Introduction to Machine Learning

Figures

Ethem Alpaydın

Chapter 1:

Introduction

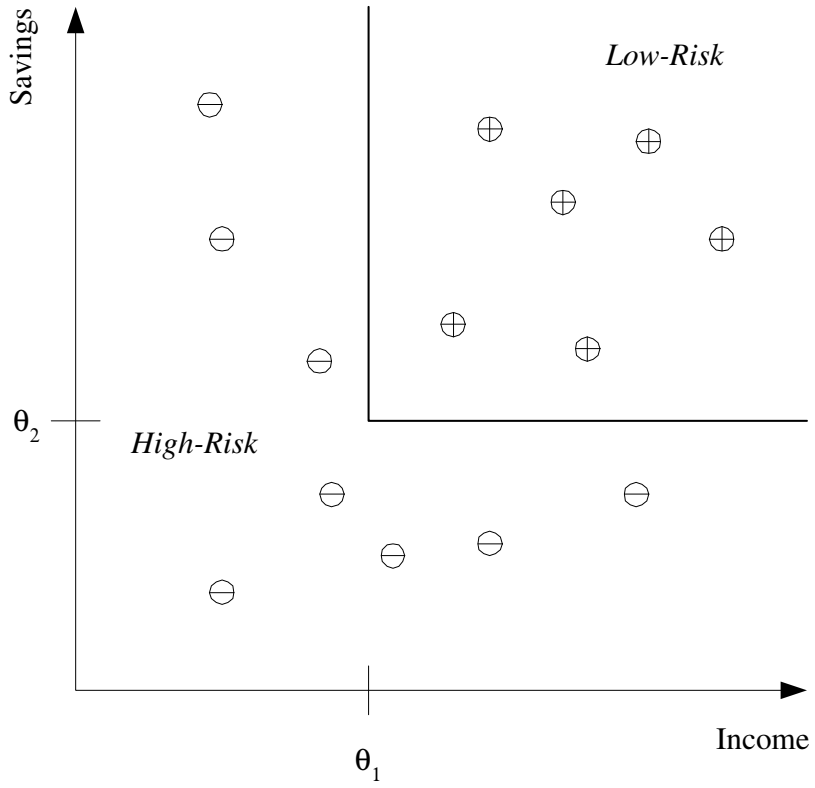


Figure 1.1: Example of a training dataset where each circle corresponds to one data instance with input values in the corresponding axes and its sign indicates the class. For simplicity, only two customer attributes, income and savings, are taken as input and the two classes are low-risk ('+') and high-risk ('−'). An example discriminant that separates the two types of examples is also shown. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

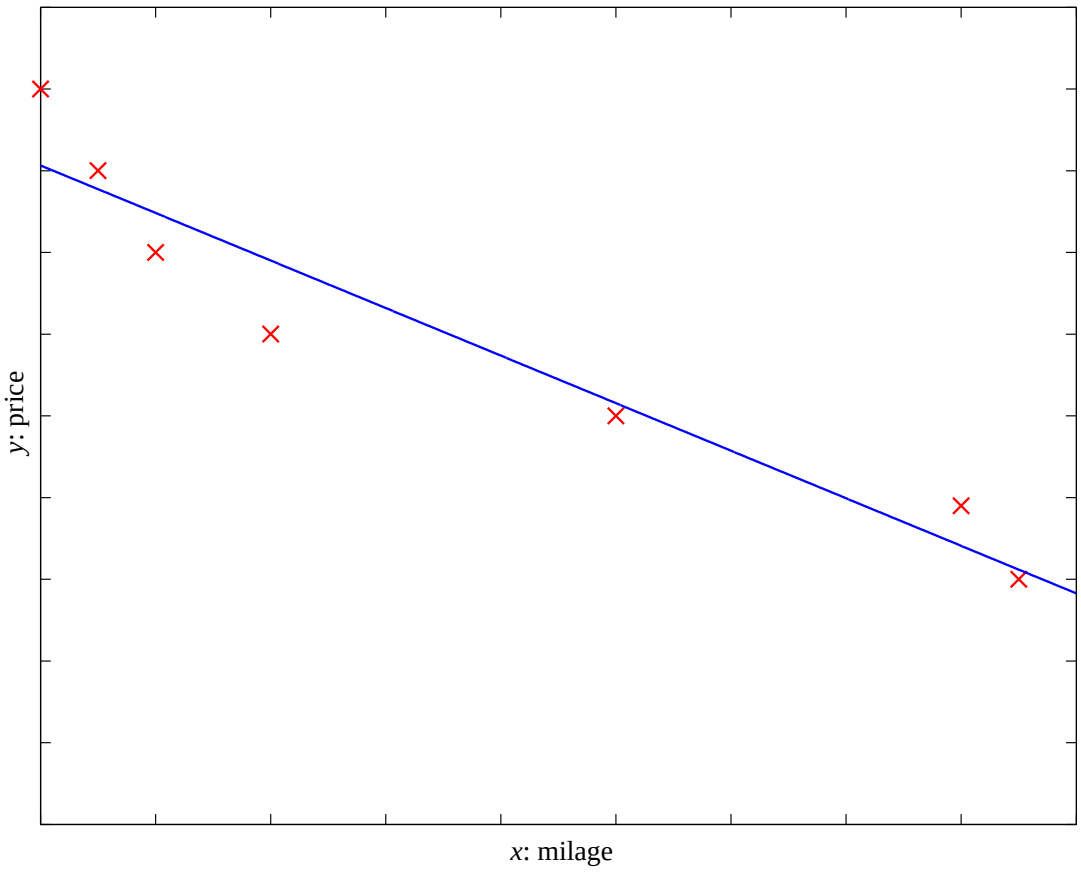


Figure 1.2: A training dataset of used cars and the function fitted. For simplicity, milage is taken as the only input attribute and a linear model is used. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Chapter 2:

Supervised Learning

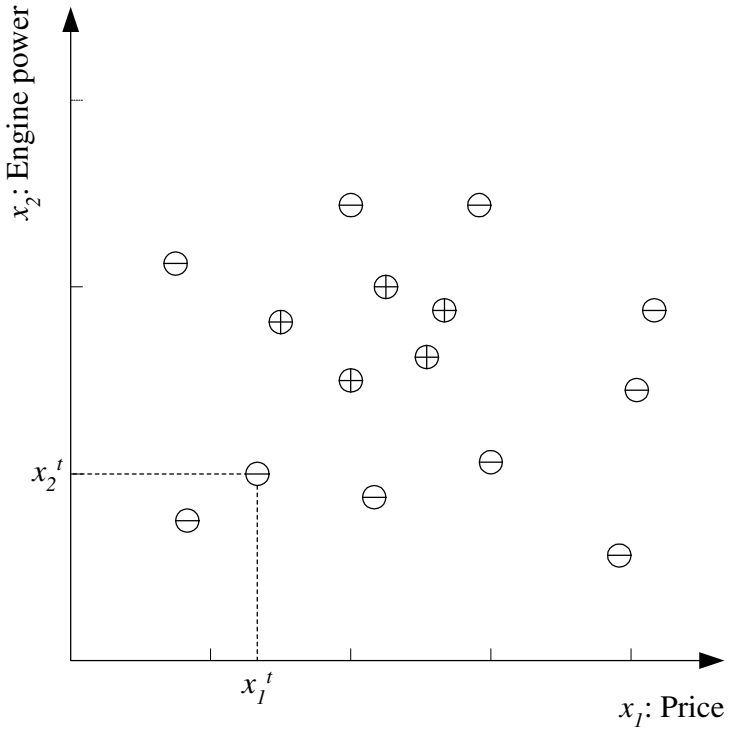


Figure 2.1: Training set for the class of a "family car." Each data point corresponds to one example car and the coordinates of the point indicate the price and engine power of that car. '+' denotes a positive example of the class (a family car), and '−' denotes a negative example (not a family car); it is another type of car. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

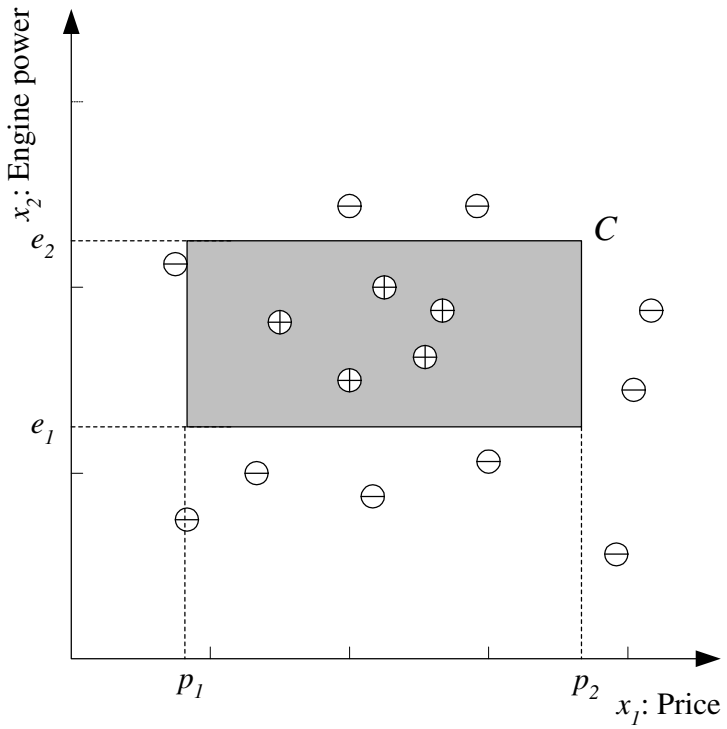


Figure 2.2: Example of a hypothesis class. The class of family car is a rectangle in the price-engine power space. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

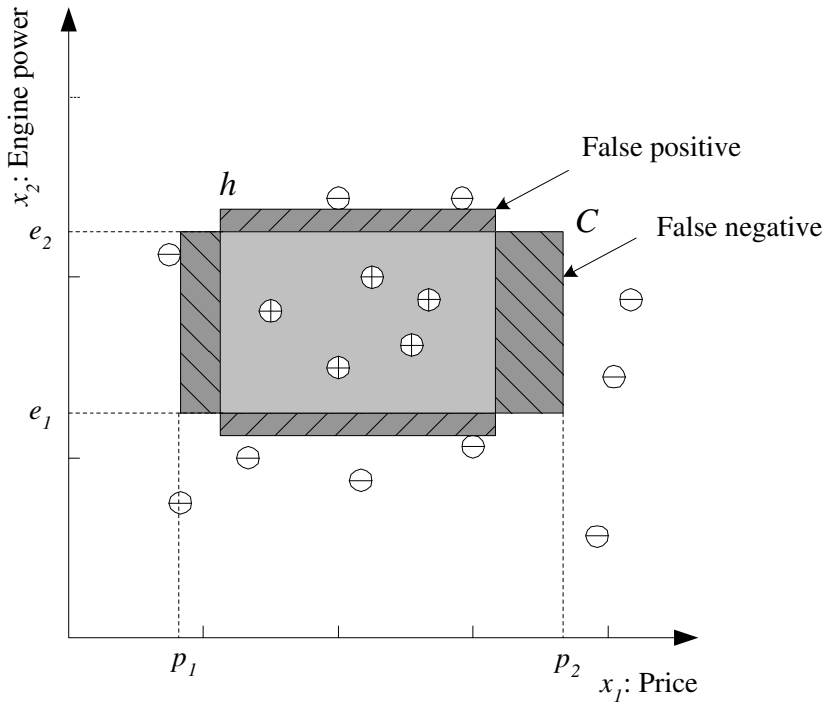


Figure 2.3: $\mathcal{C}$ is the actual class and h is our induced hypothesis. The point where $\mathcal{C}$ is 1 but h is 0 is a false negative, and the point where $\mathcal{C}$ is 0 but h is 1 is a false positive. Other points, namely true positives and true negatives, are correctly classified. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

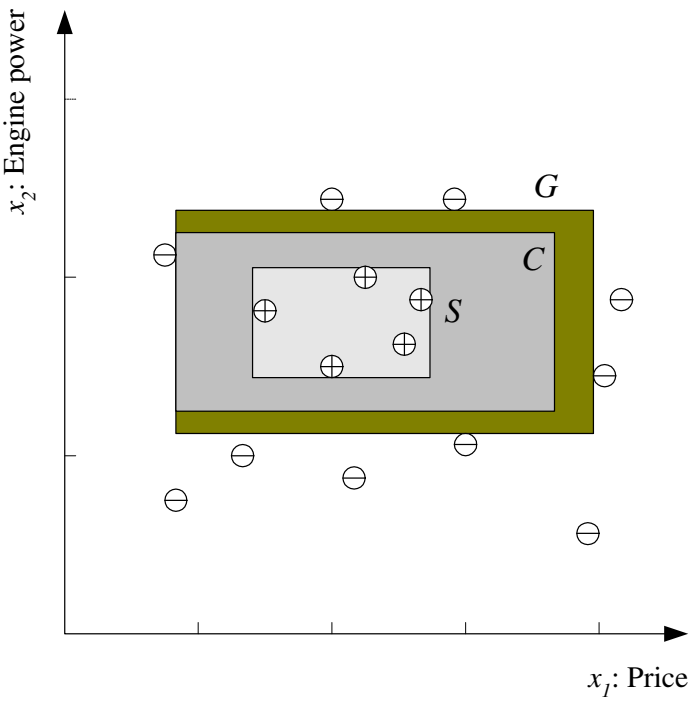


Figure 2.4: S is the most specific hypothesis and G is the most general hypothesis. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

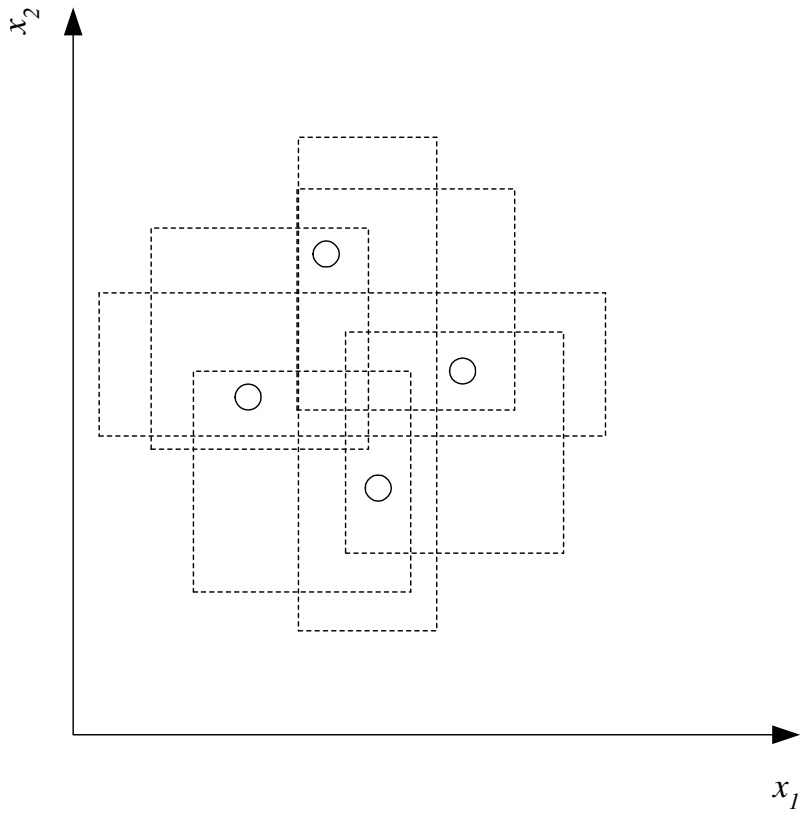


Figure 2.5: An axis-aligned rectangle can shatter four points. Only rectangles covering two points are shown. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

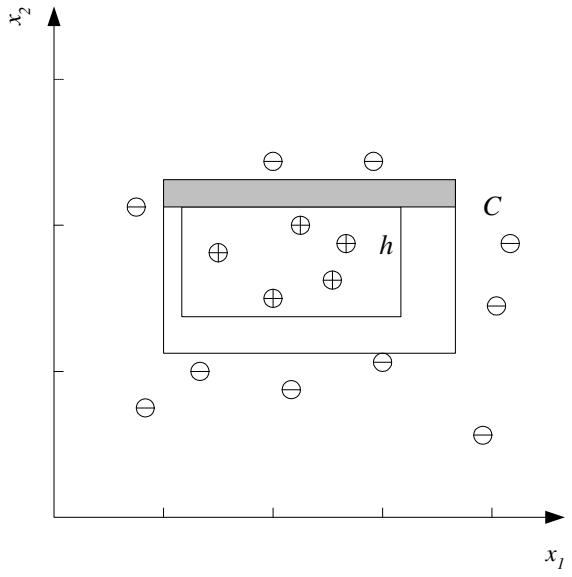


Figure 2.6: The difference between h and $\mathcal{C}$ is the sum of four rectangular strips, one of which is shaded. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

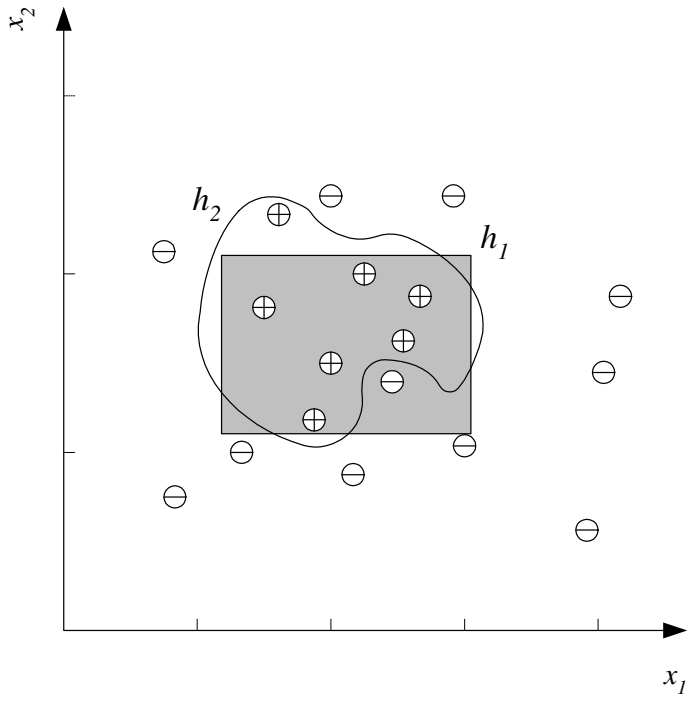


Figure 2.7: When there is noise, there is not a simple boundary between the positive and negative instances, and zero misclassification error may not be possible with a simple hypothesis. A rectangle is a simple hypothesis with four parameters defining the corners. An arbitrary closed form can be drawn by piecewise functions with a larger number of control points. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

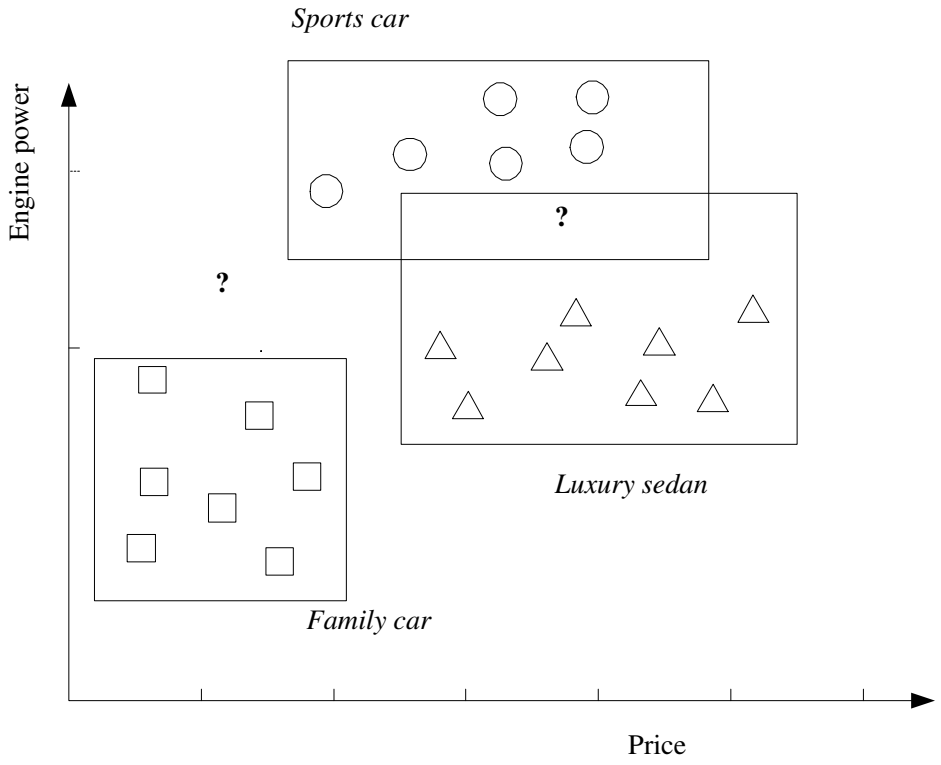


Figure 2.8: There are three classes: family car, sports car, and luxury sedan. There are three hypotheses induced, each one covering the instances of one class and leaving outside the instances of the other two classes. '?' are reject regions where no, or more than one, class is chosen. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

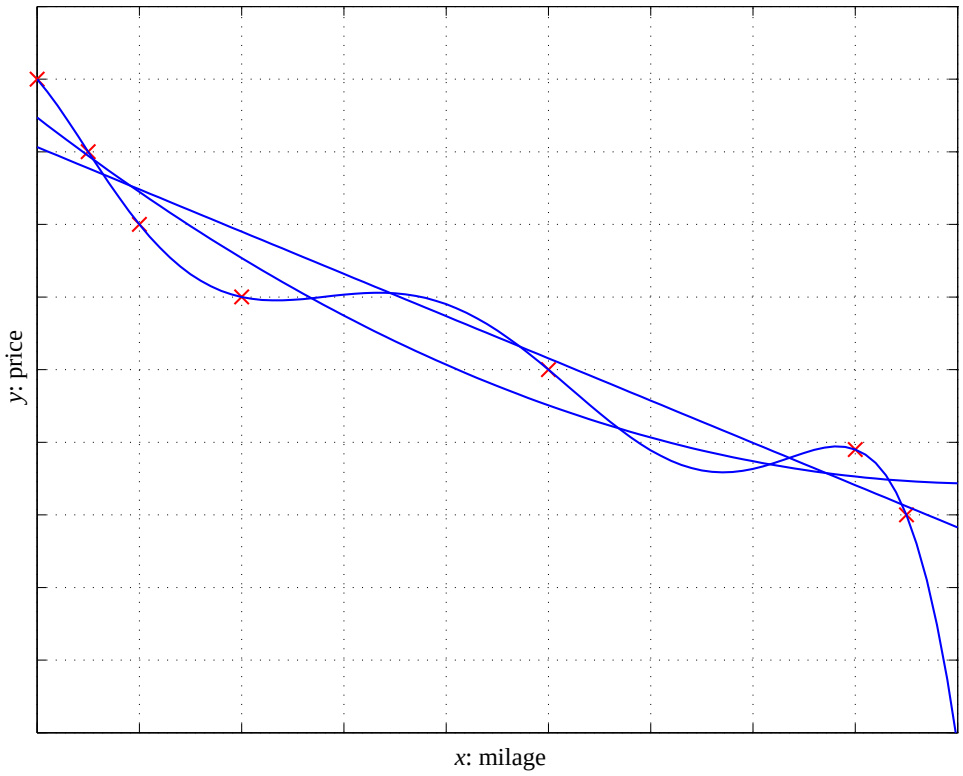


Figure 2.9: Linear, second-order, and sixth-order polynomials are fitted to the same set of points. The highest order gives a perfect fit but given this much data, it is very unlikely that the real curve is so shaped. The second order seems better than the linear fit in capturing the trend in the training data. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Chapter 3:

Bayes Decision Theory

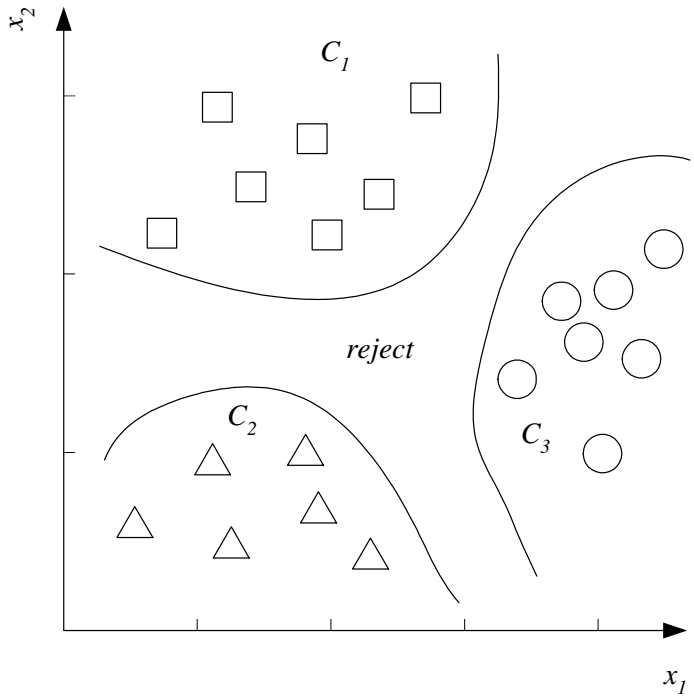


Figure 3.1: Example of decision regions and decision boundaries. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

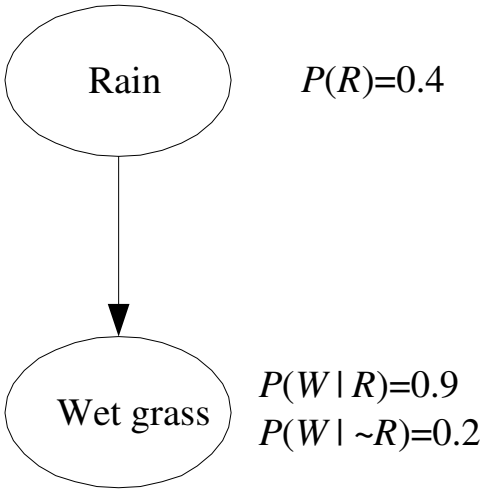


Figure 3.2: Bayesian network modeling that rain is the cause of wet grass. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

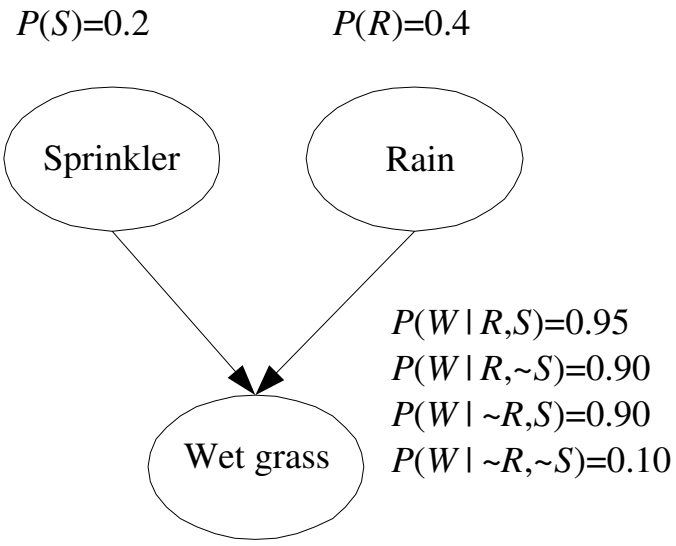


Figure 3.3: Rain and sprinkler are the two causes of wet grass. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

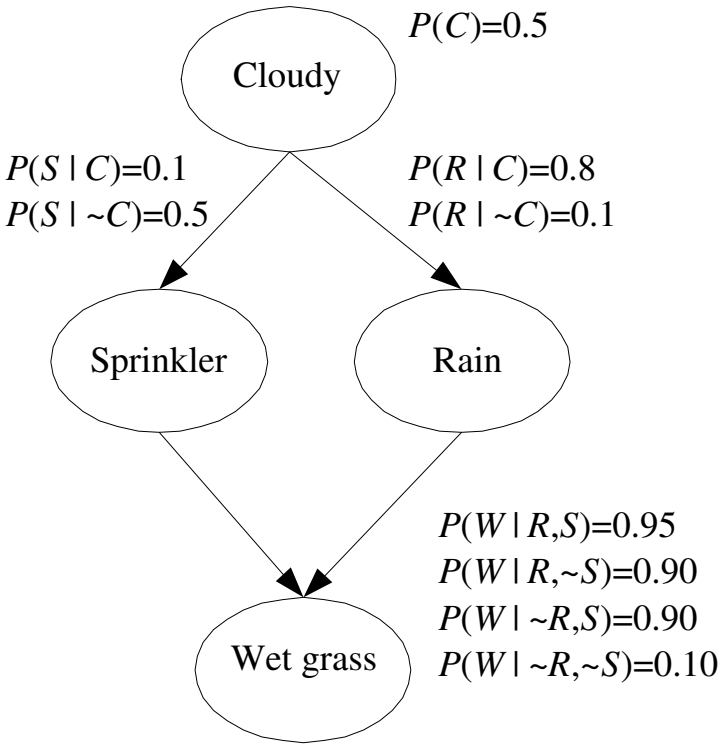


Figure 3.4: If it is cloudy, it is likely that it will rain and we will not use the sprinkler. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

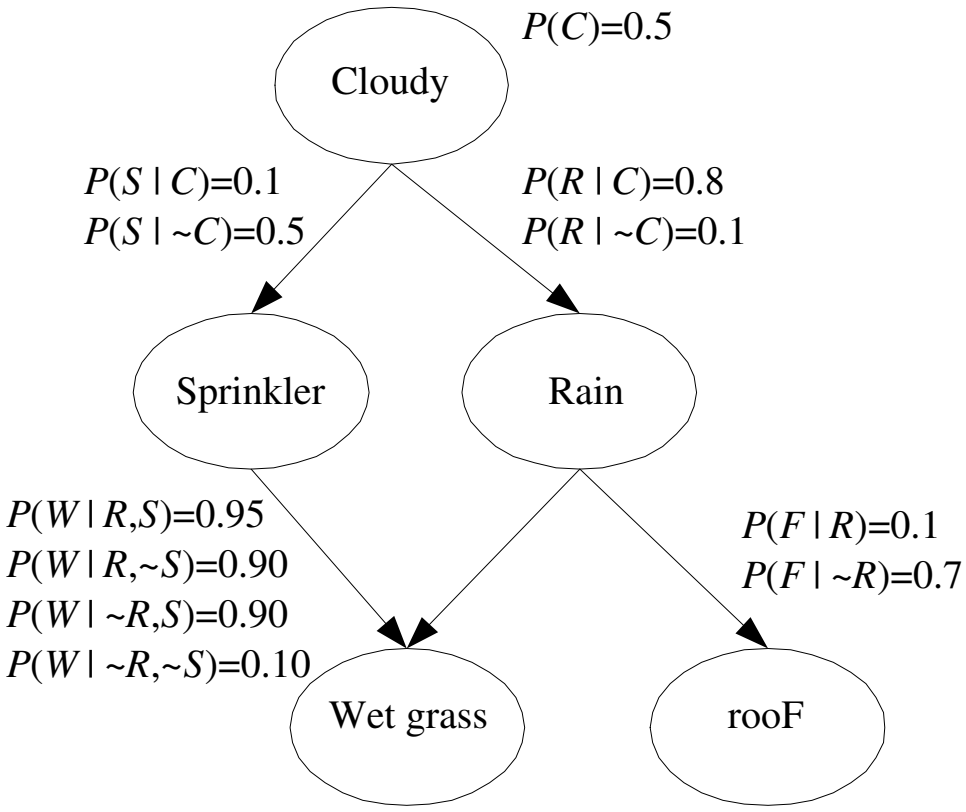


Figure 3.5: Rain not only makes the grass wet but also disturbs the cat who normally makes noise on the roof. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

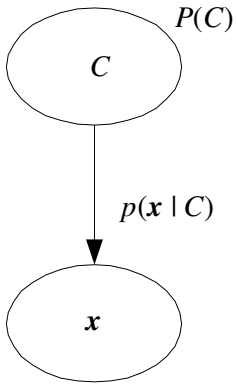


Figure 3.6: Bayesian network for classification.

