CHAPTER 3

Research Designs and Methods

This chapter explains the proposed research methodologies for data reduction in supervised learning processes. The first section describes an instance reduction method for regression data and then the method for classification data is explained in the later section.

3.1 Instance reduction for regression problems

A data reduction method is based on the relationship between output and input data. It consists of four steps as illustrated in Figure 3.1. The process starts by splitting a data set into *m* small parts. For each part, a *q*-level linear quantization is applied to quantize the magnitude of the output data in order to partition them into *q* levels. Then, for each quantized group, data separation is performed by a clustering algorithm on their associated input data in order to separate them into *c* clusters. The number of samples in each cluster can be reduced by selecting the center of each cluster as the group representative. As a result, one cluster is represented by only one center point. And finally, the center points of all *c* clusters for all *q* quantized groups and all *m* parts are combined together to create a reduced data set with smaller number of data. The detailed descriptions of each step are as follows.

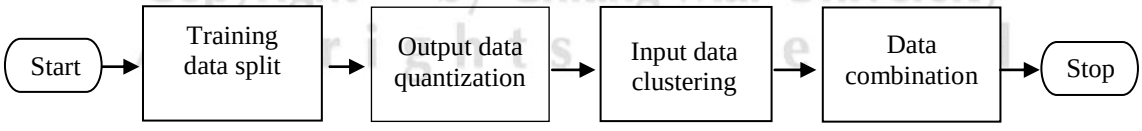

Figure 3.1 Procedure for the input-output clustering method.

3.1.1 Training data split

Working with a large data set requires a huge amount of time and memory resources. One solution to this particular problem is to separate the whole

data set into smaller parts and process them which usually require less time and memory. Therefore, the first step in instance reduction is to split the data into *m* smaller parts so that we can work with them one by one.

Let $\mathbf{T} \in \mathbb{R}^{\ell \times (N+1)}$ be the augmented matrix of a training data set where ℓ is the number of samples and N is the number of features. $\mathbf{T}$ is constructed by augmenting the input data matrix $\mathbf{X} \in \mathbb{R}^{\ell \times N}$ with the corresponding output vector $\mathbf{y} \in \mathbb{R}^{\ell}$, i.e.

$$\mathbf{T} = [\mathbf{X} \quad \mathbf{y}], \tag{3.1}$$

The input matrix $\mathbf{X}$ is the row-wise collection of all ℓ input data points. We denote the *i*-th data point as $\mathbf{x}_i \in \mathbb{R}^N$ corresponding to the output $y_i \in \mathbb{R}$ for $i = 1, 2, ..., \ell$. That is $\mathbf{X}$ and $\mathbf{y}$ are in the following form

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_\ell \end{bmatrix}^T, \tag{3.2}$$

$$\mathbf{y} = \begin{bmatrix} y_1 & y_2 & \dots & y_\ell \end{bmatrix}^T, \tag{3.3}$$

Next, let $\tilde{\mathbf{y}} = \begin{bmatrix} \tilde{y}_1 & \tilde{y}_2 & \dots & \tilde{y}_\ell \end{bmatrix}^T \in \mathbb{R}^{\ell}$ be the vector of sorted $\mathbf{y}$ in an ascending order, i.e., $\tilde{y}_1 \le \tilde{y}_2 \le \ldots \le \tilde{y}_\ell$. Also let $\tilde{\mathbf{X}}$ be the matrix of $\mathbf{X}$ that is reordered according to the vector $\tilde{\mathbf{y}}$. We denote the sorted training data matrix as $\tilde{\mathbf{T}} \in \mathbb{R}^{\ell \times (N+1)}$ where

$$\tilde{\mathbf{T}} = [\tilde{\mathbf{X}} \quad \tilde{\mathbf{y}}]. \tag{3.4}$$

An example of the sorted training samples compared with the original training samples is shown in Figure 3.2 where the vertical and horizontal axes indicate the magnitude of output data and the sample number, respectively.

