Part III

Appendices
A. Conventional Classifiers

A.1 Bayesian Classifiers

Bayesian classifiers are based on probability theory and give the theoretical basis for pattern classification.

Let $\omega$ be a random variable and take one of $n$ states: $\omega_1, \ldots, \omega_n$, where $\omega_i$ indicates class $i$, and an $m$-dimensional feature vector $\mathbf{x}$ be a random variable vector. We assume that we know the a priori probabilities $P(\omega_i)$ and conditional densities $p(\mathbf{x} | \omega_i)$. Then when $\mathbf{x}$ is observed, the a posteriori probability of $\omega_i$, $P(\omega_i | \mathbf{x})$ is calculated by the Bayesian rule:

$$P(\omega_i | \mathbf{x}) = \frac{p(\mathbf{x} | \omega_i) P(\omega_i)}{p(\mathbf{x})}, \quad (A.1)$$

where

$$p(\mathbf{x}) = \sum_{i=1}^{n} p(\mathbf{x} | \omega_i) P(\omega_i). \quad (A.2)$$

Assume that the cost $c_{ij}$ is given when $\mathbf{x}$ is classified into class $i$ although it is class $j$. Then the expected conditional cost in classifying $\mathbf{x}$ into class $i$, $C(\omega_i | \mathbf{x})$, is given by

$$C(\omega_i | \mathbf{x}) = \sum_{j=1}^{n} c_{ij} P(\omega_j | \mathbf{x}). \quad (A.3)$$

The conditional cost is minimized when $\mathbf{x}$ is classified into the class

$$\arg \min_{i=1, \ldots, n} C(\omega_i | \mathbf{x}). \quad (A.4)$$

This rule is called the Bayesian decision rule.

In diagnosis problems, usually there are normal and abnormal classes. Misclassification of normal data into the abnormal class is less fatal than misclassification of abnormal data into the normal class. In such a situation, we set a smaller cost to the former than the latter.

If we want to minimize the average probability of misclassification, we set the cost as follows:

$$c_{ij} = \begin{cases} 0 & \text{for } i = j, \\ 1 & \text{for } i \neq j, \quad i, j = 1, \ldots, n. \end{cases} \quad (A.5)$$
Then, from (A.1) and (A.2) the conditional cost given by (A.3) becomes

$$C(\omega_i | x) = \sum_{j=1}^{n} P(\omega_j | x)$$

$$= 1 - P(\omega_i | x).$$

(A.6)

Therefore, the Bayesian decision rule given by (A.4) becomes

$$\arg \max_{i=1, \ldots, n} P(\omega_i | x) = \arg \max_{i=1, \ldots, n} p(x | \omega_i) P(\omega_i).$$

(A.7)

Now, we assume that the conditional densities $p(x | \omega_i)$ are normal:

$$p(x | \omega_i) = \frac{1}{\sqrt{(2\pi)^n \det(Q_i)}} \exp \left( -\frac{(x - c_i)^t Q_i^{-1} (x - c_i)}{2} \right),$$

(A.8)

where $c_i$ is the mean vector and $Q_i$ is the covariance matrix of the normal distribution for class $i$. If the a priori probabilities $P(\omega_i)$ are the same for $i = 1, \ldots, n$, $x$ is classified into class $i$ if $p(x | \omega_i)$ given by (A.8) is the maximum.

### A.2 Nearest Neighbor Classifiers

#### A.2.1 Classifier Architecture

Nearest neighbor classifiers use all the training data as templates for classification. In the simplest form, for a given input vector, the nearest neighbor classifier searches the nearest template and classifies the input vector into the class to which the template belongs. In the complex form the classifier treats $k$ nearest neighbors. For a given input vector, the $k$ nearest templates are searched and the input vector is classified into the class with the maximum number of templates. The classifier architecture is simple but as the number of training data becomes larger, the classification time becomes longer. Therefore many methods for speeding up classification are studied [65, pp. 181–191], [129, pp. 191–201]. One uses the branch-and-bound method [108, pp. 360–362] and another edits the training data, i.e., select or replace the data with the suitable templates. It is proved theoretically that as the number of templates becomes larger, the expected error rate of the nearest neighbor classifier is bounded by twice that of the Bayesian classifier [93, pp. 159–175].

Usually the Euclidean distance is used for measuring the distance between two data $x$ and $y$:

$$d(x - y) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

(A.9)

but other distances, such as the Manhattan distance:
\[ d(x, y) = \sum_{i=1}^{m} |x_i - y_i| \]  

(A.10)

are used. It is clear from the architecture that the recognition rate of the training data for the 1-nearest neighbor classifier is 100%. But for the \( k \)-nearest neighbor classifier with \( k > 1 \), the recognition rate of the training data is not always 100%.

Since the distances such as the Euclidean and Manhattan distances are not invariant in scaling, classification performance varies according to the scaling of input ranges.

The fuzzy min-max classifier discussed in Section 9.1 is equivalent to the 1-nearest neighbor classifier with the Manhattan distance when \( \theta = 0 \), i.e., a fuzzy rule is defined for each datum, and the sensitivity parameter \( \gamma \) is set so that the degrees of membership of each datum are non-zero.