From: E. Alpaydın. 2004. Introduction to Machine Learning. ©*The MIT Press.*

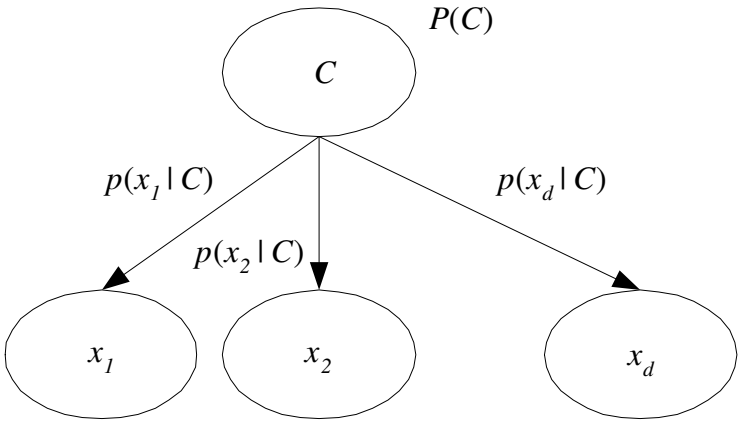


Figure 3.7: Naive Bayes' classifier is a Bayesian network for classification assuming independent inputs. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

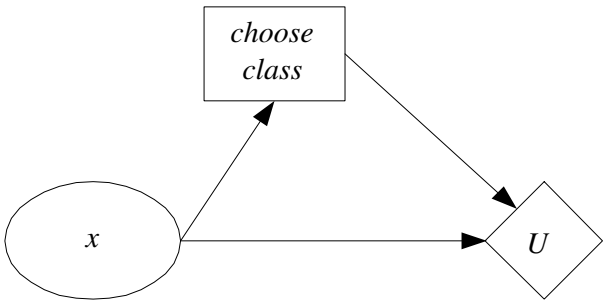


Figure 3.8: Influence diagram corresponding to classification. Depending on input x, a class is chosen that incurs a certain utility (risk). *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Chapter 4:

Parametric Methods

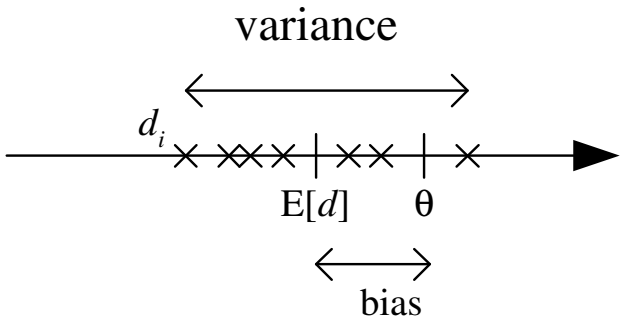


Figure 4.1: θ is the parameter to be estimated. d_i are several estimates (denoted by '$\times$') over different samples. Bias is the difference between the expected value of d and θ. Variance is how much d_i are scattered around the expected value. We would like both to be small. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

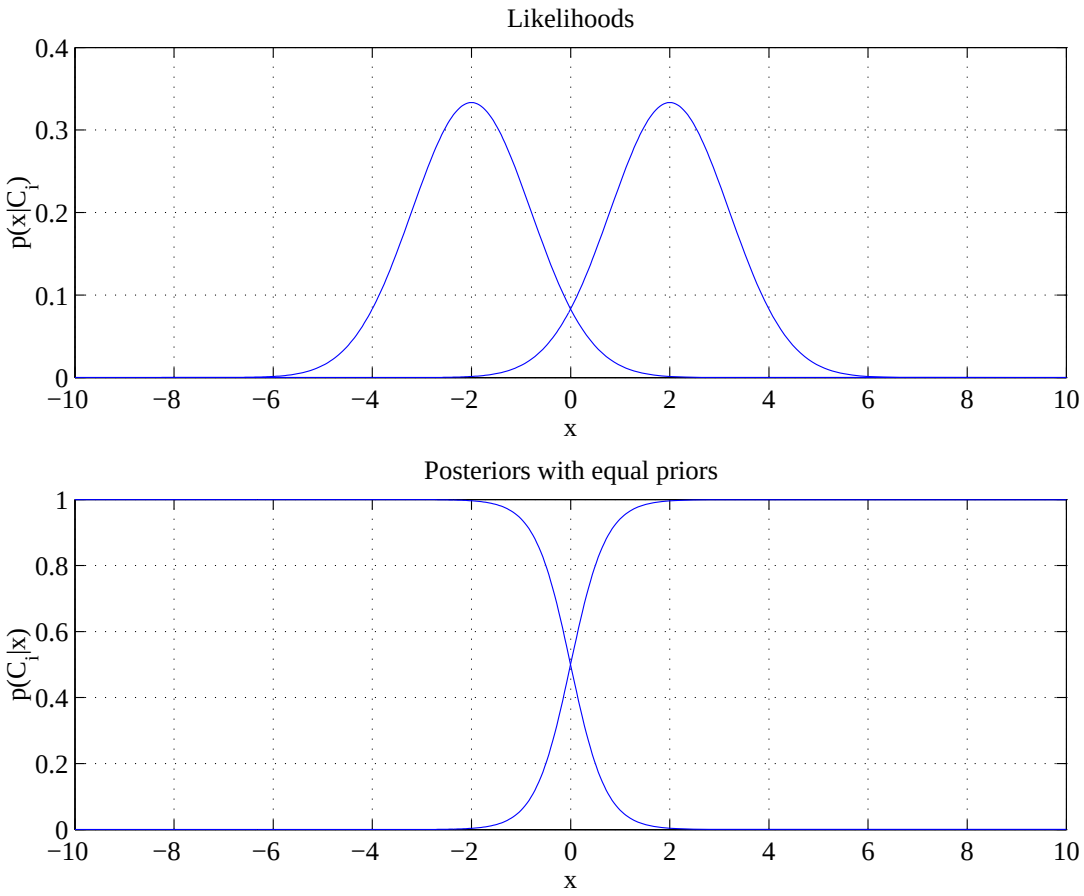


Figure 4.2: Likelihood functions and the posteriors with equal priors for two classes when the input is one-dimensional. Variances are equal and the posteriors intersect at one point, which is the threshold of decision. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

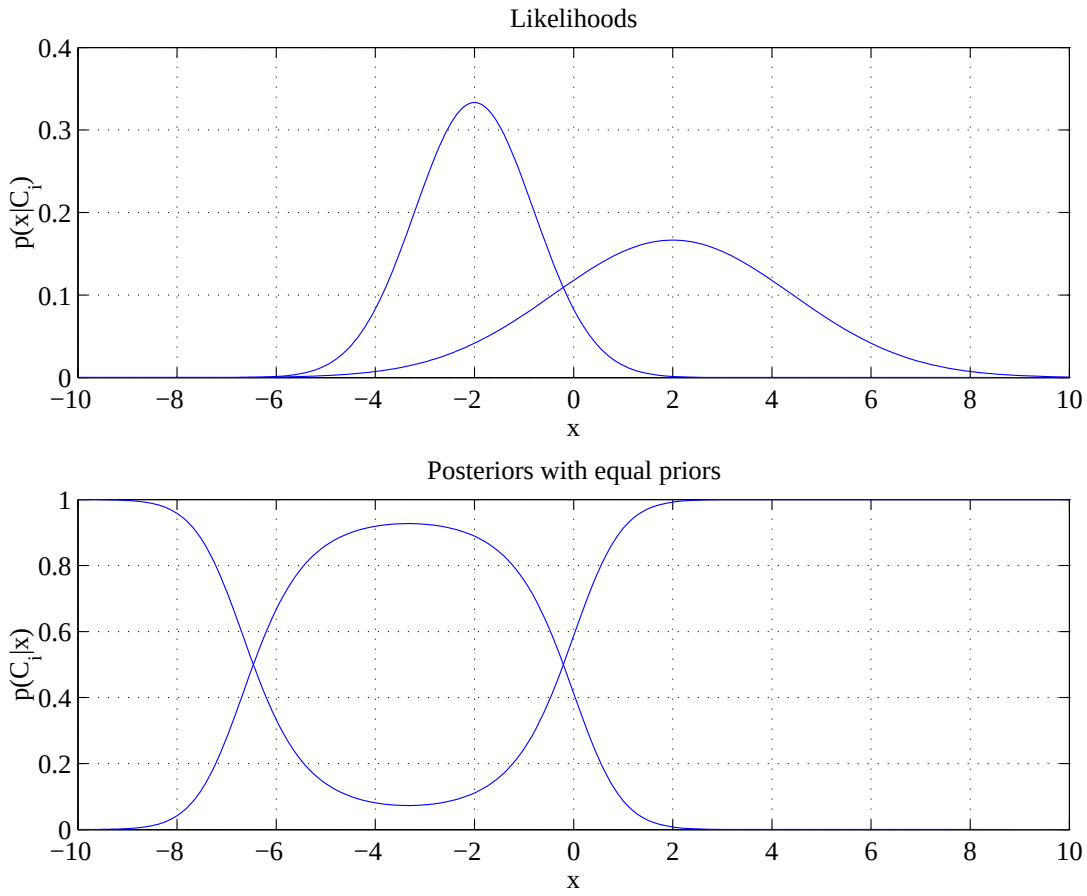


Figure 4.3: Likelihood functions and the posteriors with equal priors for two classes when the input is one-dimensional. Variances are unequal and the posteriors intersect at two points. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

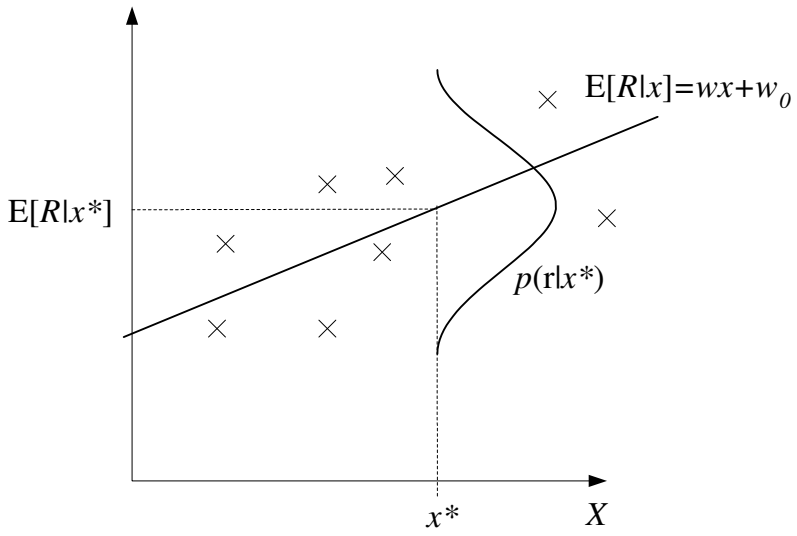


Figure 4.4: Regression assumes 0 mean Gaussian noise added to the model; here, the model is linear. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

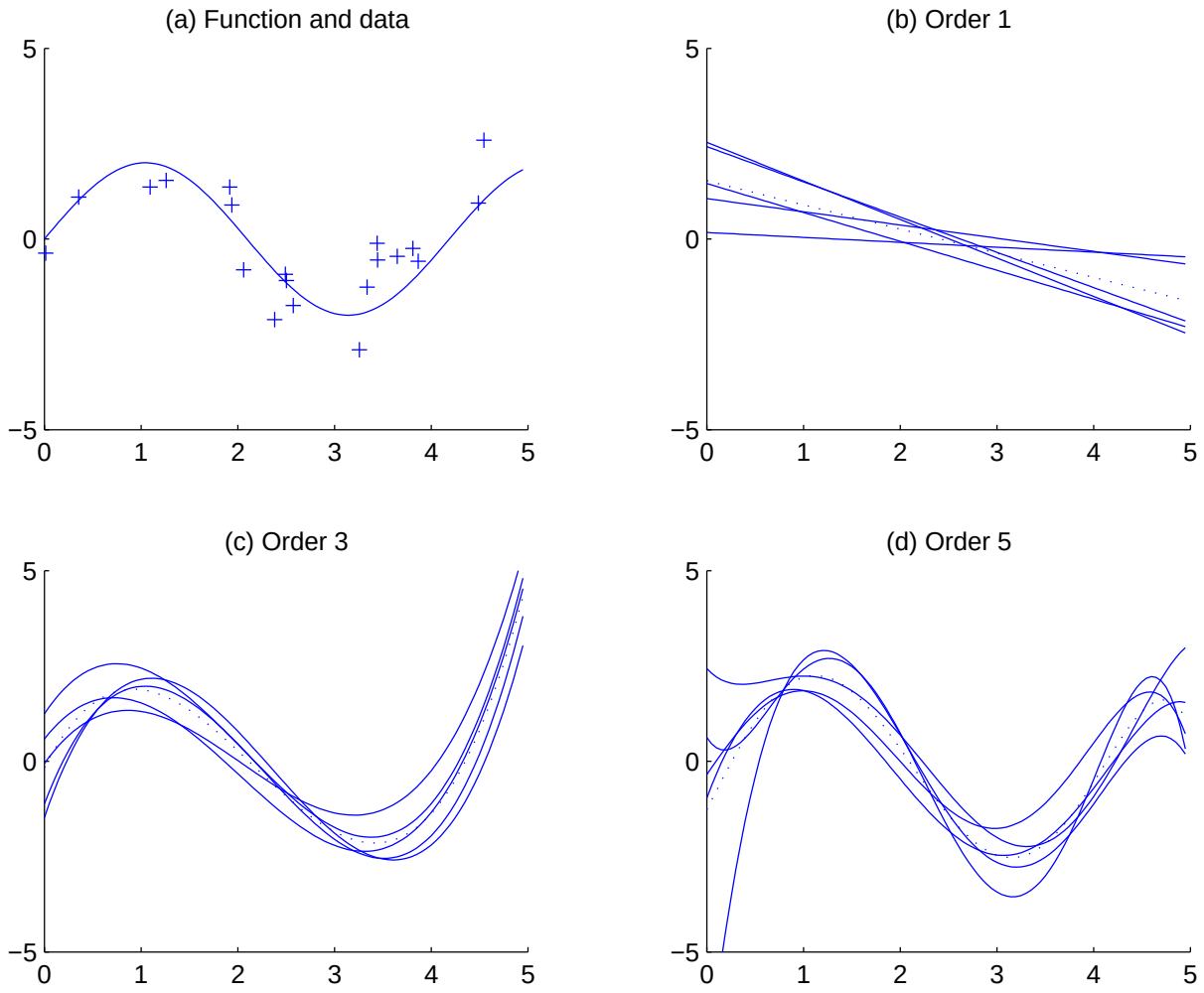


Figure 4.5: (a) Function, $f(x) = 2\sin(1.5x)$, and one noisy ($\mathcal{N}(0, 1)$) dataset sampled from the function. Five samples are taken, each containing twenty instances. (b), (c), (d) are five polynomial fits, $g_i(\cdot)$, of order 1, 3, and 5. For each case, dotted line is the average of the five fits, namely, $\overline{g}(\cdot)$. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

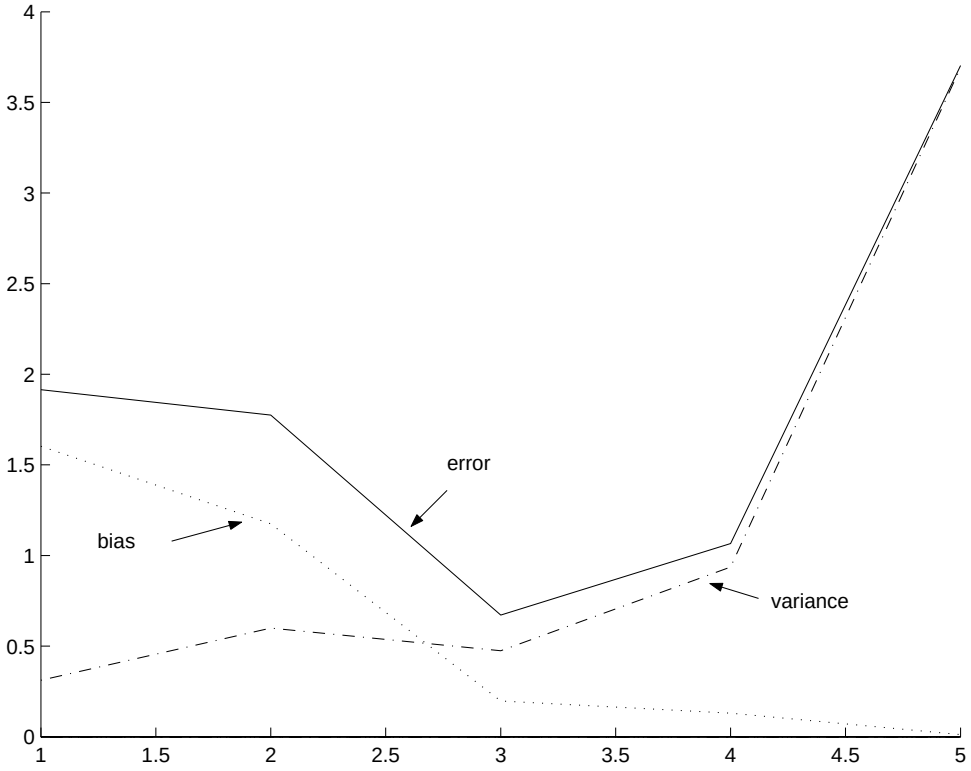


Figure 4.6: In the same setting as that of figure 4.5, using one hundred models instead of five, bias, variance, and error for polynomials of order 1 to 5. Order 1 has the smallest variance. Order 5 has the smallest bias. As the order is increased, bias decreases but variance increases. Order 3 has the minimum error. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

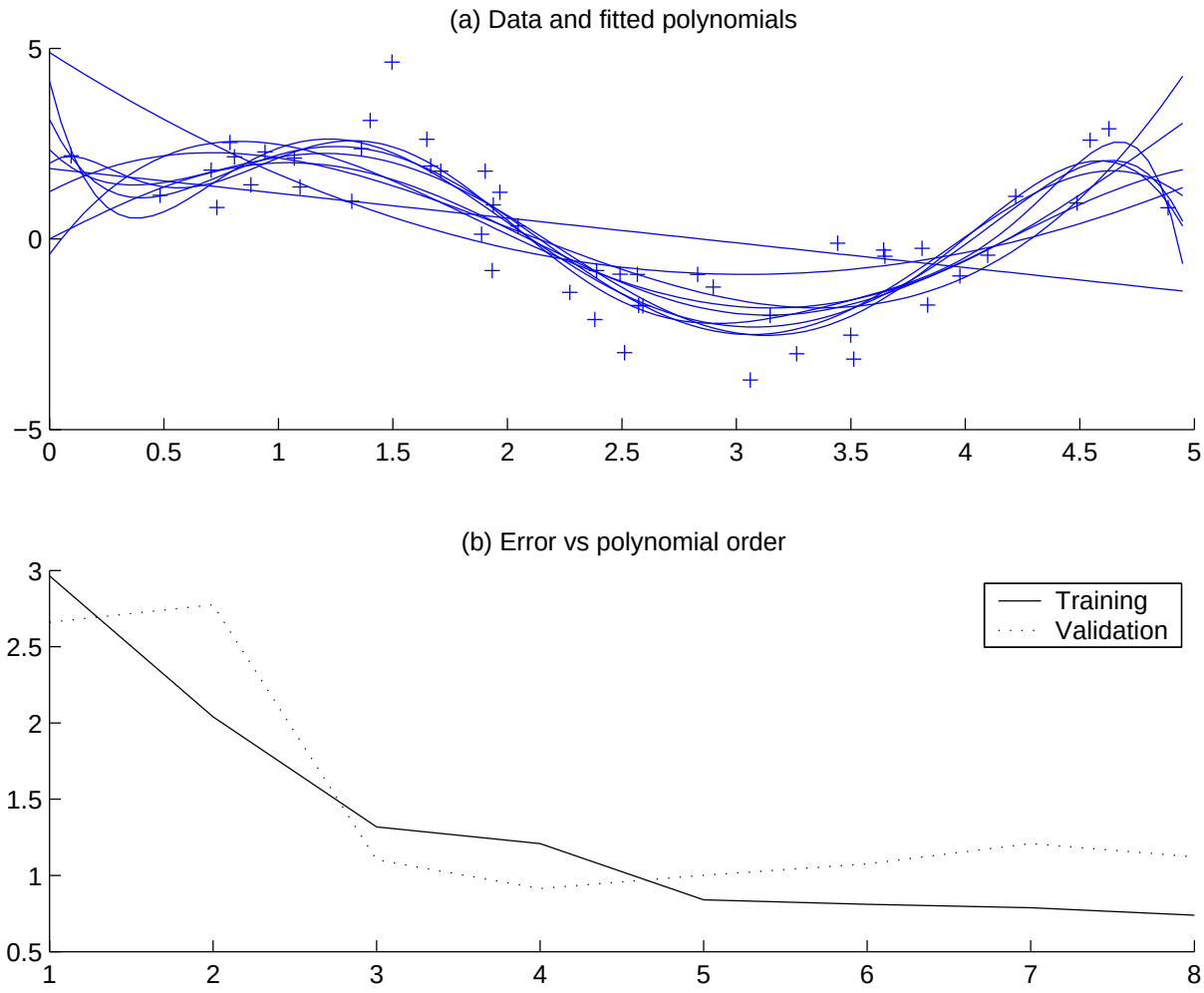


Figure 4.7: In the same setting as that of figure 4.5, training and validation sets (each containing 50 instances) are generated. (a) Training data and fitted polynomials of order from 1 to 8. (b) Training and validation errors as a function of the polynomial order. The "elbow" is at 3. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Chapter 5:

Multivariate Methods

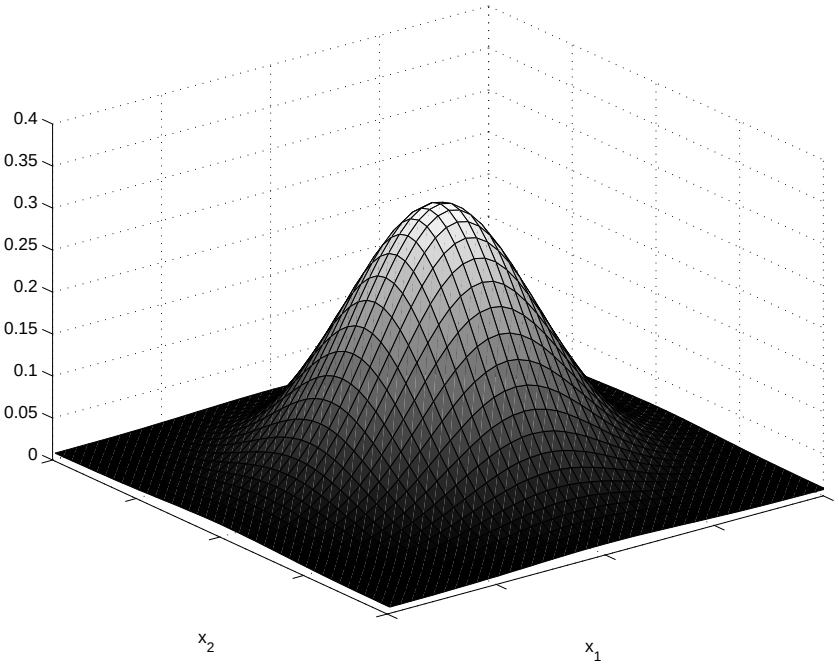


Figure 5.1: Bivariate normal distribution. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

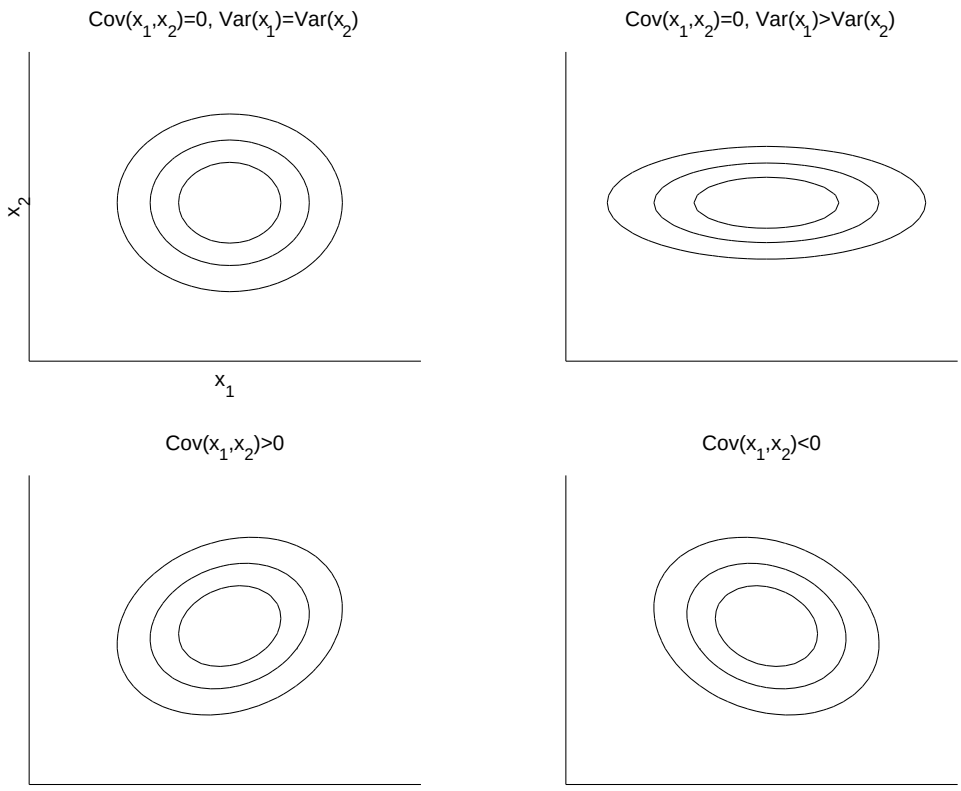


Figure 5.2: Isoprobability contour plot of the bivariate normal distribution. Its center is given by the mean, and its shape and orientation depend on the covariance matrix. *From: E. Alpaydın. 2004.* Introduction to Machine Learning.

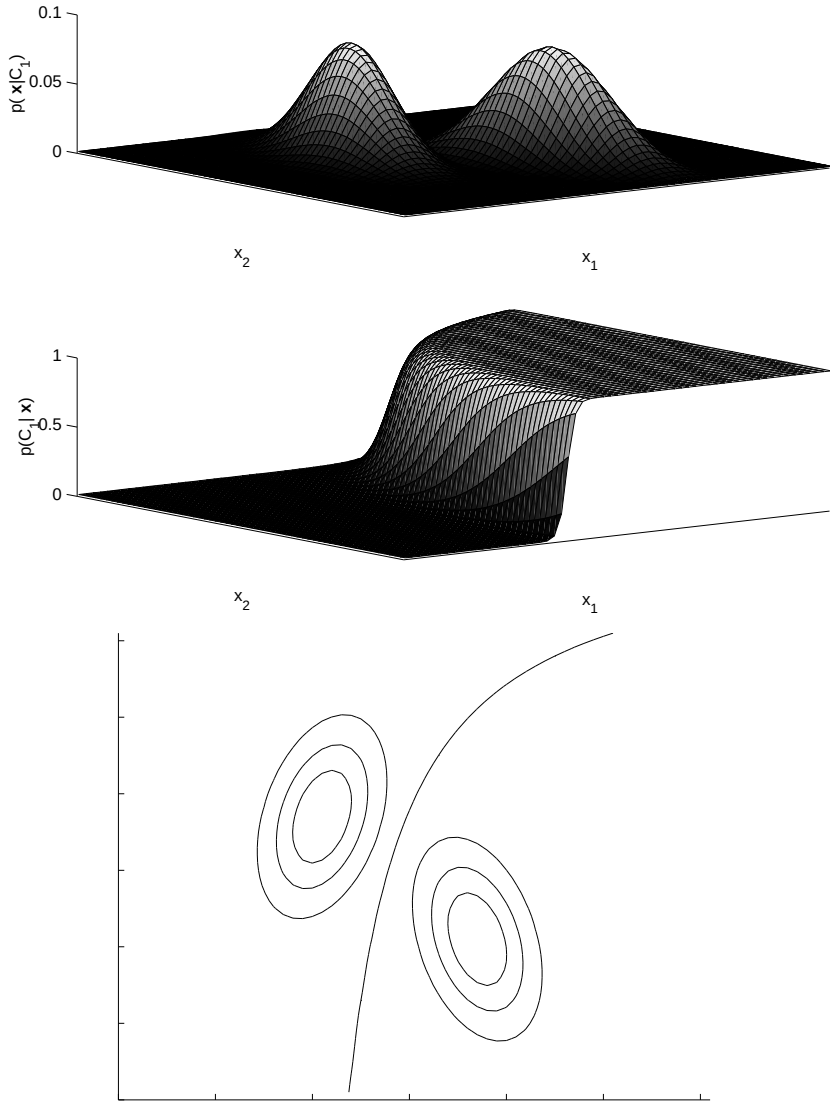


Figure 5.3: Classes have different covariance matrices. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

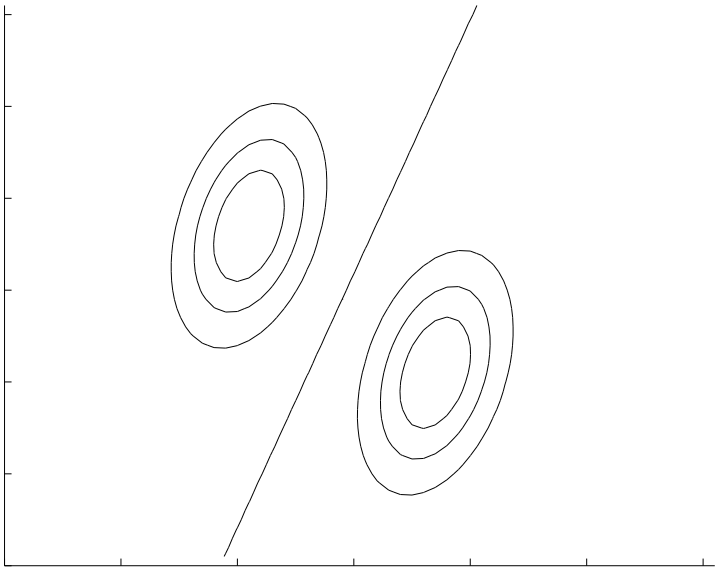


Figure 5.4: Covariances may be arbitary but shared by both classes. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

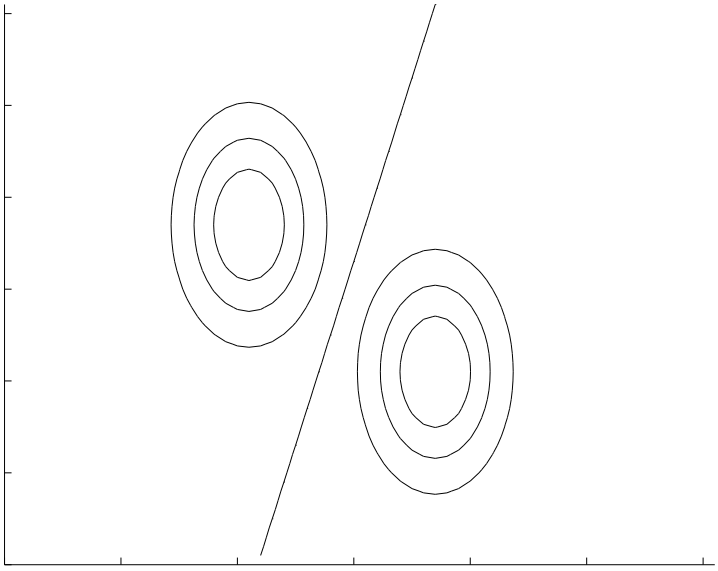


Figure 5.5: All classes have equal, diagonal covariance matrices but variances are not equal. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

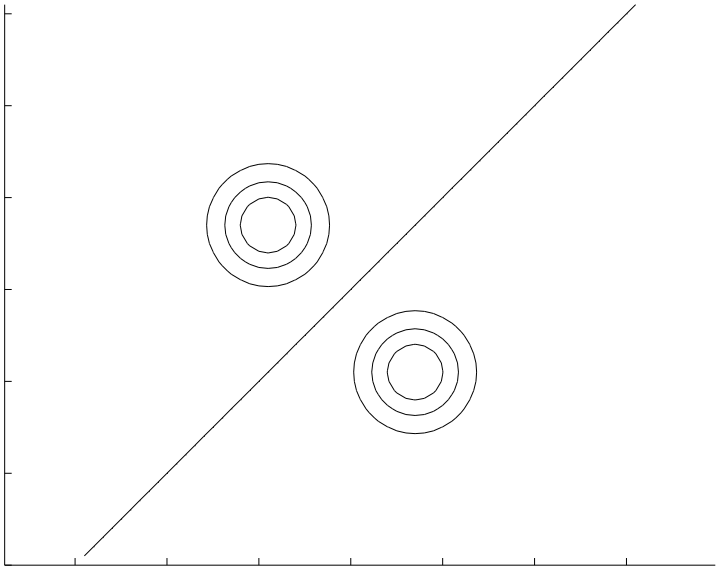


Figure 5.6: All classes have equal, diagonal covariance matrices of equal variances on both dimensions. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Chapter 6:

Dimensionality Reduction

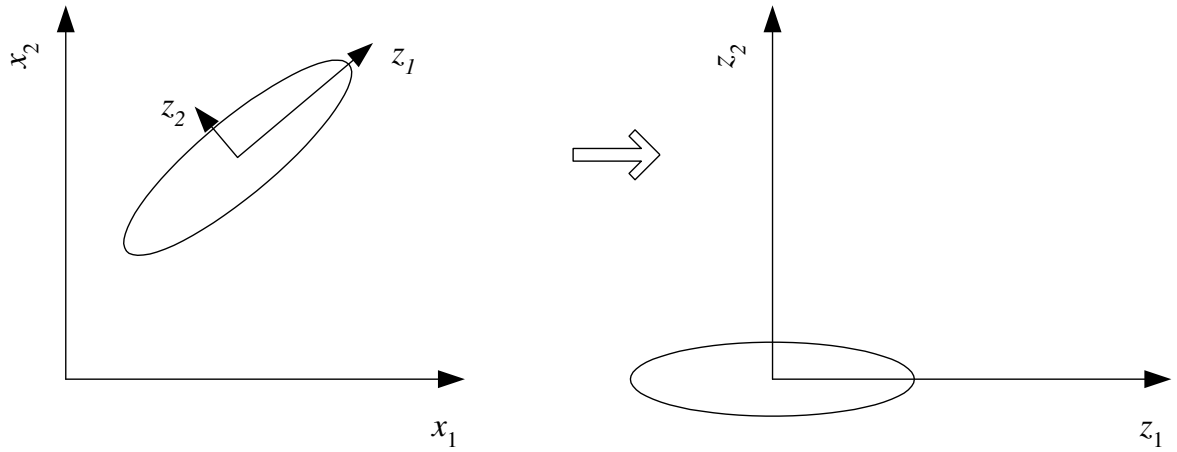


Figure 6.1: Principal components analysis centers the sample and then rotates the axes to line up with the directions of highest variance. If the variance on z_2 is too small, it can be ignored and we have dimensionality reduction from two to one. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

