# 4 *Parametric Methods*

1. *Write the code that generates a Bernoulli sample with given parameter $p$, and the code that calculates $\hat{p}$ from the sample.*

   The Matlab code is given in `ex4_1.m`.

2. *Write the log likelihood for a multinomial sample and show equation 4.6.*

   We add the constraint as a Lagrange term and maximize it:

$$
\begin{aligned}
J(p_i) &= \sum_i \sum_t x_i^t \log p_i + \lambda(1 - \sum_i p_i) \\
\frac{\partial J}{\partial p_i} &= \frac{\sum_t x_i^t}{p_i} - \lambda = 0 \\
\lambda &= \frac{\sum_t x_i^t}{p_i} \Rightarrow p_i \lambda = \sum_t x_i^t \\
\sum_i p_i \lambda &= \sum_i \sum_t x_i^t \Rightarrow \lambda = \sum_t \sum_i x_i^t \\
p_i &= \frac{\sum_t x_i^t}{\sum_t \sum_i x_i^t} = \frac{\sum_t x_i^t}{N} \text{ because } \sum_i x_i^t = 1
\end{aligned}
$$

3. *Write the code that generates a normal sample with given $\mu$ and $\sigma$, and the code that calculates $m$ and $s$ from the sample. Do the same using the Bayes' estimator assuming a prior distribution for $\mu$.*

   The Matlab code is given in `ex4_3.m` and we see an example output in figure 4.1.

4. *Given two normal distributions $p(x|C_1) \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $p(x|C_2) \sim \mathcal{N}(\mu_2, \sigma_2^2)$ and $P(C_1)$ and $P(C_2)$, calculate the Bayes' discriminant points analytically.*

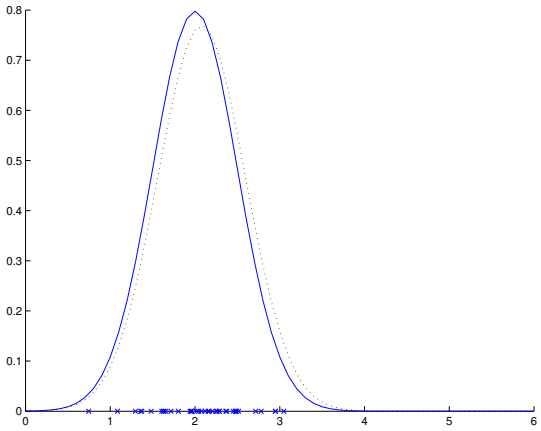


**Figure 4.1** Actual Gaussian density (continuous line), a sample drawn from it (×), and a Gaussian fitted to the sample (dotted line).

Given that

$$\begin{aligned} p(x|C_1) &\sim \mathcal{N}(\mu_1, \sigma_1^2) = \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left[-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right] \\ p(x|C_2) &\sim \mathcal{N}(\mu_2, \sigma_2^2) \end{aligned}$$

we would like to find $x$ that satisfy $P(C_1|x) = P(C_2|x)$, or

$$\begin{aligned} p(x|C_1)P(C_1) &= p(x|C_2)P(C_2) \\ \log p(x|C_1) + \log P(C_1) &= \log p(x|C_2) + \log P(C_2) \\ -\frac{1}{2}\log 2\pi - \log\sigma_1 - \frac{(x-\mu_1)^2}{2\sigma_1^2} + \log P(C_1) &= \cdots \\ -\log\sigma_1 - \frac{1}{2\sigma_1^2}\left(x^2 - 2x\mu_1 + \mu_1^2\right) + \log P(C_1) &= \cdots \\ \left(\frac{1}{2\sigma_2^2} - \frac{1}{2\sigma_1^2}\right)x^2 + \left(\frac{\mu_1}{\sigma_1^2} - \frac{\mu_2}{\sigma_2^2}\right)x + & \\ \left(\frac{\mu_2^2}{2\sigma_2^2} - \frac{\mu_1^2}{2\sigma_1^2}\right) + \log\frac{\sigma_2}{\sigma_1} + \log\frac{P(C_1)}{P(C_2)} = 0 & \end{aligned}$$

This is of the form $ax^2 + bx + c = 0$ and the two roots are

$$x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Note that if the variances are equal, the quadratic terms vanishes and there is one root, that is, the two posteriors intersect at a single $x$ value.

5. *What is the likelihood ratio*

$$\frac{p(x|C_1)}{p(x|C_2)}$$

*in the case of Gaussian densities?*

$$\frac{p(x|C_1)}{p(x|C_2)} = \frac{\frac{1}{\sqrt{2\pi}\sigma_1}\exp\left[-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right]}{\frac{1}{\sqrt{2\pi}\sigma_2}\exp\left[-\frac{(x-\mu_2)^2}{2\sigma_2^2}\right]}$$

If we have $\sigma_1^2 = \sigma_2^2 = \sigma^2$, we can simplify

$$\begin{aligned}\frac{p(x|C_1)}{p(x|C_2)} &= \exp\left[-\frac{(x-\mu_1)^2}{2\sigma^2} + \frac{(x-\mu_2)^2}{2\sigma^2}\right] \\ &= \exp\left[\frac{(\mu_1-\mu_2)}{\sigma^2}x + \frac{(\mu_2^2-\mu_1^2)}{2\sigma^2}\right] \\ &= \exp(wx + w_0)\end{aligned}$$

for $w = (\mu_1 - \mu_2)/\sigma^2$ and $w_0 = (\mu_2^2 - \mu_1^2)/2\sigma^2$.

