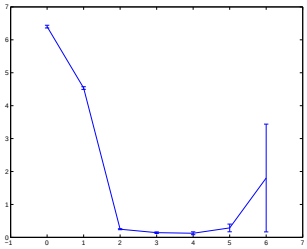
Exercise 5.21

(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.