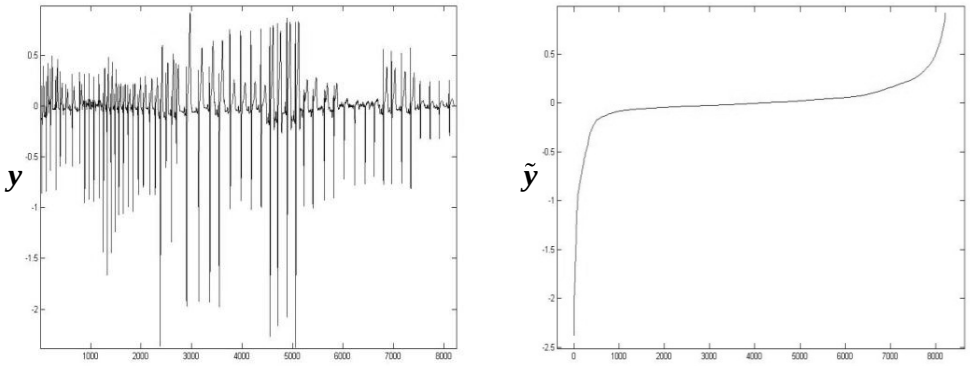

Figure 3.2 Original training data and sorted training data.

Suppose we partition the rows of the sorted training data matrix $\mathbf{T}$ into m small parts. The j-th part of the matrix is denoted as

$$\mathbf{T}_j = [\mathbf{X}_j \quad \tilde{\mathbf{y}}_j] \in \mathbb{R}^{\ell_j \times (N+1)}, \tag{3.5}$$

for $j = 1, 2, \ldots, m$ where ℓ_j is the number of samples in part j. The overall partitioned $\mathbf{T}$ is also represented as

$$\tilde{\mathbf{T}} = \begin{bmatrix} \tilde{\mathbf{T}}_1 \\ \tilde{\mathbf{T}}_2 \\ \vdots \\ \tilde{\mathbf{T}}_m \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{X}}_1 & \tilde{\mathbf{y}}_1 \\ \tilde{\mathbf{X}}_2 & \tilde{\mathbf{y}}_2 \\ \vdots & \vdots \\ \tilde{\mathbf{X}}_m & \tilde{\mathbf{y}}_m \end{bmatrix}. \tag{3.6}$$

The split data must maintain the quality of the whole data set such that the significant information is not altered when the data are later combined. From the histogram of each small part as in Figure 3.3, all histogram are similar then the separation is valid.

In the case that the output of the data set is periodic, it can be split into several parts depending on the sequence of the data where the histogram shape of each part is nearly identical. We also process a non-periodic data set based on its histogram shape. The principle is that it is important to split large data into small parts while also maintain the histogram shape of all small parts to be as similar as possible to the histogram shape of the original data.

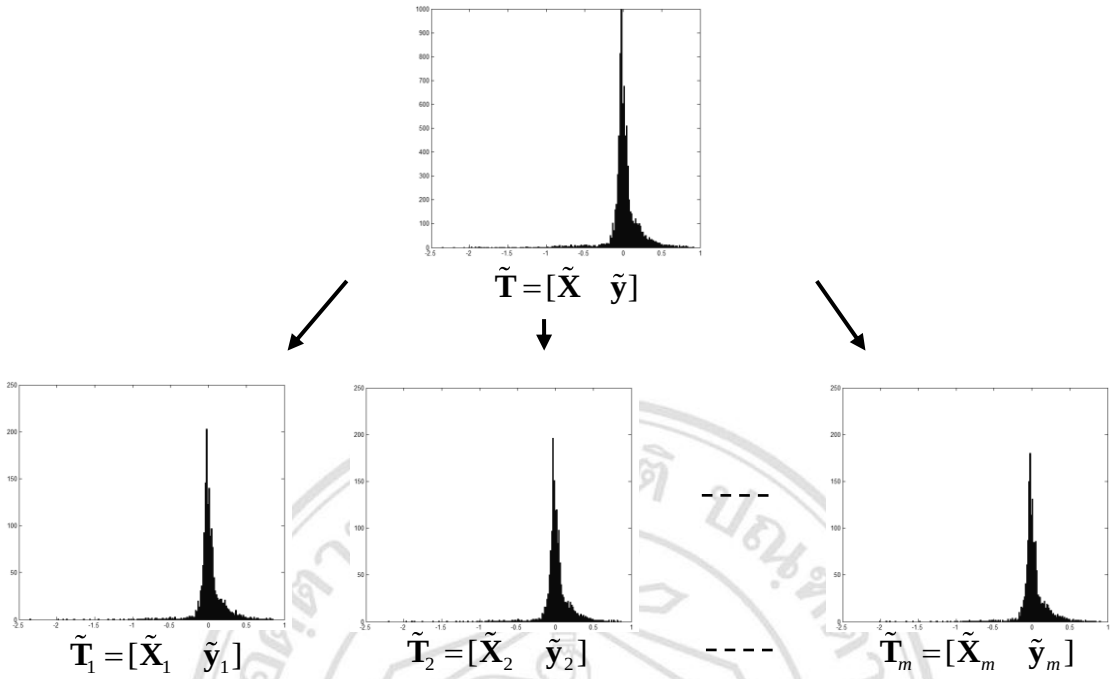

Figure 3.3 (Top) Histogram of original data, (Bottom) Histogram of each small part.