6. *For a two-class problem, generate normal samples for two classes with different variances, then use parametric classification to estimate the discriminant points. Compare these with the theoretical values.*

The Matlab code is given in `ex4_6.m`, and its output plot is in figure 4.2.

Output:

```
Real values C1:(3.0,1.0)- C2:(2.0,0.3)
Estimates C1:(3.45,1.23)- C2:(2.15,0.26)
Actual intersect pts: 2.49 1.31
Estimated intersect pts: 2.64 1.53
```

3. *Generate samples from two multivariate normal densities $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), i = 1, 2$, and calculate the Bayes' optimal discriminant for the four cases in table 5.1.*

   The Matlab code is given in `ex5_3.m` and its output is given in figure 5.7 of the book.

4. *For a two-class problem, for the four cases of Gaussian densities in table 5.1, derive*

   $$\log \frac{P(C_1|\boldsymbol{x})}{P(C_2|\boldsymbol{x})}$$

   Using Bayes' rule, we have

   $$\log \frac{P(C_1|\boldsymbol{x})}{P(C_2|\boldsymbol{x})} = \log \frac{p(\boldsymbol{x}|C_1)}{p(\boldsymbol{x}|C_2)} + \log \frac{P(C_1)}{P(C_2)}$$

   Given that

   $$p(\boldsymbol{x}C_i) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_i|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\right]$$

   we have the following four cases: (a) $\boldsymbol{\Sigma}_1 \neq \boldsymbol{\Sigma}_2$:

   $$\log \frac{p(\boldsymbol{x}|C_1)}{p(\boldsymbol{x}|C_2)} = \log \frac{|\boldsymbol{\Sigma}_1|^{-1/2}\exp[-(1/2)(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)]}{|\boldsymbol{\Sigma}_2|^{-1/2}\exp[-(1/2)(\boldsymbol{x}-\boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)]}$$

   (b) $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}$:

   $$\begin{aligned}\log \frac{p(\boldsymbol{x}|C_1)}{p(\boldsymbol{x}|C_2)} &= -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2) \\ &= \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2) + \frac{1}{2}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_2+\boldsymbol{\mu}_1)\end{aligned}$$

   We can write

   $$\log \frac{P(C_1|\boldsymbol{x})}{P(C_2|\boldsymbol{x})} = \boldsymbol{w}^T\boldsymbol{x} + w_0$$

   where

   $$\begin{aligned}\boldsymbol{w} &= \boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2) \\ w_0 &= -\frac{1}{2}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2) + \log\frac{P(C_1)}{P(C_2)}\end{aligned}$$

(c) $\mathbf{\Sigma}$ diagonal with $\sigma_j^2, j = 1, \ldots, d$:

$$\log \frac{P(C_1|\boldsymbol{x})}{P(C_2|\boldsymbol{x})} = \sum_{j=1}^{d} \left( \frac{\mu_{1j} - \mu_{2j}}{\sigma_j} \right)^2 x_j - \frac{1}{2} \sum_{j=1}^{d} \left( \frac{\mu_{1j}^2 - \mu_{2j}^2}{\sigma_j^2} \right) + \log \frac{P(C_1)}{P(C_2)}$$

(d) $\mathbf{\Sigma}$ diagonal with $\sigma^2$:

$$\log \frac{P(C_1|\boldsymbol{x})}{P(C_2|\boldsymbol{x})} = \frac{1}{\sigma^2} \sum_{j=1}^{d} (\mu_{1j} - \mu_{2j}) x_j - \frac{1}{2\sigma^2} \sum_{j=1}^{d} (\mu_{1j}^2 - \mu_{2j}^2) + \log \frac{P(C_1)}{P(C_2)}$$

5. *Another possibility using Gaussian densities is to have them all diagonal but allow them to be different. Derive the discriminant for this case.*

When the covariance matrices are diagonal, we get axis-aligned distributions. Again, we get a linear discriminant if the two matrices are equal, and otherwise, we get quadratic discriminants. Examples are given in figure 5.2, generated by `ex5_5.m`.

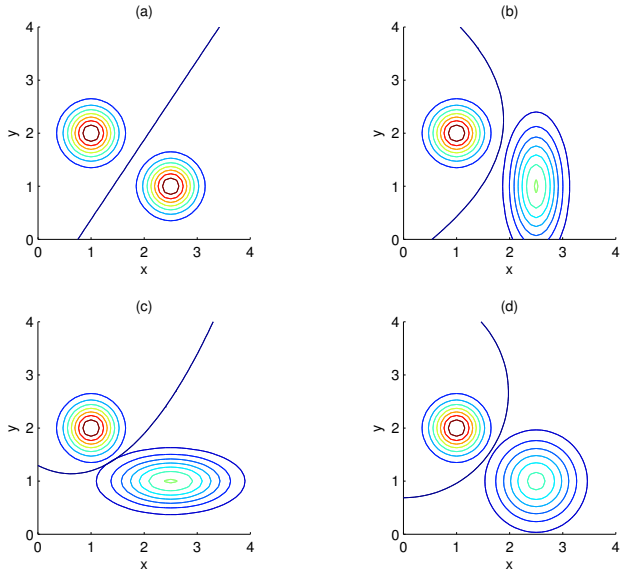


**Figure 5.2** Distributions and the discriminant for different cases of diagonal matrices. (a) The two matrices are equal and the discriminant is linear; in other cases, the matrices are different and the discriminant is quadratic.

