8-3-2006

SVM-Based Negative Data Mining to Binary Classification

Fuhua Jiang

Recommended Citation

Jiang, Fuhua, "SVM-Based Negative Data Mining to Binary Classification." Dissertation, Georgia State University, 2006.
doi: https://doi.org/10.57709/1059418

SVM-BASED NEGATIVE DATA MINING TO BINARY CLASSIFICATION

by

FUHUA JIANG

Under the Direction of A. P. Preethy

ABSTRACT

The properties of training data set such as size, distribution and number of attributes significantly contribute to the generalization error of a learning machine. A not-well-distributed data set is prone to lead to a partial overfitting model. The two approaches proposed in this paper for the binary classification enhance the useful data information by mining negative data. First, error driven compensating hypothesis approach is based on the Support Vector Machines with *1+k* times learning, where the base learning hypothesis is iteratively compensated k times. This approach produces a new hypothesis on the new data set in which, each label is a transformation of the label from the negative data set, further produces the child positive and negative data subsets in subsequent iterations. This procedure refines the model created by the base learning algorithm, creating *k* number of hypotheses over *k* iterations. A predicting method is also proposed to trace the relationships between the negative subsets and testing data set by vector similarity technique. Second, a statistical negative examples learning approach based on theoretical analysis improves the performance of base learning algorithm *learner* by creating one or two additional hypothesis *audit* and *booster* to mine the negative examples output from the learner. The *learner* employs a regular support vector

machine to classify main examples and recognize which examples are negative. The *audit* works on the negative training data created by *learner* to predict whether an instance could be negative. The negative examples are strongly imbalanced. However, boosting learning *booster* is applied when *audit* does not have enough accuracy to judge *learner* correctly. *Booster* works on the training data subset with which *learner* and *audit* do not agree. The classifier for testing is the combination of learner, audit and booster. The *classifier* for testing a specific instance returns the *learner*'s result if *audit* acknowledges *learner*'s result and *learner* agrees with *audit*'s judgment, otherwise returns the *booster*'s result. The error ε of base learning algorithm is proved to decrease from $O(\varepsilon)$ to $O(\varepsilon^2)$.

INDEX WORDS:

Data partition, Data preparation, Support vector machines, Multiple passes learning, Vector similarity, Data classification, Bioinformatics, Machine learning

SVM-BASED NEGATIVE DATA MINING TO BINARY CLASSIFICATION

by

FUHUA JIANG

A Dissertation Submitted in Partial Fulfillment of Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia Stage University

2006

SVM-BASED NEGATIVE DATA MINING TO BINARY CLASSIFICATION

by

FUHUA JIANG

Major Professor: A.P. Preethy
Committee: Yan-Qing Zhang
Yi Pan
Yichuan Zhao

Electronic Version Approved:
Office of Graduate Studies
College of Arts and Sciences
Georgia State University
August 2006

ACKNOWLEGMENTS

Firstly, my specific thanks go to my advisor, Dr. A. P. Preethy and Dr. Yan-qing Zhang, for their kind guidance and precise advisement during the process of my Ph.D. dissertation. The dissertation would not have been possible without their helps.

Secondly, I would like to thank Dr. Yi Pan and Dr. Yichuan Zhao for their well-appreciated support and assistance.

Finally, I want to thank my family and friends for their support and beliefs.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

Support Vector Machine	SVM
Negative Data Driven Compensating Hypotheses Approach	NDDCHA
Instance Based Learning	IBL
k-Nearest Neighbor	KNN
Statistical Negative Examples Learning Approach	SNELA
Independent Identically Distributed	i.i.d.
Vapnik Chervonenkis Dimension	VC Dimension
Receiver Operating Characteristics	ROC
Area Under Curve	AUC
Probably Approximately Correct	PAC

CHAPTER 1

INTRODUCTION

The approach to solve the complex problem without precise model is to learn functionality from the pairs of input and output of examples. The examples are the classification of protein types based on DNA sequence [1], the regression of the surface roughness of parts in manufacturing and so forth. In general, the problem of supervised machine learning is to search a hypothesis $h(x,\alpha)$ from a space of potential hypotheses $\mathcal{H}$ to determine the hypothesis that will best fit underlying function f and any prior knowledge as well, where x is the testing vector and α is the parameters of hypothesis [2].