3.1.2 Output data quantization

After separating the training data into *m* parts, the next step is to focus on each part. We apply a *q*-level linear quantization to the output column $\tilde{\mathbf{y}}_j$ in $\tilde{\mathbf{T}}_j$. The quantized version of $\tilde{\mathbf{y}}_j$ is denoted as $\hat{\mathbf{y}}_j \in \mathbb{R}^{\ell_j}$. In other word, $\tilde{\mathbf{T}}_j$ is divided into q_j levels and the new quantized training data matrix is denoted as $\hat{\mathbf{T}}_j \in \mathbb{R}^{\ell_j \times (N+1)}$ with

$$\hat{\mathbf{T}}_j = \begin{bmatrix} \tilde{\mathbf{X}}_j & \hat{\mathbf{y}}_j \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{X}}_{j,1} & \hat{\mathbf{y}}_{j,1} \\ \tilde{\mathbf{X}}_{j,2} & \hat{\mathbf{y}}_{j,2} \\ \vdots & \vdots \\ \tilde{\mathbf{X}}_{j,q_j} & \hat{\mathbf{y}}_{j,q_j} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{T}}_{j,1} \\ \hat{\mathbf{T}}_{j,2} \\ \vdots \\ \hat{\mathbf{T}}_{j,q_j} \end{bmatrix}, \tag{3.7}$$

where $\hat{\mathbf{y}}_{j,k} \in \mathbb{R}^{\ell_{jk}}$ is the output vector of the associated ℓ_{jk} samples whose values are quantized into level k for $k = 1, 2, \ldots, q$. $\tilde{\mathbf{X}}_{j,k} \in \mathbb{R}^{\ell_{jk} \times N}$ is the associated input to $\hat{\mathbf{y}}_{j,k}$ and $\hat{\mathbf{T}}_{j,k} = \begin{bmatrix} \tilde{\mathbf{X}}_{j,k} & \hat{\mathbf{y}}_{j,k} \end{bmatrix}$ is the matrix of input and output.

3.1.3 Input data clustering

For each $\widehat{\mathbf{T}}_{j,k}$, we turn our attention to the input part $\tilde{\mathbf{X}}_{j,k}$ which consists of ℓ_{jk} samples. The goal is to replace these samples with the best c_{jk} representative samples. This can be achieved by applying the fuzzy c-means clustering (FCM) algorithm to cluster the input data since it has a good performance for separating the overlapping data. In order to perform data clustering, the number of clusters c_{jk} can be determined by the maximum score from multi-criteria named TOPSIS. The TOPSIS calculate with five cluster validation indices such as partition index, separation index, davies bouldin index, dunn's index, and SD validity index. We calculate the TOPSIS by adjust the number of clusters from two to the number of data in this quantization level ($c = 2,\ldots,\left|\widehat{\mathbf{y}}_k^{(j)}\right|$). After calculating c_{jk} by TOPSIS, the $\tilde{\mathbf{X}}_{j,k}$ matrix in $\widehat{\mathbf{T}}_{j,k}$ can be clustered as

$$\widehat{\mathbf{T}}_{j,k} = \begin{bmatrix} \tilde{\mathbf{X}}_{j,k} & \widehat{\mathbf{y}}_{j,k} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{X}}_{j,k,1} & \widehat{\mathbf{y}}_{j,k,1} \\ \tilde{\mathbf{X}}_{j,k,2} & \widehat{\mathbf{y}}_{j,k,2} \\ \vdots & \vdots \\ \tilde{\mathbf{X}}_{j,k,c_{jk}} & \widehat{\mathbf{y}}_{j,k,c_{jk}} \end{bmatrix}, \tag{3.8}$$