### A.2.2 Performance Evaluation

We evaluated the performance of the \( k \)-nearest neighbor classifier using the data sets listed in Table 1.1 on page 19. Since the maximum recognition rates of the test data using the Euclidean distance and the Manhattan distance did not differ significantly for different \( k \)'s, we used the Euclidean distance. We coded \( k \)-nearest neighbor classifier without using any speedup method and ran the optimized Fortran code on a Sun UltraSPARC IIIi 333MHz workstation. The time listed in the following tables is the time for evaluating the recognition rate of the test data.

**Iris Data.** Table A.1 lists the recognition rates of the test (training) data and the execution time. For the training data, the numbers of misclassified data varied from 0 to 3 and for the test data the numbers varied from 4 to 6. The 1-nearest neighbor classifier showed the maximum recognition rates both for the test and the training data. Thus overfitting was not occurred.

<table>
<thead>
<tr>
<th>( k )</th>
<th>Rates (%)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>94.67 (100)</td>
<td>0.02</td>
</tr>
<tr>
<td>3</td>
<td>92.00 (96.00)</td>
<td>0.01</td>
</tr>
<tr>
<td>5</td>
<td>92.00 (97.33)</td>
<td>0.01</td>
</tr>
<tr>
<td>7</td>
<td>93.33 (98.67)</td>
<td>0.03</td>
</tr>
</tbody>
</table>

Table A.1. Performance for the iris data
Numeral Data. Table A.2 shows the results for the numeral data. The 1-nearest neighbor classifier showed the maximum recognition rates both for the test and training data; 4 data among the 820 test data were misclassified. In this case also overfitting was not occurred.

<table>
<thead>
<tr>
<th>$k$</th>
<th>Rates</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(%)</td>
<td>(s)</td>
</tr>
<tr>
<td>1</td>
<td>99.51</td>
<td>0.39</td>
</tr>
<tr>
<td>3</td>
<td>99.02</td>
<td>0.43</td>
</tr>
<tr>
<td>5</td>
<td>99.15</td>
<td>0.46</td>
</tr>
<tr>
<td>7</td>
<td>98.90</td>
<td>0.46</td>
</tr>
</tbody>
</table>

Thyroid Data. Table A.3 lists the performance for the thyroid data. The recognition rate of the test data for the 1-nearest neighbor classifier was 91.98%. Since 92% of the data belong to one class, the recognition rate was very bad. This might be due to the use of the Euclidean distance for the mostly digital features. The 5-nearest neighbor classifier showed the maximum recognition rate of 93.67% for the test data, but it was still very bad. This was the worst recognition rate among the classifiers evaluated in this book for the thyroid data as listed in Table 10.3 on page 200. For the thyroid data overfitting occurred.

<table>
<thead>
<tr>
<th>$k$</th>
<th>Rates</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(%)</td>
<td>(s)</td>
</tr>
<tr>
<td>1</td>
<td>91.98</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>93.55</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>93.67</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>93.58</td>
<td>10</td>
</tr>
</tbody>
</table>

Blood Cell Data. Table A.4 lists the performance for the blood cell data. The recognition rates of the training data decreased monotonically as $k$ increased. The drop of the recognition rate was the largest among the benchmark data sets. This might indicate the heavy overlap between classes. But for the recognition rates of the test data, there was not so much difference among the classifiers and the 5-nearest neighbor classifier showed the maximum recognition rate of 90.13%.
Table A.4. Performance for the blood cell data

<table>
<thead>
<tr>
<th>k</th>
<th>Rates (%)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>89.90 (100)</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>89.84 (95.12)</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>90.13 (93.51)</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>89.84 (93.32)</td>
<td>6</td>
</tr>
</tbody>
</table>

**Hiragana Data.** Table A.5 lists the results for the hiragana-50 data. The 1-nearest neighbor classifier showed the maximum recognition rates of the test and training data. Thus no overfitting occurred.

Table A.5. Performance for the hiragana-50 data

<table>
<thead>
<tr>
<th>k</th>
<th>Rates (%)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>97.16 (100)</td>
<td>39</td>
</tr>
<tr>
<td>3</td>
<td>96.57 (98.94)</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>95.77 (97.79)</td>
<td>41</td>
</tr>
<tr>
<td>7</td>
<td>94.49 (96.70)</td>
<td>42</td>
</tr>
</tbody>
</table>

Table A.6 lists the performance for the hiragana-105 data. For the recognition rates of the test data, although there was no much difference among the classifiers, the 3-nearest neighbor classifier showed the maximum recognition rate of 99.99%.

Table A.6. Performance for the hiragana-105 data

<table>
<thead>
<tr>
<th>k</th>
<th>Rates (%)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>99.96 (100)</td>
<td>358</td>
</tr>
<tr>
<td>3</td>
<td>99.99 (99.84)</td>
<td>363</td>
</tr>
<tr>
<td>5</td>
<td>99.90 (99.73)</td>
<td>365</td>
</tr>
<tr>
<td>7</td>
<td>99.68 (99.63)</td>
<td>359</td>
</tr>
</tbody>
</table>

Table A.7 lists the performance for the hiragana-13 data. The 1-nearest neighbor classifier showed the maximum recognition rates both for the test
and training data. The recognition rates for the test and training data decreased monotonically as $k$ increased. The hiragana-13 data were obtained from the hiragana-105 and the maximum recognition rate of the test data was comparable with that of the hiragana-105 data.