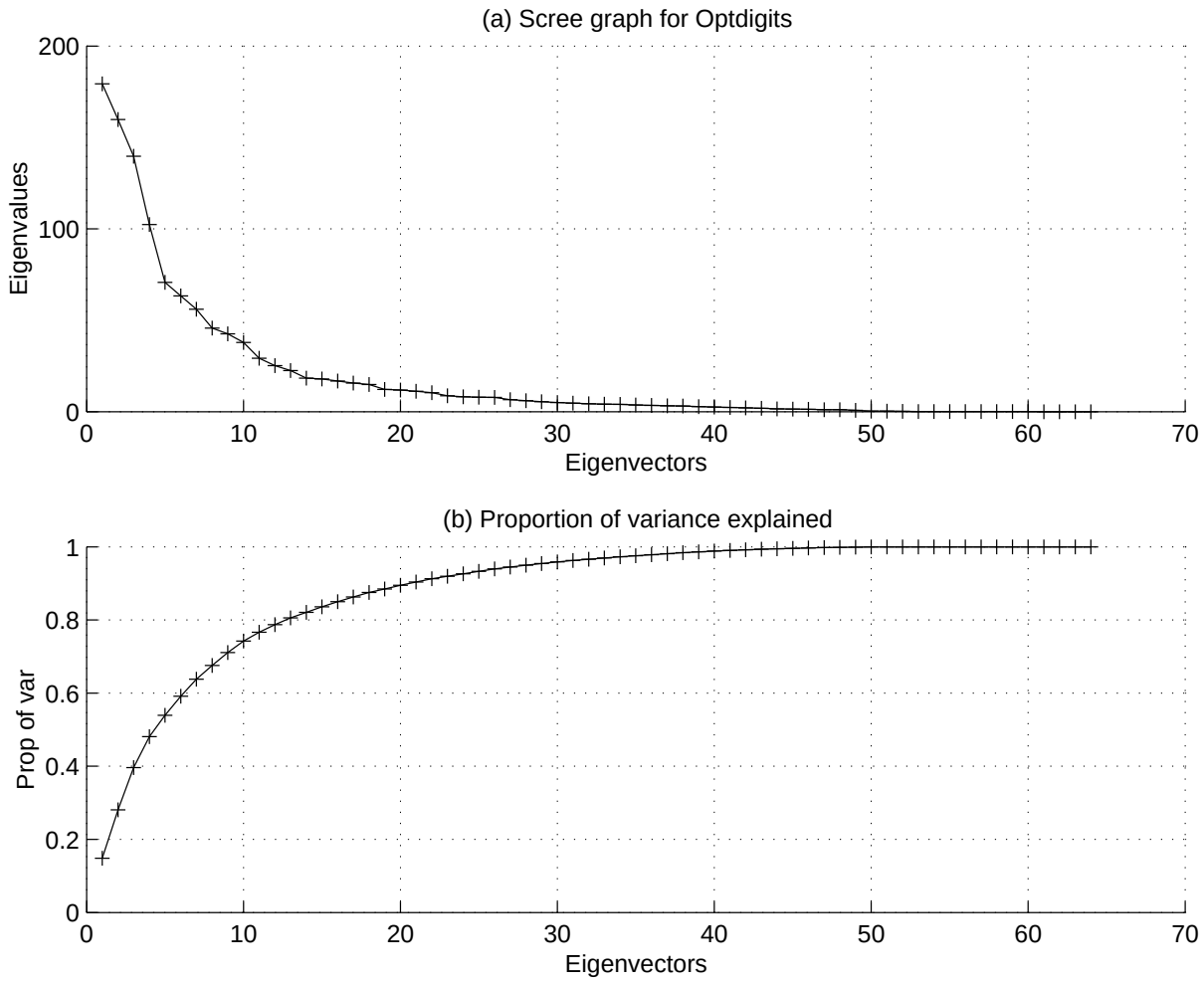


Figure 6.2: (a) Scree graph. (b) Proportion of variance explained is given for the Optdigits dataset from the UCI Repository. This is a handwritten digit dataset with ten classes and sixty-four dimensional inputs. The first twenty eigenvectors explain 90 percent of the variance. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

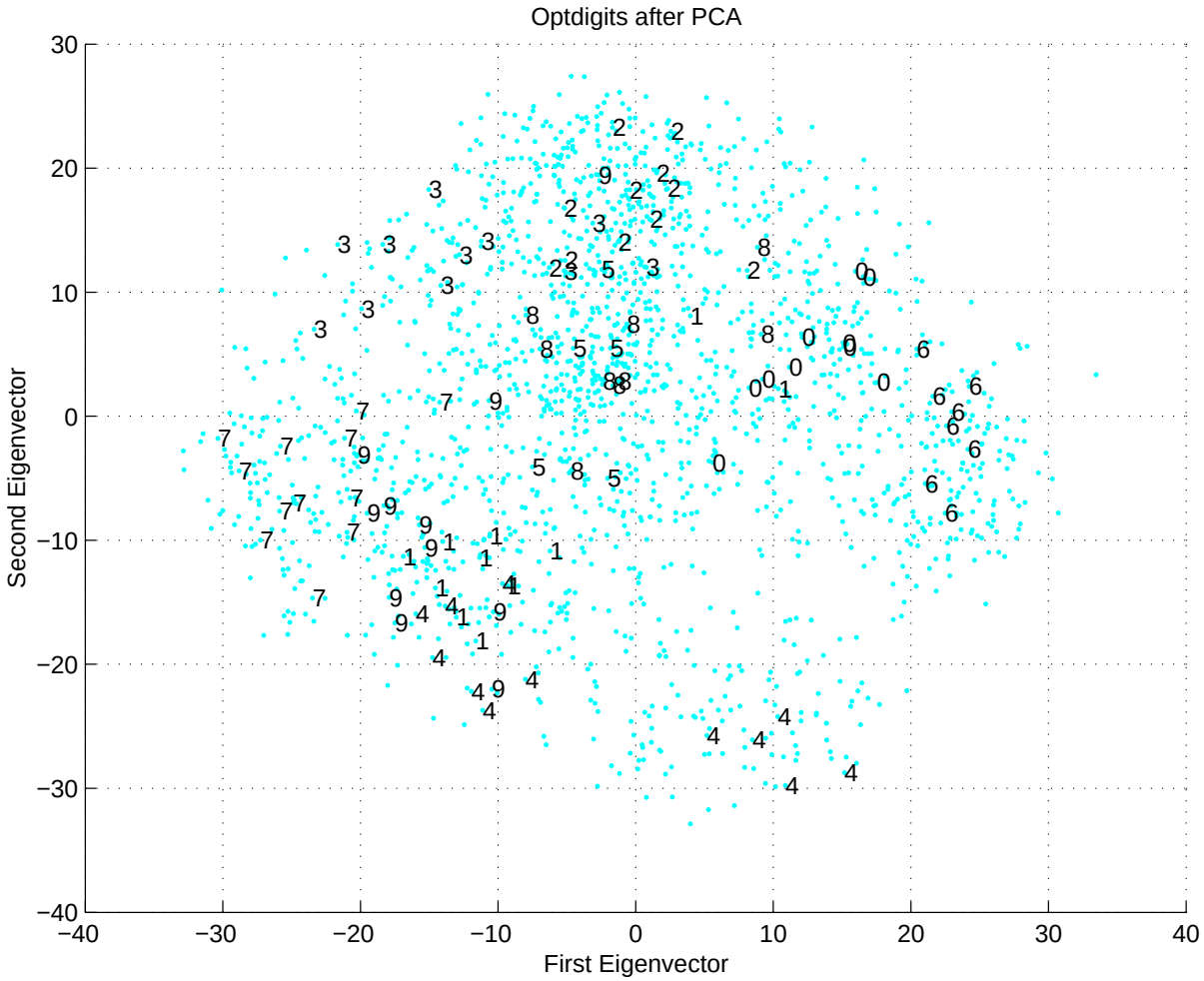


Figure 6.3: Optdigits data plotted in the space of two principal components. Only the labels of hundred data points are shown to minimize the ink-to-noise ratio. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

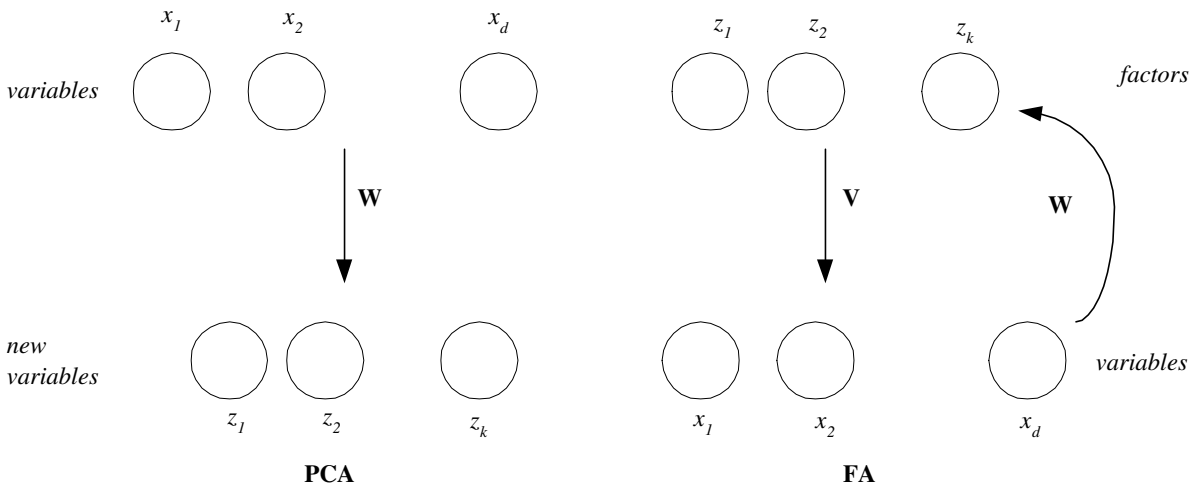


Figure 6.4: Principal components analysis generates new variables that are linear combinations of the original input variables. In factor analysis, however, we posit that there are factors that when linearly combined generate the input variables. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

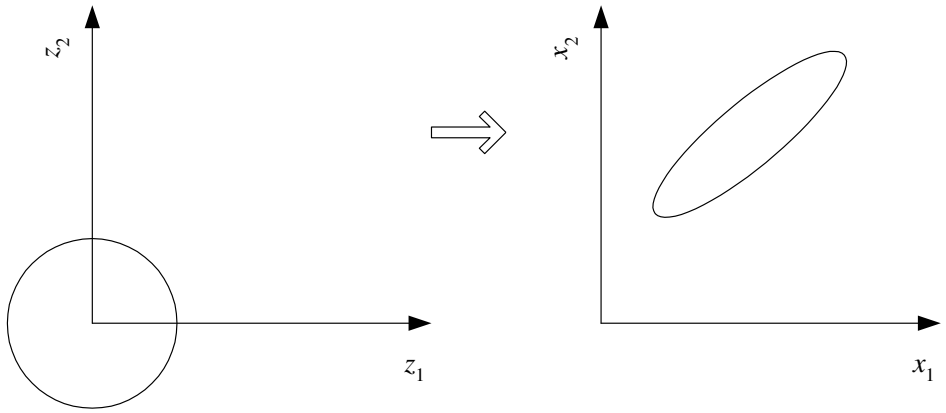


Figure 6.5: Factors are independent unit normals that are stretched, rotated, and translated to make up the inputs. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

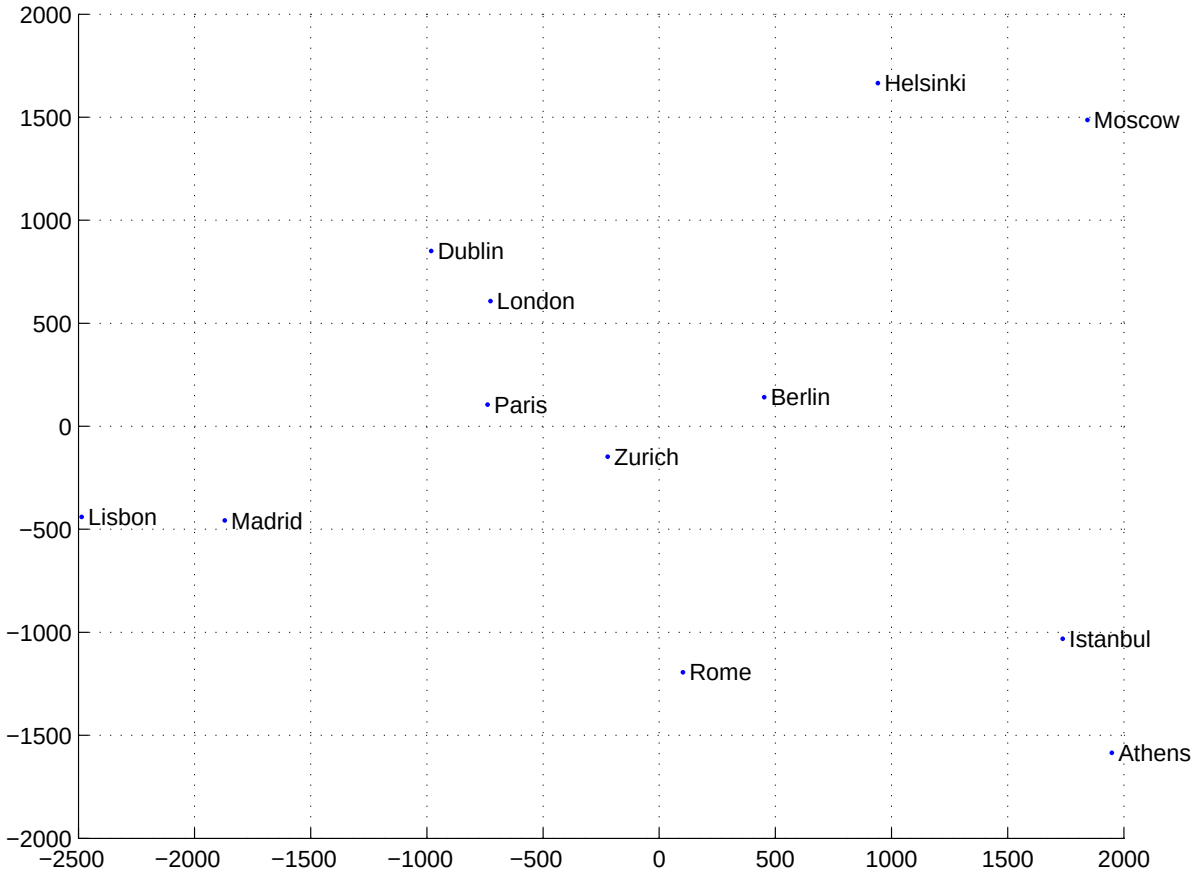


Figure 6.6: Map of Europe drawn by MDS. Cities include Athens, Berlin, Dublin, Helsinki, Istanbul, Lisbon, London, Madrid, Moscow, Paris, Rome, and Zurich. Pairwise road travel distances between these cities are given as input, and MDS places them in two dimensions such that these distances are preserved as well as possible. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

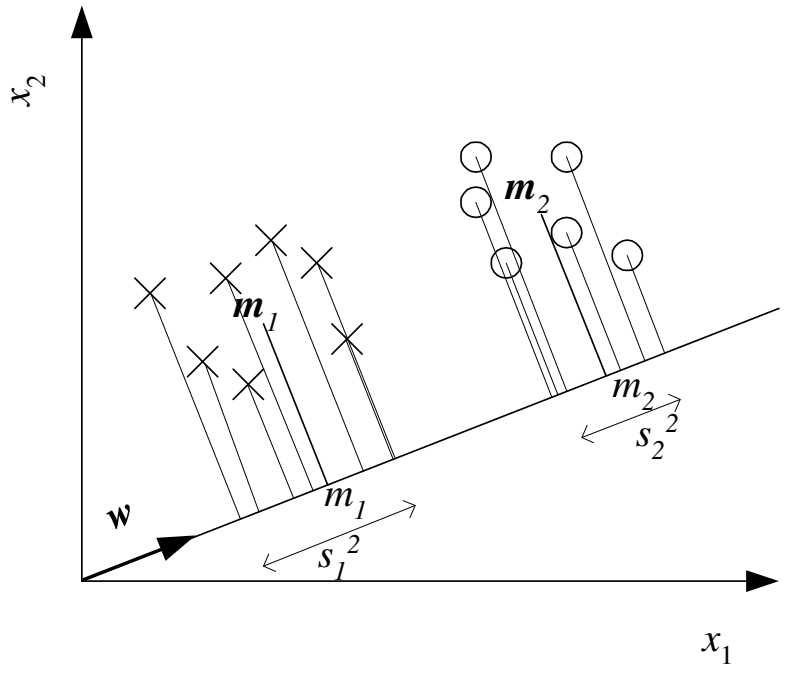


Figure 6.7: Two-dimensional, two-class data projected on $\boldsymbol{w}$. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. © *The MIT Press.*

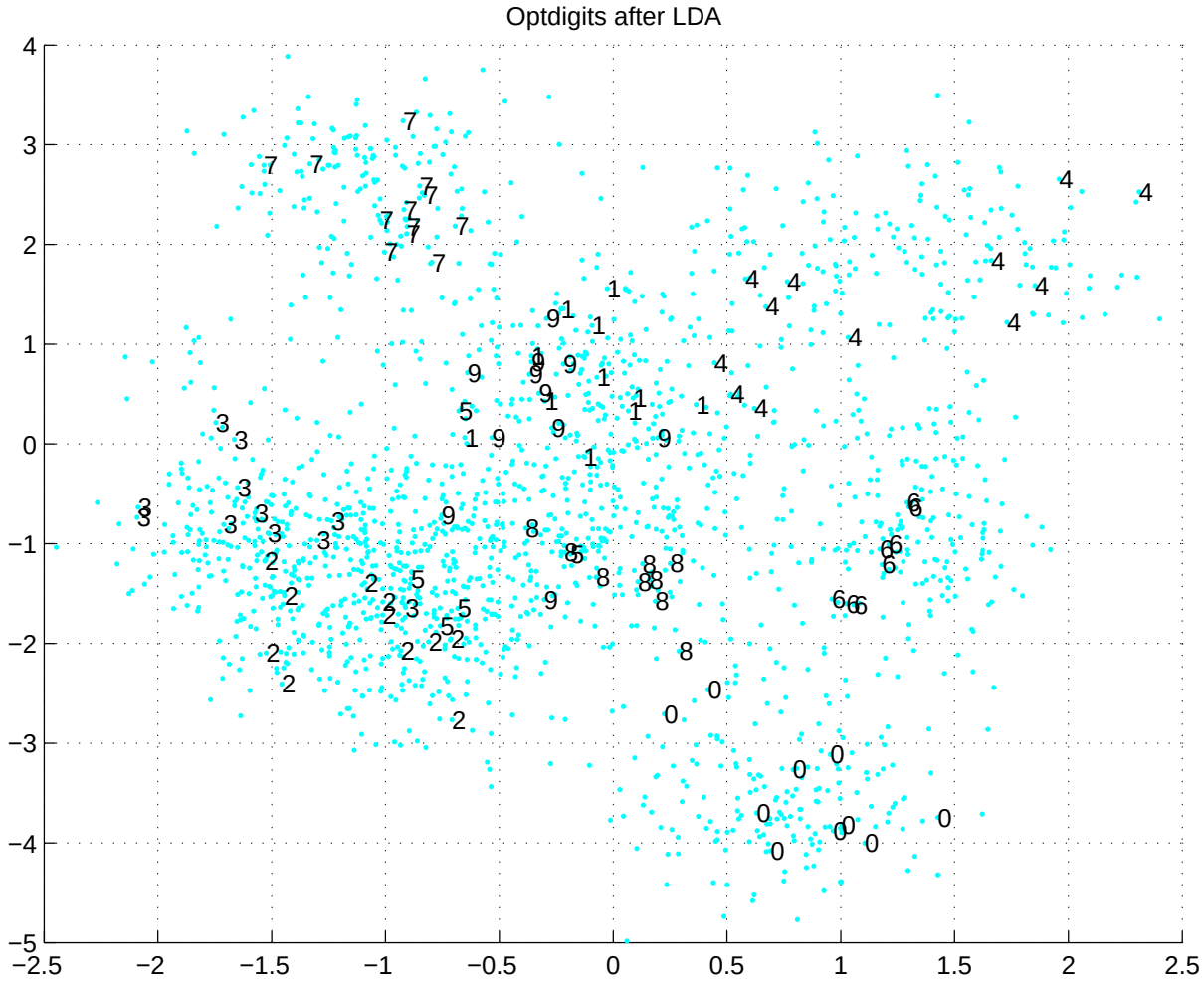


Figure 6.8: Optdigits data plotted in the space of the first two dimensions found by LDA. Comparing this with figure 6.3, we see that LDA, as expected, leads to a better separation of classes than PCA. Even in this two-dimensional space (there are nine), we can discern separate clouds for different classes. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Chapter 7:

Clustering

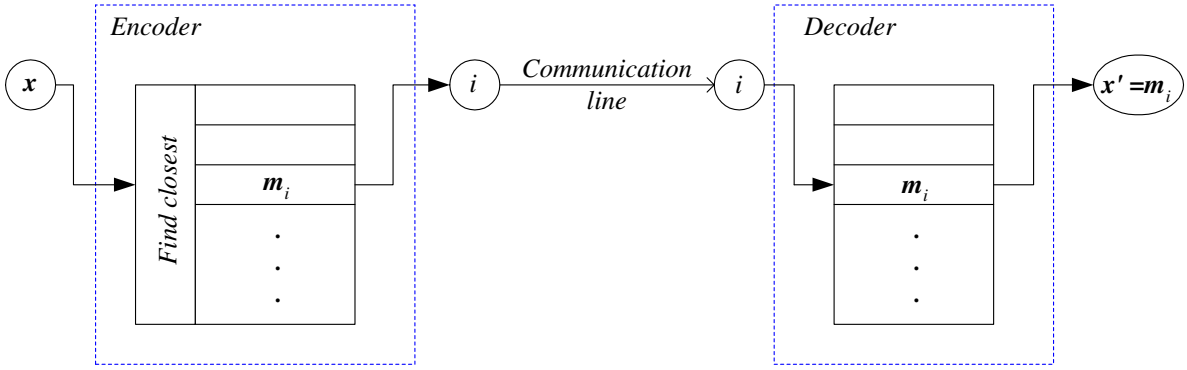


Figure 7.1: Given $\boldsymbol{x}$, the encoder sends the index of the closest code word and the decoder generates the code word with the received index as $\boldsymbol{x}'$. Error is $\|\boldsymbol{x}' - \boldsymbol{x}\|^2$. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

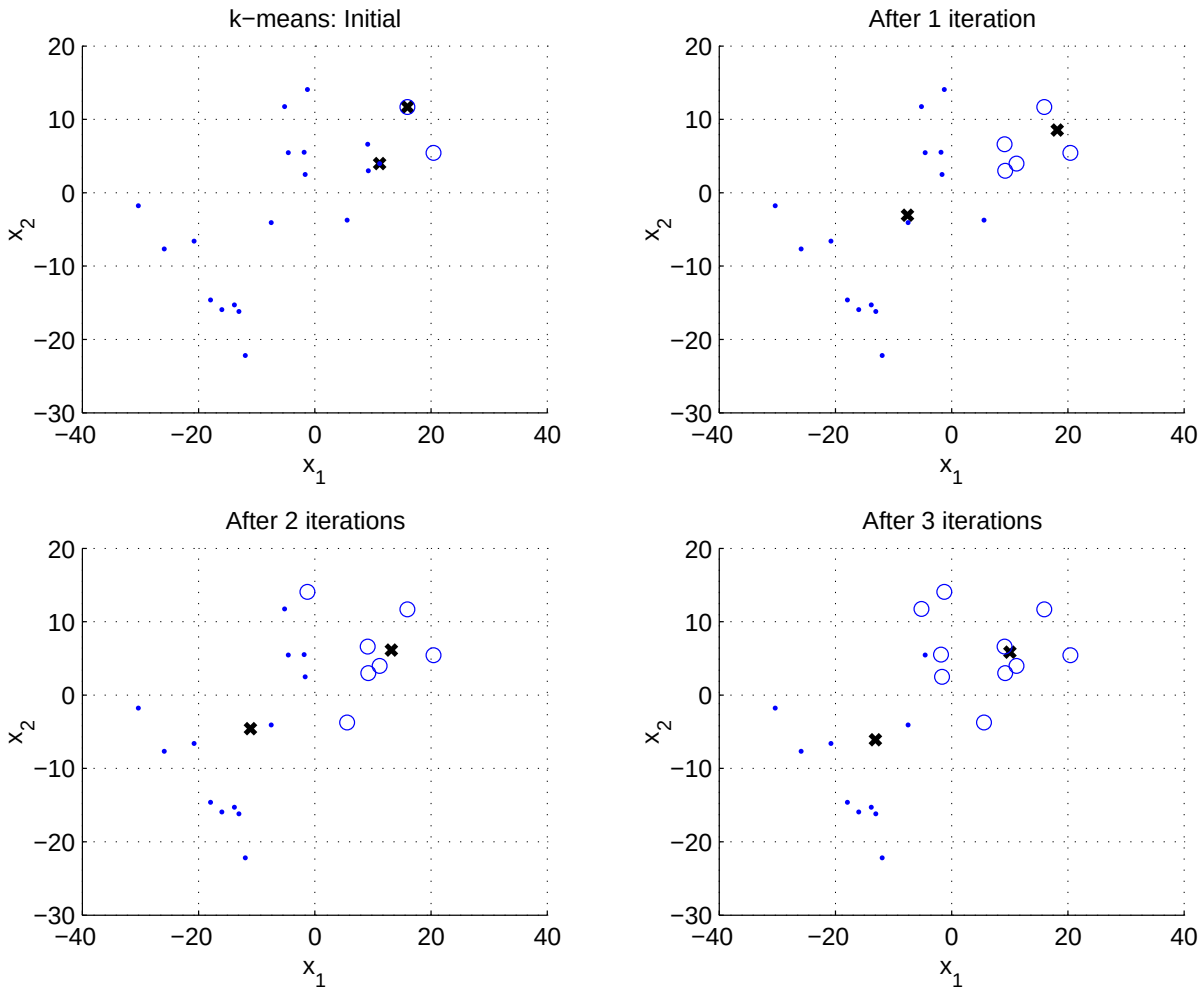


Figure 7.2: Evolution of k-means. Crosses indicate center positions. Data points are marked depending on the closest center. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Initialize $\boldsymbol{m}_i, i = 1, \ldots, k$, for example, to k random $\boldsymbol{x}^t$

Repeat

For all $\boldsymbol{x}^t \in \mathcal{X}$

$$b_i^t \leftarrow \begin{cases} 1 & \text{if } \|\boldsymbol{x}^t - \boldsymbol{m}_i\| = \min_j \|\boldsymbol{x}^t - \boldsymbol{m}_j\| \\ 0 & \text{otherwise} \end{cases}$$

For all $\boldsymbol{m}_i, i = 1, \ldots, k$

$$\boldsymbol{m}_i \leftarrow \sum_t b_i^t \boldsymbol{x}^t / \sum_t b_i^t$$

Until $\boldsymbol{m}_i$ converge

Figure 7.3: k-means algorithm. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

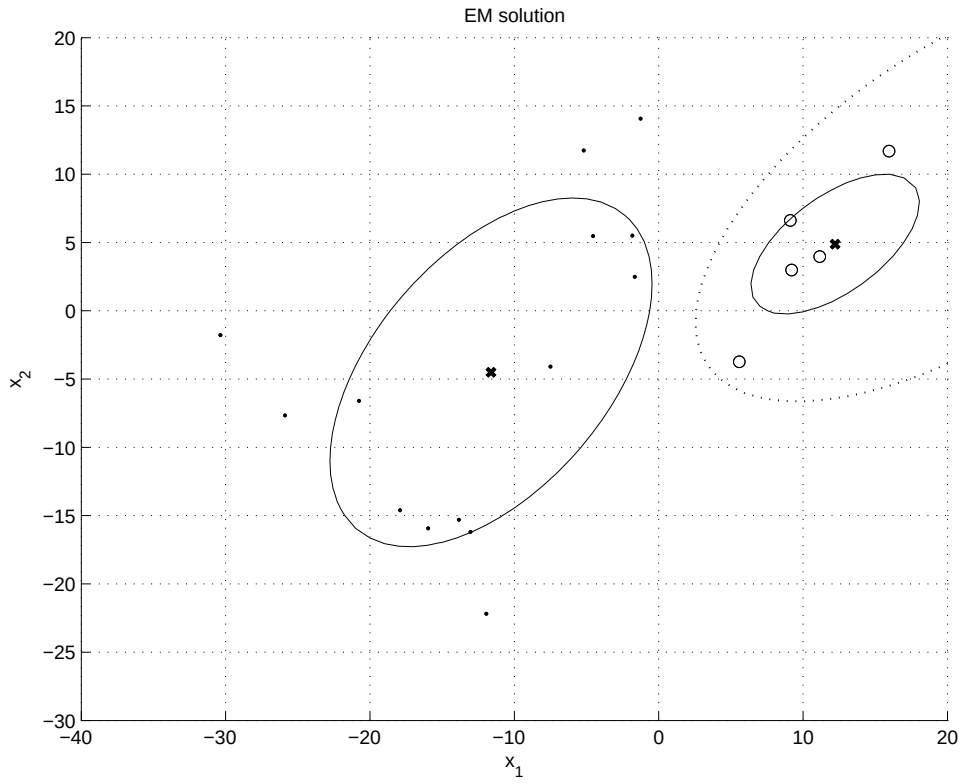


Figure 7.4: Data points and the fitted Gaussians by EM, initialized by one k-means iteration of figure 7.2. Unlike in k-means, as can be seen, EM allows estimating the covariance matrices. The data points labeled by greater h_i, the contours of the estimated Gaussian densities, and the separating curve of $h_i = 0.5$ (dashed line) are shown. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

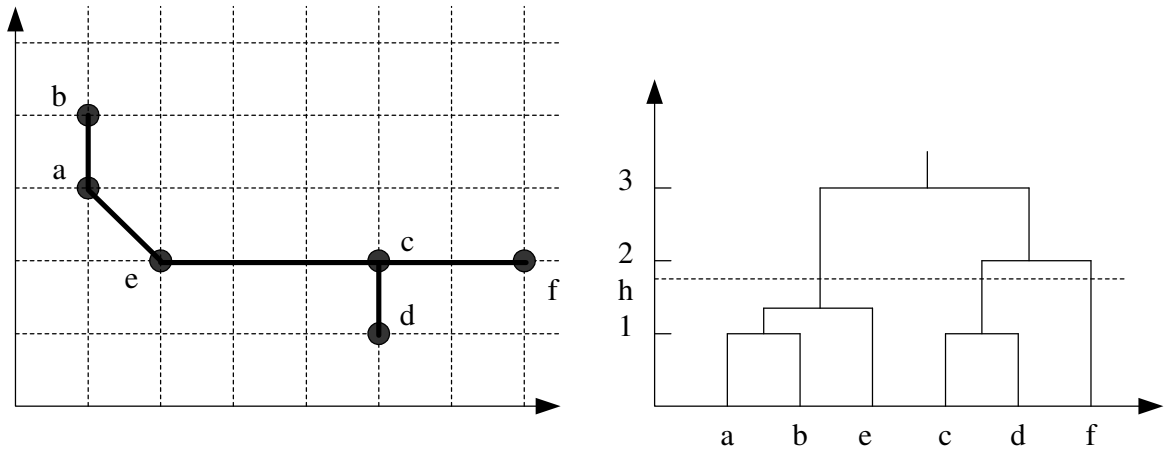


Figure 7.5: A two-dimensional dataset and the dendrogram showing the result of single-link clustering is shown. Note that leaves of the tree are ordered so that no branches cross. The tree is then intersected at a desired value of h to get the clusters. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Chapter 8:

Nonparametric Methods

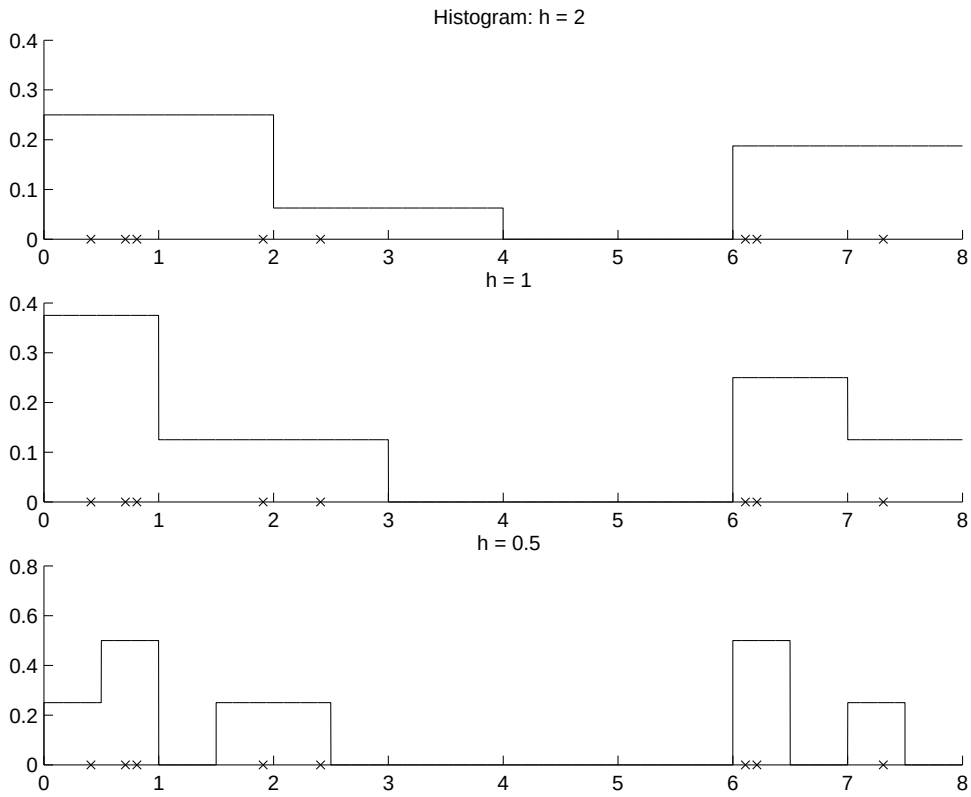


Figure 8.1: Histograms for various bin lengths. '×' denote data points. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

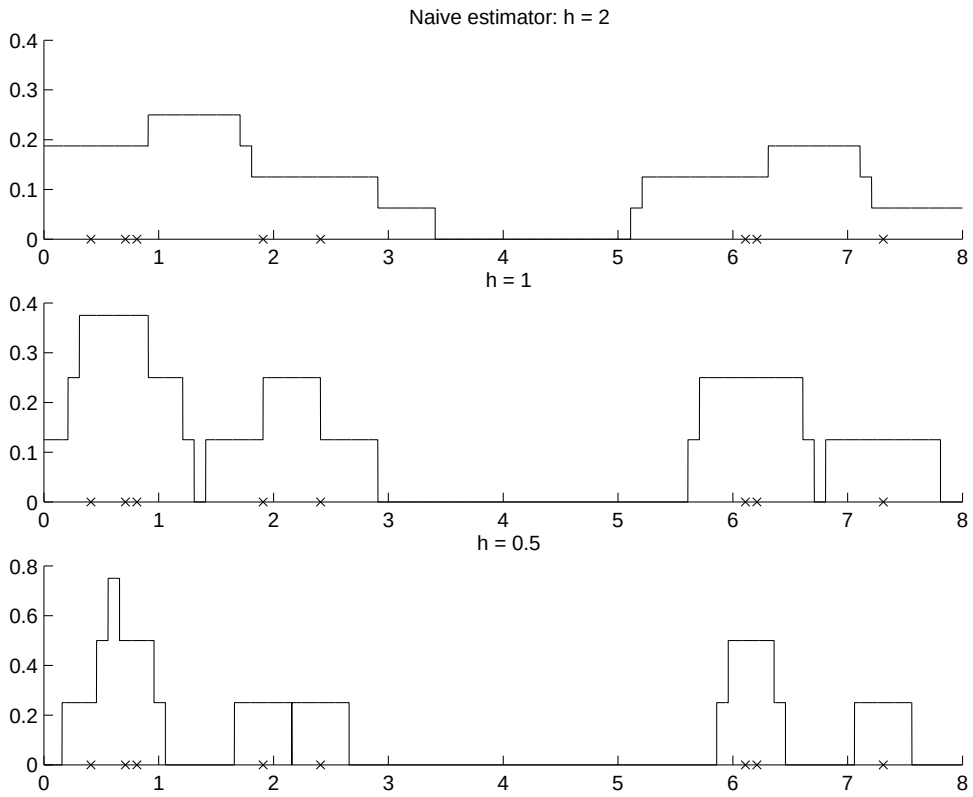


Figure 8.2: Naive estimate for various bin lengths.