The learning has training and testing phases. The training is to estimate the parameter α to the hypothesis or model $h(x, \alpha)$. The testing is to use the model to predict the labels of testing data. A hypothesis $h(x, \alpha)$ can be abbreviated by $h(x)$ once α is determined. A hypothesis $h(x)$ can be considered as a ***hyper-surface*** in the n-dimensional input space, where $n=|x|$, by geometric interpretation. For example, a hypothesis of fuzzy controller or the support vector machine (SVM) [1, 3-7] is a hyper-surface, although the hypothesis created by instance-based learning[8, 9] is not in this case. A hypothesis $h(x)$ can be also considered as a ***hyperplane*** in the feature linear space of SVM.

A hypothesis learned from training examples is not perfect to fit underlying function, because the computational errors of approximation and estimation are inevitable to overcome; training data includes noises and does not well distributed. Not well distributed examples means that these examples are not well represented the whole input

space. Some areas may have more examples and other areas may only have a few examples. Therefore some examples are not negative contribution to the hypothesis learned. These negative examples can be mined to improve the accuracy of hypothesis. This chapter describes basic concepts and briefly introduces the main approach proposed.

1.1 Learning Problem Terminology

There is an *instance* vector $\boldsymbol{x}$ from an input space $\boldsymbol{X}$, a response or label $\boldsymbol{y}$ from output space $\boldsymbol{Y}$ and a hypothesis h form hypotheses space $\mathcal{H}$ for a learner L. We have

$$x = (x^{(1)}, x^{(2)},..., x^{(n)}),\ X \subseteq \mathcal{R}^n,\ x \in X,\ x^{(i)} \in \mathcal{R} \tag{1-1}$$

where $\mathcal{R}$ is a set of real numbers, integer n>0 is the size of vector $\boldsymbol{x}$. $\boldsymbol{Y} = \{-1,+1\}$ or $\boldsymbol{Y} \subseteq \mathcal{R}$ is in binary classification, $\boldsymbol{Y} = \{1,2,...,m\}$ is m-class classification, and $\boldsymbol{Y} \subseteq \mathcal{R}$ is in regression. The learned hypothesis h returns a predicting label, $\boldsymbol{y'}=h(\boldsymbol{x})$, of an instance $\boldsymbol{x}$, a real number. In the binary classification, if $h(x)$ returns a confidence value then $y'>0$, means y' is in the class +1 whereas $y'<0$ means in the class -1.

A training data set S is a collection of training examples or observations given by $z_i=(x_i,y_i)$. It is denoted by

$$S = \{(x_1, y_1), (x_2, y_2),..(x_l, y_l)\}\ \ i = 1..l \tag{1-2}$$

where $\ell = |S|$ is the size of the training set. In this paper the label of binary classification $\boldsymbol{Y}$ is extended to $\boldsymbol{Y} \subseteq \mathcal{R}$, the final output of binary classification is the sign of label.

There exists a true functional relationship or underlying function f: $\boldsymbol{X} \subseteq \mathcal{R}^n \rightarrow \boldsymbol{Y}$, which is often based on the knowledge of the essential mechanism. These types of models are called mechanistic models. A hypothesis h is an approximation to the underlying functional relationship f between variables of interest. The problem for the

learner L is to learn an unknown target function $h: \boldsymbol{X} \rightarrow \boldsymbol{Y}$ drawn from $\mathcal{H}$ and output a maximum likelihood hypothesis.

