Chapter 15
Data Preprocessing

Data preprocessing converts raw data and signals into data representation suitable for application through a sequence of operations. The objectives of data preprocessing include size reduction of the input space, smoother relationships, data normalization, noise reduction, and feature extraction. Several data preprocessing algorithms, such as data values averaging, input space reduction, and data normalization, will be briefly discussed in this chapter. Computer programs for data preprocessing are also provided.

15.1 Introduction

A pattern is an entity to represent an abstract concept or a physical object. It may contain several attributes (features) to characterize an object. Data preprocessing is to remove the irrelevant information and extract key features of the data to simplify a pattern recognition problem without throwing away any important information. It is crucial to the success of fuzzy modeling and neural network processing and when the quantity of available data is a limiting factor. In fact, data preprocessing converts raw data and signals into data representation suitable for application through a sequence of operations. It can simplify the relationship inferred by a model. Though preprocessing is an important role, the development of an effective preprocessing algorithm usually involves a combination of problem-specific knowledge and iterative experiments. In particular, the process is very time consuming and the quality of the preprocessing may vary from case to case.

The objectives of data preprocessing have five folds [3]: size reduction of the input space, smoother relationships, data normalization, noise reduction, and feature extraction.

Size Reduction of the Input Space: Reducing the number of input variables or the size of the input space are a common goal of the preprocessing. The objective is to get a reasonable generalization with a lower dimensionality of the data set.
without losing the most significant relationship of the data. If the input space is large, one may identify the most important input variables and eliminate the insignificant or independent variables by combining several variables as a single variable. This approach can reduce the number of inputs and the input variances, and therefore improve results if there are only limited data.

**Smoothen Relationships:** Another commonly used type of preprocessing is problem transformation. The original problem is transformed into a simpler problem. It means that the associated mappings become smoother. The transformations can be obtained from intuition about the problem.

**Normalization:** For many practical problems, the units used to measure each of the input variables can skew the data and make the range of values along some axes much larger than others. This results in unnecessarily complex relationships by making the nature of the mapping along some dimensions much different from others. This difficulty can be circumvented by normalizing (or scaling) each of the input variables so that the variance of each variable is equal. A large value input can dominate the input effect and influence the model accuracy of the fuzzy learning system or the neural network system. Data scaling depends on the data distribution.

**Noise Reduction:** A sequence of data may involve useful data, noisy data, and inconsistent data. Preprocessing may reduce the noisy and inconsistent data. The data corrupted with noise can be recovered with preprocessing techniques.

**Feature Extraction:** The input data is a pattern in per se. If the key attributes or features characterizing the data can be extracted, the problem encountered can be easily solved. However, feature extractions are usually dependent upon the domain-specific knowledge.

### 15.2 Data Preprocessing Algorithms

For data preprocessing, the original raw data used by the preprocessor is denoted as a **raw input vector**. The transformed data output produced by the preprocessor is termed a **preprocessed input vector or feature vector**. The block diagram of the data preprocessing is shown in Figure 1.

![Figure 1 Block diagram of data preprocessing](image)

In general, problem-specific knowledge and generic dimensionality reduction techniques will be used to construct the preprocessor. A better preprocessing algorithm can be arrived at by exercising several different forms of preprocessing techniques. Several data preprocessing algorithms are discussed below [1-6].
15.2.1 Data Values Averaging

Since the averaging effect can reduce the data sensitivity with respect to fluctuation, a noisy data set can be enhanced by taking average of the data. In a time series analysis, the moving average method can be adopted for filtering the small data fluctuation. Note that root-mean-square error between an average and the true mean will decrease at a rate of \( \frac{1}{\sqrt{N}} \) for noisy data with a standard deviation of \( \sigma \). The averaging result of noisy data is given in Figure 2.

![Figure 2](image-url)

**Figure 2**  Noise reduction averaging (see MATLAB program in Appendix) A. Original noisy time-series data (the mean is 0; the std is 1), B. Time-series data averaged over a window of 5 data points, C. Time-series data averaged over a window of 10 data points.