From: E. Alpaydın. 2004. Introduction to Machine Learning. ©*The MIT Press.*

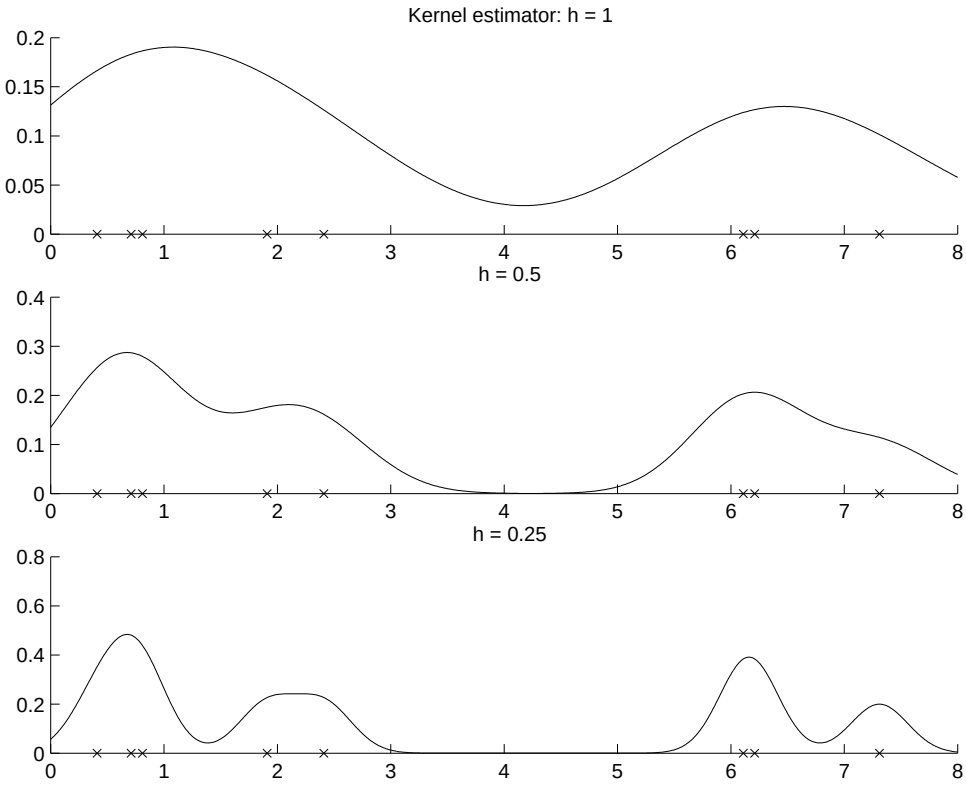


Figure 8.3: Kernel estimate for various bin lengths. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

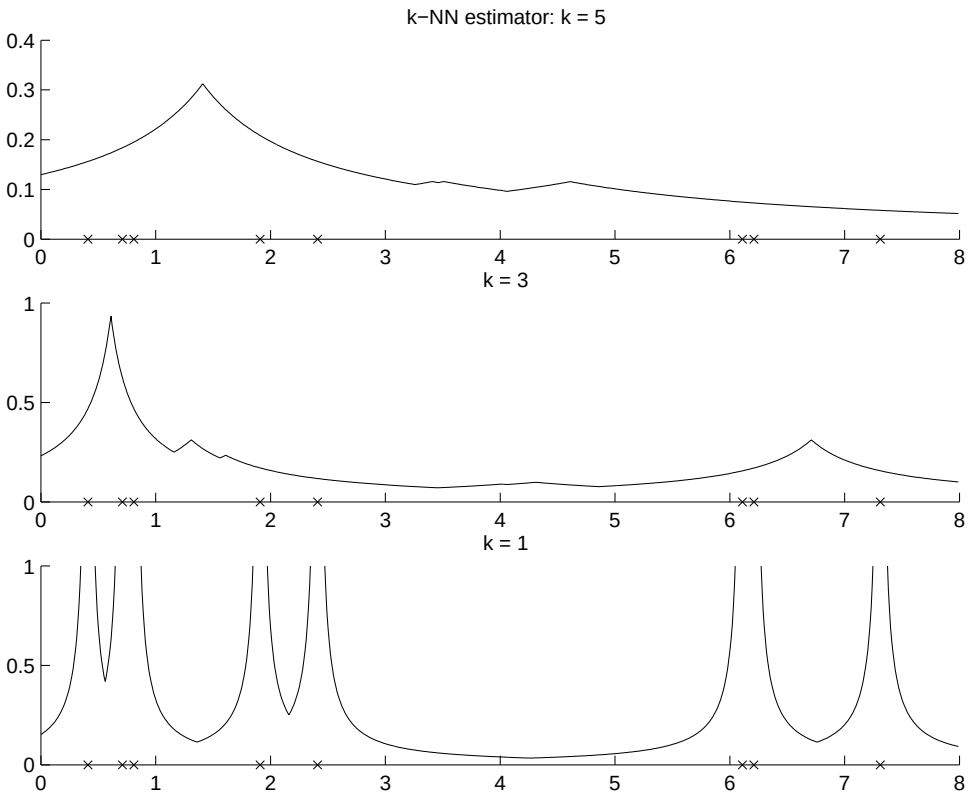


Figure 8.4: k-nearest neighbor estimate for various k values. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

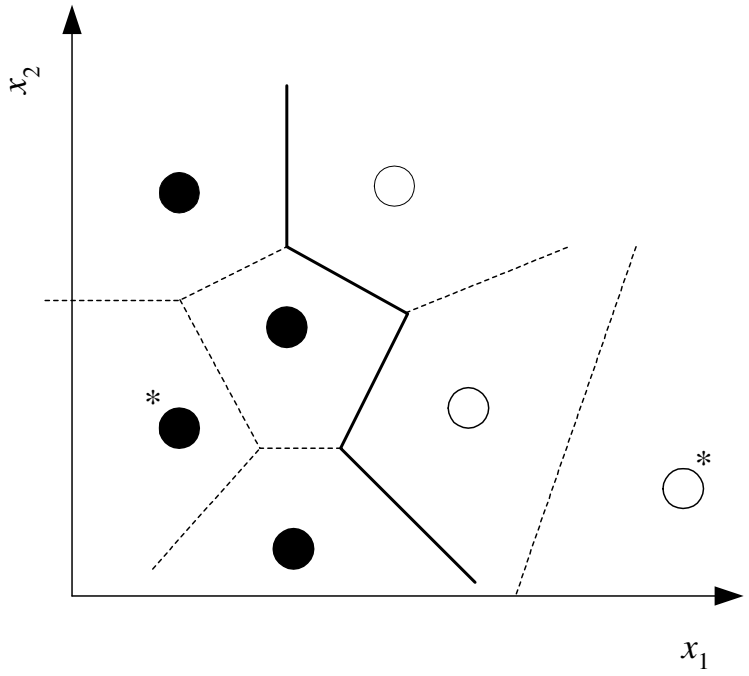


Figure 8.5: Dotted lines are the Voronoi tesselation and the straight line is the class discriminant. In condensed nearest neighbor, those instances that do not participate in defining the discriminant (marked by '*') can be removed without increasing the training error. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

$\mathcal{Z} \leftarrow \emptyset$
Repeat
    For all $\boldsymbol{x} \in \mathcal{X}$ (in random order)
        Find $\boldsymbol{x}' \in \mathcal{Z}$ s.t. $\|\boldsymbol{x} - \boldsymbol{x}'\| = \min_{\boldsymbol{x}^j \in \mathcal{Z}} \|\boldsymbol{x} - \boldsymbol{x}^j\|$
        If class($\boldsymbol{x}$)$\neq$class($\boldsymbol{x}'$) add $\boldsymbol{x}$ to $\mathcal{Z}$
Until $\mathcal{Z}$ does not change

Figure 8.6: Condensed nearest neighbor algorithm.

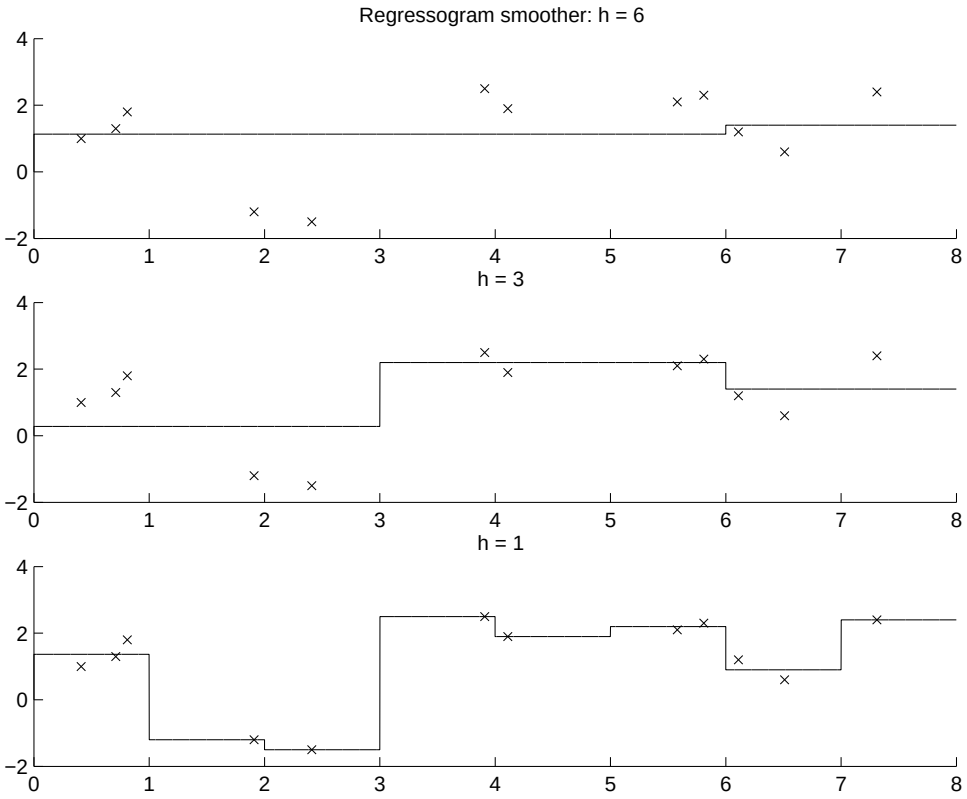


Figure 8.7: Regressograms for various bin lengths. '×' denote data points. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

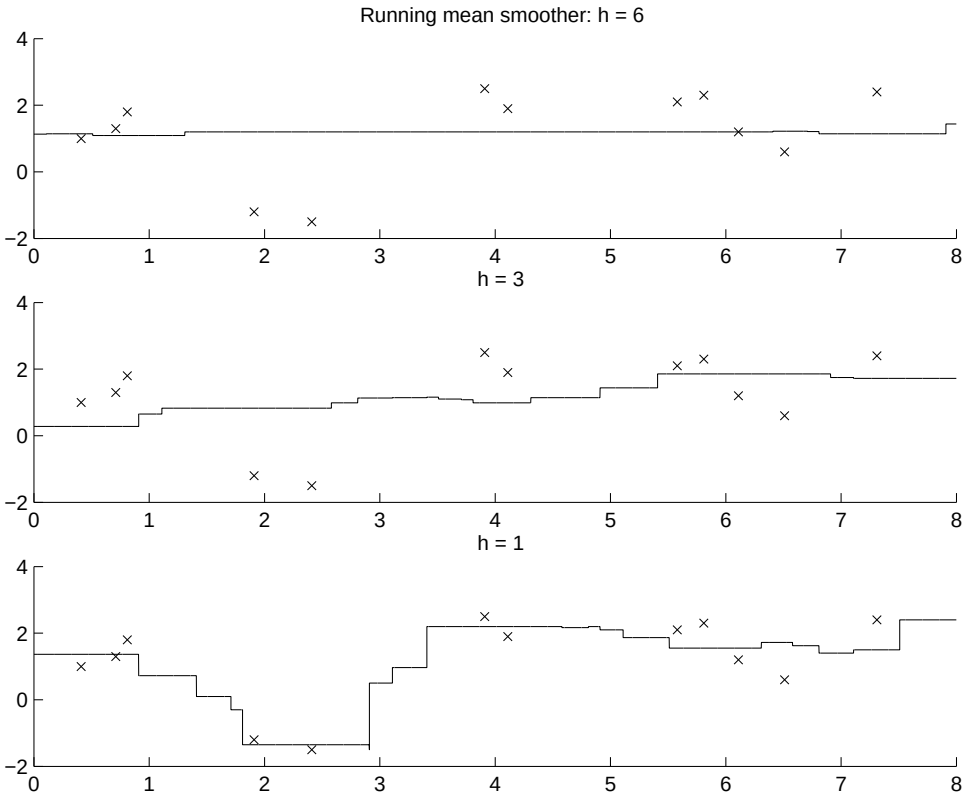


Figure 8.8: Running mean smooth for various bin lengths. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

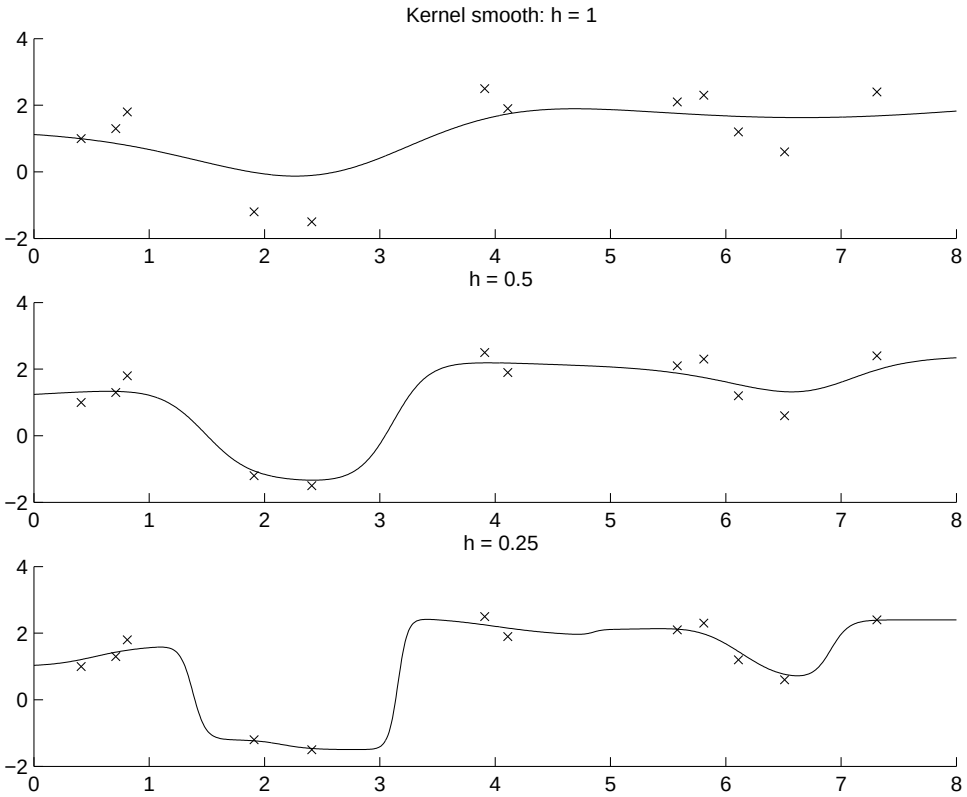


Figure 8.9: Kernel smooth for various bin lengths. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

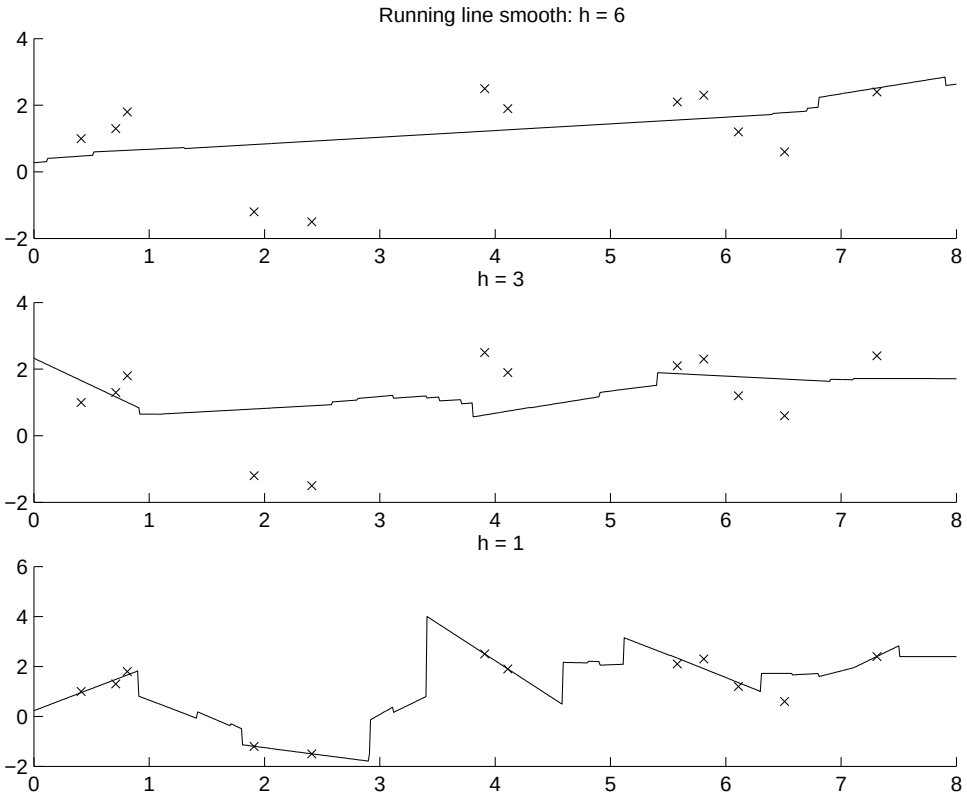


Figure 8.10: Running line smooth for various bin lengths. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

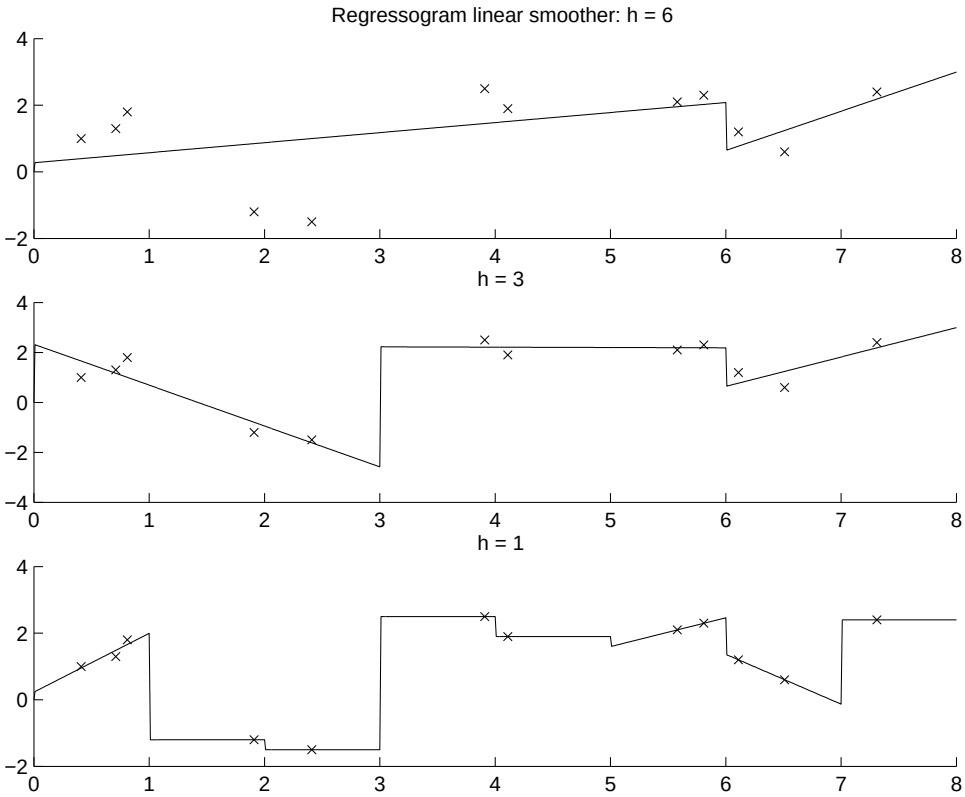


Figure 8.11: Regressograms with linear fits in bins for various bin lengths. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Chapter 9:

Decision Trees

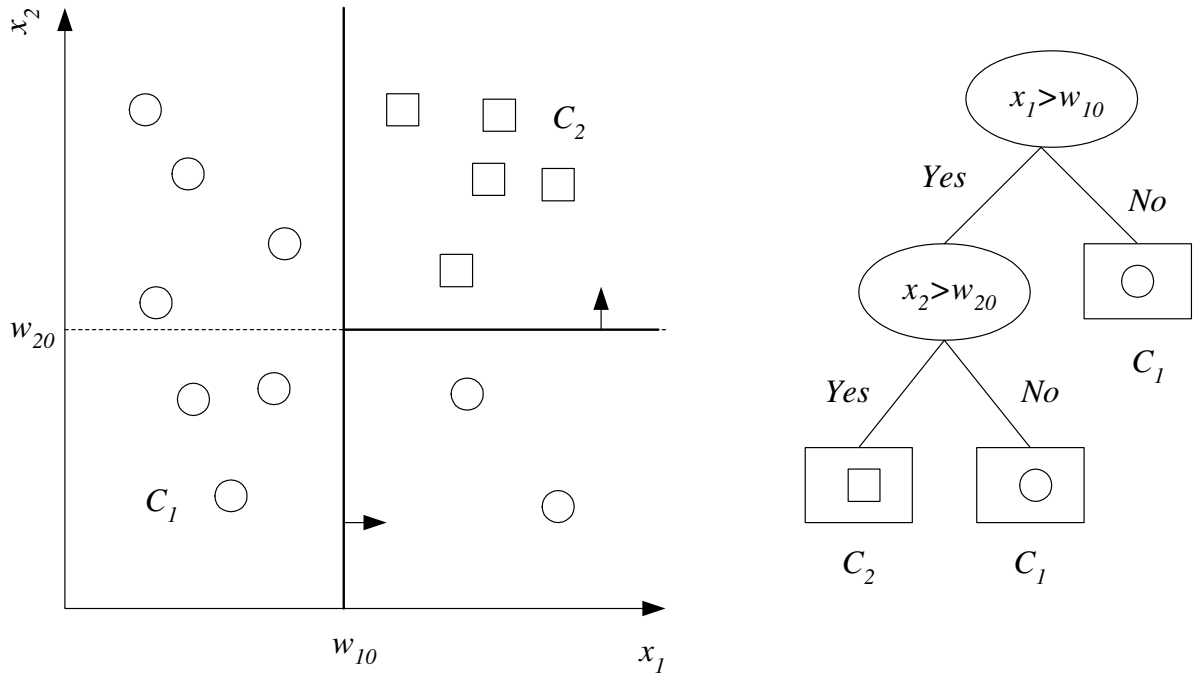


Figure 9.1: Example of a dataset and the corresponding decision tree. Oval nodes are the decision nodes and rectangles are leaf nodes. The univariate decision node splits along one axis, and successive splits are orthogonal to each other. After the first split, $\{\boldsymbol{x}|x_1 < w_{10}\}$ is pure and is not split further. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

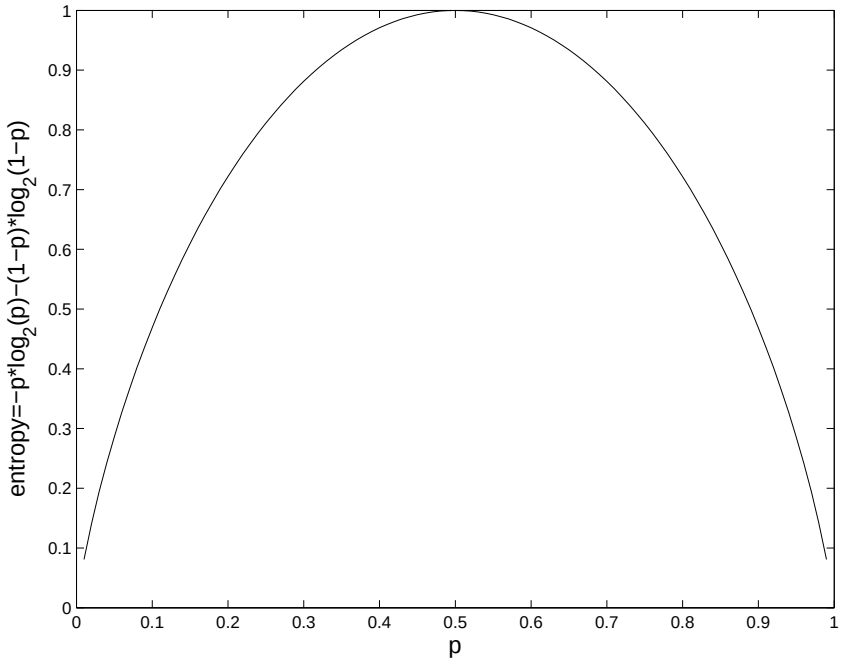


Figure 9.2: Entropy function for a two-class problem. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

```
GenerateTree(X)
    If NodeEntropy(X)< θ_I /* eq. 9.3
       Create leaf labelled by majority class in X
       Return
    i ← SplitAttribute(X)
    For each branch of x_i
       Find X_i falling in branch
       GenerateTree(X_i)
SplitAttribute(X)
    MinEnt← MAX
    For all attributes i = 1,...,d
        If x_i is discrete with n values
           Split X into X_1,...,X_n by x_i
           e ← SplitEntropy(X_1,...,X_n) /* eq. 9.8 */
           If e<MinEnt MinEnt ← e; bestf ← i
        Else /* x_i is numeric */
           For all possible splits
                Split X into X_1, X_2 on x_i
                e←SplitEntropy(X_1, X_2)
                If e<MinEnt MinEnt ← e; bestf ← i
    Return bestf
```

Figure 9.3: Classification tree construction. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

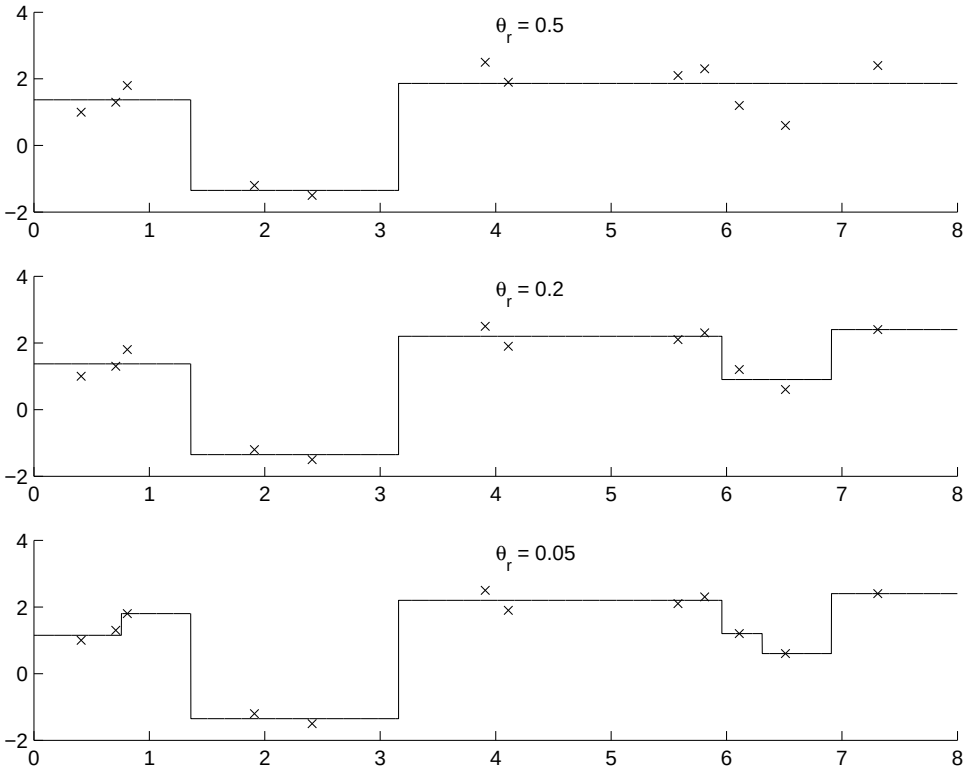


Figure 9.4: Regression tree smooths for various values of θ_r. The corresponding trees are given in figure 9.5. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

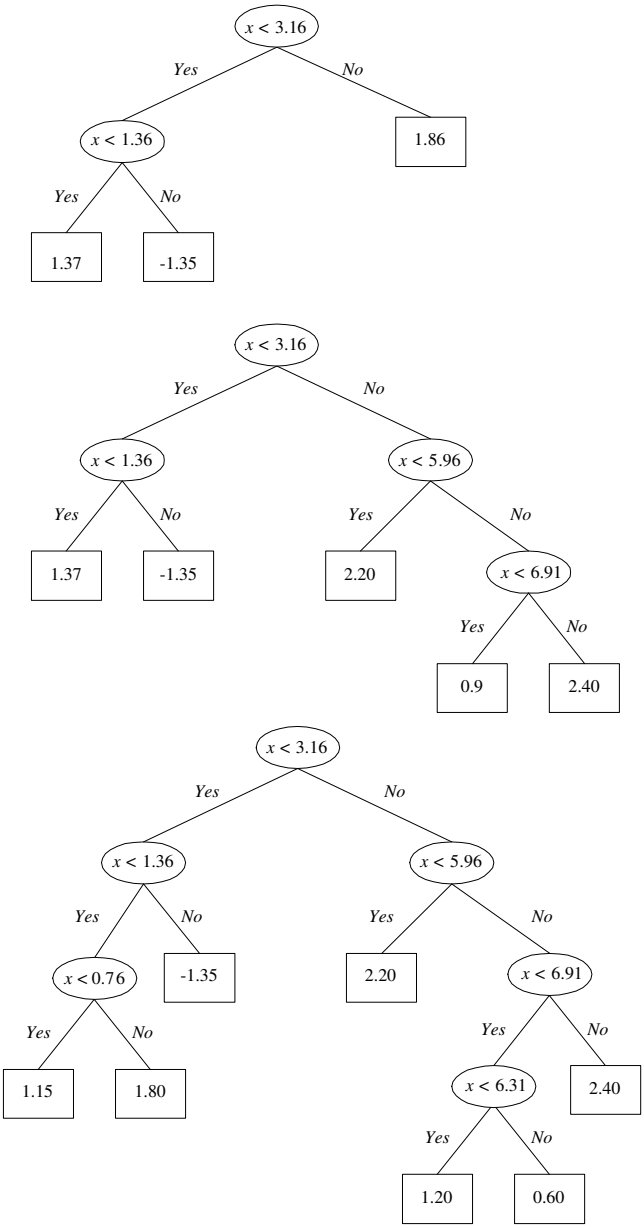


Figure 9.5: Regression trees implementing the smooths of figure 9.4 for various values of θ_r. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

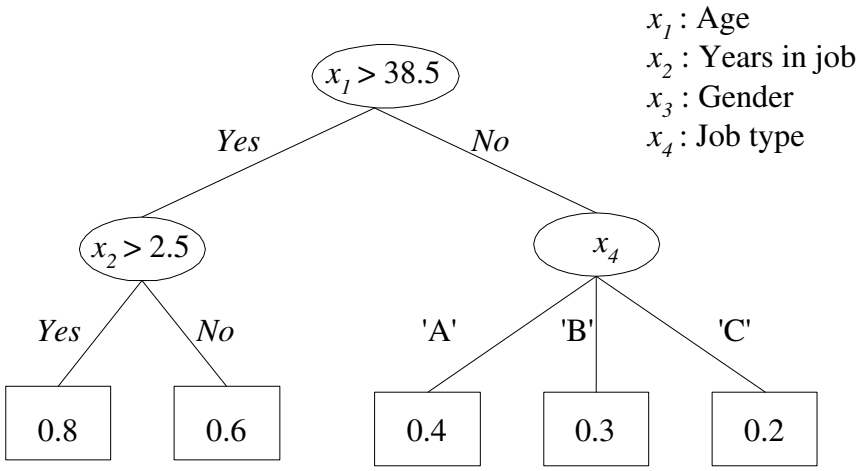


Figure 9.6: Example of a (hypothetical) decision tree. Each path from the root to a leaf can be written down as a conjunctive rule, composed of conditions defined by the decision nodes on the path. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

```
Ripper(Pos,Neg,k)
  RuleSet ← LearnRuleSet(Pos,Neg)
  For k times
    RuleSet ← OptimizeRuleSet(RuleSet,Pos,Neg)
LearnRuleSet(Pos,Neg)
  RuleSet ← ∅
  DL ← DescLen(RuleSet,Pos,Neg)
  Repeat
    Rule ← LearnRule(Pos,Neg)
    Add Rule to RuleSet
    DL' ← DescLen(RuleSet,Pos,Neg)
    If DL'>DL+64
      PruneRuleSet(RuleSet,Pos,Neg)
      Return RuleSet
    If DL'<DL DL ← DL'
      Delete instances covered from Pos and Neg
  Until Pos = ∅
  Return RuleSet
```

Figure 9.7: Ripper algorithm for learning rules.