1.2 Evaluating the Performance of Binary Classification

In a binary classification, examples of class +1 and class -1 are usually said to be positives and negatives respectively. Traditionally, three metrics, named accuracy, sensitivity and specificity, are used to evaluate the performance of hypothesis based on the confusion matrix in Table 1.1:

$$accuracy = \frac{TN + TP}{TN + FN + FP + TP} \tag{1-3}$$

$$sensitivity = \frac{TP}{TP + FN} \tag{1-4}$$

$$specificity = \frac{TN}{FP + TN} \tag{1-5}$$

Sensitivity is the proportion of true positives and specificity is the proportion of true negatives. The *predictive value positive* and *predictive value negative* is evaluated accuracies of the positive and negative examples respectively.

$$predictive\ value\ positive = \frac{TP}{TP + FP} \tag{1-6}$$

$$predictive\ value\ negative = \frac{TN}{TN + FN} \tag{1-7}$$

The sum of FP and FN is the number of misclassification examples on the unseen testing dataset whereas the sum of TP and TN is the number of correctly classified examples. ***Predictive positives*** are consisted of true positives (TP) and true negatives (TN). ***Predictive negatives*** include false positives (FP) and false negatives (FN).

TABLE 1.1 Confusion matrix

		Real		
		Positive	Negative	
Test	Positive	True Positive TP	False Positive FP	TP + FP
	Negative	False Negative FN	True Negative TN	FN + TN
		TP + FN	FP + TN	

The accuracy ρ is usually used as metric to evaluate whether a model is good or not in the binary classification. Training accuracy ρ_t and the testing accuracy ρ_p are used to evaluate the performance of learning machine. Sometimes a high ρ_t results in a high ρ_p, otherwise a high ρ_t results in a low ρ_p which is called ***overfitting***. The testing accuracy is a measure of generalization capacity of a model. If there exist two models for the same learning problem with the same training accuracy, how can we determine which model has a higher probability of performance without predicting. The support vector machine (SVM) uses maximal margin as the metric. A hypothesis *h(x)* could be considered as a predicting ***confidence*** of an instance *x*. The relationship between confidence and label is shown on Figiure1.1, which is an example predicting task with 30 testing instances. Although some instances are labeled as class +1, their confidences are quite different. It can be considered that a predictive data with high confidence is true in high probability.

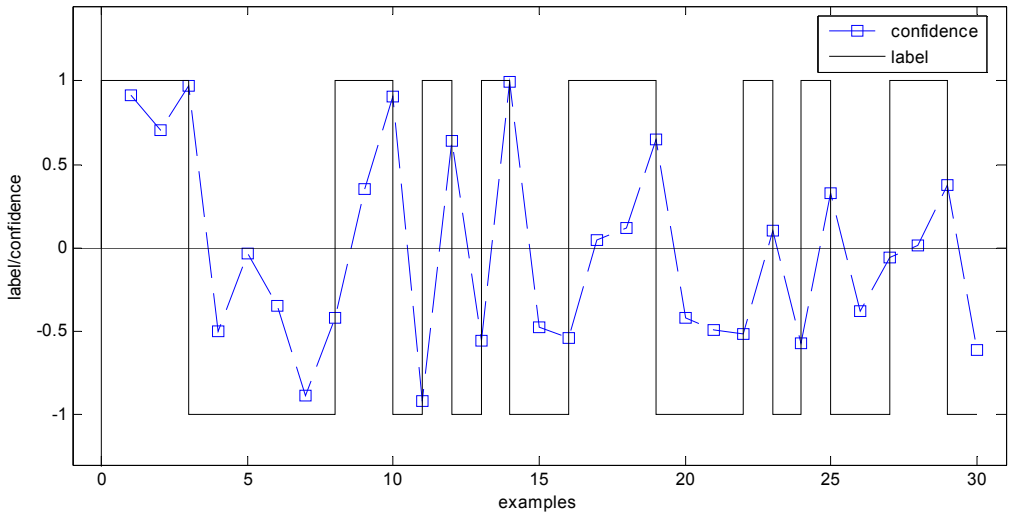

Figure 1.1 The relationship of label and predicting confidence

A metric ***average residual*** (AR) to evaluate the model is shown on (1-8) based on the generalization theory [5]

$$AR = \frac{1}{l}\sum_{i=1}^{l} y_i h(x_i) \tag{1-8}$$

where l is the size of training data set and y_i is the label of input vector x_i. The high AR means high training accuracy. AR could be negative if there are many misclassified examples. The hyper-surface $h(x)$ separates the hyperspace into two sides in the binary classification $y = \{-1,+1\}$. In SVM, the predictive value of $y'=h(x)$ is proportional to the geometric distance from point x to the hyperplane in the feature space. y' is the geometric distance if maximal margin is normalized to 2. Then the new metric is the average distance to hyper-surface as shown on Figure 1.2. One side $h(x)>0$ is in the class +1 and the other side $h(x)<0$ is in the class -1. The hyperplane $h(x)=0$ is the ***separator***. The hypothesis $h(x)$ becomes a measure of performances to separate vector x.