15.2.2 Input Space Reduction

If the available data is not rich enough compared to the size of the input variables, the input space reduction should be employed. Several algorithms can be applied.
PCA is used to determine a \( m \)-dimensional "most significant" subspace from the \( n \)-dimensional input space. Then the data is projected onto this \( m \)-dimensional subspace. Therefore, the number of input variables can be reduced from \( n \) to \( m \). We shall discuss the PCA Theorem [2] next.

For a given data set (a training set) \( T_{\text{tra}} = \mathbf{x} = \{ \mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N \} \), containing \( N \) \( n \)-dimensional zero-mean randomly generated patterns (i.e., \( \mathbf{x}^i \in \mathbb{R}^n, i = 1, \ldots, N \)) with real-valued elements, let \( \mathbf{R}_x \in \mathbb{R}^{n \times n} \) be a symmetric, real-valued \( n \times n \) covariance matrix. Let the eigenvalues of the covariance matrix \( \mathbf{R}_x \) be arranged in the decreasing order \( \lambda_1 \geq \lambda_2 \geq \cdots \lambda_n \geq 0 \) (with \( \lambda_1 = \lambda_{\text{max}} \)). Assume that the corresponding orthonormal eigenvectors (orthogonal with unit length \( \|e\| = 1 \)) \( \mathbf{e}^1, \mathbf{e}^2, \ldots, \mathbf{e}^n \) consist of the \( n \times n \) orthonormal matrix

\[
\mathbf{E} = [\mathbf{e}^1, \mathbf{e}^2, \ldots, \mathbf{e}^n],
\]

with columns being orthonormal eigenvectors. Then the optimal linear transformation

\[
\mathbf{y} = \mathbf{W}^i \mathbf{x}
\]

transforms the original \( n \)-dimensional patterns \( \mathbf{x} \) into \( m \)-dimensional (\( m \leq n \)) feature patterns by minimizing the mean least square reconstruction error,

\[
J(F, T_{\text{tra}}) = \frac{1}{2} \sum_{i}^N \|\mathbf{x}^i - \mathbf{y}^i\| = \frac{1}{2} \sum_{i}^N \sum_{j}^n (x_j^i - y_j^i)^2.
\]

The \( m \times n \) optimal transformation matrix \( \mathbf{W} \), (under the constraints \( \mathbf{W}^T \mathbf{W} = \mathbf{I} \)), is given by

\[
\mathbf{W} = \begin{bmatrix}
\mathbf{e}^1 \\
\mathbf{e}^2 \\
\ldots \\
\mathbf{e}^m
\end{bmatrix},
\]

where the \( m \) rows are composed of the first \( m \) orthonormal eigenvectors of the original data covariance matrix \( \mathbf{R}_x \).

**Remarks**

(1) The \( n \)-dimension mean vector is

\[
\mu = \mathbb{E}[\mathbf{x}] = [\mathbb{E}[\mathbf{x}^1], \mathbb{E}[\mathbf{x}^2], \ldots, \mathbb{E}[\mathbf{x}^n]]^T
\]

and the square \( n \times n \) dimensional covariance matrix is

\[
\mathbf{R}_x = \sum = \mathbb{E}[(\mathbf{x} - \mu)(\mathbf{x} - \mu)^T]
\]

where \( \mathbb{E}[.\] is the expectation operator, and \( \mu \) is the mean vector of patterns \( \mathbf{x} \).
(2) The square, semipositive definite, symmetric, real-valued covariance matrix $\mathbf{R}_x$ describes the correlations between the elements of pattern vectors (treated as random variables).