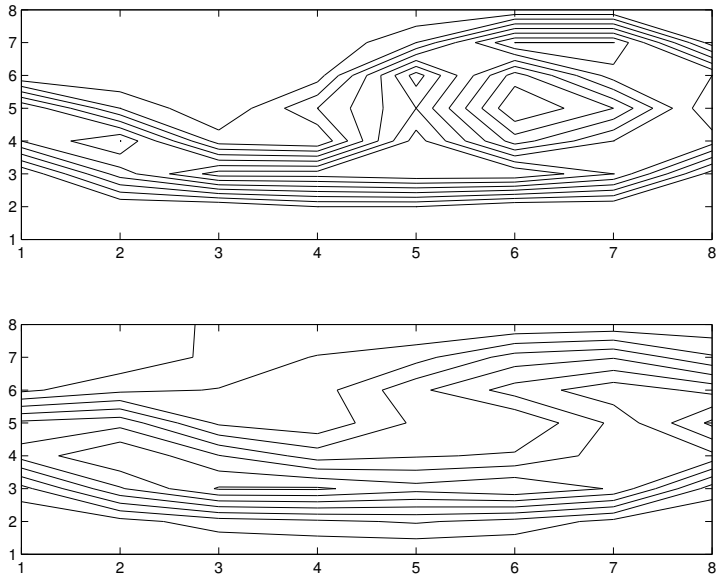


**Figure 6.1** A '6' and its reconstruction using the top two eigenvectors.

3. *Plot the map of your state/country using MDS, given the road travel distances as input.*

   The Matlab code that generates figure 6.9 in the book is given in `ex6_3.m`. It generates the map of Europe from pairwise distances between 12 cities.

4. *In Sammon mapping, if the mapping is linear, namely, $g(\boldsymbol{x}|\mathbf{W}) = \mathbf{W}^T\boldsymbol{x}$, how can $\mathbf{W}$ that minimizes the Sammon stress be calculated?*

   MDS already does this linear mapping. Another possibility is to plug in the linear model in Sammon stress (equation 6.37) and find $\mathbf{W}$ that minimizes it. One can for example use an iterative, gradient-descent procedure; this is especially interesting when the mapping is nonlinear. We discuss this in chapter 11 when we use multilayer perceptrons (p. 304).

5. *In figure 6.11, we see a synthetic two-dimensional data where LDA does a better job than PCA. Draw a similar dataset where PCA and LDA find*

*the same good direction. Draw another where neither PCA nor LDA find a good direction.*

See figures 6.2a and b; the Matlab code is in `ex6_5.m`.

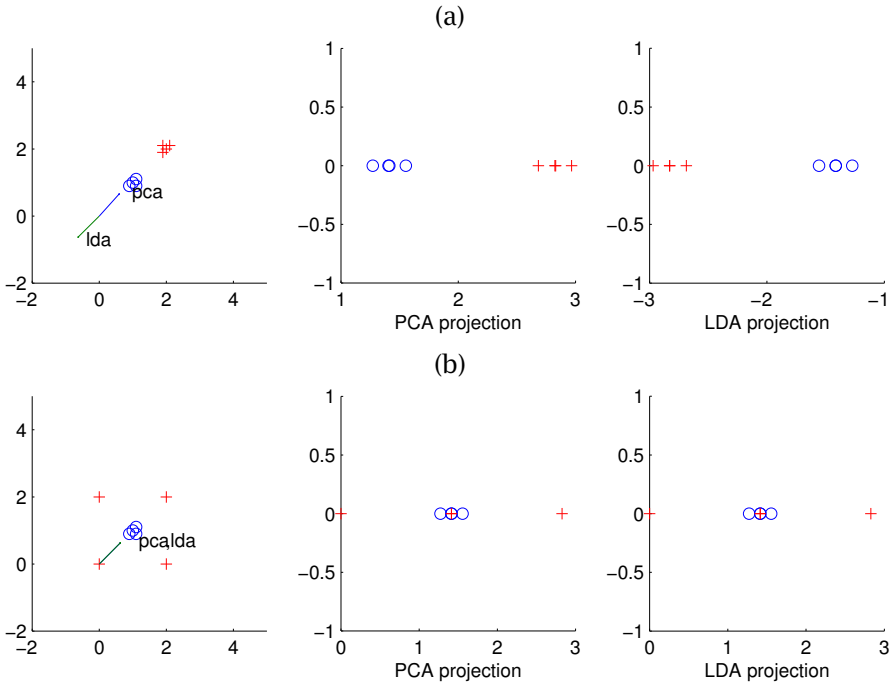


**Figure 6.2** (a) Both PCA and LDA find the good direction; (b) Neither work well.

6. *Redo exercise 3, this time using Isomap where two cities are connected only if there is a direct road between them that does not pass through any other city.*

   That is actually how intercity distances are calculated, namely by connecting the distances between the cities that lie on the shortest path.

7. In Isomap, instead of using Euclidean distance, we can also use Mahalanobis distance between neighboring points. What are the advantages and disadvantages of this approach, if any?