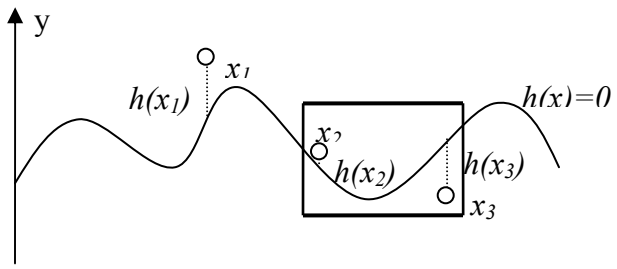

Figure 1.2 Model $h(x)$ is a hyper-surface. Instance x_1 is well-separated; instance x_2 is not well-separated and instance x_3 is misclassified where their labels y_1, y_2 and $y_3>0$.

Furthermore, when the training data are strongly imbalanced, ***accuracy*** may mislead because the all positive or all negative classifiers may have a very good accuracy. The ***Receiver Operating Characteristics*** (*ROC)* curve has been introduced by the signal detection theory to evaluate the capability of a human operator of distinguishing signal and noise[10]. *ROC* analysis is now being acknowledged as a practical tool to evaluate classifiers of imbalanced data, even when the prior distribution of the classes is not known[11]. *ROC* curve is a two-dimensional measure of classification performance. It can be understood as a plot of the probability of correctly classifying the positive examples against the rate of incorrectly classifying negative examples as shown below.

The *AUC* is defined at the area under an *ROC* curve. Processing the *AUC* would need the computation of an integral in the continuous case. The following equation is the *AUC* on discrete case such as in the classification.

$$AUC(h) = \frac{1}{l^+ l^-} \sum_{i=1}^{l^+} \sum_{j=1}^{l^-} 1_{h(x_i^+) > h(x_j^-)} \tag{1-9}$$

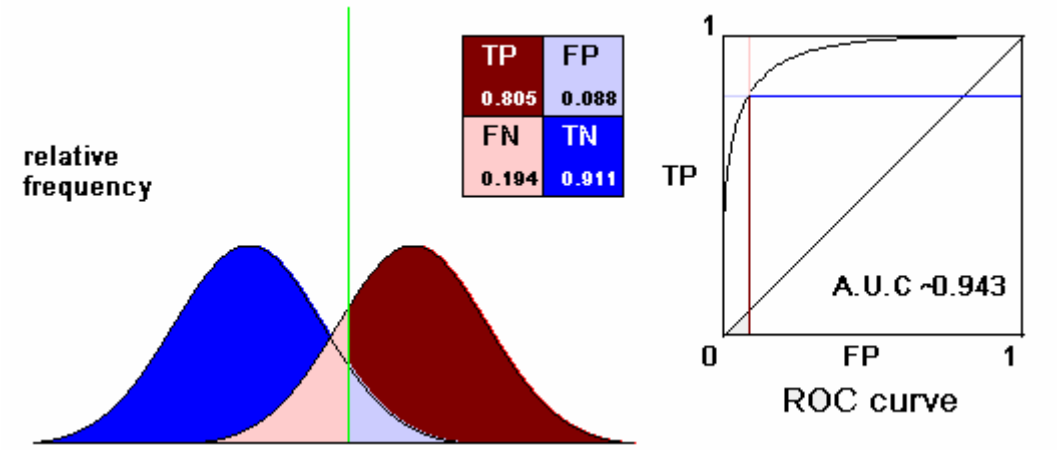

Figure 1.3 An example of *ROC* Curve for a given hypothesis[12], y-axis is *sensitivity* and x-axis is the 1-*specificity*. The diagonal line from (0,0) to (1,1) is drawn for random classifier as a reference.