where $\tilde{\mathbf{X}}_{j,k,l} \in \mathbb{R}^{\ell_{jkl} \times N}$ is the matrix of input samples belonging to cluster l for $l = 1,2,...,c_{jk}$ with the associated output $\widehat{\mathbf{y}}_{j,k,l} \in \mathbb{R}^{\ell_{jkl}}$. Suppose $\tilde{\mathbf{X}}_{j,k,l}$ and $\widehat{\mathbf{y}}_{j,k,l}$ have the center at $\hat{\mathbf{x}}_{j,k,l} \in \mathbb{R}^N$ and $\hat{y}_{j,k,l} \in \mathbb{R}$, respectively. Then the reduced training matrix $\hat{\mathbf{T}}_{j,k}$ can be constructed in the form of

$$\hat{\mathbf{T}}_{j,k} = \begin{bmatrix} \hat{\mathbf{x}}_{j,k,1}^T & \hat{y}_{j,k,1} \\ \hat{\mathbf{x}}_{j,k,2}^T & \hat{y}_{j,k,2} \\ \vdots & \vdots \\ \hat{\mathbf{x}}_{j,k,c_{jk}}^T & \hat{y}_{j,k,c_{jk}} \end{bmatrix}. \tag{3.9}$$

3.1.4 Data combination

After following the first three steps to generate the reduced fractions of the data sets, $\hat{\mathbf{T}}_{j,k}$, for all *j* and *k*, The last step is to combine all of them together into a single training data matrix $\hat{\mathbf{T}}$ so that it is ready to use in further supervised learning processes for regression problem. $\hat{\mathbf{T}}$ is in the form of

$$\hat{\mathbf{T}} = \left[\hat{\mathbf{T}}_{1,1}^T \quad \hat{\mathbf{T}}_{1,2}^T \quad \cdots \quad \hat{\mathbf{T}}_{1,\ell_1}^T \;\middle|\; \hat{\mathbf{T}}_{2,1}^T \quad \hat{\mathbf{T}}_{2,2}^T \quad \cdots \quad \hat{\mathbf{T}}_{2,\ell_2}^T \;\middle|\; \cdots \right.$$
$$\left. \cdots \;\middle|\; \hat{\mathbf{T}}_{m,1}^T \quad \hat{\mathbf{T}}_{m,2}^T \quad \cdots \quad \hat{\mathbf{T}}_{m,\ell_m}^T \right]^T. \qquad (3.10)$$

3.2 Instance reduction for classification problems

A classification problem differs from a regression one due to its discrete nature of the output data. In regression, the domain of the output data is continuous real numbers while, in classification, it is discrete and usually finite. Therefore, in this section, we treat data reductions differently for classification. A goal of our reduction method is to maintain the data points which are near the decision boundary and affect the classification accuracy. Data points far from the decision boundary are also needed to be removed.

Let $\mathcal{T}$ be a training data set containing ℓ samples $\mathbf{x}_i \in \mathbb{R}^N$ for $i = 1, 2, \ldots, \ell$. These samples belong to one of C class. By applying the proposed reduction technique, we hope to reduce the number of samples in $\mathcal{T}$. The reduced version of $\mathcal{T}$ will be called $\mathcal{T}_{\text{Reduced}}$, which has the same value of N and C but with the number of samples $\ell_r \leq \ell$. The parameter ℓ_r represents the number of data points located near the decision boundary and thus, the reduced data maintain classification accuracy similar to the original data set. The procedure of data reduction for classification problems consists of five steps as shown in Figure 3.4. More details on each step are given as follows.

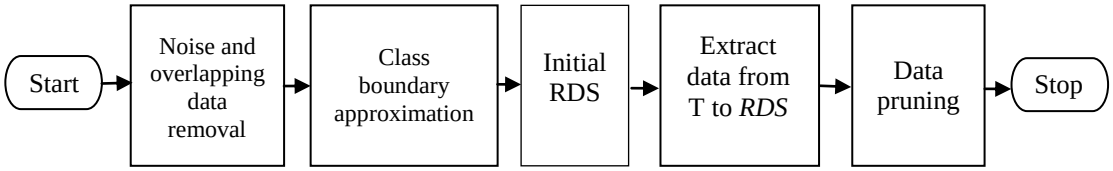

Figure 3.4 Procedure for data reduction for classification problems.

3.2.1 Noise and overlapping data removal

The overlapping data and noises affect the accuracy in determining decision boundaries. One solution to solve this problem is smoothing or removing the data points in the affected areas. In this research, we incorporate a data editing method called 1-NN rules, to remove noise and overlapping data. And after this step, the generalization accuracy will be improved.