   The disadvantage is the need to calculate, store and process the covariance matrix used in Mahalanobis distance calculation. There can be one global matrix or better still, for each instance, instances in

The update equation for the shared variance is

$$s^2 = \frac{\sum_t \sum_i \sum_{k=1}^{d} h_i^t (x_k^t - m_{ik})^2}{Nd}$$

4. *Define a multivariate Bernoulli mixture where inputs are binary and derive the EM equations.*

   When the components are multivariate Bernouilli, we have binary vectors that are $d$-dimensional. Assuming that the dimensions are independent, we have (see section 5.7)

$$p_i(\boldsymbol{x}^t|\Phi) = \prod_{j=1}^{d} p_{ij}^{x_j^t} (1 - p_{ij})^{1-x_j^t}$$

   where $\Phi^l = \{p_{i1}^l, p_{i2}^l, \ldots, p_{id}^l\}_{i=1}^k$. The E-step does not change (equation 7.9). In the M-step, for the component parameters $p_{ij}, i = 1, \ldots, k, j = 1, \ldots, d$, we maximize

$$\begin{aligned} \mathcal{Q}' &= \sum_t \sum_i h_i^t \log p_i(\boldsymbol{x}^t|\phi^l) \\ &= \sum_t \sum_i h_i^t \sum_j x_j^t \log p_{ij}^l + (1 - x_j^t) \log(1 - p_{ij}^l) \end{aligned}$$

   Taking the derivative with respect to $p_{ij}$ and setting it equal to 0, we get

$$p_{ij}^{l+1} = \frac{\sum_t h_i^t x_j^t}{\sum_t h_j^t}$$

   Note that this is the same as in equation 5.31, except that estimated "soft" labels $h_i^t$ replace the supervised labels $r_i^t$.

5. *In the mixture of mixtures approach for classification, how can we finetune $k_i$, the number of components for class $C_i$?*

   As usual, by cross-validation. We can try different $k_i$ and choose the one that performs best on the validation data. We can start with small $k_i$ for all $i$ and then, looking at the confusion matrix, we can pinpoint which classes are frequently confused and increase their $k_i$ to give them more flexibility.

# 8 *Nonparametric Methods*

1. *How can we have a smooth histogram?*

   We can interpolate between the two nearest bin centers. We can consider the bin centers as $x^t$, consider the histogram values as $r^t$, and use any interpolation scheme, linear or kernel-based.

2. *Show equation 8.16.*

   Given that

   $$\hat{p}(\boldsymbol{x}|C_i) = \frac{k_i}{N_i V^k(\boldsymbol{x})} \qquad \hat{P}(C_i) = \frac{N_i}{N}$$

   we can write

   $$\begin{aligned}\hat{P}(C_i|\boldsymbol{x}) &= \frac{\hat{p}(\boldsymbol{x}|C_i)\hat{P}(C_i)}{\sum_j \hat{p}(\boldsymbol{x}|C_j)\hat{P}(C_j)} = \frac{\frac{k_i}{N_i V^k(\boldsymbol{x})}\frac{N_i}{N}}{\sum_j \frac{k_j}{N_j V^k(\boldsymbol{x})}\frac{N_j}{N}} \\ &= \frac{k_i}{\sum_j k_j} = \frac{k_i}{k}\end{aligned}$$

3. *Parametric regression (section 5.8) assumes Gaussian noise and hence is not robust to outliers; how can we make it more robust ?*

   One approach would be to use a robust error function, such as the one used in support vector regression, that adds up differences linearly rather than quadratically.
   Another approach—and this is related to our discussion in this chapter—would be to try to find the outliers and remove them. Towards this aim, one possibility is to use the local outlier factor of section 8.7 in

a regression setting—prune instances that are far away (in the input space) from all others. Another possibility is to make an initial fit, find instances on which we have a bad fit, remove those, and refit. In both methods, there are thresholds that need to be adjusted carefully by cross-validation.
Note that on small datasets, deciding whether an instance is an outlier becomes a very difficult and critical problem.

4. *How can we detect outliers after hierarchical clustering (section 7.8) ?*
After hierarchical clustering, we would expect outliers to form separate leaves, far from the other leaves and containing very few instances.

5. *How does condensed nearest neighbor behave if $k > 1$?*

   When $k > 1$, to get full accuracy without any misclassification, it may be necessary to store an instance multiple times so that the correct class gets the majority of the votes. For example, if $k = 3$ and $\boldsymbol{x}$ has two neighbors both belonging to a different class, we need to store $\boldsymbol{x}$ twice (i.e., it gets added in two epochs), so that if $\boldsymbol{x}$ is seen during test, the majority (two in this case) out of three neighbors belong to the correct class.

6. *In condensed nearest neighbor, an instance previously added to $\mathcal{Z}$ may no longer be necessary after a later addition. How can we find such instances that are no longer necessary?*

   We pick randomly an instance from $\mathcal{Z}$ and check if we can correctly classify it using the rest of $\mathcal{Z}$; if we can, this means that we would not have added it if we were seeing it now for the first time and therefore can remove it from $\mathcal{Z}$; if we misclassify it, we keep it. I used to call this the “sleep” mode and alternated it with the “awake” mode that added instances from the training set to $\mathcal{Z}$; see E Alpaydın (1994) "GAL: Networks that Grow when they Learn and Shrink when they Forget," *International Journal of Pattern Recognition and Artificial Intelligence*, **8**, 391–414.