where *h(x)* is the hypothesis. $\boldsymbol{x}^+$ and $\boldsymbol{x}^-$ respectively denote the class +1 and -1 examples and l+ and l− are respectively the numbers of class +1 and -1 examples and 1_π is defined to be 1 if the predicate π holds and 0 otherwise. AUC value is the probability $P(Y_1>Y_2)$ where Y_1 is the random variable corresponding to the distribution of the outputs for the positives and Y_2 is the one corresponding to the negatives[13]. The average of *AUC* is monotonically increasing as the accuracy of hypothesis, but the standard deviation for imbalanced distributions is grown[14]. Therefore AUC is a better metric than accuracy in the case of imbalanced examples distribution. Alain Rakotomamonjy proposed an AUC maximization algorithm and show that under certain conditions 2-norm soft margin SVMs can maximize AUC[15]. AUC is not as an optimization objective in this dissertation but as an evaluation metric. Genuine SVMs assume that misclassification costs are equal for both classes of binary classification. So SVMs are not suitable in detecting a small size class in the imbalanced data set. For example, it is very hard to

detect the negatives when the size of negative is far less than that of positives. Thus, ROC curve approach is of interest because it reflects both true positive and false positive information. However, if training examples are separable, any hypotheses in the version space will maximize AUC.

In order not to confuse the terminologies of positives and negatives in the following chapters and sections, positives and negatives are called class +1 and class -1 examples respectively because labels belong to $\{-1,+1\}$. That is very convenient to extend binary classification into multiple classifications. For example, 4-category classifications problem have class 1, 2, 3, 4 examples because labels belong to $\{1,2,3,4\}$.

1.3 Relatively Performance Evaluation

Most literatures evaluated a hypothesis by using absolutely evaluation hypothesis method (AEHM). The examples include accuracy, AUC, least square sum. All AEHMs have to assume the data distribution is i.i.d. These methods are necessary in the evaluation of generalization capacity in the unseen data. When the size of data set is small, AEHMs is not meaningful because a small size of examples is not capable to show the whole picture of data distribution. Then a relatively evaluation hypothesis method (REHM) is introduced.

To show how REHMs works, assume a data set $\boldsymbol{D}$ includes l examples and n-fold cross-validation are used. We have n pairs of training and testing data set. The size of training data and testing examples are listed below:

$$\left|S_i\right| = \frac{n-1}{n} \times \left|D\right|,\ i = 1..n \tag{1-10}$$

$$\left|T_i\right| = \frac{1}{n} \times \left|D\right|,\ i = 1..n \tag{1-11}$$

Firstly, data set ***D*** is trained and hypothesis $h_D(x)$ is gotten. We get performance value η_D by testing data set ***D*** using $h_D(x)$. The performance value could be accuracy, AUC and etc. Secondly, data set $\boldsymbol{S_i}$ is trained and hypothesis $h_i(x)$ is gotten. We get performance value η_i by testing data set $\boldsymbol{T_i}$ using $h_i(x)$. Let the average of η_i be η_S. Then REHM is defined below:

$$REHM = \frac{\eta_S}{\eta_D} \tag{1-12}$$

The performance value η_D is the best value of given hypothesis in terms of data set D. Any other η_i cannot be better than η_D because the testing examples are included in the training data in the AEHM. Thereby REHM is a value less or equal than 1.0. To the separatable examples, REHM is exactly the same as AEHM.