From: E. Alpaydın. 2004. Introduction to Machine Learning. ©*The MIT Press.*

```
PruneRuleSet(RuleSet,Pos,Neg)
   For each Rule ∈ RuleSet in reverse order
      DL ← DescLen(RuleSet,Pos,Neg)
      DL' ← DescLen(RuleSet-Rule,Pos,Neg)
      IF DL'¡DL Delete Rule from RuleSet
   Return RuleSet
OptimizeRuleSet(RuleSet,Pos,Neg)
   For each Rule ∈ RuleSet
        DL0 ← DescLen(RuleSet,Pos,Neg)
        DL1 ← DescLen(RuleSet-Rule+
         ReplaceRule(RuleSet,Pos,Neg),Pos,Neg)
        DL2 ← DescLen(RuleSet-Rule+
         ReviseRule(RuleSet,Rule,Pos,Neg),Pos,Neg)
        If DL1=min(DL0,DL1,DL2)
         Delete Rule from RuleSet and
            add ReplaceRule(RuleSet,Pos,Neg)
        Else If DL2=min(DL0,DL1,DL2)
         Delete Rule from RuleSet and
            add ReviseRule(RuleSet,Rule,Pos,Neg)
   Return RuleSet
```

Figure 9.7: Ripper algorithm for learning rules (cont'd). *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

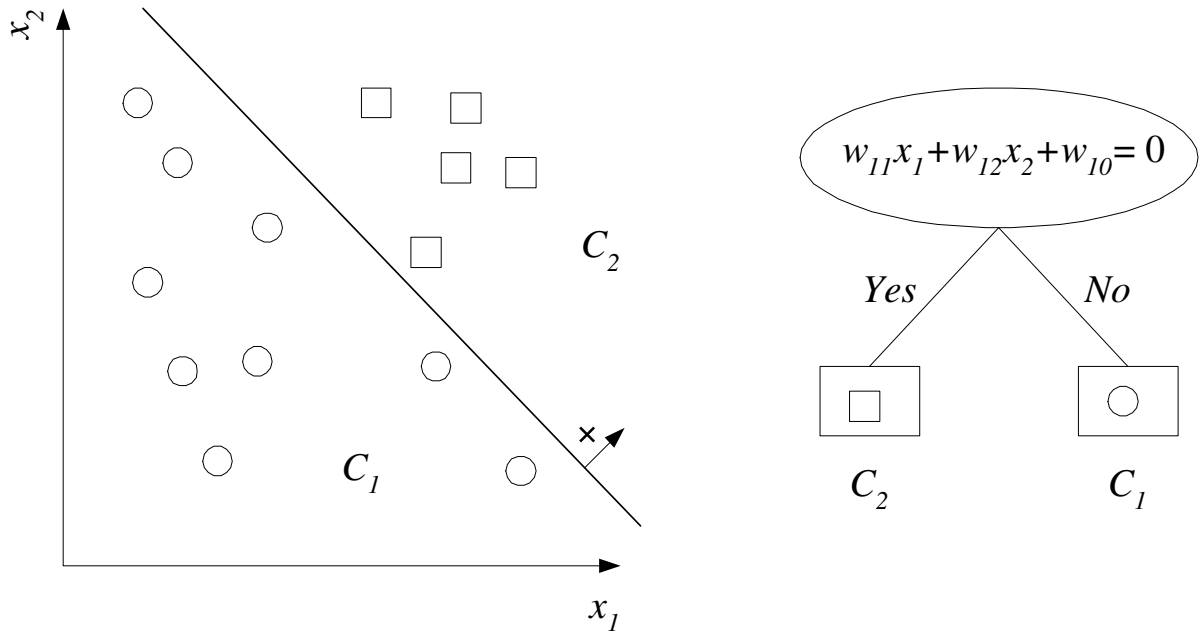


Figure 9.8: Example of a linear multivariate decision tree. The linear multivariate node can place an arbitrary hyperplane thus is more general, whereas the univariate node is restricted to axis-aligned splits. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Chapter 10:

Linear Discrimination

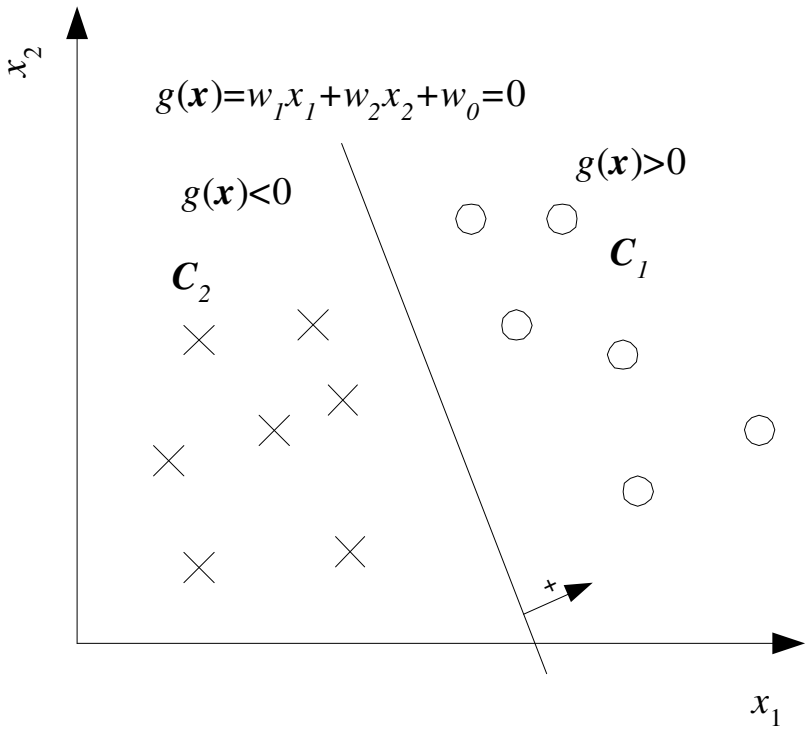


Figure 10.1: In the two-dimensional case, the linear discriminant is a line that separates the examples from two classes. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

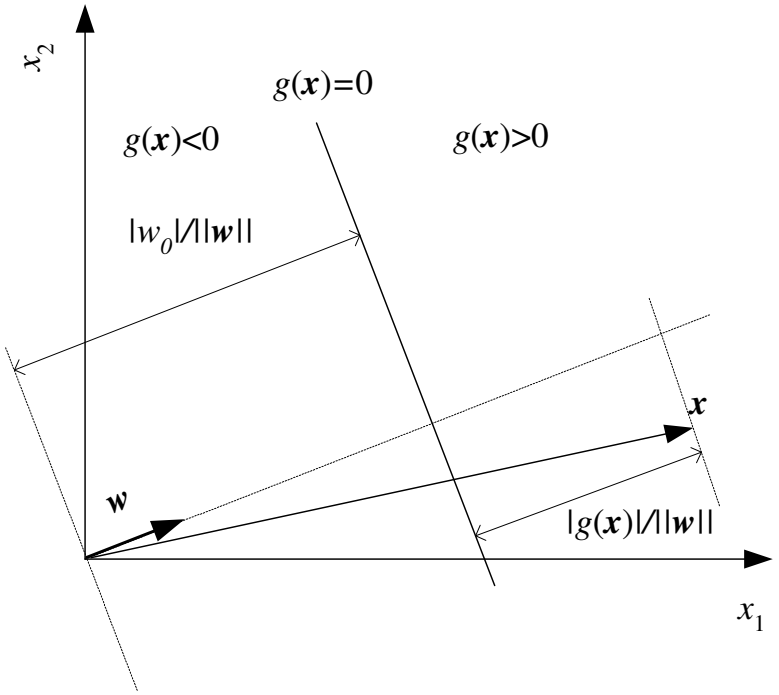


Figure 10.2: The geometric interpretation of the linear discriminant. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

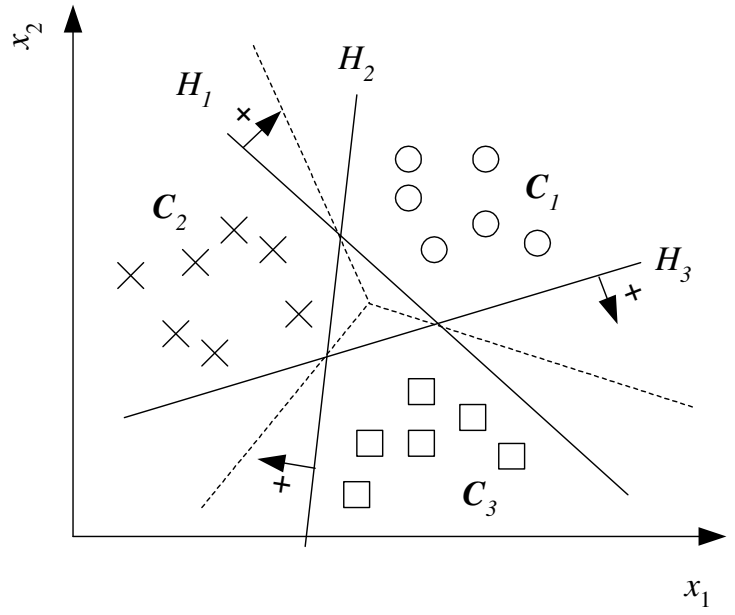


Figure 10.3: In linear classification, each hyperplane H_i separates the examples of $\mathcal{C}_i$ from the examples of all other classes. Thus for it to work, the classes should be linearly separable. Dotted lines are the induced boundaries of the linear classifier. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

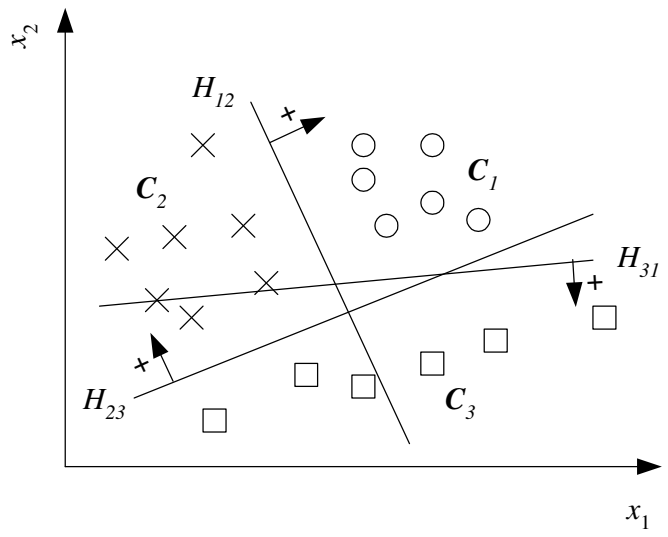


Figure 10.4: In pairwise linear separation, there is a separate hyperplane for each pair of classes. For an input to be assigned to $\mathcal{C}_1$, it should be on the positive side of H_{12} and H_{13} (which is the negative side of H_{31}); we do not care for the value of H_{23}. In this case, $\mathcal{C}_1$ is not linearly separable from other classes but is pairwise linearly separable. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

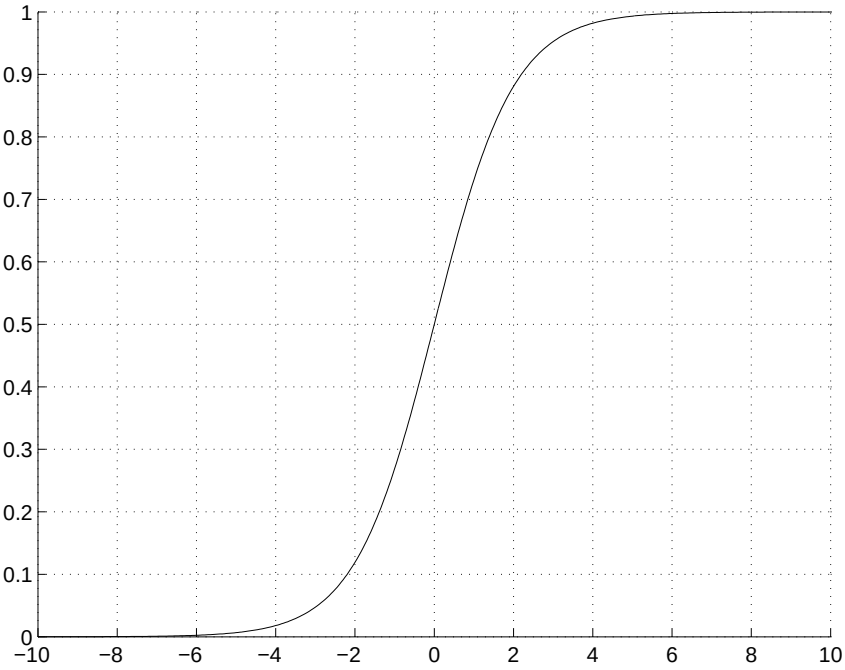


Figure 10.5: The logistic, or sigmoid, function.

From: E. Alpaydın. 2004. Introduction to Machine Learning. ©*The MIT Press.*

For $j = 0, \ldots, d$
    $w_j \leftarrow$rand(-0.01,0.01)
Repeat
    For $j = 0, \ldots, d$
        $\Delta w_j \leftarrow 0$
    For $t = 1, \ldots, N$
        $o \leftarrow 0$
        For $j = 0, \ldots, d$
            $o \leftarrow o + w_j x_j^t$
        $y \leftarrow \text{sigmoid}(o)$
        $\Delta w_j \leftarrow \Delta w_j + (r^t - y) x_j^t$
    For $j = 0, \ldots, d$
        $w_j \leftarrow w_j + \eta \Delta w_j$
Until convergence

Figure 10.6: Logistic discrimination algorithm implementing gradient-descent for the single output case with two classes. For w_0, we assume that there is an extra input x_0, which is always $+1$: $x_0^t \equiv +1, \forall t$. *From: E. Alpaydın. 2004.* Introduction to Machine Learning.

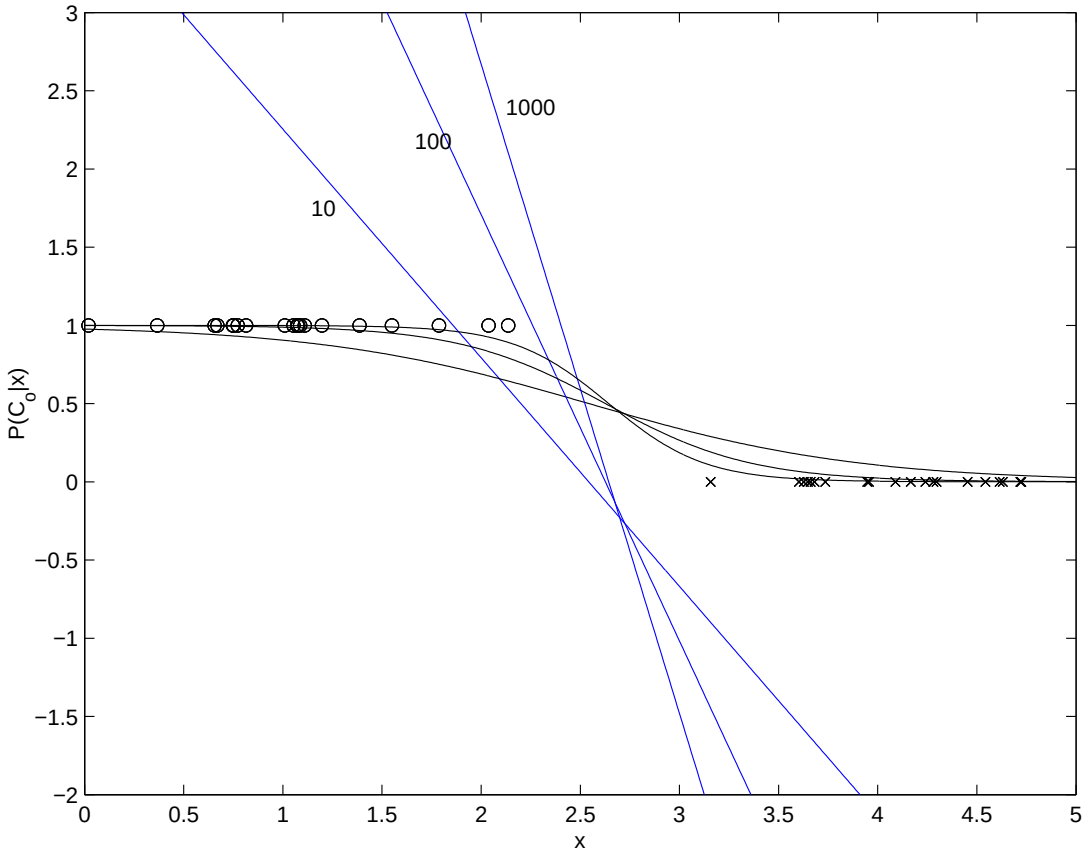


Figure 10.7: For a univariate two-class problem (shown with '$\circ$' and '$\times$'), the evolution of the line $wx + w_0$ and the sigmoid output after 10, 100, and 1,000 iterations over the sample. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

For $i = 1, \ldots, K$, For $j = 0, \ldots, d$, $w_{ij} \leftarrow$ rand$(-0.01, 0.01)$
Repeat
    For $i = 1, \ldots, K$, For $j = 0, \ldots, d$, $\Delta w_{ij} \leftarrow 0$
    For $t = 1, \ldots, N$
        For $i = 1, \ldots, K$
            $o_i \leftarrow 0$
            For $j = 0, \ldots, d$
                $o_i \leftarrow o_i + w_{ij} x_j^t$
        For $i = 1, \ldots, K$
            $y_i \leftarrow \exp(o_i) / \sum_k \exp(o_k)$
        For $i = 1, \ldots, K$
            For $j = 0, \ldots, d$
                $\Delta w_{ij} \leftarrow \Delta w_{ij} + (r_i^t - y_i) x_j^t$
    For $i = 1, \ldots, K$
        For $j = 0, \ldots, d$
            $w_{ij} \leftarrow w_{ij} + \eta \Delta w_{ij}$
Until convergence

Figure 10.8: Logistic discrimination algorithm implementing gradient-descent for the case with $K > 2$ classes. For generality, we take $x_0^t \equiv 1, \forall t$.

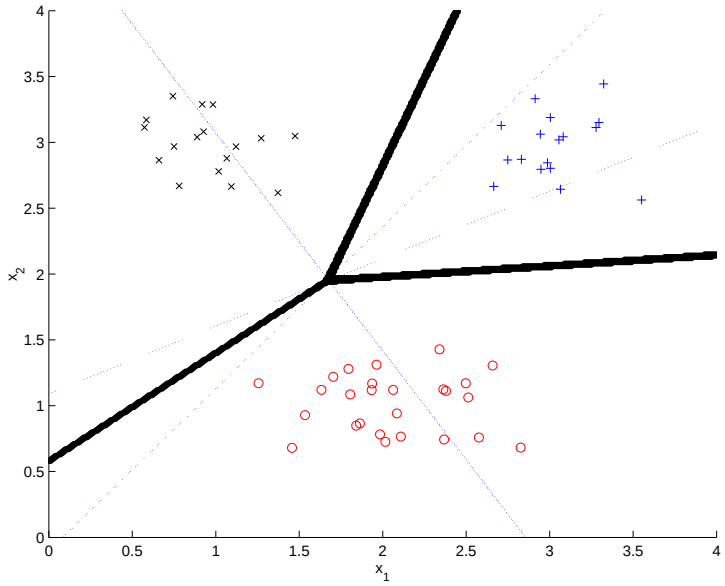


Figure 10.9: For a two-dimensional problem with three classes, the solution found by logistic discrimination. Thin lines are where $g_i(\boldsymbol{x}) = 0$, and the thick line is the boundary induced by the linear classifier choosing the maximum. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

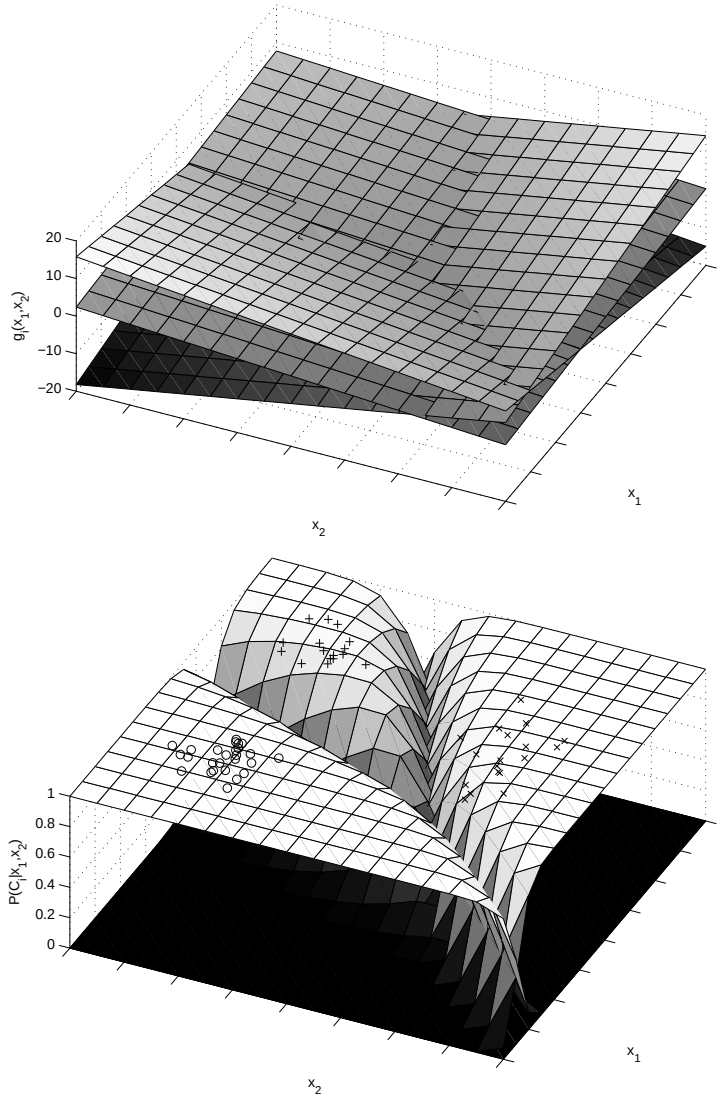


Figure 10.10: For the same example in figure 10.9, the linear discriminants (top), and the posterior probabilities after the softmax (bottom). *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

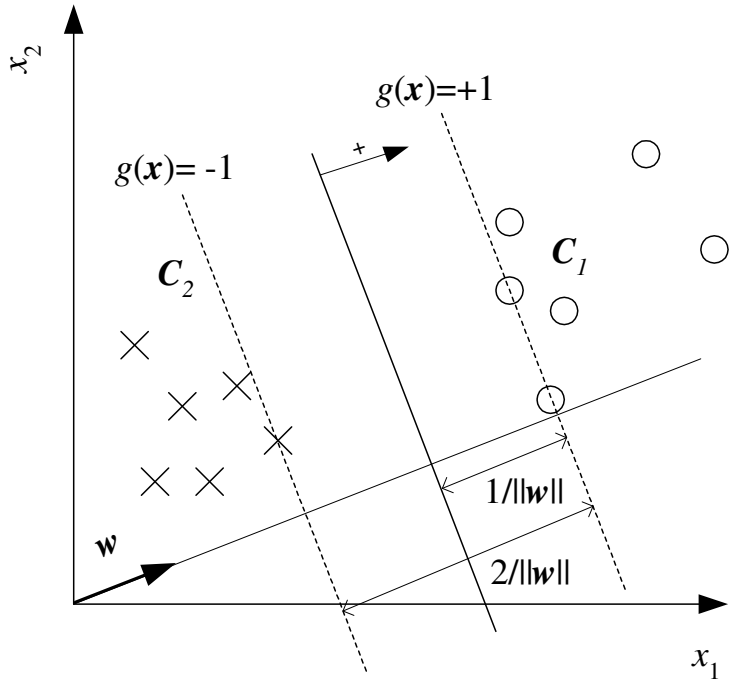


Figure 10.11: On both sides of the optimal separating hyperplance, the instances are at least $1/\|\boldsymbol{w}\|$ away and the total margin is $2/\|\boldsymbol{w}\|$. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

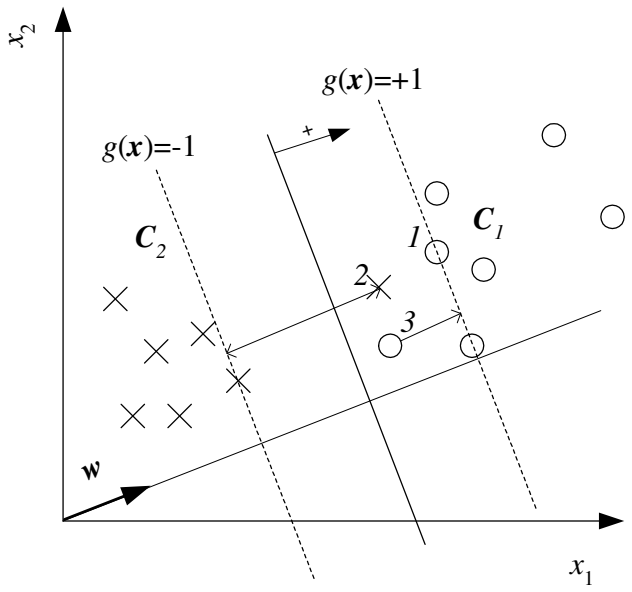


Figure 10.12: In classifying an instance, there are three possible cases: In (1), $\xi = 0$; it is on the right side and sufficiently away. In (2), $\xi = 1 + g(\boldsymbol{x}) > 1$; it is on the wrong side. In (3), $\xi = 1 - g(\boldsymbol{x}), 0 < \xi < 1$; it is on the right side but is in the margin and not sufficiently away. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

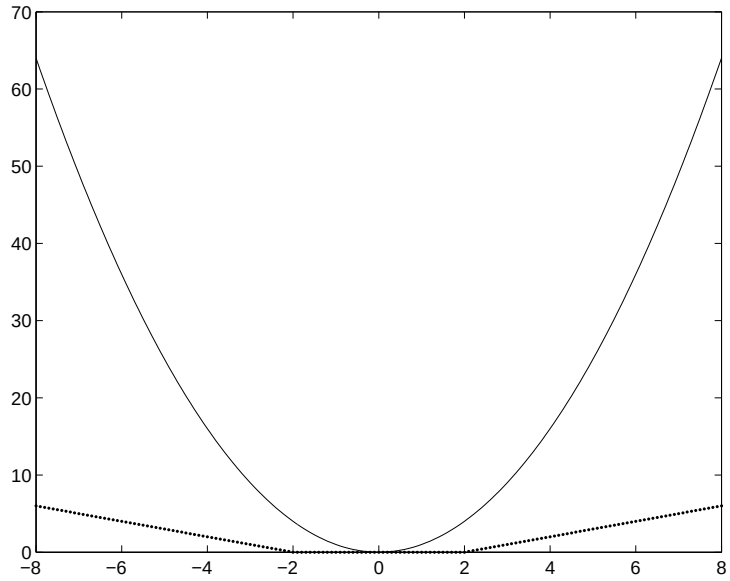


Figure 10.13: Quadratic and ϵ-sensitive error functions. We see that ϵ-sensitive error function is not affected by small errors and also is affected less by large errors and thus is more robust to outliers. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Chapter 11:

Multilayer Perceptrons

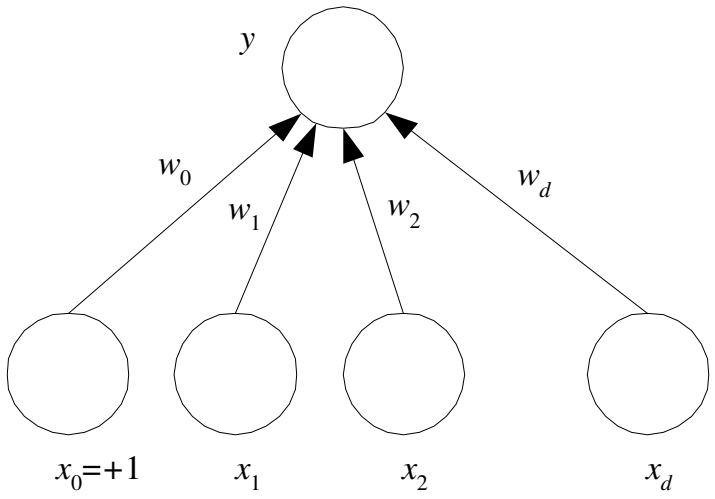


Figure 11.1: Simple perceptron. $x_j, j = 1, \ldots, d$ are the input units. x_0 is the bias unit that always has the value 1. y is the output unit. w_j is the weight of the directed connection from input x_j to the output. *From: E. Alpaydın. 2004.* Introduction to Machine Learning.

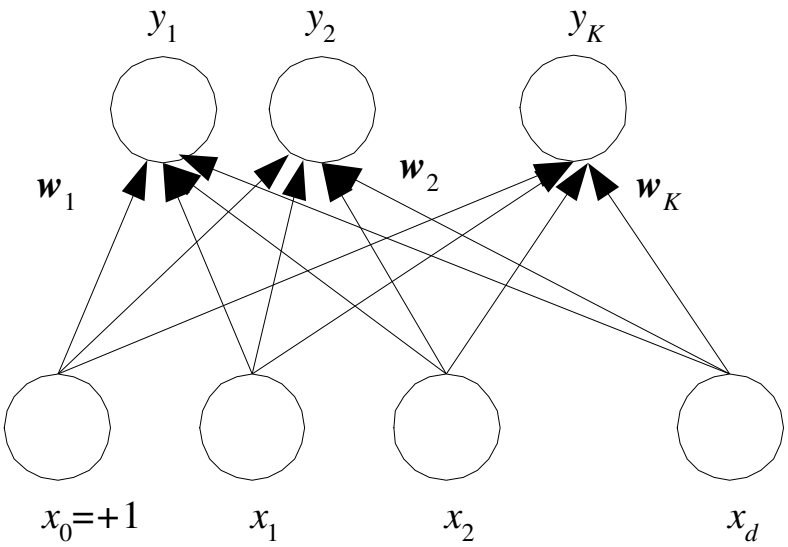


Figure 11.2: K parallel perceptrons. $x_j, j = 0, \ldots, d$ are the inputs and $y_i, i = 1, \ldots, K$ are the outputs. w_{ij} is the weight of the connection from input x_j to output y_i. Each output is a weighted sum of the inputs. When used for K-class classification problem, there is a postprocessing to choose the maximum, or softmax if we need the posterior probabilities. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

For $i = 1, \ldots, K$
    For $j = 0, \ldots, d$
        $w_{ij} \leftarrow \text{rand}(-0.01, 0.01)$
Repeat
    For all $(\boldsymbol{x}^t, r^t) \in \mathcal{X}$ in random order
        For $i = 1, \ldots, K$
            $o_i \leftarrow 0$
            For $j = 0, \ldots, d$
                $o_i \leftarrow o_i + w_{ij} x_j^t$
        For $i = 1, \ldots, K$
            $y_i \leftarrow \exp(o_i) / \sum_k \exp(o_k)$
        For $i = 1, \ldots, K$
            For $j = 0, \ldots, d$
                $w_{ij} \leftarrow w_{ij} + \eta(r_i^t - y_i) x_j^t$
Until convergence

Figure 11.3: Percepton training algorithm implementing stochastic online gradient-descent for the case with $K > 2$ classes. This is the online version of the algorithm given in figure 10.8. *From: E. Alpaydın. 2004.* Introduction to Machine Learning.