7. *In a regressogram, instead of averaging in a bin and doing a constant fit, one can use the instances falling in a bin and do a linear fit (see figure 8.14). Write the code and compare this with the regressogram proper.*

for $i = j$, we have

$$\begin{aligned}\frac{\partial y_i}{\partial a_i} &= \frac{\exp a_i \left(\sum_j \exp a_j\right) - \exp a_i \exp a_j}{\left(\sum_j \exp a_j\right)^2} \\ &= \frac{\exp a_i}{\sum_j \exp a_j}\left(\frac{\sum_j \exp a_j - \exp a_i}{\sum_j \exp a_j}\right) \\ &= y_i(1 - y_i)\end{aligned}$$

and for $i \neq j$, we have

$$\begin{aligned}\frac{\partial y_i}{\partial a_j} &= \frac{-\exp a_i \exp a_j}{\left(\sum_j \exp a_j\right)^2} \\ &= -\left(\frac{\exp a_i}{\sum_j \exp a_j}\right)\left(\frac{\sum_j \exp a_j}{\sum_j \exp a_j}\right) \\ &= y_i(0 - y_j)\end{aligned}$$

which we can combine in one equation as

$$\frac{\partial y_i}{\partial a_j} = y_i(\delta_{ij} - y_j)$$

4. With $K = 2$, show that using two softmax outputs is equal to using one sigmoid output.

$$\begin{aligned}y_1 &= \frac{\exp o_1}{\exp o_1 + \exp o_2} = \frac{1}{1 + \exp(o_2 - o_1)} = \frac{1}{1 + \exp(-(o_1 - o_2))} \\ &= \text{sigmoid}(o_1 - o_2)\end{aligned}$$

For example, if we have $o_1 = \boldsymbol{w}_1^T \boldsymbol{x}$, we have

$$y_1 = \frac{\exp \boldsymbol{w}_1^T \boldsymbol{x}}{\exp \boldsymbol{w}_1^T \boldsymbol{x} + \exp \boldsymbol{w}_2^T \boldsymbol{x}} = \text{sigmoid}(\boldsymbol{w}_1^T \boldsymbol{x} - \boldsymbol{w}_2^T \boldsymbol{x}) = \text{sigmoid}(\boldsymbol{w}^T \boldsymbol{x})$$

where $\boldsymbol{w} \equiv \boldsymbol{w}_1 - \boldsymbol{w}_2$ and $y_2 = 1 - y_1$.

# 11 *Multilayer Perceptrons*

1. *Show the perceptron that calculates NOT of its input.*

   $y = s(-x + 0.5)$

2. *Show the perceptron that calculates NAND of its two inputs.*

   $y = s(-x_1 - x_2 + 1.5)$

3. *Show the perceptron that calculates the parity of its three inputs.*

$$
\begin{aligned}
h_1 &= s(-x_1 - x_2 + 2x_3 - 1.5) \quad 001 \\
h_2 &= s(-x_1 + 2x_2 - x_3 - 1.5) \quad 010 \\
h_3 &= s(2x_1 - x_2 - x_3 - 1.5) \quad 100 \\
h_4 &= s(x_1 + x_2 + x_3 - 2.5) \quad 111 \\
y &= s(h_1 + h_2 + h_3 + h_4 - 0.5)
\end{aligned}
$$

   The four hidden units corresponding to the four cases of $(x_1, x_2, x_3)$ values where the parity is 1, namely, 001, 010, 100, and 111. They are then OR'd to calculate the overall output. Note that another possibility is to calculate the three bit parity is in terms of two-bit parity (XOR) as: $(x_1$ XOR $x_2)$ XOR $x_3$.

4. Derive the update equations when the hidden units use tanh, instead of the sigmoid. Use the fact that $\tanh' = (1 - \tanh^2)$.

   The only difference from the case where hidden units use sigmoid is that the derivative of the hidden unit activations will change. In

being trained together as usual. The perceptron that contains the direct weights from the inputs to the outputs is a linear model and can be thought of as implemening the linear component of whatever we are trying to learn; the MLP then learns to correct this "default" linear model. We can also train from that point of view if we like: That is, first we train the linear model, calculate the residual, and then train later the MLP on these residuals. We may need to alternate these two steps. Then we will have a linear rule plus some nonlinear exceptions or corrections to the rule (see also equation 12.22 where we have Gaussian hidden units instead of sigmoid).

7. *Parity is cyclic shift invariant, for example, "0101" and "1010" have the same parity. Propose a multilayer perceptron to learn the parity function using this hint.*

   One can generate virtual examples by adding shifted versions of instances to the training set. Or, one can define local hidden units with weight sharing to keep track of local parity which are then combined to calculate the overall parity.

8. *In cascade correlation, what are the advantages of freezing the previously existing weights?*

   The main advantage is that it allows us to train a single layer at each step which is faster than training multiple hidden layers. Keeping all but one of the set of parameters fixed is similar to *backfitting algorithms*; we go over a set of additive models, while keeping all the others fixed, we update the parameters of one model to minimize the residual from the other models. This makes sense in cascade correlation because each new unit is adde to learn what has not yet been learned by the previous layer(s).

9. *Derive the update equations for an MLP implementing Sammon mapping that minimizes Sammon stress (equation 11.40).*

$$
\begin{aligned}
E(\theta|\mathcal{X}) &= \sum_{r,s}\left[\frac{\|\boldsymbol{g}(\boldsymbol{x}^r|\theta)-\boldsymbol{g}(\boldsymbol{x}^s|\theta)\|-\|\boldsymbol{x}^r-\boldsymbol{x}^s\|}{\|\boldsymbol{x}^r-\boldsymbol{x}^s\|}\right]^2 \\
&= \frac{1}{\sum_{r,s}\|\boldsymbol{x}^r-\boldsymbol{x}^s\|^2}\sum_{r,s}\left(\|\boldsymbol{g}(\boldsymbol{x}^r|\theta)-\boldsymbol{g}(\boldsymbol{x}^s|\theta)\|-\|\boldsymbol{x}^r-\boldsymbol{x}^s\|\right)^2
\end{aligned}
$$

The MLP is written as

$$g(x|\boldsymbol{v},\boldsymbol{W}) = \sum_h v_h z_h + v_0 \text{ and } z_h = \text{sigmoid}\left(\sum_j w_{hj}x_j + w_{h0}\right)$$