Table A.7. Performance for the hiragana-13 data

<table>
<thead>
<tr>
<th>$k$</th>
<th>Rates (%)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>99.55 (100)</td>
<td>42</td>
</tr>
<tr>
<td>3</td>
<td>99.21 (99.50)</td>
<td>47</td>
</tr>
<tr>
<td>5</td>
<td>98.68 (98.88)</td>
<td>50</td>
</tr>
<tr>
<td>7</td>
<td>98.24 (98.33)</td>
<td>48</td>
</tr>
</tbody>
</table>

Discussions. The nearest neighbor classifier uses all the training data as templates and thus training is not necessary. Thus the classification time even for the hiragana-105 was not so long (about 6 minutes). But the problem is that a large number of templates must be stored for classification.

For four data sets among seven data sets, the 1-nearest neighbor classifier performed best without overfitting. But for the thyroid and blood cell data the 5-nearest neighbor classifier performed best and for the hiragana-105 data the 3-nearest neighbor classifier did. Thus the best classifier depends on the classification problem.

As compared in Chapter 10, the classification performance of the $k$-nearest neighbor classifier was not stable. Namely, the classification performance was good for the hiragana data sets but was the worst for the iris and thyroid data sets. To improve classification performance, proper scaling might be necessary [129, pp.197-198], [130, p. 71].
B. Matrices

B.1 Matrix Properties

In this section, we summarize the matrix properties used in this book. For more detailed explanation, see, e.g. [47].

Vectors $x_1, \ldots, x_n$ are \textit{linearly independent} if

$$a_1 x_1 + \cdots + a_n x_n = 0$$

(B.1)

holds only when $a_1 = \cdots = a_m = 0$. Otherwise, namely, at least one $a_i$ is nonzero, $x_1, \ldots, x_n$ are \textit{linearly dependent}.

Let $A$ be an $m \times m$ matrix:

$$A = \begin{pmatrix}
a_{11} & \cdots & a_{1m} \\
\vdots & \ddots & \vdots \\
a_{m1} & \cdots & a_{mm}
\end{pmatrix}.$$ (B.2)

Then the \textit{transpose} of $A$ denoted by $A^t$ is

$$A^t = \begin{pmatrix}
a_{11} & \cdots & a_{m1} \\
\vdots & \ddots & \vdots \\
a_{1m} & \cdots & a_{mm}
\end{pmatrix}.$$ (B.3)

If $A$ satisfies $A = A^t$, $A$ is a \textit{symmetric matrix}. If $A$ satisfies $A^t A = A A^t = I$, $A$ is an \textit{orthogonal matrix}.

The $m \times m$ unit matrix $I$ is

$$I = \begin{pmatrix}
1 & 0 & \cdots & 0 \\
0 & 1 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1
\end{pmatrix}.$$ (B.4)

If $m \times m$ matrices $A$ and $B$ satisfies $AB = I$, $B$ is called the \textit{inverse} of $A$ and is denoted by $A^{-1}$. If $A$ has the inverse, $A$ is \textit{nonsingular}. Otherwise, $A$ is singular.

The \textit{determinant} of an $m \times m$ matrix $A = \{a_{ij}\}$, $\det(A)$, is defined recursively by

$$\det(A) = \sum_{i=1}^{m} (-1)^{i+1} a_{1i} \det(A_{1i}),$$ (B.5)
where $A_{11}$ is the $(m - 1) \times (m - 1)$ matrix obtained by deleting the first row and the ith column from $A$. When $m = 1$, $\det(A) = a_{11}$.

If the $m \times m$ matrix $A$ satisfies

$$Ax = \lambda x, \quad (B.6)$$

where $x$ is a non-zero, $m$-dimensional vector, $\lambda$ is a constant and is called an eigenvalue, and $x$ is called an eigenvector. Rearranging (B.6) gives

$$(A - \lambda I)x = 0. \quad (B.7)$$

Thus, (B.7) has nonzero $x$, when

$$\det(A - \lambda I) = 0, \quad (B.8)$$

which is called a characteristic equation.

**Theorem B.1.1.** All the eigenvalues of a real symmetric matrix are real.

**Theorem B.1.2.** Eigenvectors associated with different eigenvalues for a real symmetric matrix are orthogonal.

For an $m$-dimensional vector $x$ and an $m \times m$ symmetric matrix $A$, $Q = x^tAx$ is called a quadratic form. If for any nonzero $x$, $Q = x^tAx \geq 0$, $Q$ is positive semi-definite. Matrix $Q$ is positive definite, if the strict inequality holds. Let $L$ be an $m \times m$ orthogonal matrix. By $y = Lx$, $x$ is transformed into $y$. This is the transformation from one orthonormal base into another orthonormal basis. The quadratic form $Q$ is

$$Q = x^tAx$$

$$= y^tLA^tL^t y. \quad (B.9)$$

**Theorem B.1.3.** The characteristic equations for $A$ and $LAL^t$ are the same.

**Theorem B.1.4.** If an $m \times m$ real symmetric matrix $A$ is diagonalized by $L$:

$$LAL^t = \begin{pmatrix} 
\lambda_1 & 0 & \cdots & 0 \\
0 & \lambda_2 & \cdots & 0 \\
\cdots & \cdots & \cdots & \cdots \\
0 & 0 & \cdots & \lambda_m 
\end{pmatrix}, \quad (B.10)$$

$\lambda_1, \ldots, \lambda_m$ are the eigenvalues of $A$ and the $i$th row of $L$ is the eigenvector associated with $\lambda_i$.