3.2.2 Class boundary approximation

For each class i of data in set $\mathcal{T}$ for $i = 1, 2, \ldots, C$, the second step begins with finding three nearest neighbor enemies of each data point, and then these enemies are stored in the i-th approximated class boundary set, denoted as $\mathcal{B}_i$. For example, if $\mathcal{T}$ has two classes, i.e. $C = 2$, then we store all of three nearest neighbor enemies of class 1 in $\mathcal{B}_1$ and the all of three nearest neighbor enemies of class 2 in $\mathcal{B}_2$. According to Figure 3.5, the black dots represent the samples in class 1 and the gray dots represent the samples in class 2. The members of $\mathcal{B}_1$ are represented by the gray points located within the dotted line and indicated by arrows.

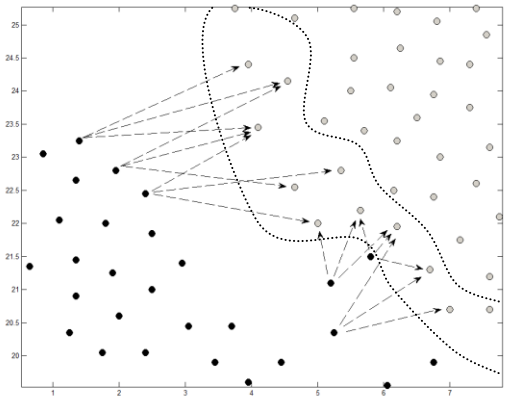

Figure 3.5 Demonstration of class boundary approximation.

3.2.3 Reduced data set ($\mathcal{RS}$) initialization

The third step starts by setting $\mathcal{RS} = \varnothing$, then searches for the data points $\mathbf{x} \in \mathbb{R}^N$ in $\mathcal{T}$ satisfying the condition $\mathbf{x} = \xi(\xi(\mathbf{x}))$ where $\xi(\mathbf{x})$ is the nearest enemy function of $\mathbf{x}$. After that, all of the data point $\mathbf{x}$ are added to $\mathcal{RS}$ which will be used as the initial data set for the next step.

3.2.4 Extracting instances in $\mathcal{T}$ to $\mathcal{RS}$

The forth step starts by removing data in set $\mathcal{T}$ with $\mathcal{RS}$ and the result is kept for the next step. Suppose the resulted set is denoted as $\mathcal{X}_{\text{process}}$ where $\mathcal{X}_{\text{process}} = \mathcal{T} \setminus \mathcal{RS}$. Then, we calculate the distance between every data points in $\mathcal{X}_{\text{process}}$ and the nearest enemies, $\xi(\mathbf{x})$. After that, the data in $\mathcal{X}_{\text{process}}$ whose distance is the shortest are selected, and thus combined to $\mathcal{RS}$. Then, $\mathcal{RS}$ is used as the training set for 1NN classification. Here, we used the $\mathcal{X}_{\text{process}}$ as the testing set. The data points whose classification results are correct will be removed from $\mathcal{X}_{\text{process}}$. We put the data points which missed classification into $\mathcal{X}_{\text{process}}$ until $\mathcal{X}_{\text{process}}$ is empty, i.e. $\left|\mathcal{X}_{\text{process}}\right| = 0$, or the number of correct classification is equal to zero. After that, this process is terminated. The remaining data points in $\mathcal{X}_{\text{process}}$ are combined with $\mathcal{RS}$.

This is simply $\mathcal{X}' = \mathcal{RS} \cup \mathcal{X}_{\text{process}}$ then $\mathcal{RS} = \mathcal{X}'$, otherwise the process is repeated again.

3.2.5 Data pruning

The result from the above step may show some inessential data points or, in orther words, the data points which are located far from the decision boundary. Thus, the pruning step is used to eliminate the data points that does not effect classification accuracy. This process starts by setting a temporary set equal to $\mathcal{RS}$ $\left(\mathcal{T}_{\text{pruning}} = \mathcal{RS}\right)$. Next, we process 1NN classification by using a training set from the class boundary approximation in step 2 and data class in $\mathcal{T}_{\text{pruning}}$ which reduced one data point. The $\mathcal{RS}$ is the testing set. If the number of incorrect classifications is zero, then we remove the next data point from $\mathcal{T}_{\text{pruning}}$, otherwise bring the removed data point back to $\mathcal{T}_{\text{pruning}}$ and repeat this process from the beginning until all data points are tested. Finally, the $\mathcal{T}_{\text{reduced}}$ is equal to $\mathcal{T}_{\text{pruning}}$.