The update rules are found using gradient-descent:

$$\begin{aligned}
\Delta v_h &= -\eta \frac{1}{\sum_{r,s}\|\boldsymbol{x}^r - \boldsymbol{x}^s\|^2} \sum_{r,s} \left(\|\boldsymbol{g}(\boldsymbol{x}^r|\theta) - \boldsymbol{g}(\boldsymbol{x}^s|\theta)\| - \|\boldsymbol{x}^r - \boldsymbol{x}^s\|\right)(z_h^r - z_h^s) \\
\Delta w_{hj} &= -\eta \frac{1}{\sum_{r,s}\|\boldsymbol{x}^r - \boldsymbol{x}^s\|^2} \sum_{r,s} \left(\|\boldsymbol{g}(\boldsymbol{x}^r|\theta) - \boldsymbol{g}(\boldsymbol{x}^s|\theta)\| - \|\boldsymbol{x}^r - \boldsymbol{x}^s\|\right) \\
&\cdot \quad v_h\left[z_h^r(1-z_h^r)x_j^r - z_h^s(1-z_h^s)x_j^s\right]
\end{aligned}$$

10. *In section 11.6, we discuss how a MLP with two hidden layers can implement piecewise constant approximation. Show that if the weight in the last layer is not a constant but a linear function of the input, we can implement piecewise linear approximation.*

    As before, we can have the second hidden layer sigmoid units detect a local region in the input space, then the input-dependent weight in the final layer defines the line (or hyperplane in general) that will be used there. As we will see in chapter 12, we can have a single layer of Gaussians instead of two layers of sigmoids to detect a local input region; in such a case when we have input-dependent weights in the final layer, we will have a *mixture of experts* (section 12.8).

11. *Derive the update equations for soft weight sharing.*

    Assume a one-layer model for two-class classification for simplicity,

    $$y^t = \text{sigmoid}(\sum_i w_i x_i^t)$$

    the augmented error is

    $$E' = \log \sum_t r^t \log y^t + \lambda \sum_i \log \sum_{j=1}^M \alpha_j p_j(w_i)$$

where $p_j(w_i) \sim \mathcal{N}(m_j, s_j^2)$. In the general case of a multilayer perceptron, $\{w_i\}_i$ includes all the weights in all layers including the bias weights. When we use gradient-descent, we get

$$\Delta w_i^t = \eta(r^t - y^t)x_i^t - \eta\lambda \sum_j \pi_j(w_i) \frac{(w_i - m_j)}{s_j^2}$$

where

$$\pi_j(w_i) = \frac{\alpha_j p_j(w_i)}{\sum_l \alpha_l p_l(w_i)}$$

is the posterior probability. The weight is updated both to decrease the cross-entropy and to move it closer to the mean of the nearest Gaussian. Using such a scheme, we can also update the mixture parameters, for example

$$\Delta m_j = \eta\lambda \sum_i \pi_j(w_i) \frac{(w_i - m_j)}{s_j^2}$$

$\pi_j(w_i)$ is close to 1 if it is highly probable that $w_i$ comes from component $j$; in such a case, the mean $m_j$ is updated to be closer to the weight $w_i$ it represents. This is an iterative clustering procedure, and we will discuss such methods in more detail in chapter 12, see for example, equation 12.5.

12. *In the autoassociator network, how can we decide on the number of hidden units?*

    We can do cross-validation on the reconstruction error.
    For an autoencoder with one hidden layer, we can also use the number of eigenvectors that explain, for example 95 percent of the variance.

13. *Incremental learning of the structure of a MLP can be viewed as a state space search. What are the operators? What is the goodness function? What type of search strategies are appropriate? Define these in such a way that dynamic node creation and cascade-correlation are special instantiations.*

    We need to define an error function in terms of misclassification error or regression. We need to have a training set and a separate validation set on which we calculate the error to make sure that we do not overfit. We can do a forward search starting from an initial state which

# 12 *Local Models*

1. *Show an RBF network that implements XOR.*

   There are two possibilities: (a) We can have two circular Gaussians centered on the two positive instances and the second layer ORs them, (b) We can have one elliptic Gaussian centered on (0.5, 0.5) with negative correlation to cover the two positive instances.

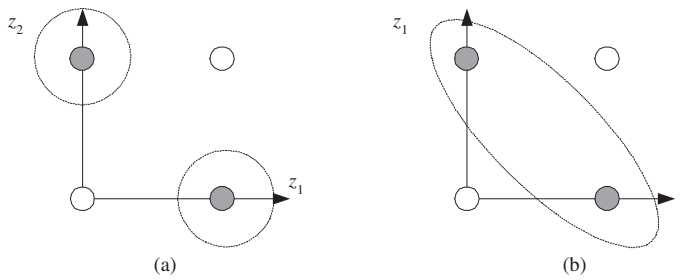


**Figure 12.1** Two ways of implementing XOR with RBF.

2. *Write down the RBF network that uses elliptic units instead of radial units as in equation 12.13.*

$$p_h^t = \exp\left[-\frac{1}{2}(\boldsymbol{x}^t - \boldsymbol{m}_h)^T \mathbf{S}_h^{-1}(\boldsymbol{x}^t - \boldsymbol{m}_h)\right]$$

   where $\mathbf{S}_h$ is the local covariance matrix.

3. *Derive the update equations for the RBF network for classification (equations 12.20 and 12.21).*

The easiest is to use a linear model:

$$\eta_i(\boldsymbol{x}|\{\boldsymbol{V}_i\}_i) = \frac{\exp \boldsymbol{V}_i^T \boldsymbol{x}}{\sum_k \exp \boldsymbol{V}_k^T \boldsymbol{x}}$$

It is also possible to use a kernel function here as well, to have nonlinear gating.