If all the eigenvalues of $A$ are positive, $A$ is positive definite. If all the eigenvalues are non-negative, $A$ is positive semi-definite.
B.2 Least-squares Method and Singular Value Decomposition

Assume that we have $M$ input-output pairs $\{(a_1', b_1), \ldots, (a_M', b_M')\}$ in the $(n-1)$-dimensional input space $x'$ and one-dimensional output space $y$. Now using the least-squares method, we determine the linear relation of the input-output pairs:

$$y = p^t x' + q,$$

where $p$ is the $(n-1)$-dimensional vector, $q$ is a scalar constant, and $M \geq n$.

Rewriting (B.11), we get

$$(x'^t, 1) \begin{pmatrix} p \\ q \end{pmatrix} = y. \quad \text{(B.12)}$$

Substituting $a_i'$ and $b_i$ into $x'$ and $y$ of (B.12), respectively, and replacing $(p^t, q)^t$ with the $n$-dimensional parameter vector $x$, we obtain

$$a_i^t x = b_i \quad \text{for} \quad i = 1, \ldots, M, \quad \text{(B.13)}$$

where $a_i = (a_i^t, 1)^t$.

We determine the parameter vector $x$ so that the sum of squared errors:

$$E = (Ax - b)^t (Ax - b) \quad \text{(B.14)}$$

is minimized, where $A$ is an $M \times n$ matrix and $b$ is an $M$-dimensional vector:

$$A = \begin{pmatrix} a_1^t \\ a_2^t \\ \vdots \\ a_M^t \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{pmatrix}. \quad \text{(B.15)}$$

Here, if the rank of $A$ is smaller than $n$, there is no unique solution. In that situation, we determine $x$ so that the Euclidean norm of $x$ is minimized.

Matrix $A$ is decomposed into singular values [47]:

$$A = USV^t, \quad \text{(B.16)}$$

where $U$ and $V$ are $M \times M$ and $n \times n$ orthogonal matrices, respectively, and $S$ is an $M \times n$ diagonal matrix given by

$$S = \begin{pmatrix} \sigma_1 & & 0 \\ & \ddots & \vdots \\ 0 & & \sigma_n \end{pmatrix}. \quad \text{(B.17)}$$

Here, $\sigma_i$ are singular values and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$, and $0_{M-n,n}$ is the $(M-n) \times n$ zero matrix.

It is known that the columns of $U$ and $V$ are the eigenvectors of $AA^t$ and $A^tA$, respectively, and the singular values correspond to the square roots of
the eigenvalues of $A^t A$ which are the same with those of $A^t A$ [17, pp. 434-435]. Thus when $A$ is a symmetric square matrix, $U = V$ and $A = USU^t$. This is similar to the diagonalization of the square matrix given by Theorem B.1.4 on page 308. The difference is that the singular values $A$ are the absolute values of the eigenvalues of $A$. Thus, if $A$ is a positive (semi-)definite matrix, the both decompositions are the same.

Rewriting (B.14), we get [47, p. 256]

\[
E = (Ax - b)^t (Ax - b) \\
= (USV^t x - UU^t b)^t (Ax - b) \\
= (SV^t x - U^t b)^t (SV^t x - U^t b) \\
= \sum_{i=1}^{n} (\sigma_i v_i^t x - u_i^t b)^2 + \sum_{i=n+1}^{M} (u_i^t b)^2, \tag{B.18}
\]

where $U = (u_1, \ldots, u_M)$ and $V = (v_1, \ldots, v_M)$. Assuming the rank of $A$ is $r (\leq n)$, (B.18) is minimized when

\[
\sigma_i v_i^t x = u_i^t b \quad \text{for} \quad i = 1, \ldots, r, \tag{B.19}
\]

\[
v_i^t x = 0 \quad \text{for} \quad i = r + 1, \ldots, n. \tag{B.20}
\]

Equation (B.20) is imposed to obtain the minimum Euclidean norm solution. From (B.19) and (B.20), we obtain

\[
x = VS^+U^t b = A^+ b, \tag{B.21}
\]

where $S^+$ is the $n \times M$ diagonal matrix given by

\[
S^+ = \begin{pmatrix}
\frac{1}{\sigma_1} & 0 \\
0 & \ddots & 0 \\
0 & \frac{1}{\sigma_r} & 0 \\
0 & 0 & 0
\end{pmatrix}. \tag{B.22}
\]

We call $A^+$ the pseudo-inverse of $A$. We must bear in mind that in calculating the pseudo-inverse, we replace the reciprocal of 0 with 0, not infinity. This ensures the minimum norm solution.