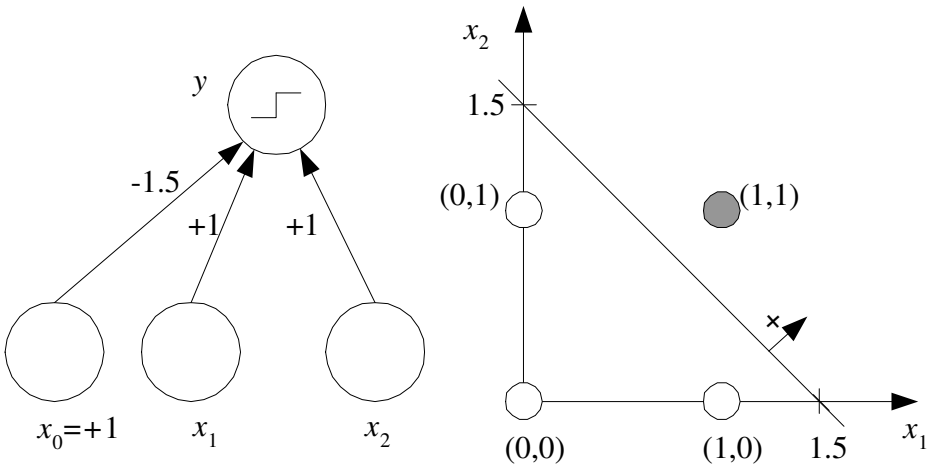


Figure 11.4: The perceptron that implements AND and its geometric interpretation. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

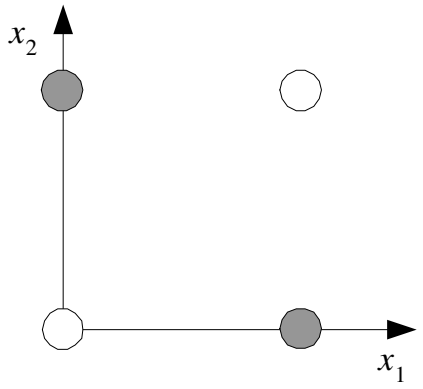


Figure 11.5: XOR problem is not linearly separable. We cannot draw a line where the empty circles are on one side and the filled circles on the other side. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

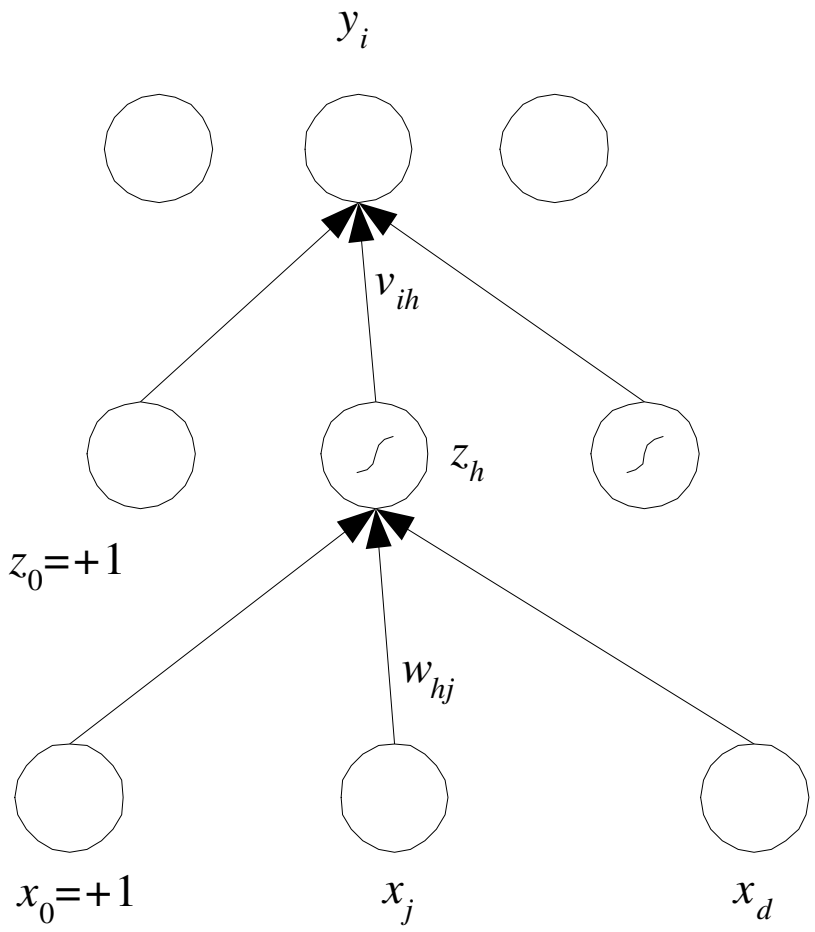


Figure 11.6: The structure of a multilayer perceptron. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

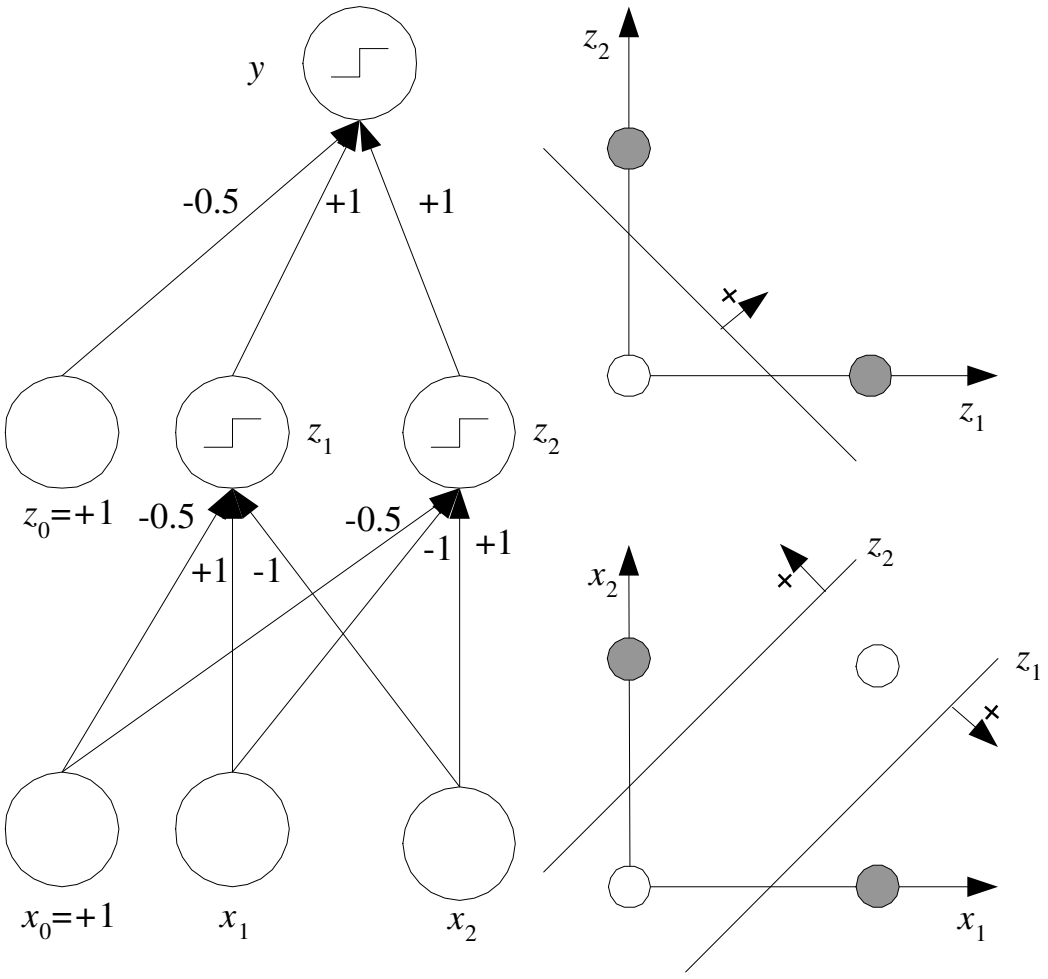


Figure 11.7: The multilayer perceptron that solves the XOR problem. The hidden units and the output have the threshold activation function with threshold at 0. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

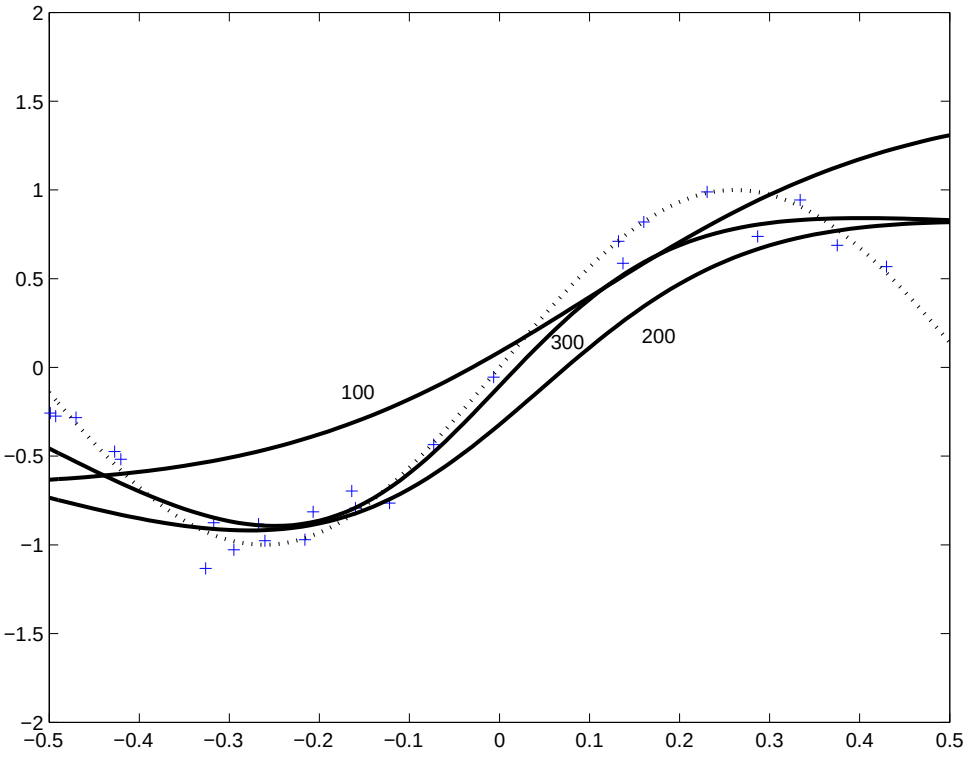


Figure 11.8: Sample training data shown as '+', where $x^t \sim U(-0.5, 0.5)$, and $y^t = f(x^t) + \mathcal{N}(0, 0.1)$. $f(x) = \sin(6x)$ is shown by a dashed line. The evolution of the fit of an MLP with two hidden units after 100, 200, and 300 epochs is drawn. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

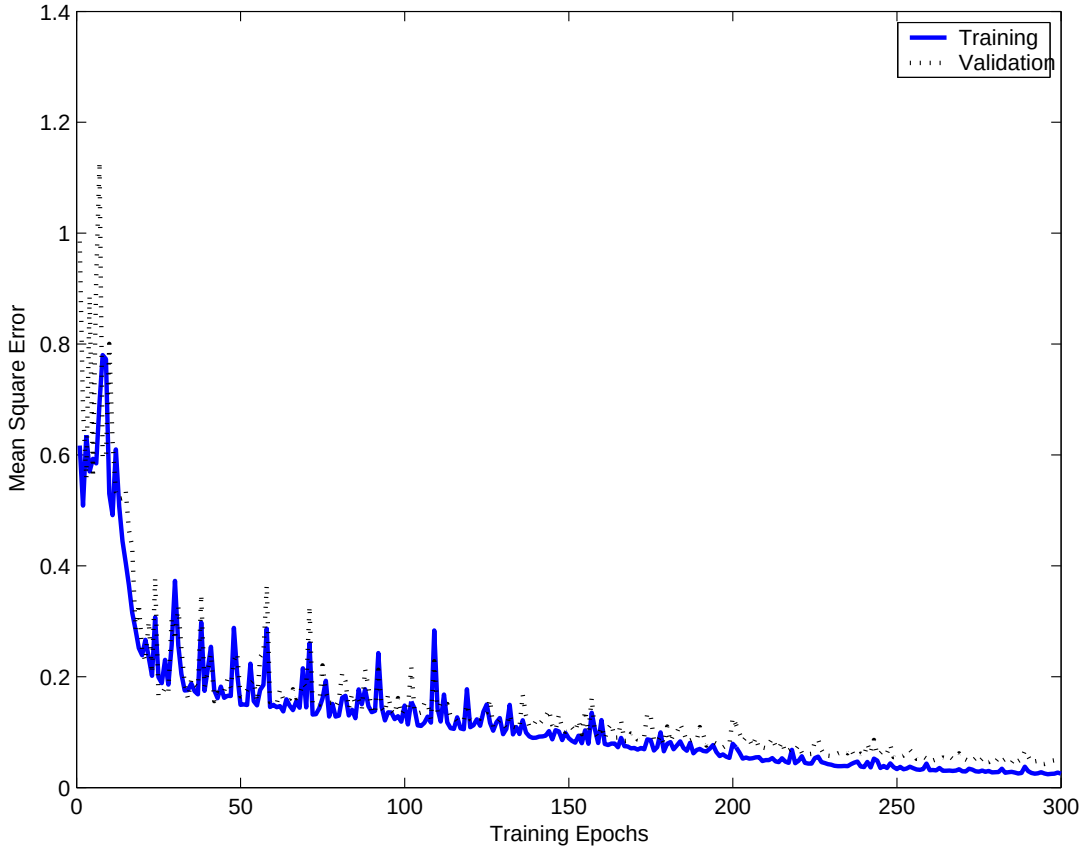


Figure 11.9: The mean square error on training and validation sets as a function of training epochs. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

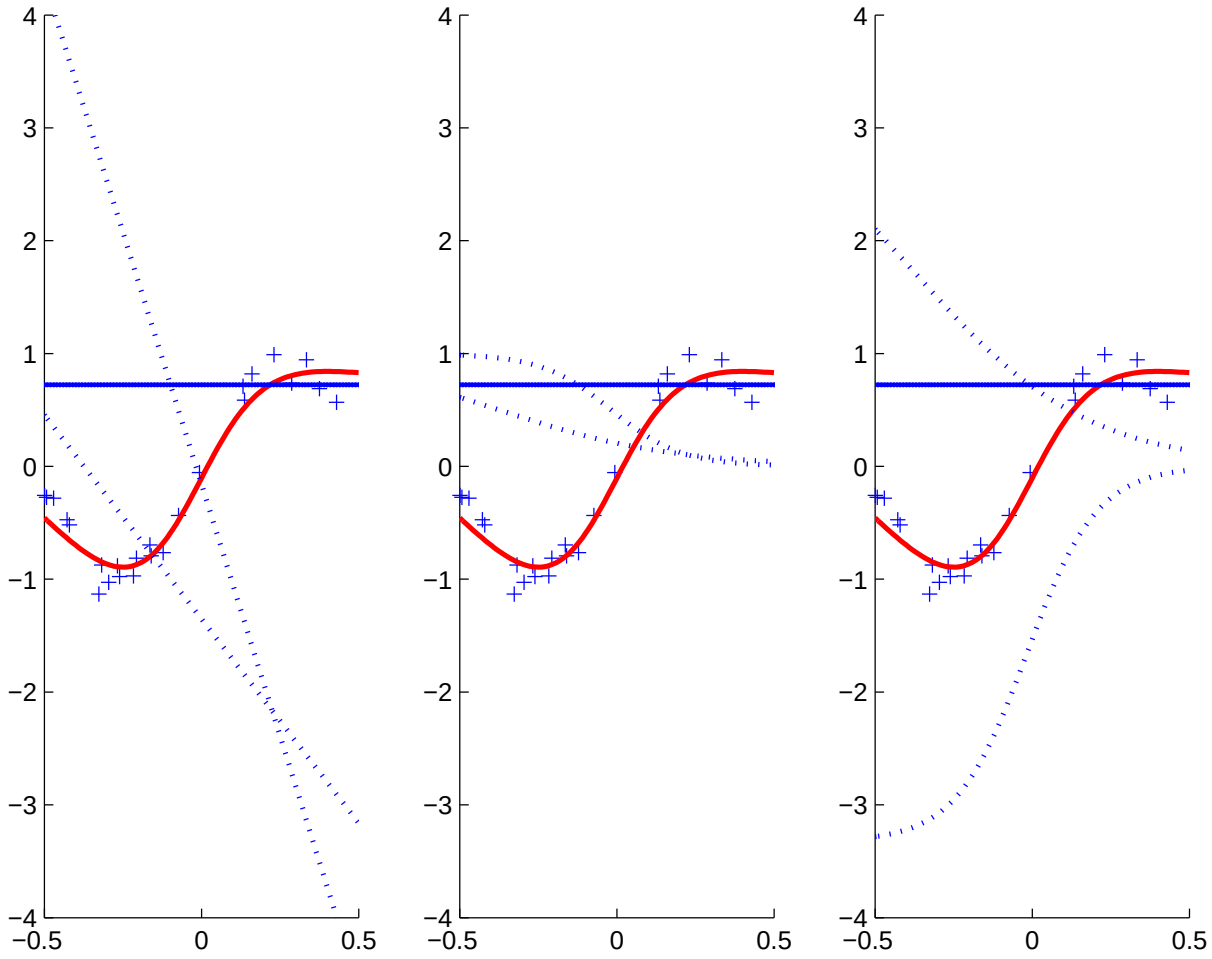


Figure 11.10: (a) The hyperplanes of the hidden unit weights on the first layer, (b) hidden unit outputs, and (c) hidden unit outputs multiplied by the weights on the second layer. Two sigmoid hidden units slightly displaced, one multiplied by a negative weight, when added, implement a bump. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Initialize all v_{ih} and w_{hj} to rand$(-0.01, 0.01)$
Repeat
    For all $(\boldsymbol{x}^t, r^t) \in \mathcal{X}$ in random order
        For $h = 1, \ldots, H$
            $z_h \leftarrow \text{sigmoid}(\boldsymbol{w}_h^T \boldsymbol{x}^t)$
        For $i = 1, \ldots, K$
            $y_i = \boldsymbol{v}_i^T \boldsymbol{z}$
        For $i = 1, \ldots, K$
            $\Delta \boldsymbol{v}_i = \eta(r_i^t - y_i^t)\boldsymbol{z}$
        For $h = 1, \ldots, H$
            $\Delta \boldsymbol{w}_h = \eta(\sum_i (r_i^t - y_i^t) v_{ih}) z_h (1 - z_h) \boldsymbol{x}^t$
        For $i = 1, \ldots, K$
            $\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \Delta \boldsymbol{v}_i$
        For $h = 1, \ldots, H$
            $\boldsymbol{w}_h \leftarrow \boldsymbol{w}_h + \Delta \boldsymbol{w}_h$
Until convergence

Figure 11.11: Backpropagation algorithm for training a multilayer perceptron for regression with K outputs. This code can easily be adapted for two-class classification (by setting a single sigmoid output) and to $K > 2$ classification (by using softmax outputs). *From: E. Alpaydın. 2004.* Introduction to Machine Learning.

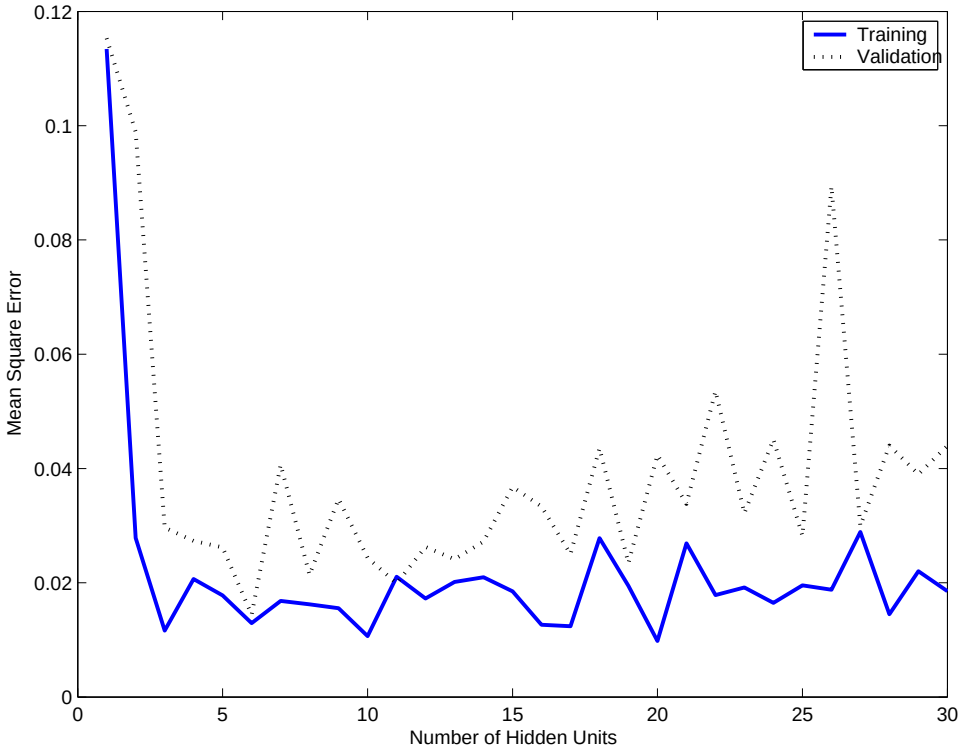


Figure 11.12: As complexity increases, training error is fixed but the validation error starts to increase and the network starts to overfit. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

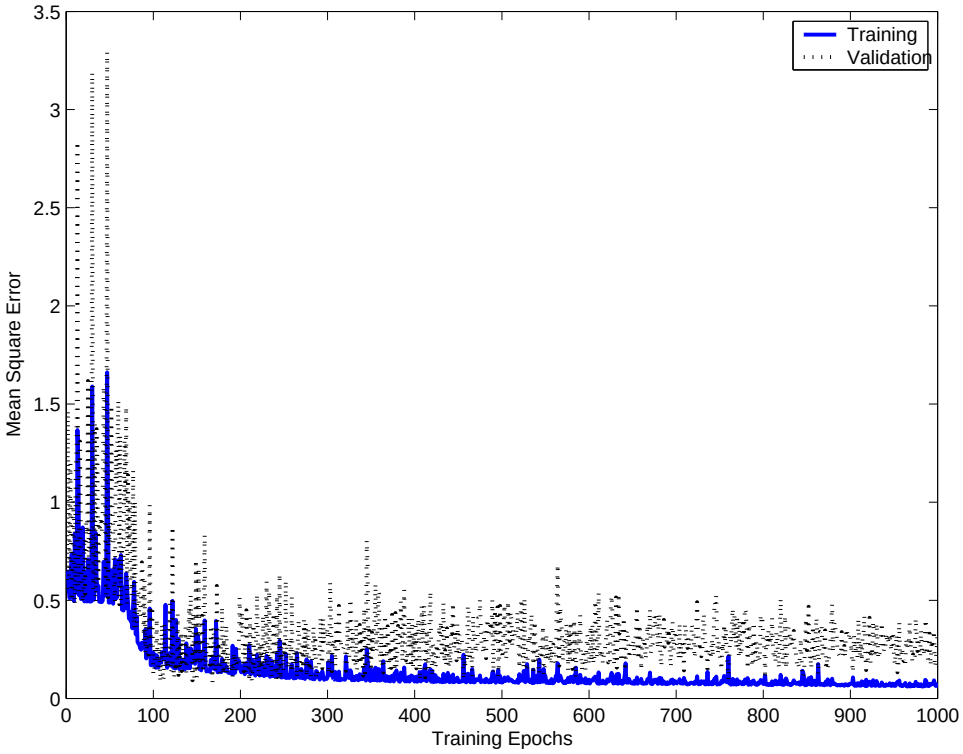


Figure 11.13: As training continues, the validation error starts to increase and the network starts to overfit. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

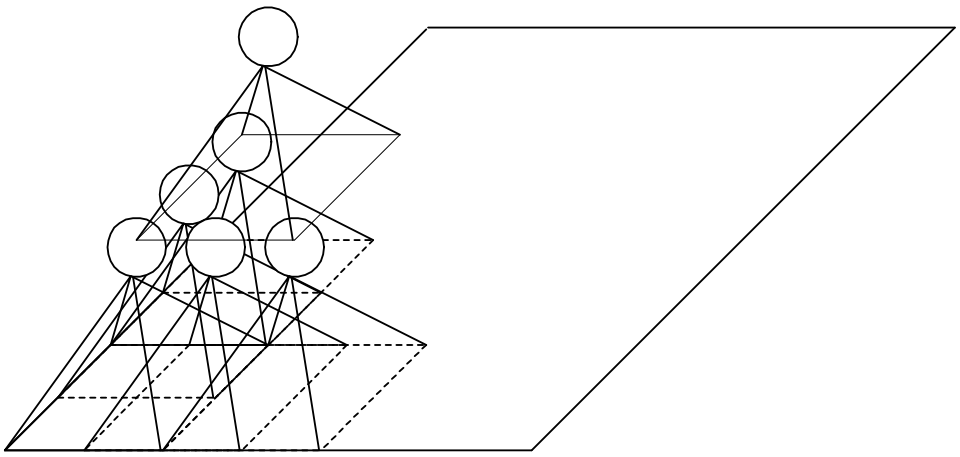


Figure 11.14: A structured MLP. Each unit is connected to a local group of units below it and checks for a particular feature—for example, edge, corner, and so forth—in vision. Only one hidden unit is shown for each region; typically there are many to check for different local features. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

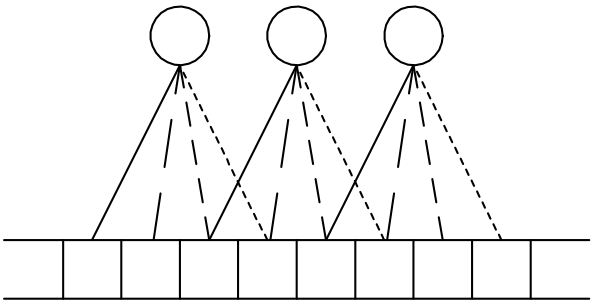


Figure 11.15: In weight sharing, different units have connections to different inputs but share the same weight value (denoted by line type). Only one set of units is shown; there should be multiple sets of units, each checking for different features. *From: E. Alpaydın. 2004.* Introduction to Machine Learning.

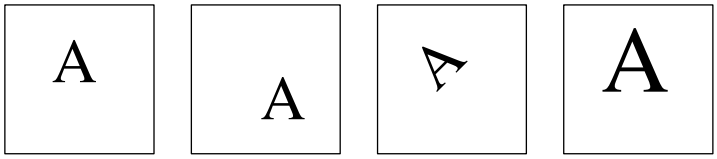
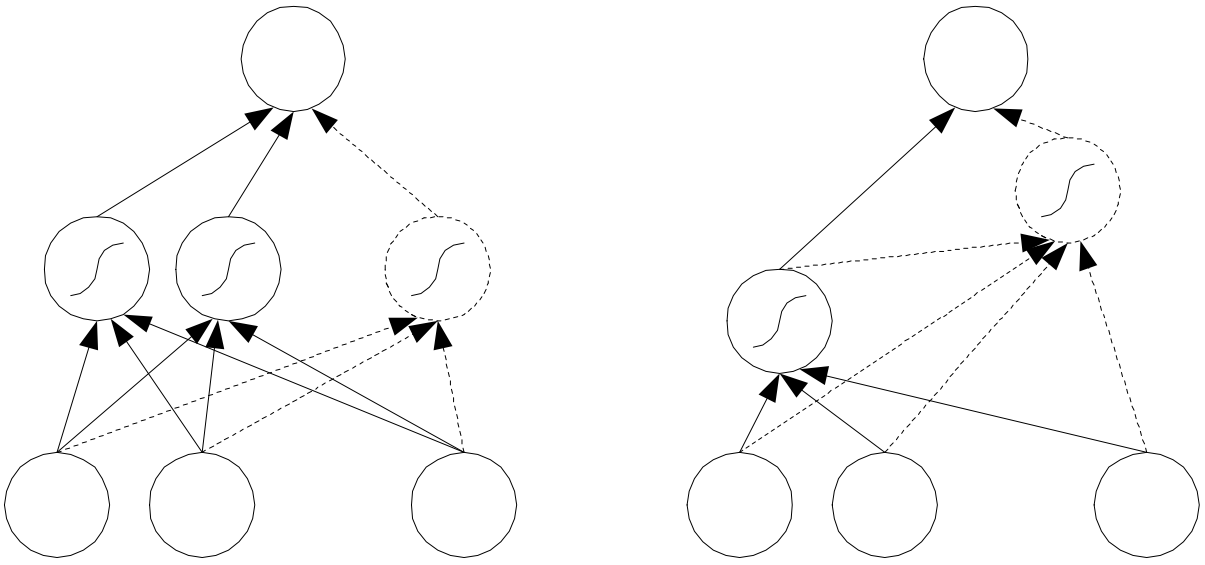


Figure 11.16: The identity of the object does not change when it is translated, rotated, or scaled. Note that this may not always be true, or may be true up to a point: 'b' and 'q' are rotated versions of each other. These are hints that can be incorporated into the learning process to make learning easier. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*



Dynamic Node Creation *Cascade Correlation*

Figure 11.17: Two examples of constructive algorithms: Dynamic node creation adds a unit to an existing layer. Cascade correlation adds each unit as new hidden layer connected to all the previous layers. Dashed lines denote the newly added unit/connections. Bias units/weights are omitted for clarity. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

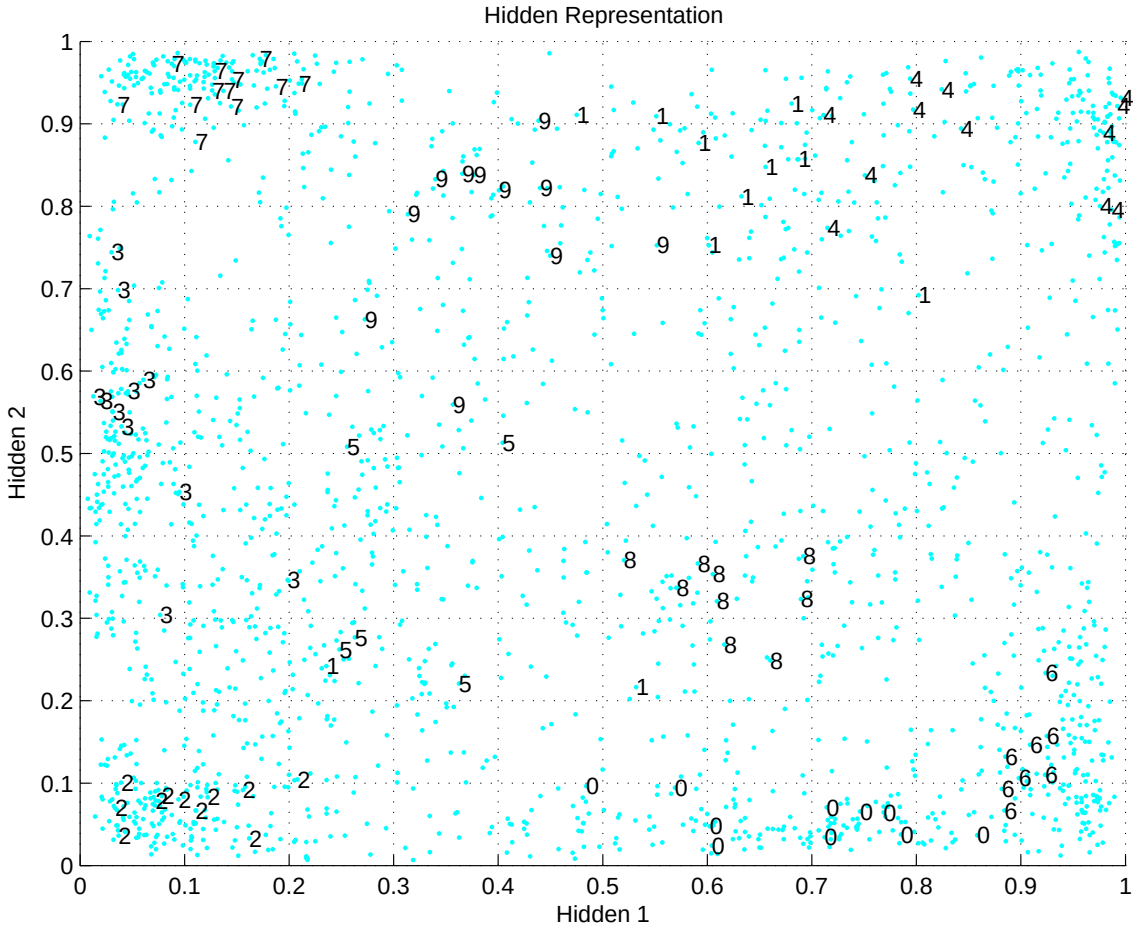


Figure 11.18: Optdigits data plotted in the space of the two hidden units of an MLP trained for classification. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