// Remove noise and overlapping data.

$$\mathcal{T}_{1\text{NN}} = \left\{ \mathbf{x} \in \mathcal{T} \middle| \begin{array}{l} 1\text{NN}(\mathbf{x}) \in \text{Correctly classified} \\ 1\text{NN trained by } \mathcal{T} \end{array} \right\}$$

$$\mathcal{T} = \mathcal{T}_{1NN}$$

// Approximation the class boundary.

$$\mathcal{B}_i = \{\mathbf{x} \in \text{ the set of 3 nearest enemies of class } i \mid i = 1, 2, ..., C\}$$

// Initial RDS.

$$\mathcal{RS} = \left\{\mathbf{x} \mid \mathbf{x} \in \mathcal{T} \wedge \mathbf{x} = \xi\left(\xi\left(\mathbf{x}\right)\right)\right\}$$

// Extract instances in $\mathcal{T}$ to $\mathcal{RS}$.

$$\mathcal{X}_{\text{process}} = \mathcal{T} \setminus \mathcal{RS}$$

$$n_{\text{correct}} = \left|\mathcal{X}_{\text{correct}}\right|$$

While $\left(\left|\mathcal{X}_{\text{process}}\right| = 0\right) \vee \left(n_{\text{correct}} = 0\right)$

$$\mathbf{x}_{dmin} = \left\{\mathbf{x} \mid \mathbf{x} \in \mathcal{X}_{\text{Process}} \wedge \min_{\mathbf{x}} d\left(\mathbf{x}, \xi\left(\xi\left(\mathbf{x}\right)\right)\right)\right\}$$

$$\mathcal{X}' = \mathcal{RS} \cup \mathbf{x}_{dmin}$$

Figure 3.6 Proposed data reduction algorithm for classification problems.

$\mathcal{RS} = \mathcal{X}'$

$\mathcal{X}_{\text{process}} = \mathcal{X}_{\text{process}} \setminus \mathbf{x}_{dmin}$

$$\mathcal{X}_{\text{process}} = \left\{ \mathbf{x} \in \mathcal{X}_{\text{process}} \middle| \begin{array}{l} \text{1NN}(\mathbf{x}) \in \text{Missed classified} \\ \text{1NN trained by } \mathcal{RS} \end{array} \right\}$$

$$\mathcal{X}_{\text{correct}} = \left\{ \mathbf{x} \in \mathcal{X}_{\text{process}} \middle| \begin{array}{l} \text{1NN}(\mathbf{x}) \in \text{Correct classified} \\ \text{1NN trained by } \mathcal{RS} \end{array} \right\}$$

$n_{\text{correct}} = \left| \mathcal{X}_{\text{correct}} \right|$

End While

If $\left| \mathcal{X}_{\text{process}} \right| > 0$

$\mathcal{X}' = \mathcal{RS} \cup \mathcal{X}_{\text{process}}$

$\mathcal{RS} = \mathcal{X}'$

End If

// Pruning Data

$\mathcal{T}_{\text{pruning}} = \mathcal{RS}$

$\mathbf{x}_{\text{test}} = \left\{ \mathbf{x} \,|\, \mathbf{x} \in \mathcal{T}_{\text{pruning}} \right\}$

For all $\mathbf{x}_{\text{test}}$ instances in $\mathcal{T}_{\text{pruning}}$

$\mathcal{X}' = \mathcal{T}_{\text{pruning}} \setminus \mathbf{x}_{\text{test}}$

$\mathcal{T}_{\text{pruning}} = \mathcal{X}'$

$$\mathcal{X}_{\text{miss}} = \left\{ \mathbf{x} \in \mathcal{RS} \middle| \begin{array}{l} \text{1NN}(\mathbf{x}) \in \text{Missed classified} \\ \text{1NN trained by } \mathcal{T}_{\text{pruning}} \text{ and } \mathcal{B}_i \end{array} \right\}$$

If $\mathcal{X}_{\text{miss}} > 0$

$\mathcal{X}' = \mathcal{T}_{\text{pruning}} \cup \mathbf{x}_{\text{test}}$

$\mathcal{T}_{\text{pruning}} = \mathcal{X}'$

End If

End For

$\mathcal{T}_{\text{reduced}} = \mathcal{T}_{\text{pruning}}$

Figure 3.6 Proposed data reduction algorithm for classification problems (cont.).