From (B.16) and (B.21),

\[
A^+ A = VS^+U^t USV^t \\
= VS^+SV^t \\
= V \begin{pmatrix}
I_r & 0_{r,n-r} \\
0_{n-r,r} & 0_{n-r}
\end{pmatrix} V^t \\
= \begin{pmatrix}
I_r & 0_{r,n-r} \\
0_{n-r,r} & 0_{n-r}
\end{pmatrix}, \tag{B.23}
\]

\[
AA^+ = USS^+U^t \\
= \begin{pmatrix}
I_r & 0_{r,M-r} \\
0_{M-r,r} & 0_{M-r}
\end{pmatrix}. \tag{B.24}
\]
where \( I_r \) is the \( r \times r \) unit matrix, \( O_i \) is the \( i \times i \) zero matrix, \( O_{i,j} \) is the \( i \times j \) zero matrix. Therefore, if \( A \) is a square matrix with rank \( n \), \( A^+ A = AA^+ = I \). Namely, the pseudo-inverse of \( A \) coincides with the inverse of \( A \), \( A^{-1} \). If \( M > n \) and the rank of \( A \) is \( n \), \( A^+ A = I \) but \( AA^+ \neq I \). In this case \( A^+ \) is given by

\[
A^+ = (A^t A)^{-1} A^t.
\] (B.25)

This is obtained by taking the derivative of (B.14) with respect to \( x \) and equating the result to zero.

When \( M > n \) and the rank of \( A \) is smaller than \( n \), \( A^+ A \neq I \) and \( AA^+ \neq I \).

Even when \( A^t A \) is nonsingular, it is recommended to calculate the pseudo-inverse by singular value decomposition, not using (B.25). Because if \( A^t A \) is near singular, \( (A^t A)^{-1} A^t \) is vulnerable to the small singular values [131, pp. 59–70].

### B.3 Covariance Matrix

Let \( x_1, \ldots, x_M \) be \( M \) samples of the \( m \)-dimensional random variable \( X \). Then the sample covariance matrix of \( X \) is given by

\[
Q = \frac{1}{M} \sum_{i=1}^{M} (x_i - c) (x_i - c)^t,
\] (B.26)

where \( c \) is the mean vector:

\[
c = \frac{1}{M} \sum_{i=1}^{M} x_i.
\] (B.27)

To get the unbiased covariance matrix, we replace \( M \) with \( M - 1 \) in (B.26), but in this book we use (B.26) as the sample covariance matrix.

Let

\[
y_i = x_i - c.
\] (B.28)

Then, (B.26) becomes

\[
Q = \frac{1}{M} \sum_{i=1}^{M} y_i y_i^t.
\] (B.29)

From (B.27) and (B.28), \( y_1, \ldots, y_M \) are linearly dependent. According to the definition, the covariance matrix \( Q \) is symmetric. Matrix \( Q \) is positive (semi-)definite as the following theorem shows.

**Theorem B.3.1.** The covariance matrix \( Q \) given by (B.29) is positive definite if \( y_1, \ldots, y_M \) have at least \( m \) linearly independent data. Matrix \( Q \) is positive semi-definite, if any \( m \) data from \( y_1, \ldots, y_M \) are linearly dependent.
Proof. Let $z$ be an $m$-dimensional nonzero vector. From (B.29),
\[
zs'Qz = z' \left( \frac{1}{M} \sum_{i=1}^{M} y_i y_i' \right) z
\]
\[
= \frac{1}{M} \sum_{i=1}^{M} (z' y_i) (z' y_i)'
\]
\[
= \frac{1}{M} \sum_{i=1}^{M} (z' y_i)^2 \geq 0.
\]
(B.30)

Thus $Q$ is positive semi-definite. If there are $m$ linearly independent data in $\{y_1, \ldots, y_M\}$, they span the $m$-dimensional space. Since any $z$ is expressed by a linear combination of these data, the strict inequality holds for (B.30).

Since $y_1, \ldots, y_M$ are linearly dependent, at least $m + 1$ samples are necessary so that $Q$ becomes positive definite. (Q.E.D.)

Assuming that $Q$ is positive definite, the following theorem holds.

**Theorem B.3.2.** If $Q$ is positive definite, the mean square weighted distance for $\{y_1, \ldots, y_M\}$ is $m$:
\[
\frac{1}{M} \sum_{i=1}^{M} y_i' Q^{-1} y_i = m.
\]
(B.31)

Proof. Let $P$ be the orthogonal matrix that diagonalizes $Q$. Namely,
\[
PQ P^t = \text{diag}(\lambda_1, \ldots, \lambda_m),
\]
(B.32)

where $\text{diag}$ denotes the diagonal matrix, and $\lambda_1, \ldots, \lambda_m$ are the eigenvalues of $Q$. From (B.32),
\[
Q = P^t \text{diag}(\lambda_1, \ldots, \lambda_m) P,
\]
(B.33)
\[
Q^{-1} = P^t \text{diag}(\lambda_1^{-1}, \ldots, \lambda_m^{-1}) P.
\]
(B.34)