(3) The original data patterns are assumed to be zero-mean random vectors

$$\mu = E[x] = 0 .$$

(15.6)

If this condition is not satisfied, the original pattern $x$ can be converted into the zero mean representation by the operation $x - \mu$. For zero mean patterns, the covariance (autocorrelation) matrix is defined as

$$\mathbf{R}_x = \sum = E[xx^T] .$$

(15.7)

(4) Since the exact probability distribution of the patterns is not known, the true values $\mu$ and $\mathbf{R}_x$ (the mean vectors and the covariance matrix) are usually not available. The given data set $T_{tra}$ contains a finite number of $N$ patterns $\{x^1, x^2, \ldots, x^N\}$. Therefore, the mean can be estimated by

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} x^i$$

(15.8)

and the covariance matrix (unbiased estimate) by

$$\hat{\mathbf{R}}_x = \frac{1}{N-1} \sum_{i=1}^{N} (x^i - \hat{\mu})(x^i - \mu)^T$$

(15.9)

based on a given limited sample.

For zero-mean data, the covariance estimate becomes

$$\hat{\mathbf{R}}_x = \frac{1}{N-1} \sum_{i=1}^{N} x^i(x^i)^T = \frac{1}{N-1} \mathbf{x}^T \mathbf{x} ,$$

(15.10)

where $\mathbf{x}$ is a whole $N \times n$ original data pattern matrix.

The $n$ eigenvalues $\lambda_i$ and the corresponding eigenvectors $\mathbf{e}^i$ can be solved by

$$\mathbf{R}_x \mathbf{e}^i = \lambda_i \mathbf{e}^i , i = 1, 2, \ldots, n ,$$

(15.11)

The orthonormal eigenvectors are considered since the covariance matrix $\mathbf{R}_x$ is symmetric and real-valued. In other words, the eigenvectors are orthogonal $(\mathbf{e}^i)^T \mathbf{e}^j = 0$ ($i, j = 1, 2, \ldots, n, i \neq j$) with unit length.

(5) The eigenvalues and corresponding eigenvectors must be in descending order since only the first $m$ dominant eigenvalues will be considered as performing the dimensionality reduction.

© 2001 by CRC Press LLC
Eliminating Correlated Input Variables

The input space reduction can also be achieved by removing highly correlated input variables. The correlation among input variables can be examined through statistical correlation tests (e.g., s-test) and visual inspection. All highly correlated variables should be eliminated except one. Unimportant input variables can also be eliminated.

Combining Noncorrelated Input Variables

Several dependent input variables can be combined to form a single input variable. Therefore, the input space and the complexity of the system modeling can be reduced.

15.2.3 Data Normalization (Data Scaling)

Data normalization can provide a better modeling and avoid numerical problems. Several algorithms can be used to normalize the data.

Min-Max Normalization

Min-max normalization is a linear scaling algorithm. It transforms the original input range into a new data range (typically 0-1). It is given as

$$Y_{\text{new}} = \left( \frac{y_{\text{old}} - \text{min}_1}{\text{max}_1 - \text{min}_1} \right) (\text{max}_2 - \text{min}_2) + \text{min}_2,$$

where $y_{\text{old}}$ is the old value, $y_{\text{new}}$ is the new value, $\text{min}_1$ and $\text{max}_1$ are the minimum and maximum of the original data range, and $\text{min}_2$ and $\text{max}_2$ are the minimum and maximum of the new data range.

Since the min-max normalization is a linear transformation, it can preserve all relationships of the data values exactly, as shown in Figure 3. The two diagrams in Figure 3 resemble to each other except the scaling on y-axis.

Zscore Normalization

In Zscore normalization, the input variable data is converted into zero mean and unit variance. The mean and standard deviation of the input data should be calculated first. The algorithm is shown below

$$y_{\text{new}} = \frac{y_{\text{old}} - \text{mean}}{\text{std}},$$

where $y_{\text{old}}$ is the original value, $y_{\text{new}}$ is the new value, and $\text{mean}$ and $\text{std}$ are the mean and standard deviation of the original data range, respectively.
For the case that the actual minimums and maximums of the input variables are unknown, the Zscore normalization can be used. The algorithm is based on the normalization of the standard deviation of the sample population. The example is shown in Figure 4.