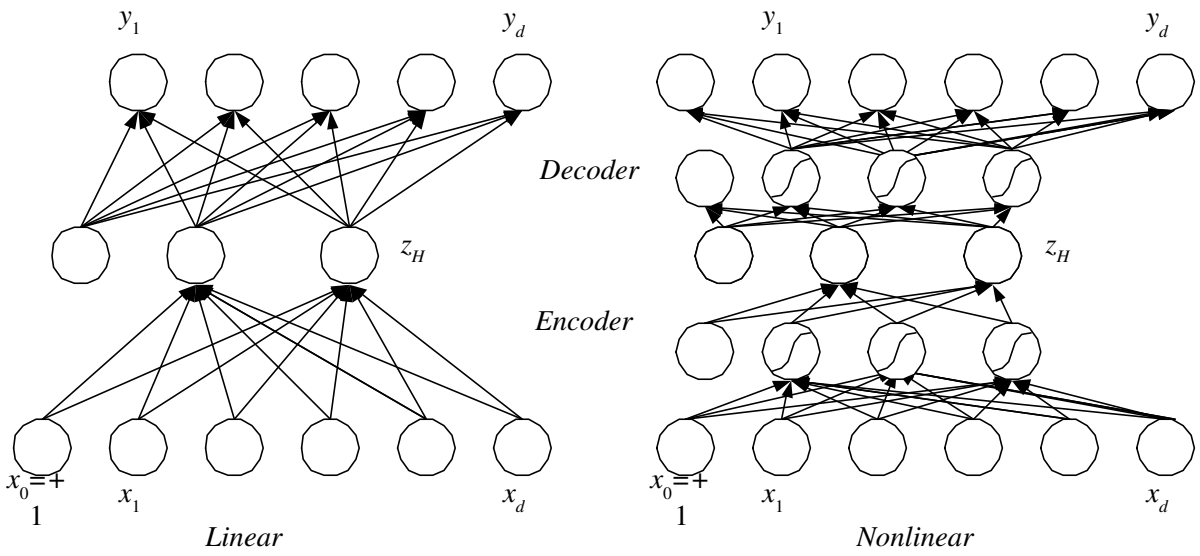


Figure 11.19: In the autoassociator, there are as many outputs as there are inputs and the desired outputs are the inputs. When the number of hidden units is less than the number of inputs, the MLP is trained to find the best coding of the inputs on the hidden units, performing dimensionality reduction. On the left, the first layer acts as an encoder and the second layer acts as the decoder. On the right, if the encoder and decoder are multilayer perceptrons with sigmoid hidden units, the network performs nonlinear dimensionality reduction. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

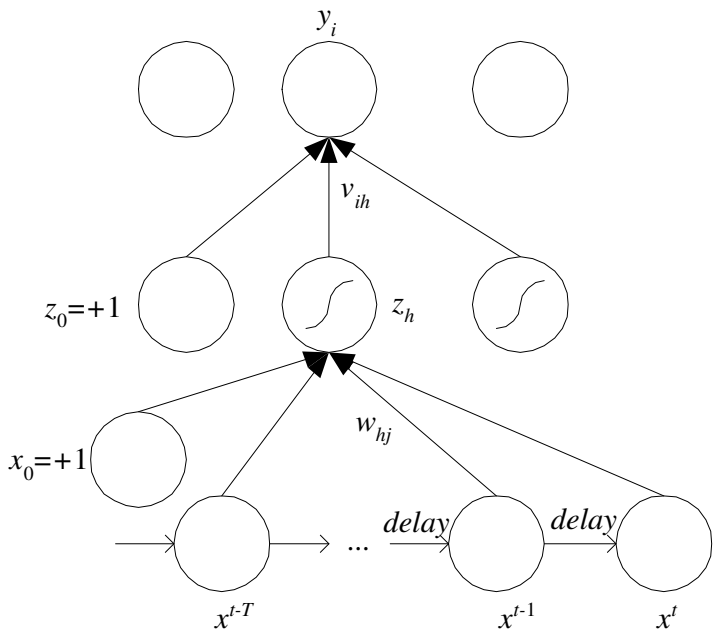


Figure 11.20: A time delay neural network. Inputs in a time window of length T are delayed in time until we can feed all T inputs as the input vector to the MLP. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

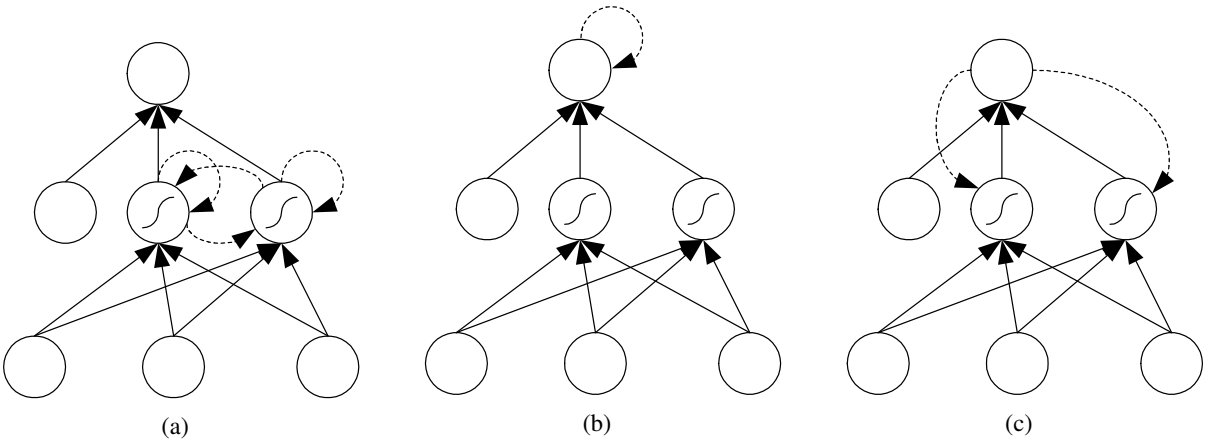


Figure 11.21: Examples of MLP with partial recurrency. Recurrent connections are shown with dashed lines: (a) self-connections in the hidden layer, (b) self-connections in the output layer, and (c) connections from the output to the hidden layer. Combinations of these are also possible. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

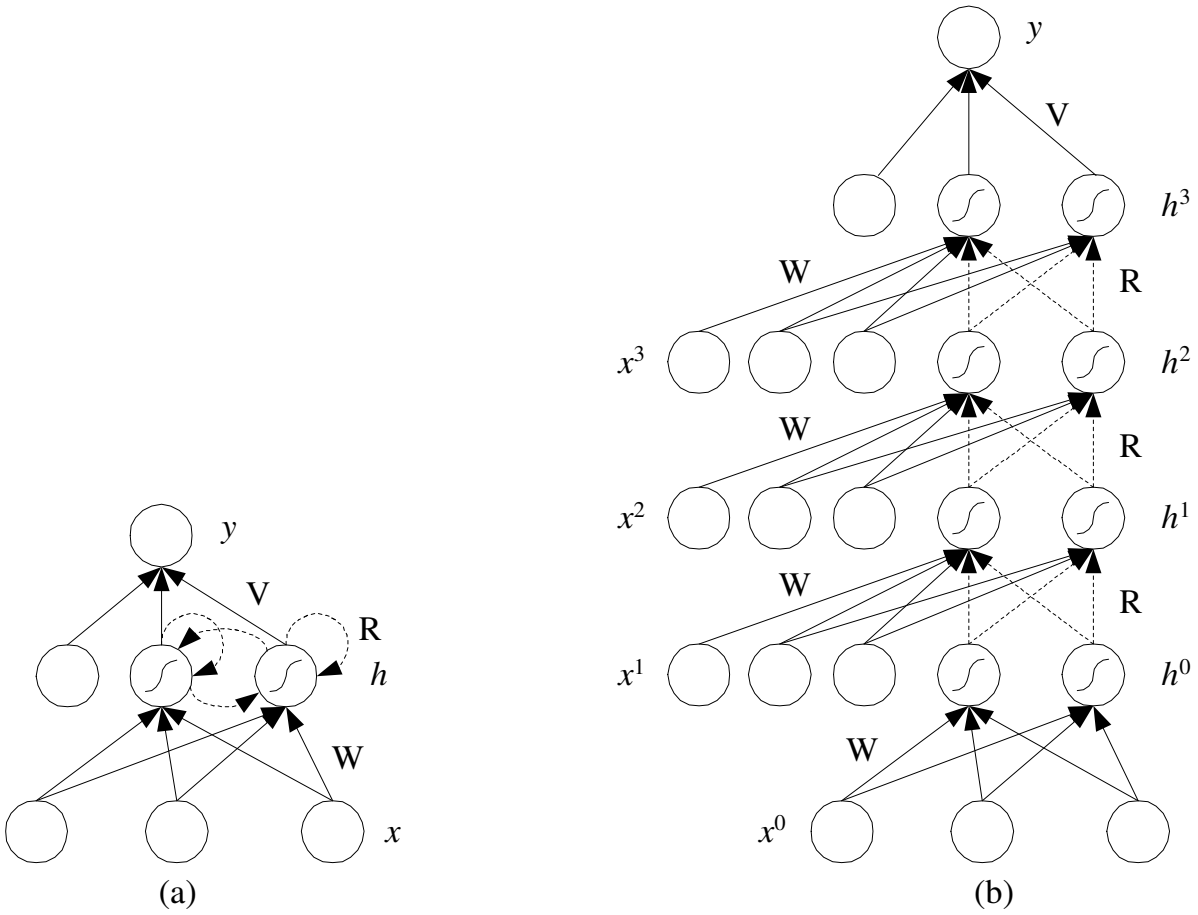


Figure 11.22: Backpropagation through time: (a) recurrent network and (b) its equivalent unfolded network that behaves identically in four steps. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Chapter 12:

Local Models

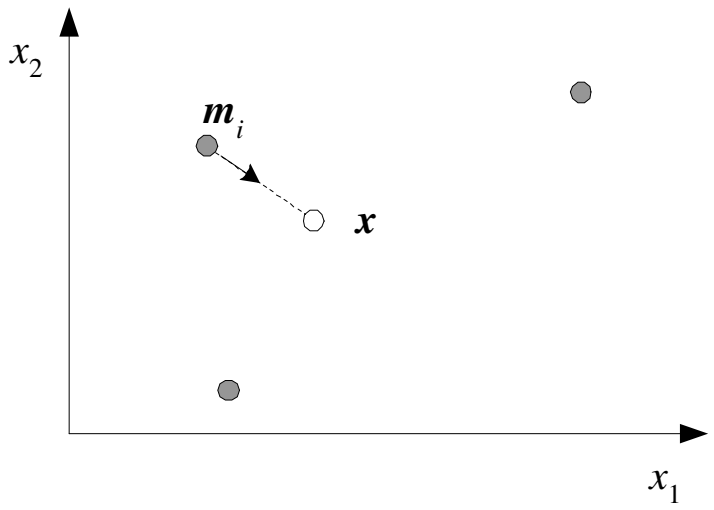


Figure 12.1: Shaded circles are the centers and the empty circle is the input instance. The online version of k-means moves the closest center along the direction of $(\boldsymbol{x} - \boldsymbol{m}_i)$ by a factor specified by η. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Initialize $\boldsymbol{m}_i, i = 1, \ldots, k$, for example, to k random $\boldsymbol{x}^t$
Repeat
    For all $\boldsymbol{x}^t \in \mathcal{X}$ in random order
        $i \leftarrow \arg\min_j \|\boldsymbol{x}^t - \boldsymbol{m}_j\|$
        $\boldsymbol{m}_i \leftarrow \boldsymbol{m}_i + \eta(\boldsymbol{x}^t - \boldsymbol{m}_j)$
Until $\boldsymbol{m}_i$ converge

Figure 12.2: Online k-means algorithm. The batch version is given in figure 7.3. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

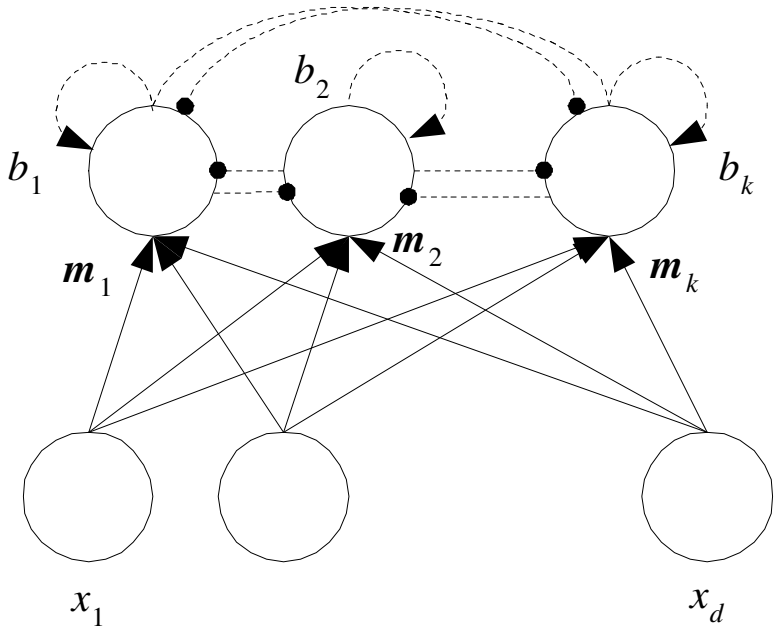


Figure 12.3: The winner-take-all competitive neural network, which is a network of k perceptrons with recurrent connections at the output. Dashed lines are recurrent connections, of which the ones that end with an arrow are excitatory and the ones that end with a circle are inhibitory. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

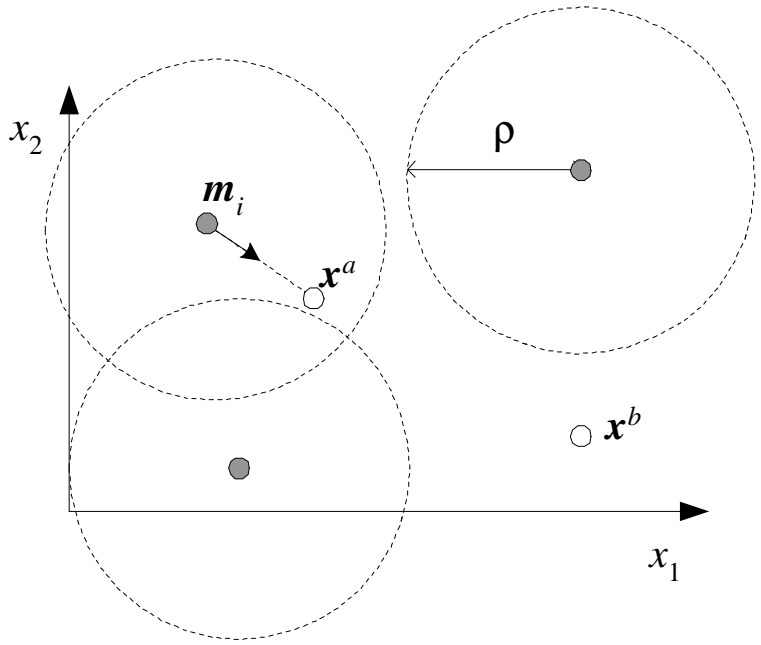


Figure 12.4: The distance from $\boldsymbol{x}^a$ to the closest center is less than the vigilance value ρ and the center is updated as in online k-means. However, $\boldsymbol{x}^b$ is not close enough to any of the centers and a new group should be created at that position. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

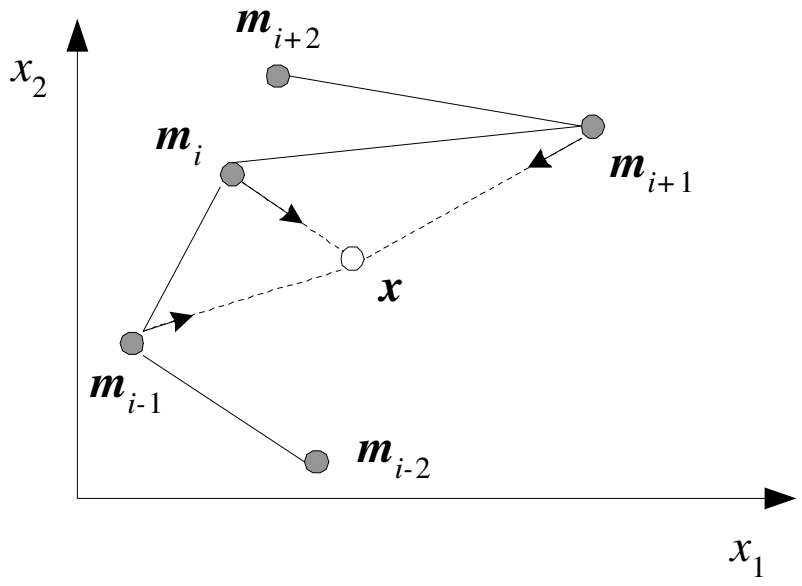


Figure 12.5: In the SOM, not only the closest unit but also its neighbors, in terms of indices, are moved toward the input. Here, neighborhood is 1; $\boldsymbol{m}_i$ and its 1-nearest neighbors are updated. Note here that $\boldsymbol{m}_{i+1}$ is far from $\boldsymbol{m}_i$, but as it is updated with $\boldsymbol{m}_i$, and as $\boldsymbol{m}_i$ will be updated when $\boldsymbol{m}_{i+1}$ is the winner, they will become neighbors in the input space as well. *From: E. Alpaydın. 2004.* Introduction to Machine Learning.

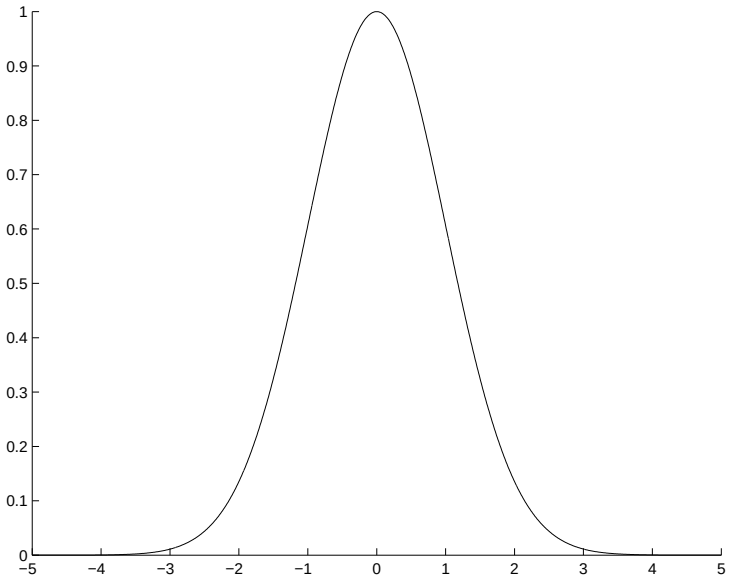
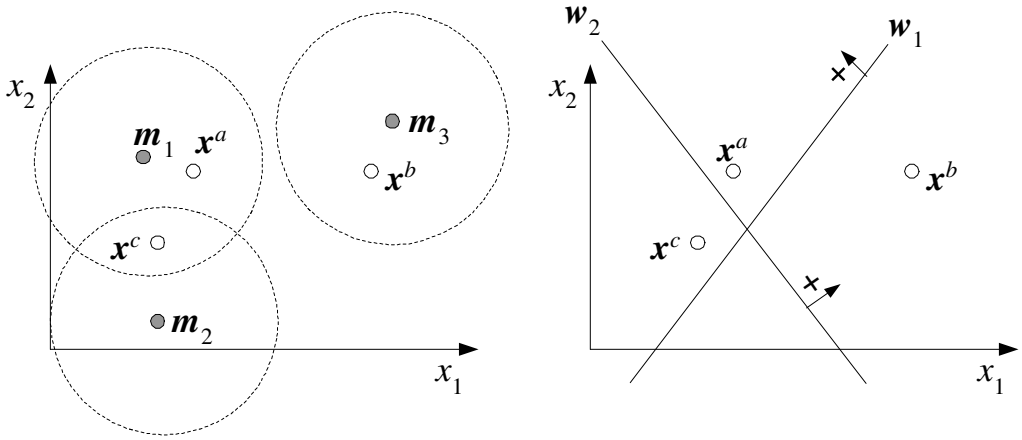


Figure 12.6: The one-dimensional form of the bell-shaped function used in the radial basis function network. This one has $m = 0$ and $s = 1$. It is like a Gaussian but it is not a density; it does not integrate to 1. It is nonzero between $(m - 3s, m + 3s)$, but a more conservative interval is $(m - 2s, m + 2s)$. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*



Local representation in the space of (p_1, p_2, p_3)

$\boldsymbol{x}^a$: (1.0, 0.0, 0.0)
$\boldsymbol{x}^b$: (0.0, 0.0, 1.0)
$\boldsymbol{x}^c$: (1.0, 1.0, 0.0)

Distributed representation in the space of (h_1, h_2)

$\boldsymbol{x}^a$: (1.0, 1.0)
$\boldsymbol{x}^b$: (0.0, 1.0)
$\boldsymbol{x}^c$: (1.0, 0.0)

Figure 12.7: The difference between local and distributed representations. The values are hard, 0/1, values. One can use soft values in $(0, 1)$ and get a more informative encoding. In the local representation, this is done by the Gaussian RBF that uses the distance to the center, $\boldsymbol{m}_i$, and in the distributed representation, this is done by the sigmoid that uses the distance to the hyperplane, $\boldsymbol{w}_i$. *From: E. Alpaydın. 2004.* Introduction to Machine Learning.

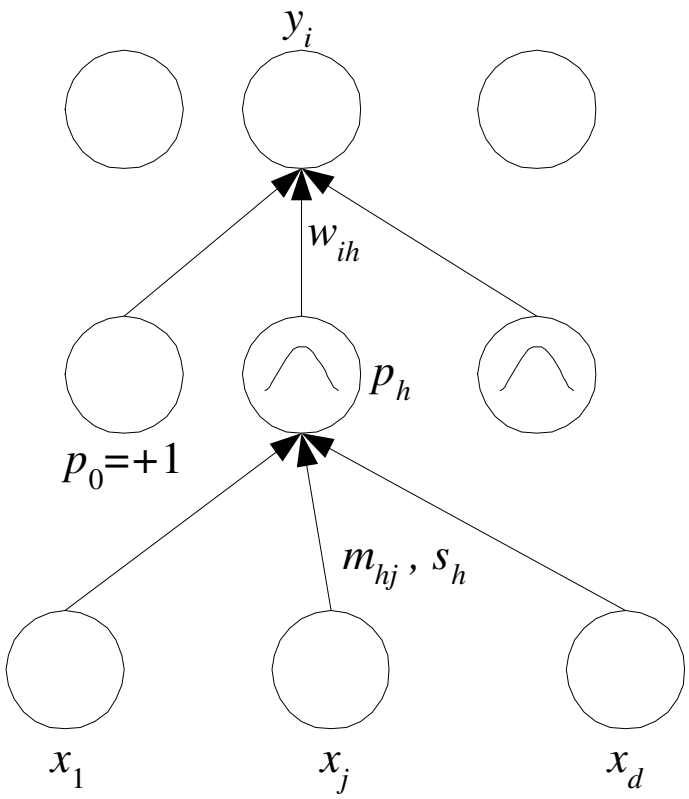


Figure 12.8: The RBF network where p_h are the hidden units using the bell-shaped activation function. $\boldsymbol{m}_h, s_h$ are the first-layer parameters, and $\boldsymbol{w}_i$ are the second-layer weights. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

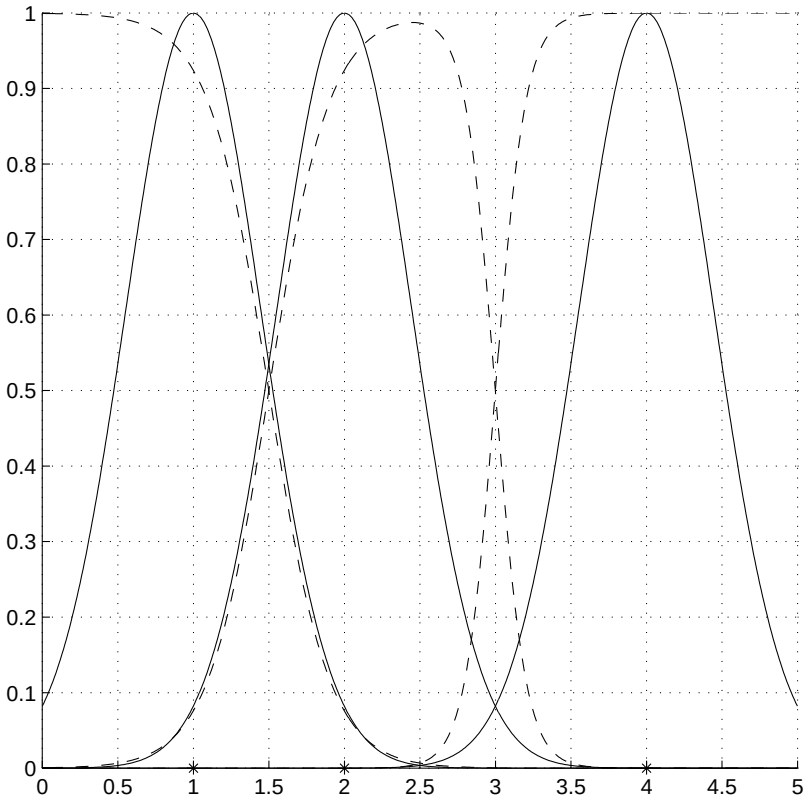


Figure 12.9: (-) Before and (- -) after normalization for three Gaussians whose centers are denoted by '*'. Note how the nonzero region of a unit depends also on the positions of other units. If the spreads are small, normalization implements a harder split; with large spreads, units overlap more. *From:* *E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

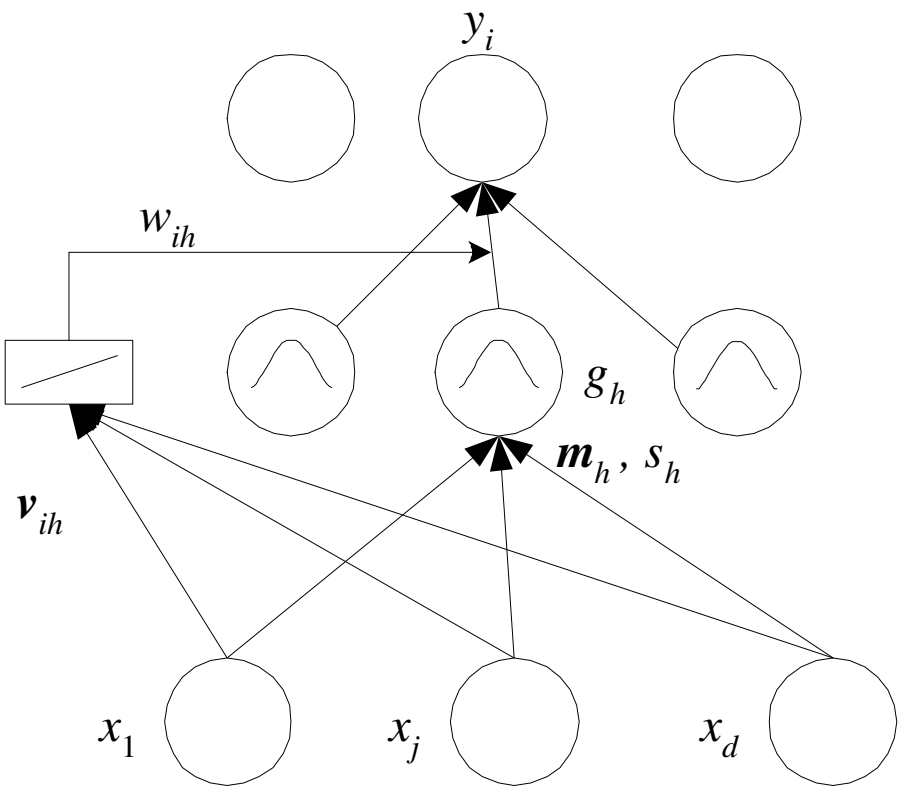


Figure 12.10: The mixture of experts can be seen as an RBF network where the second-layer weights are outputs of linear models. Only one linear model is shown for clarity. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

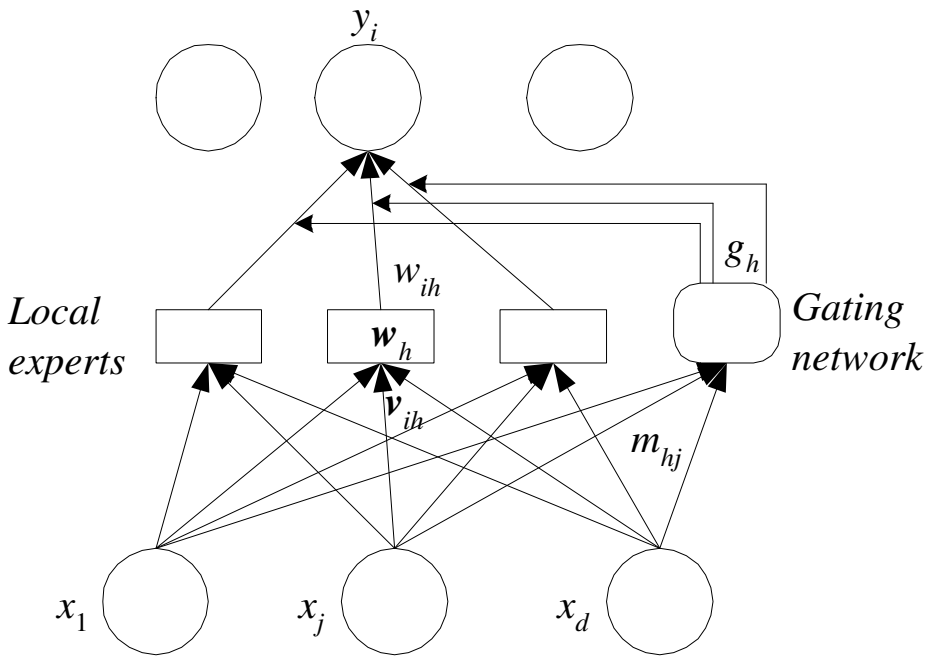


Figure 12.11: The mixture of experts can be seen as a model for combining multiple models. $\boldsymbol{w}_h$ are the models and the gating network is another model determining the weight of each model, as given by g_h. Viewed in this way, neither the experts nor the gating are restricted to be linear. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Chapter 13:

Hidden Markov Models

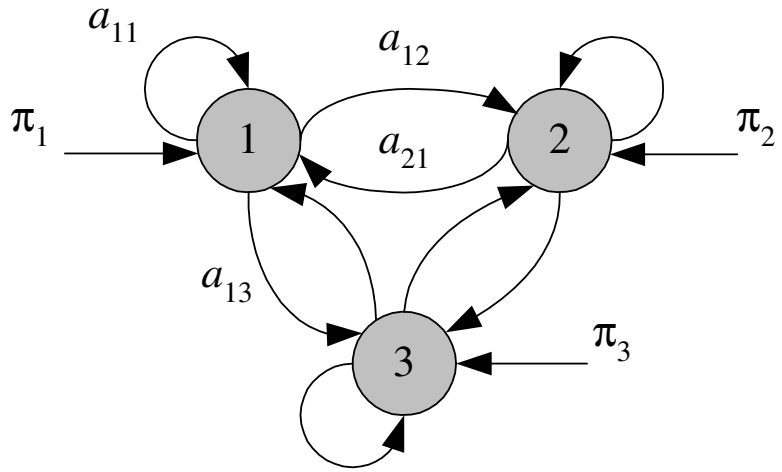


Figure 13.1: Example of a Markov model with three states is a stochastic automaton. π_i is the probability that the system starts in state S_i, and a_{ij} is the probability that the system moves from state S_i to state S_j. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