Let
\[
\tilde{y}_i = Py_i.
\]
(B.35)

Then from (B.29) and (B.35), (B.32) becomes
\[
\frac{1}{M} \sum_{i=1}^{M} \tilde{y}_i \tilde{y}_i' = \text{diag}(\lambda_1, \ldots, \lambda_m).
\]
(B.36)

Thus for the diagonal elements of (B.36),
\[
\frac{1}{M} \sum_{i=1}^{M} \tilde{y}_{ik}^2 = \lambda_k \quad \text{for } k = 1, \ldots, m,
\]
(B.37)

where $\tilde{y}_{ik}$ is the $k$th element of $\tilde{y}_i$. From (B.34) and (B.35), the left hand side of (B.31) becomes
\[
\frac{1}{M} \sum_{i=1}^{M} y_i^t Q^{-1} y_i = \frac{1}{M} \sum_{i=1}^{M} \tilde{y}_i^t \text{diag}(\lambda_1^{-1}, \ldots, \lambda_m^{-1}) \tilde{y}_i \\
= \frac{1}{M} \sum_{i=1}^{M} \sum_{k=1}^{m} \lambda_k^{-1} \tilde{y}_{ik}^2.
\] (B.38)

Thus from (B.37) and (B.38), the theorem holds. (Q.E.D.)
References


318 References


# Index

Activation function, 7
AND operator, 10, 252
Associative memory, 7
Average operator, 70, 72, 73, 75, 77, 80, 83, 84, 93, 115, 119, 129, 178, 258

Back-propagation, 34, 216
- error, 34
Back-propagation algorithm, 9, 21, 33, 34, 41, 251
- training, 34
Backward substitution, 31
Batch mode, 35
Bayesian classifier, 301
Bayesian decision rule, 301
Bayesian rule, 301
Bias neuron, 21, 24, 113
Blood cell data, 19, 43, 58, 143, 153, 182, 193, 200, 227, 234, 245, 304
Bootstrap, 240
Breakdown point, 161

Category, 3
Center, 16, 36, 55, 68, 82, 85, 106, 110, 114, 117, 130, 133, 140, 151, 155, 159, 162, 164, 169, 203, 209, 221, 239, 245, 260, 263, 268, 270, 281, 286
- vector, 36, 77, 90, 103
Center-of-gravity defuzzification, 264, 287
Center-of-gravity method, 253, 260
Central moment, 3, 20, 45, 59, 183, 195
Characteristic equation, 308
Cholesky factorization
- symmetric, 92, 226, 235
Class, 3
- plural separated, 25, 27
- singly separated, 25, 27, 36
Class boundary, 5, 63, 93, 99, 105, 162, 209
Class region, 23, 85, 116, 184, 195, 216, 221, 222, 237
- approximation, 149, 218, 221
- overlap, 4, 104
Classification performance, 17
Cluster, 109
Clustering, 14, 81, 109, 115, 118, 148, 153, 290
- batch, 109
- dynamic, 13, 14, 177, 254
- fuzzy c-means, 109
- fuzzy min-max, 115, 143, 146, 147
- minimum volume, 113
- on-line, 109
- overlap resolving, 116
Competitive leaning, 111
Conditional cost, 301
Confidence interval, 49
Convex hull, 85, 90, 149
- dynamic generation method, 85
- initial, 85, 87, 90
- between-class, 212
- diagonal, 80, 91, 148, 204, 275, 276, 278, 281
- within-class, 212
Cross-validation, 197, 240

DDP, see Division of data by pairing
Decision function, 48
Degree of membership, 10–12
- maximum, 12, 63, 81, 116, 179
- negative, 63, 66, 70, 72, 77, 94
Determinant, 212, 307
Deviation ratio, 162, 163
- threshold, 164
Dimensionality
- curse, 251, 253
Direct method, 15, 119, 120, 129, 139, 141, 150, 196
Discriminant analysis, 35, 212, 227
Distance
- mean square weighted, 221, 312
- squared tuned, 120, 130
Division of data by pairing, 241
Dot product space, 47, 55
Dual problem, 51, 54

Eigenvalue, 210, 244, 308, 312
- accumulation, 211, 228
Eigenvector, 209, 244, 308, 309
Empirical error, 49
Entropy, 216
- conditional, 216
Epoch, 34
- maximum number, 35, 41
- tuning, 274
- tuning centers, 138
- tuning slopes, 125
Error function, 35, 216, 288
- cross-entropy, 216
- Hampel, 288
- squared, 216, 288
Euclidean distance, 50, 110, 227, 302
Exception ratio, 5, 220–222, 224–226, 229, 233, 236
- approximated by ellipsoids, 221
- approximated by hyperboxes, 217
- monotonicity, 222
Expansion
- hyperbox, 179
Expansion parameter, 186, 196, 227