The main problem here is that with these new parameters $\boldsymbol{V}_i$, we have a problem which is no longer convex. It becomes convex once we fix $\boldsymbol{V}_i$. So we use an alternating procedure where we start with random $\boldsymbol{V}_i$, solve for SVM parameters, then having fixed them, we to gradient-descent on the dual to optimize $\boldsymbol{V}_i$, fix them and solve for SVM again and so on. See (Gönen and Alpaydın 2008) for details.

5. *In kernel regression, what is the relation, if any, between $\epsilon$ and noise variance?*

   The advantage of having $\epsilon$ is that if for an instance, the prediction error is less than $\epsilon$, we can ignore it. Noise is what we have by chance and is not due to the underlying function, so we need to have $\epsilon$ just large enough to cover what can be due to noise. For example, if we know that noise is Gaussian with variance $\sigma^2$ and if we know $\sigma^2$, then a good $\epsilon$ value is $1.96\sigma$.

6. *In kernel regression, what is the effect of using different $\epsilon$ on bias and variance?*

   $\epsilon$ is a smoothing parameter. When it is too large, we smooth too much which reduces variance but risks increasing bias. If it is too small, the variance may be large and bias would be small.

7. *Derive the kernelized version of the primal, dual, and the score functions for ranking.*

   The primal is

$$L_p = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_t \xi^t$$

   subject to

$$\begin{aligned} \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}^u - \boldsymbol{x}^v) &\geq 1 - \xi^t \\ \xi^t &\geq 0 \end{aligned}$$

The dual is

$$L_d = \sum_t \alpha^t - \frac{1}{2}\sum_t \sum_s \alpha^t \alpha^s K(\boldsymbol{x}^u - \boldsymbol{x}^v, \boldsymbol{x}^k - \boldsymbol{x}^l)$$

where $K(\boldsymbol{x}^u - \boldsymbol{x}^v, \boldsymbol{x}^k - \boldsymbol{x}^l) = \boldsymbol{\phi}(\boldsymbol{x}^u - \boldsymbol{x}^v)^T \boldsymbol{\phi}(\boldsymbol{x}^k - \boldsymbol{x}^l)$. For new test instance $\boldsymbol{x}$, the score is calculated as

$$g(\boldsymbol{x}) = \sum_t \alpha^t K(\boldsymbol{x}^u - \boldsymbol{x}^v, \boldsymbol{x})$$

8. *How can we use one-class SVM for classification?*

   We can use a separate one-class SVM for each class and then combine them to make a decision. For example, for each class $C_i$, we fit a one-class SVM to find parameters $\alpha_i^t$:

   $$\sum_t \alpha_i^t K_G(\boldsymbol{x}, \boldsymbol{x}^t)$$

   and this then can be taken as an estimator for $p(\boldsymbol{x}|C_i)$. If the priors are more or less equal, we can simply choose the class having the largest value; otherwise we can use Bayes' rule for classification.

9. In a setting such as that in figure 13.12, use kernel PCA with a Gaussian kernel.

   See figure 13.1.

10. *Let us say we have two representations for the same object and associated with each, we have a different kernel. How can we use both to implement a joint dimensionality reduction using kernel PCA?*

    There are two possibilities: (1) For each kernel, we can do a separate kernel PCA and get a lower-dimensional vector and then at the end, concatenate these vectors, or (2) We can take a sum of the two kernel functions and then use kernel PCA once on the sum. When taking the sum, we need to make sure that kernel matrices contain comparable values.

# 14 *Graphical Models*

1. *With two independent inputs in a classification problem, that is, $p(x_1, x_2|C) = p(x_1|C)p(x_2|C)$, how can we calculate $p(x_1|x_2, C)$? Derive the formula for $p(x_j|C_i) \sim \mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$.*

   This is a trick question. If $x_1$ and $x_2$ are independent, $p(x_1|x_2, C) = p(x_1|C)$. If the two are dependent, $p(x_1, x_2|C)$ is a bivariate density and we have

   $$p(x_1|x_2, C) = \frac{p(x_1, x_2|C)}{p(x_2|C)}$$

   When the two are independent, the bivariate is a product of two univariates of which one cancels with the denominator.

2. *For a head-to-head node, show that equation 14.10 implies $P(X, Y) = P(X) \cdot P(Y)$.*

   We know that $P(X, Y, Z) = P(Z|X, Y)P(X, Y)$, and if we also know that $P(X, Y, Z) = P(X)P(Y)P(Z|X, Y)$, we see that $P(X, Y) = P(X)P(Y)$.

3. In figure 14.4, calculate $P(R|W)$, $P(R|W, S)$, and $P(R|W, \sim S)$.

$$\begin{aligned}
P(R|W) &= \frac{P(R, W)}{P(W)} = \frac{\sum_S P(R, W, S)}{\sum_R \sum_S P(R, W, S)} \\
&= \frac{\sum_S P(R)P(S)P(W|R, S)}{\sum_R \sum_S P(R)P(S)P(W|R, S)} \\
P(R|W, S) &= \frac{P(R, W, S)}{P(W, S)} = \frac{P(R)P(S)P(W|R, S)}{\sum_R P(R)P(S)P(W|R, S)} \\
P(R|W, \sim S) &= \frac{P(R, W, \sim S)}{P(W, \sim S)} = \frac{P(R)P(\sim S)P(W|R, \sim S)}{\sum_R P(R)P(\sim S)P(W|R, \sim S)}
\end{aligned}$$