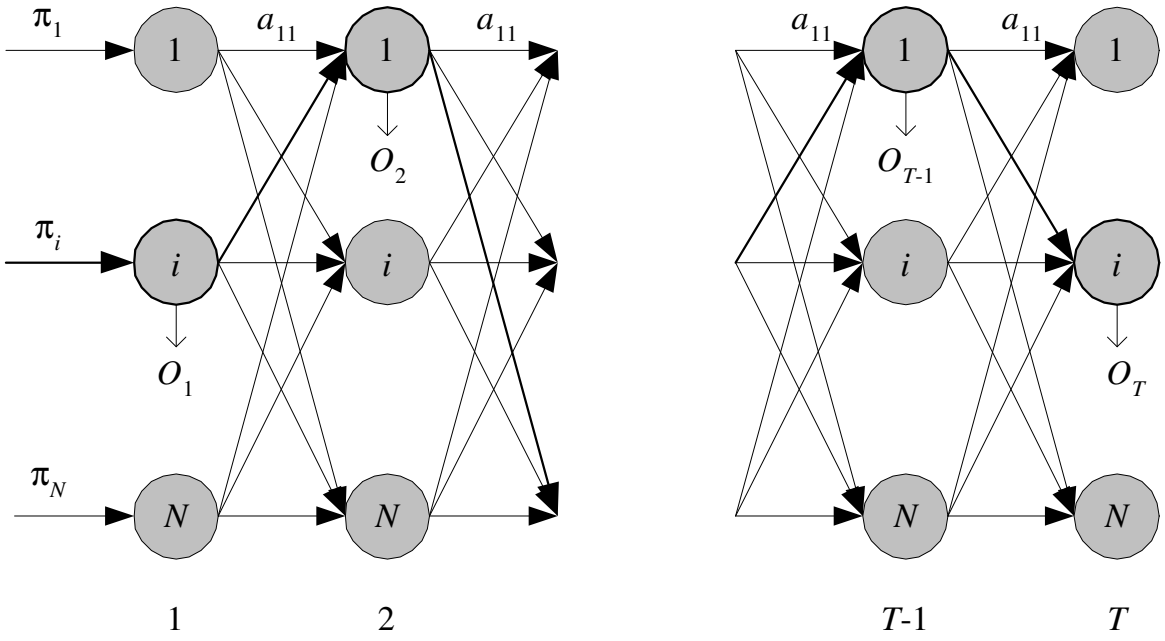
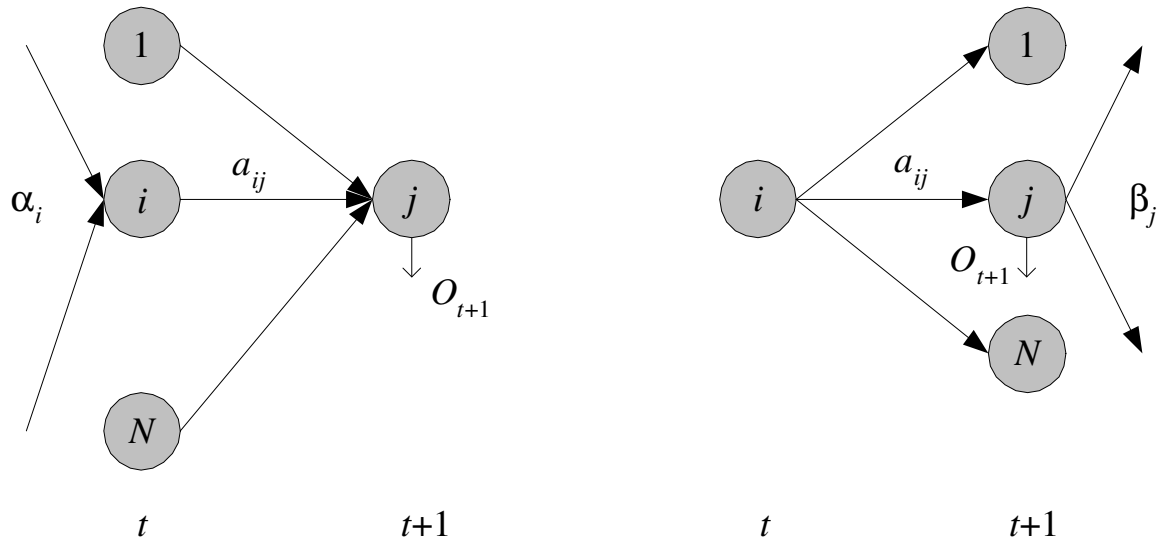


Figure 13.2: An HMM unfolded in time as a lattice (or trellis) showing all the possible trajectories. One path, shown in thicker lines, is the actual (unknown) state trajectory that generated the observation sequence. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*



(a) Forward (b) Backward

Figure 13.3: Forward-backward procedure: (a) computation of $\alpha_t(j)$ and (b) computation of $\beta_t(i)$. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

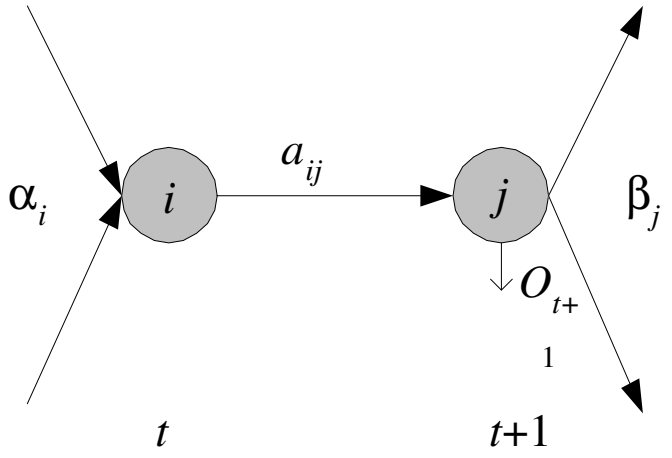


Figure 13.4: Computation of arc probabilities, $\xi_t(i,j)$.
From: E. Alpaydın. 2004. Introduction to Machine Learning. ©*The MIT Press.*

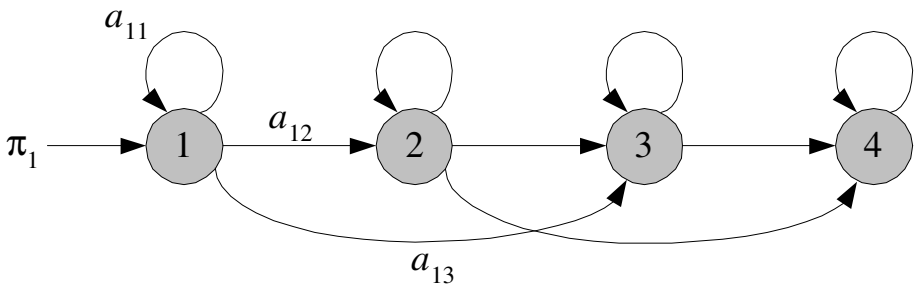


Figure 13.5: Example of a left-to-right HMM. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Chapter 14:

Assessing and Comparing

Classification Algorithms

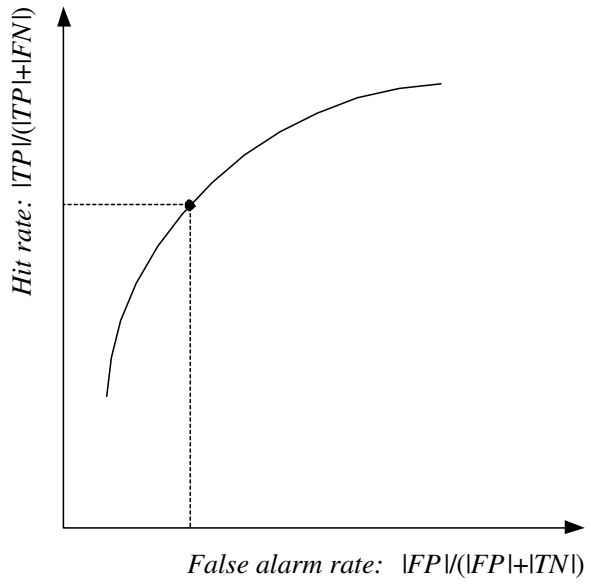


Figure 14.1: Typical roc curve. Each classifier has a parameter, for example, a threshold, which allows us to move over this curve, and we decide on a point, based on the relative importance of hits versus false alarms, namely, true positives and false positives. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

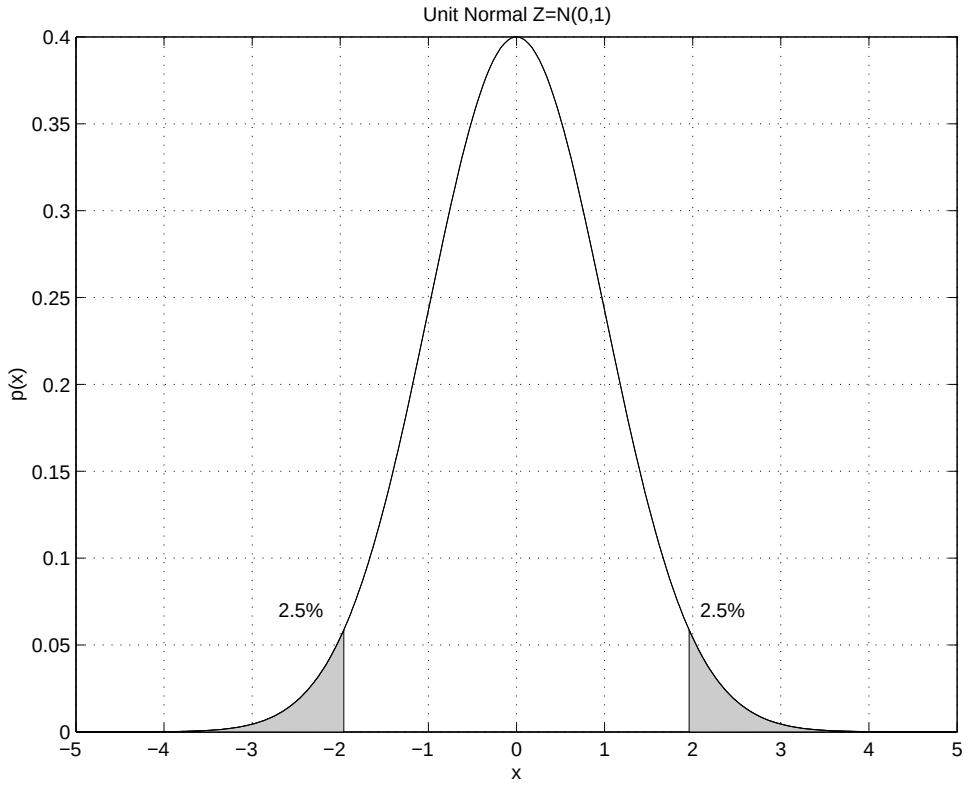


Figure 14.2: 95 percent of the unit normal distribution lies between -1.96 and 1.96. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

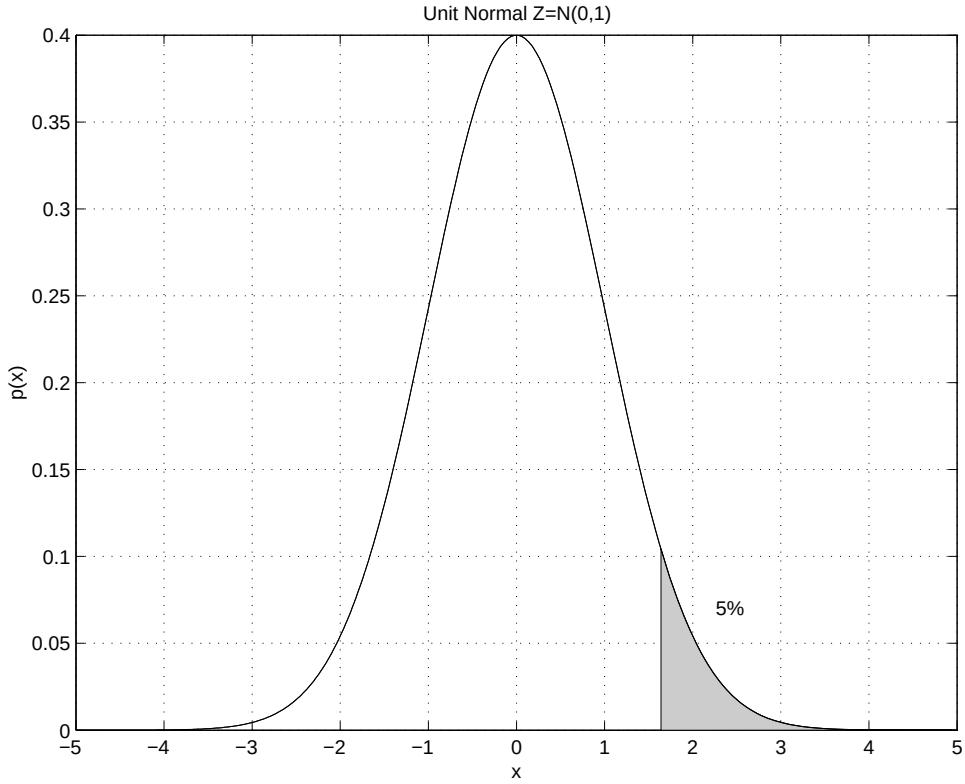


Figure 14.3: 95 percent of the unit normal distribution lies before 1.64. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Chapter 15:

Combining Multiple Learners

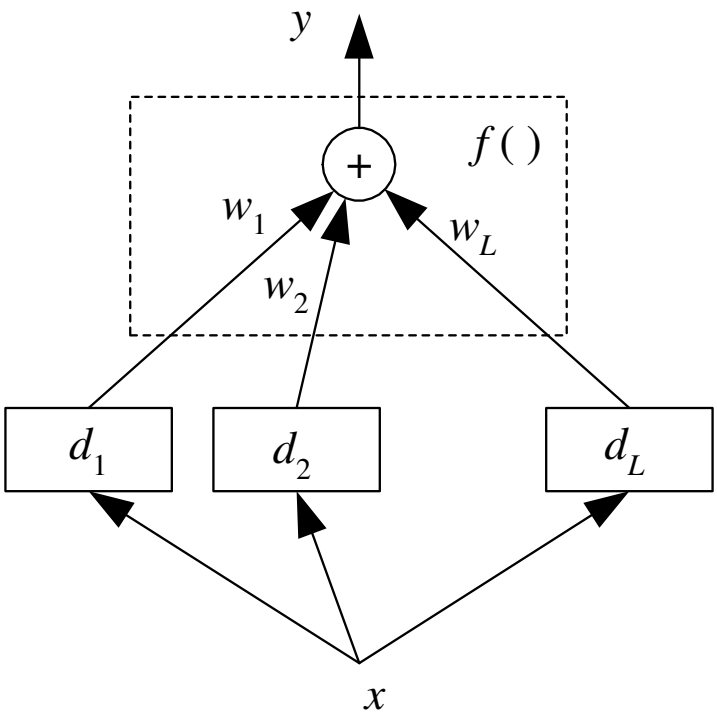


Figure 15.1: In voting, the combiner function $f(\cdot)$ is a weighted sum. d_j are the multiple learners, and w_j are the weights of their votes. y is the overall output. In the case of multiple outputs, for example, classification, the learners have multiple outputs d_{ji} whose weighted sum gives y_i. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Training:

For all $\{x^t, r^t\}_{t=1}^N \in \mathcal{X}$, initialize $p_1^t = 1/N$

For all base-learners $j = 1, \ldots, L$

Randomly draw $\mathcal{X}_j$ from $\mathcal{X}$ with probabilities p_j^t

Train d_j using $\mathcal{X}_j$

For each (x^t, r^t), calculate $y_j^t \leftarrow d_j(x^t)$

Calculate error rate: $\epsilon_j \leftarrow \sum_t p_j^t \cdot 1(y_j^t \neq r^t)$

If $\epsilon_j > 1/2$, then $L \leftarrow j - 1$; stop

$\beta_j \leftarrow \epsilon_j / (1 - \epsilon_j)$

For each (x^t, r^t), decrease probabilities if correct:

If $y_j^t = r^t$ $p_{j+1}^t \leftarrow \beta_j p_j^t$ Else $p_{j+1}^t \leftarrow p_j^t$

Normalize probabilities:

$Z_j \leftarrow \sum_t p_{j+1}^t; \quad p_{j+1}^t \leftarrow p_{j+1}^t / Z_j$

Testing:

Given x, calculate $d_j(x), j = 1, \ldots, L$

Calculate class outputs, $i = 1, \ldots, K$:

$$y_i = \sum_{j=1}^{L} \left(\log \frac{1}{\beta_j} \right) d_{ji}(x)$$

Figure 15.2: AdaBoost algorithm. *From:*

E. Alpaydın. 2004. Introduction to Machine

Learning. ©*The MIT Press.*

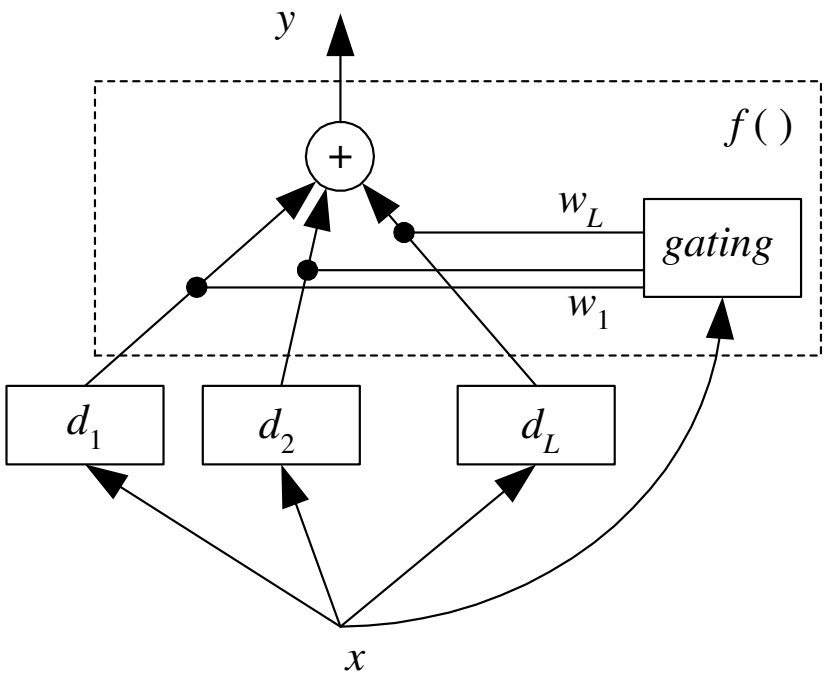


Figure 15.3: Mixture of experts is a voting method where the votes, as given by the gating system, are a function of the input. The combiner system f also includes this gating system. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

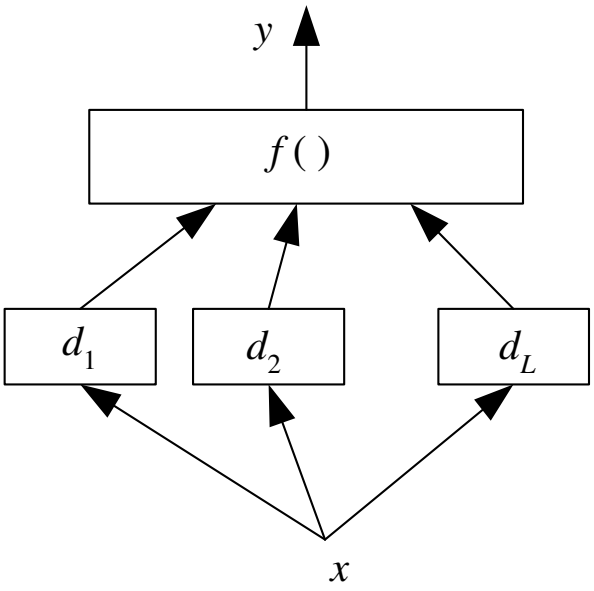


Figure 15.4: In stacked generalization, the combiner is another learner and is not restricted to being a linear combination as in voting. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

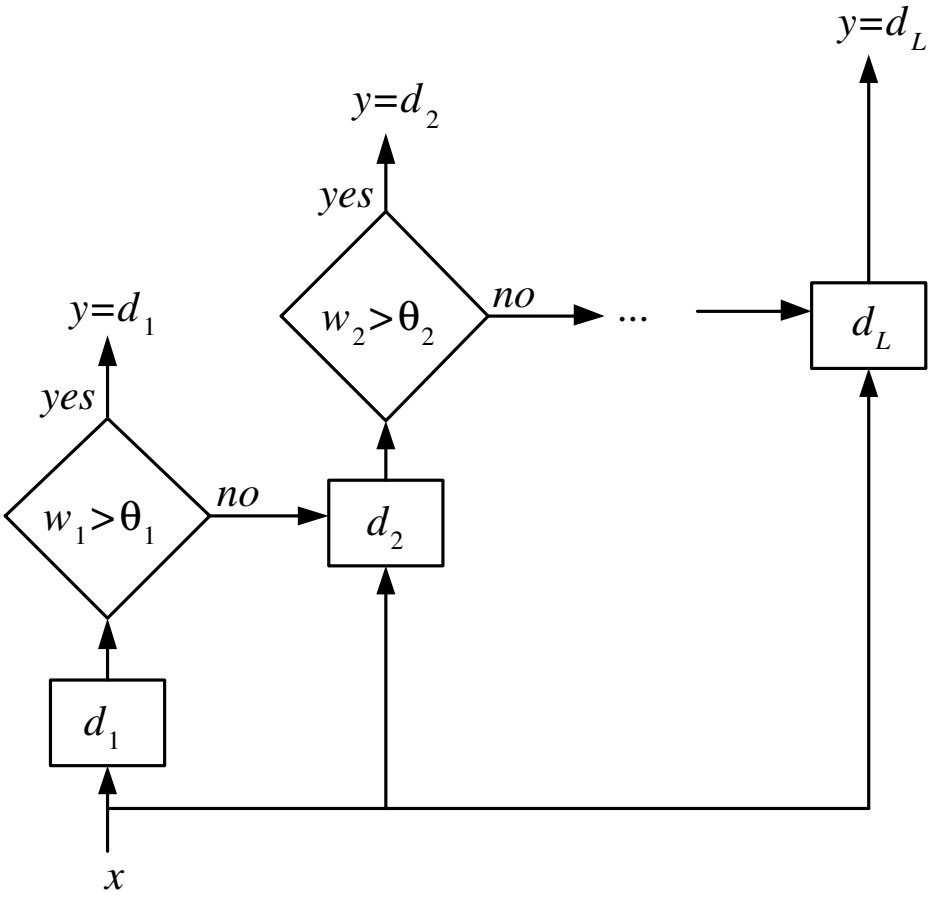


Figure 15.5: Cascading is a multistage method where there is a sequence of classifiers, and the next one is used only when the preceding ones are not confident. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Chapter 16:

Reinforcement Learning

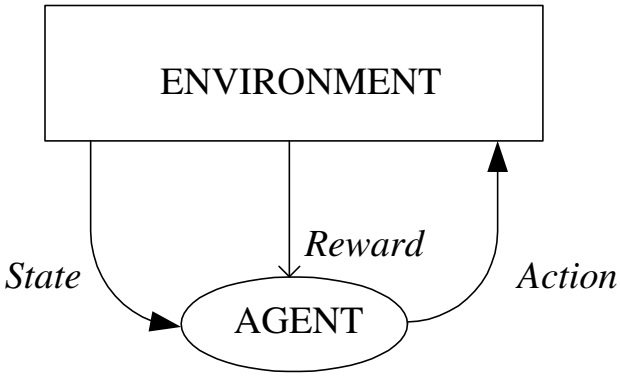


Figure 16.1: The agent interacts with an environment. At any state of the environment, the agent takes an action that changes the state and returns a reward. *From: E. Alpaydın. 2004.* Introduction to Machine Learning.

Initialize $V(s)$ to arbitrary values
Repeat
  For all $s \in \mathcal{S}$
    For all $a \in \mathcal{A}$
      $Q(s,a) \leftarrow E[r|s,a] + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a)V(s')$
    $V(s) \leftarrow \max_a Q(s,a)$
Until $V(s)$ converge

Figure 16.2: Value iteration algorithm for model-based learning. *From: E. Alpaydın. 2004.* Introduction to Machine Learning.

Initialize a policy π arbitrarily
Repeat
  $\pi \leftarrow \pi'$
  Compute the values using π by
    solving the linear equations
      $V^{\pi}(s) = E[r|s, \pi(s)] + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \pi(s)) V^{\pi}(s')$
  Improve the policy at each state
    $\pi'(s) \leftarrow \arg\max_a (E[r|s, a] + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) V^{\pi}(s'))$
Until $\pi = \pi'$

Figure 16.3: Policy iteration algorithm for model-based learning. *From: E. Alpaydın. 2004.* Introduction to Machine Learning.

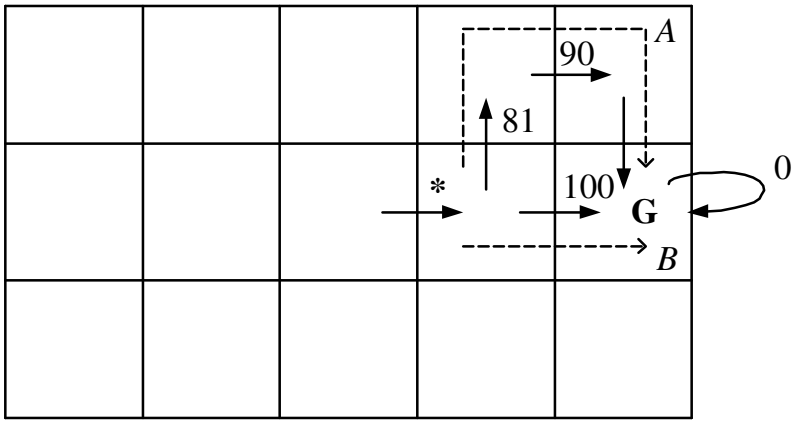


Figure 16.4: Example to show that Q values increase but never decrease. This is a deterministic grid-world where G is the goal state with reward 100, all other immediate rewards are 0 and $\gamma = 0.9$. Let us consider the Q value of the transition marked by asterisk, and let us just consider only the two paths A and B. Let us say that path A is seen before path B, then we have $\gamma \max(0, 81) = 72.9$. If afterward B is seen, a shorter path is found and the Q value becomes $\gamma \max(100, 81) = 90$. If B is seen before A, the Q value is $\gamma \max(100, 0) = 90$. Then when B is seen, it does not change because $\gamma \max(100, 81) = 90$. *From: E. Alpaydın. 2004.* Introduction to Machine Learning.

Initialize all $Q(s, a)$ arbitrarily
For all episodes
  Initalize s
  Repeat
    Choose a using policy derived from Q, e.g., ϵ-greedy
    Take action a, observe r and s'
    Update $Q(s, a)$:
      $Q(s, a) \leftarrow Q(s, a) + \eta(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$
    $s \leftarrow s'$
  Until s is terminal state

Figure 16.5: Q learning, which is an off-policy temporal difference algorithm. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Initialize all $Q(s,a)$ arbitrarily
For all episodes
  Initalize s
  Choose a using policy derived from Q, e.g., ϵ-greedy
  Repeat
    Take action a, observe r and s'
    Choose a' using policy derived from Q, e.g., ϵ-greedy
    Update $Q(s,a)$:
      $Q(s,a) \leftarrow Q(s,a) + \eta(r + \gamma Q(s',a') - Q(s,a))$
    $s \leftarrow s',\ a \leftarrow a'$
  Until s is terminal state

Figure 16.6: Sarsa algorithm, which is an on-policy version of Q learning. *From: E. Alpaydın. 2004.* Introduction to Machine Learning.

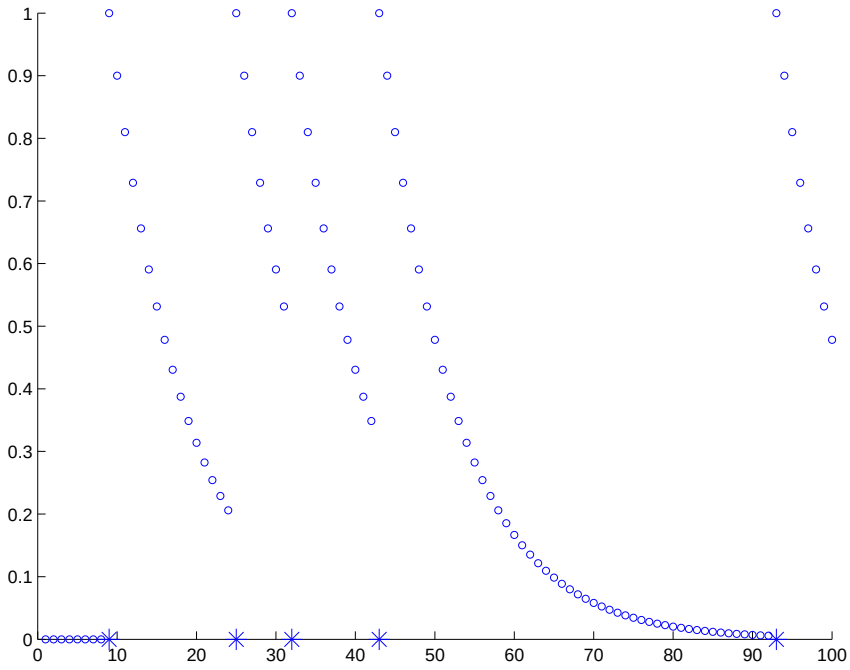


Figure 16.7: Example of an eligibility trace for a value. Visits are marked by an asterisk. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Initialize all $Q(s,a)$ arbitrarily, $e(s,a) \leftarrow 0, \forall s,a$
For all episodes
  Initalize s
  Choose a using policy derived from Q, e.g., ϵ-greedy
  Repeat
    Take action a, observe r and s'
    Choose a' using policy derived from Q, e.g., ϵ-greedy
    $\delta \leftarrow r + \gamma Q(s',a') - Q(s,a)$
    $e(s,a) \leftarrow 1$
    For all s,a:
      $Q(s,a) \leftarrow Q(s,a) + \eta \delta e(s,a)$
      $e(s,a) \leftarrow \gamma \lambda e(s,a)$
    $s \leftarrow s',\ a \leftarrow a'$
  Until s is terminal state

Figure 16.8: Sarsa(λ) algorithm. *From: E. Alpaydın. 2004.* Introduction to Machine Learning.

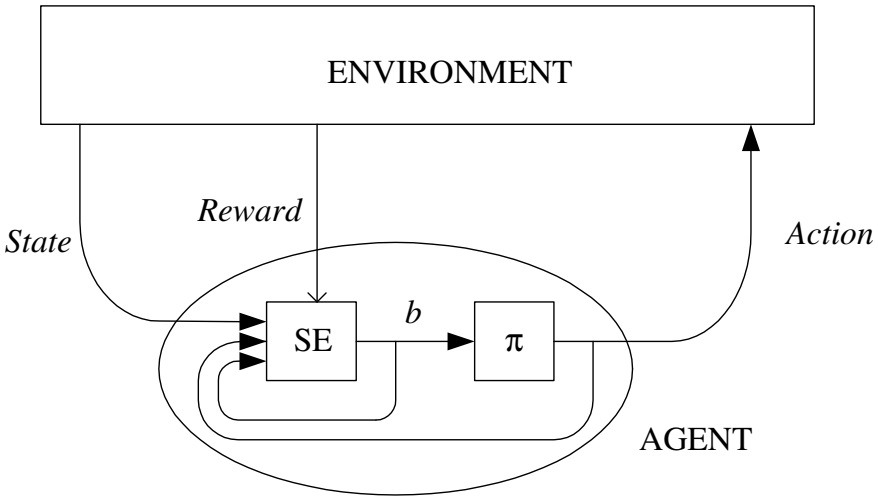


Figure 16.9: In the case of a partially observable environment, the agent has a state estimator (SE) that keeps an internal belief state b and the policy π generates actions based on the belief states. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

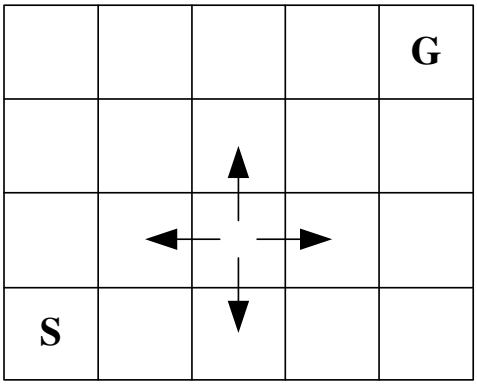


Figure 16.10: The grid world. The agent can move in the four compass directions starting from S. The goal state is G. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Appendix: Probability

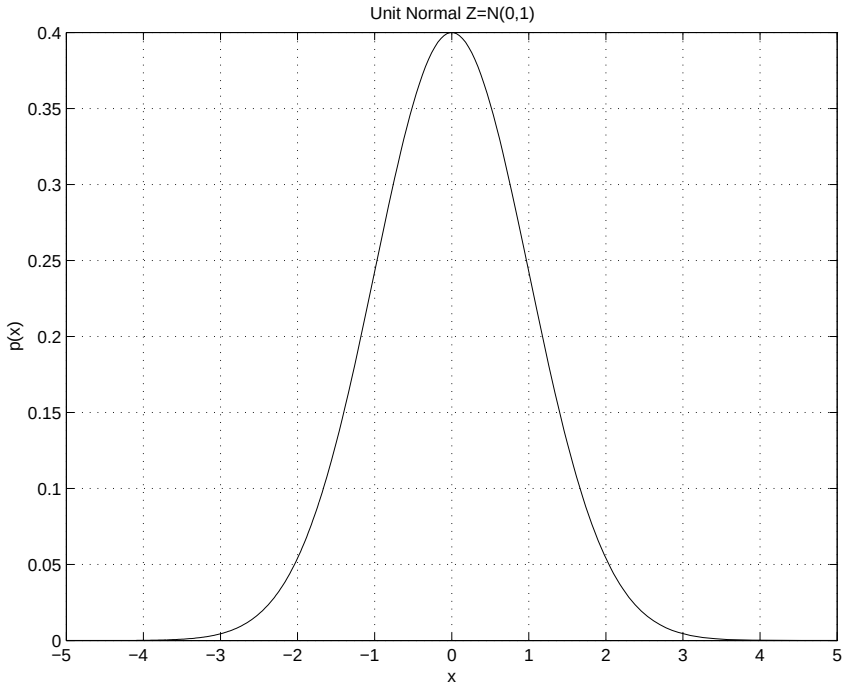


Figure A.1: Probability density function of $\mathcal{Z}$, the unit normal distribution.