Face, 86, 89, 90
Facet, 86, 89, 148
FACG, 264, 273, 277, 278, 280, 288
FALC, 264, 269, 272, 273, 276, 278, 280, 286
FAMI, 264, 275, 280, 281, 283
Feature, 3
- extraction, 4, 5, 157, 209
- optimizing, 205
- optimum, 4, 5
- robust, 6
- scaling, 205
- selection, 4, 5, 215
- selection criterion, 216
- monotonicity, 222
- space, 5, 55, 60
Firing neuron, 111
Forward propagation, 34
Fuzzy classifier
- ellipsoidal regions, 12, 91, 141, 147, 150, 160, 169, 197, 200, 202, 207, 213, 227, 229, 234, 245
- hyperbox regions, 12, 177, 201, 204, 207
- invariance, 207, 208
- learning capability, 12
- polyhedral regions, 12, 90, 141, 148
- pyramidal membership functions, 85, 141, 197, 199, 201, 203, 204, 206
Fuzzy function approximator, 251
- ellipsoidal regions, 289
- learning capability, 251, 254
Fuzzy min-max classifier, 115, 159, 177, 177, 184, 199, 200, 204, 207, 303
- inhibition, 184, 191, 193, 207, 217, 227, 229, 233, 264
Fuzzy region, 10, 12, 15, 81, 129, 217, 264
Fuzzy rule, 12, 81, 82, 179
- ellipsoidal, 90, 95, 100, 257
- hyperbox, 82, 257
- polyhedral, 85, 99
- pyramidal membership function, 82
- tuning, 15
Fuzzy rule generation
- dynamic, 81, 177
- Postclustering, 105
- preclustering, 103
- static, 81
Fuzzy set, 252

Gaussian function, 7, 112, 252
General position, 86, 87
Generalization ability, 9, 17, 47, 50, 91, 106, 125, 154, 161, 183, 192, 203, 205, 219, 239, 252, 287
Generalization region, 48, 54
Global function, 9, 252
Gram-Schmidt orthogonalization, 91
Hidden layer, 9, 21, 31, 35, 212
Hidden neuron, 9, 21, 23, 27, 29, 41, 56, 160, 198, 212, 228, 252
- input, 25
- output, 25, 29, 35, 37
- target value, 36, 39, 213
Hilbert-Schmidt theory, 55
Hiragana data, 20, 45, 59, 145, 154, 182, 194, 200, 214, 246, 305
Ho-Kashyap algorithm, 36, 40, 213, 214
Hopfield neural network, 7
Hyperbox
  – activation, 184, 185, 187, 188, 218
  – contraction, 179, 180, 207
  – expanded inhibition, 186–188, 191, 208, 217, 223
  – expansion, 179, 186, 207
  – inhibition, 184, 185, 219
Hyperplane
  – negative side, 24, 28, 31, 36
  – optimal separating, 47–50
  – positive side, 24, 27, 30, 36
  – separating, 25, 27, 29, 33, 36, 48, 51, 85
  – soft margin, 53
If-part, 10, 179, 252, 264
If-then rule, 11, 252
Indirect method, 15, 16, 119, 139
Input layer, 9, 21, 34, 212
Input neuron, 21, 27, 29, 56
Interclass tuning, 165, 170
  – parameter, 159, 162, 165
Invariance
  – limited scale, 207
  – linear, 4, 206
  – rotation, 206
  – scale, 4, 206, 303
  – translation, 206
Iris data, 19, 41, 56, 97, 142, 148, 151, 181, 192, 198, 227, 228, 303
K-nearest neighbor classifier, 197, 204, 302, 306
Kohonen network, 111, 113
Kuhn-Tucker’s theorem, 52
Lagrange multiplier, 51, 53, 210
Layer
  – hidden, see Hidden layer
  – input, see Input layer
  – number, 9
  – output, see Output layer
Learning, 9
Learning rate, 33, 34, 41, 140, 141, 228
Least-median-of-squares method, 287, 288
Least-squares method with partial data, 288, 292
Leave-one-out method, 18, 240
Linear dependence, 307
Linear independence, 86, 88, 307
Linear regression, 251, 265
Linear transformation, 206
LMS, see Least-median-of-squares method
Local function, 9
Local maximum, 122
Local minimum, 138, 270
LS, see Least-squares method
LSPD, see Least-squares method with partial data
Mackey-Glass differential equation, 274, 275, 285, 292
Mamdani’s method, 252
Manhattan distance, 302
Margin, 48
  – parameter, 36, 41
  – vector, 40, 46
Matrix
  – covariance, see Covariance matrix
  – inverse, 307
  – nonsingular, 307
  – orthogonal, 79, 92, 130, 210, 307
  – singular, 91, 154, 307
  – symmetric, 307
  – transpose, 307
  – triangular, 92, 226
  – unit, 307
Maximum operator, 253
Median, 160, 161, 163, 291
  – absolute deviation, 160, 163
Membership function, 10, 12, 14, 16, 63, 119
  – activation hyperbox, 188
  – bell-shaped, 68, 77, 119, 130, 257
  – inhibition hyperbox, 188
  – multi-dimensional, 69
  – piecewise linear, 66, 120, 141, 178
  – polyhedral pyramidal, 74, 79, 90, 119, 129
  – truncated, 76, 79, 119
  – quadratic, 66, 72, 115, 120, 139, 141, 142, 147, 178, 259
  – rectangular pyramidal, 71, 79, 82, 119, 130, 132, 178, 257
  – truncated, 73, 79, 115, 119, 129, 178, 188, 204
  – trapezoidal, 66, 69, 120
  – triangular, 15, 64, 67, 69, 120, 270
  – tuning, 119
Mercer's condition, 55, 56
Min-max inference, 191
Minimum operator, 70, 73, 78, 79, 84, 90, 94, 119, 130, 188, 253, 258, 259
Model selection, 239
Momentum coefficient, 33, 35, 41, 228
Mutual information, 216