Sigmoidal Normalization

Sigmoidal normalization is a nonlinear transformation. It transforms the input data into the range -1 to 1, using a sigmoid function. Again, the mean and standard deviation of the input data should be calculated first. The linear (or quasi-linear) region of the sigmoid function corresponds to those data points within a standard deviation of the mean, while those outlier points are compressed along the tails of the sigmoid function.
Figure 4 Zscore Normalization (see MATLAB program in Appendix) A. Original unnormalized data B. Data normalized using Zscore normalization

The algorithm is given below:

\[ Y_{new} = \frac{1 - e^{-\alpha}}{1 + e^{-\alpha}}, \]  

(15.14)

where

\[ \alpha = \frac{val - \text{mean}}{\text{std}}. \]  

(15.15)

The outliers of the data points usually have large values. In order to represent those large outlier data, the sigmoidal normalization is an appropriate approach. The data shown in Figure 5 have two large outlier points. Clearly, the sigmoidal normalization can still capture the very large outlier values while mapping the input data to the range -1 to +1.
15.3 Conclusions

Data preprocessing is very important and useful for data modeling. The key issue is to maintain the most significant data while throwing away unimportant data. The overprocessing of the data may result in catastrophe. One should always start with minimal preprocessing and incrementally add more preprocessing while evaluating the result. In particular, there exist many other algorithms for data preprocessing.

15.4 Appendix: Matlab Programs

15.4.1 Example of Noise Reduction Averaging

clear;
y1=randn(1,200); % the mean is 0; the stand deviation is 1
y2=y1;
y3=y2;

© 2001 by CRC Press LLC
for i=1:195
    y2(i+4)=(y1(i)+y1(i+1)+y1(i+2)+y1(i+3)+y1(i+4))/5;
end
for i=1:190
    y3(i+9)=(y1(i)+y1(i+1)+y1(i+2)+y1(i+3)+y1(i+4)+...
    y1(i+5)+y1(i+6)+y1(i+7)+y1(i+8)+y1(i+9))/10;
end
figure(4)
%title('Noise Reduction Averaging')
subplot(221),plot(y1),xlabel('A')
subplot(222),plot(y2),xlabel('B')
subplot(223),plot(y3),xlabel('C')

15.4.2 Example of Min-Max Normalization

clear;
x=5:0.005:6;
y1=exp(x)+25*sin(100*x)+43*randn(1,201)+36*cos(10*x);
min1=min(y1); max1=max(y1);
min2=0; max2=1;
y2=((y1-min1)/(max1-min1))*(max2-min2)+min2;
subplot(121),plot(y1(1:200)),xlabel('A')
subplot(122),plot(y2(1:200)),xlabel('B')

15.4.3 Example of Zscore Normalization

clear;
x=5:0.005:6;
y1=exp(x)+25*sin(100*x)+43*randn(1,201)+36*cos(10*x);
mean1=sum(y1)/length(y1);
std1=sqrt((norm(y1)^2)/length(y1)-mean1);
y3=(y1-mean1)/std1;
subplot(121),plot(y1(1:200)),xlabel('A')
subplot(122),plot(y3(1:200)),xlabel('B')

15.4.4 Example of Sigmoidal Normalization

clear;
x=5:0.005:6;
y1=exp(x)+25*sin(100*x)+43*randn(1,201)+36*cos(10*x);
y(50)=y(50)+986;y(150)=y(150)+1286;
mean1=sum(y1)/length(y1);
std1=sqrt((norm(y1)^2)/length(y1)-mean1);
alpha=(y1-mean1)/std1;
y4=(1-exp(-alpha))/(1+exp(-alpha));
15.4.5 The Definitions of Mean and Standard Deviation

mean:
let $n$ values be $x_1, x_2, \ldots, x_n$, then the mean

$$\overline{X} = \frac{1}{n} (x_1 + x_2 + \cdots + x_n) = \frac{1}{n} \sum_{i=1}^{n} x_i$$

standard deviation:

$$std = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \overline{X})^2} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2 - \overline{X}^2}$$
References