4. *In equation 14.30, $X$ is binary. How do we need to modify it if $X$ can take one of $K$ discrete values?*

   Let us say there are $j = 1, \ldots, K$ states. Then, keeping the model linear, we need to parameterize each by a separate $\boldsymbol{w}_j$ and use softmax to map to probabilities:

$$P(X = j|U_1, \ldots, U_k, \{w_{ji}\}) = \frac{\exp \sum_{i=1}^{k} w_{ji} U_i + w_{j0}}{\sum_{l=1}^{K} \exp \sum_{i=1}^{k} w_{li} U_i + w_{l0}}$$

5. *Show that in a directed graph where the joint distribution is written as equation 14.12, $\sum_{\boldsymbol{X}} p(\boldsymbol{x}) = 1$.*

   The terms cancel when we sum up over all possible values because these are probabilities. Let us, for example, take figure 14.3a:

$$\begin{aligned}
P(X, Y, Z) &= P(X)P(Y|X)P(Z|X) \\
\sum_X \sum_Y \sum_Z P(X, Y, Z) &= \sum_X \sum_Y \sum_Z P(X)P(Y|X)P(Z|X) \\
&= \sum_X \sum_Y P(X)P(Y|X) \sum_Z P(Z|X) \\
&= \sum_X \sum_Y P(X)P(Y|X) \sum_Z \frac{P(Z, X)}{P(X)} \\
&= \sum_X \sum_Y P(X)P(Y|X) \frac{P(X)}{P(X)} \\
&= \sum_X \sum_Y P(X)P(Y|X) \\
&= \sum_X P(X) \sum_Y P(Y|X) = \sum_X P(X) = 1
\end{aligned}$$

6. *Draw the Necker cube as a graphical model defining links to indicate mutually reinforcing or inhibiting relations between different corner interpretations.*

   We are going to have nodes corresponding to corners and they take values depending on the interpretation; there will be positive, enforcing, excitatory connections between corners that are part of the same interpretation, and negative, inhibitory connections between corners that are part of different interpretations (see figure 14.1).

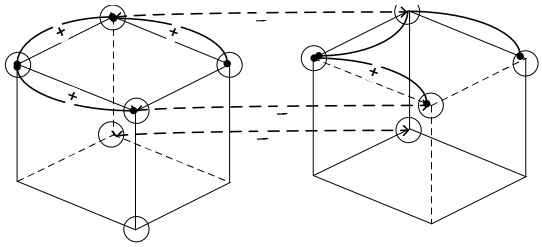

**Figure 14.1** Two different interpretations of the Necker cube.

7. *Write down the graphical model for linear logistic regression for two classes in the manner of figure 14.7.*

   It is shown in figure 14.2 where $q$ corresponds to priors, and $\mu$ and $\Sigma$ parameterize $p(x|C)$. We can add priors for these if we like to get Bayesian logistic classification, as we discuss in chapter 16.

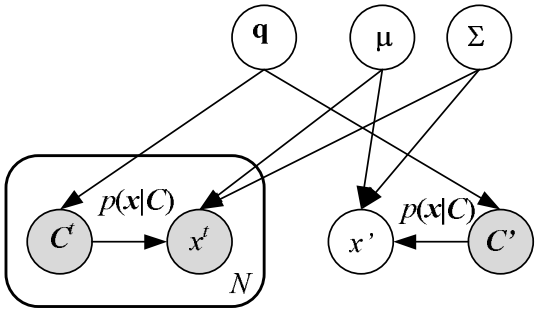


**Figure 14.2** Graphical model for classification.

8. *Propose a suitable goodness measure that can be used in learning graph structure as a state-space search. What are suitable operators?*

   We need a *score function* that is the sum of two parts, one quantifying a goodness of fit, that is, how likely is the data given the model, and one quantifying the complexity of the graph, to alleviate overfitting. In measuring complexity, we must take into account the total number of nodes and the number of parameters needed to represent the conditional probability distributions. For example, we should try to have nodes with as few parents as possible. Possible operators are there to add/remove an edge and add/remove a hidden node.

# 16 *Bayesian Estimation*

1. *For the setting of figure 16.3, observe how the posterior changes as we change $N$, $\sigma^2$, and $\sigma_0^2$.*

   Matlab code that generates this is `ex14_3.m`. See figures 16.1, 16.2, 16.3, and 16.4.

2. *Let us denote by $x$ the number of spam emails I receive in a random sample of $n$. Assume that the prior for $q$, the proportion of spam emails is uniform in $[0,1]$. Find the posterior distribution for $p(q|x)$.*

   $x$ is binomial and unless $n$ is small, it can be approximated by a Gaussian. Uniform prior is a flat prior and the posterior will have the same shape as the likelihood, and peaks at the same point.

3. *As above, except that assume that $p(q) \sim \mathcal{N}(\mu_0, \sigma_0^2)$. Also assume $n$ is large so that you can use central limit theorem and approximate binomial by a Gaussian. Derive $p(q|x)$.*

   In this case both the likelihood and prior are Gaussian and we can use equations 16.6 and 16.7.

4. *What is $\mathrm{Var}(r')$ when the maximum likelihood estimator is used? Compare it with equation 16.25.*

   We have maximum likelihood when $\alpha$ goes to 0 in equation 16.23, which we can then plug in equation 16.25.

5. *In figure 16.10, how does the fit change when we change $s^2$?*

   As usual, $s$ is the smoothing parameter and we get smoother fits as we increase $s$.