Nearest neighbor classifier, 224, 302
Neighborhood function, 112
Neighborhood relation, 111, 112
Neural network
- four-layer, 23, 27
- multilayer, 9, 21, 24, 31, 63, 113, 216, 251, 254, 280, 283, 287
- three-layer, 9, 21, 23, 25, 27, 33, 35, 56, 197, 212, 227
- two-layer, 25, 33, 111, 209, 212
Neuron, 7
Normal distribution, 69, 91, 302
Normalization, 205
Normalized root-mean-square error, 275, 292
NRMSE, see Normalized root-mean-square error
Numerical data, 19, 42, 56, 142, 151, 181, 192, 198, 227, 230, 304

Off-diagonal element, 91, 206
Online mode, 35, 41
OR operator, 10, 253
Outlier, 49, 126, 159, 161, 165, 169, 175, 287, 290
Output function, 7, 22, 33, 252
Output layer, 9, 21, 34, 40, 213
Output neuron, 9, 21, 23, 24, 26, 33, 39, 111, 212, 252
- output, 39
Overfitting, 17, 47, 61, 142, 151, 203, 285, 303, 304
Overlap
- checking, 180
- degree, 5, 217, 218, 221, 222
Overlapping region, 5, 82, 100, 105, 116, 184, 219

PCA, see Principal component analysis
Perceptron, 33
Piecewise linear function, 7
Polynomial, 55
Positive definite, 91, 210, 308, 312
Positive semi-definite, 91, 210, 308, 308
Postclustering, 14, 81, 103, 105, 107, 142, 145, 152, 156, 170, 254, 263, 264, 286
Precision parameter, 92
Preclustering, 14, 81, 102, 142, 143, 145, 196, 254, 263, 286
Principal axis, 91, 129
Principal component, 211, 233
Principal component analysis, 157, 209, 210, 228
Probability
- a posteriori, 301
- a priori, 301
Product operator, 63, 70, 73, 76, 78, 258, 259
Proposition, 252
Pseudo-inverse, 40, 92, 154, 267, 310
Quadratic form, 308
Quadratic programming, 51
Radial basis function, 55
- neural network, 7, 91, 252
-- classifier, 9
-- robust, 160
Random division, 239, 246
Rank, 86, 92
Recognition rate, 17
Resampling technique, 239
Rotation, 206
Saddle point, 51
Scaling, 206, 211
Scaling problem, 205
Selection
- backward, 225
-- ellipsoid, 148, 227
-- hyperbox, 227
- forward, 224
-- ellipsoid, 227
-- hyperbox, 227
Self-organizing network, 111
Sensitivity, 84, 216
Sensitivity parameter, 178, 181, 188, 191, 228, 303
Separability, 5, 157
- class, 211
- linear, 48
SI, see Solving inequalities
Sigmoid function, 7, 22, 24, 33, 252
Similarity measure, 243
Singleton, 252
Singular value, 92, 154, 155, 157, 203, 214
- decomposition, 40
Solving inequalities, 35, 41, 197, 199
- training, 35
Squared error, 34, 40, 291
- median, 287, 288, 291
- sum, 22, 291
Statistical learning theory, 47
Steepest descent method, 9, 17, 34, 91, 140, 255, 260
Supervised learning, 9, 111
Support vector, 51
Support vector machine, 7, 47, 54, 58, 60, 197
Surface, 12, 74, 129
SVM, see Support vector machine
Synapse, 9

Takagi-Sugeno model, 257, 264, 287, 289
Test data, 4, 12, 17
Then-part, 81, 252, 253, 257
Threshold, 162, 164
Threshold function, 7, 33
Thyroid data, 19, 43, 58, 142, 149, 152, 181, 192, 198, 213, 227, 232, 304
Tolerance of convergence, 41
- hidden neuron output, 39, 41
- output neuron output, 23, 24, 41
Trace, 211

Training, 9
- layer by layer, 35
Training data, 4, 7, 9, 13, 15, 16
Translation, 6, 205–207
Tuning parameter, 68, 75, 77, 83, 91, 93, 95, 104, 107, 120, 124, 141, 148, 151, 161, 167, 201, 203, 258, 270
Two-class problem, 47
Two-dimensional lattice, 111

Underfitting, 81
Universal approximator, 23, 251
Universal classifier, 23, 28
Unsupervised learning, 9, 111, 255
Upper bound, 53, 56

Validation data, 18
VC dimension, 49, 54

Water purification, 279
Weight, 9, 21, 23, 27, 29, 33, 35, 37, 40, 56, 159, 161, 163, 170, 205, 213, 228
- negative, 9
- positive, 9
- vector, 26, 111, 112
Weight decay, 216
Weight function, 163, 164
Weight matrix, 148