1.4 Challenges of Machine Learning

The goal of learning is to have high testing or predicting accuracy rather than training accuracy. The underlying function *f* of the practical problem is unknown and even hard to be described. What can be known in the problem are the training data set *S,* and the limit and not full prior knowledge of the problem. The general purpose learning algorithm does not even take advantage of domain knowledge, such as statistical learning methods. They only consider or assume the distribution of data where all data are drawn from this distribution possibility, although such assumption is not realistic. The underlying function *f* of problem is in the *target space TS*; the model is a hypothesis *h* from the *hypotheses space H*. Therefore, three types of error are inevitable in the process of machine learning [16]. The first is the *approximation error* from the number of

hypotheses in the hypotheses space less than that of target space, $H < TS$. The underlying function f may be beyond the hypothesis space if $h \neq f$. The second is the *estimation error* for a training algorithm from selecting a non-optimal model or hypothesis due to the technique of computation, for example, the back propagation algorithm cannot produce the optimal solution because of local minima problem. The last one is the *generalization error* jointly from the approximation and estimation error.

In addition to those, the properties of data such as a small size, dirty, imbalanced or not well-distributed training data set, which means that the training data set does not well reflect the real problem, contribute to the generalization error. Hence, the generalization error is the composite error from all aspects. In the supervised machine learning, the hypotheses space is selected by human, and the number of types of hypotheses spaces that are available to human is limited. The hypotheses space in the artificial neural network (ANN) is the topology of network and the approximation functions, such as sigmoid functions, in the neuron[17-20]. In the support vector machine (SVM), the hypotheses spaces could be regarded as the kernel functions such as polynomial kernel; radial basis function kernel (RBF) and etc. Therefore the approximation error cannot be reduced once the hypotheses space is chosen. How to choose a suitable hypothesis space depends on human's a priori knowledge of identifying characteristics of a real learning problem and the learning accuracy.

It is known from above discussion, the performance of testing or capacity of generalization relies on the shape of the hyper-surface or model. Sometimes the hypotheses spaces are larger enough than the target spaces; the model is still prone to overfitting due to not well distributed training data as shown on Figure 1.4**.** Not well

distributed data are scattered on the input space un-uniformly. One challenge is there exists a general method to compensate a hypothesis and let it fall in hypothesis space to reduce approximation error. Another is how to reduce estimation error. SVMs are proved as global optimization method once kernel is chosen.

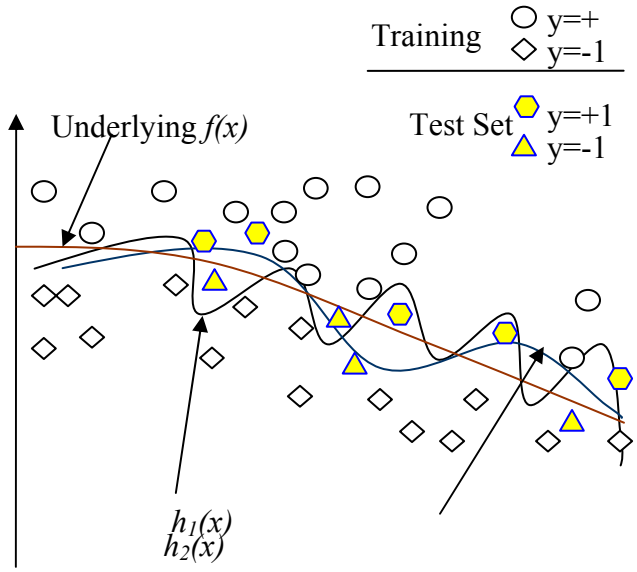

Figure 1.4 Comparing to classifier $h_1(x)$ and $h_2(x)$ of the binary classification, the model with high degree is prone to overfitting, where $f(x)$ is the underlying function.

1.5 Negative Data Mining

According to traditional Chinese philosophy, Yin and Yang are the two primal cosmic principles of the universe. Yin is the passive, female principle while Yang is the active, masculine principle. The best state for everything in the universe is a state of harmony represented by a balance of Yin and Yang. True harmony requires Yang to be dominant. It's just the natural phenomena. As show on Figure 1.5 Yin-Yang symbol, when Yin and Yang are in harmony with one another, they are two halves of the circle,

one dark and the other light. The small circle within each half shows that the part of each opposite is always found within the other. They are not really opposites at all. Yin and Yang is interrelated. Partial Yin is inside of Yang whereas partial Yang is inside of Yin. Yin and Yang should be respected to an equal extent.

Figure 1.5 The Yin-Yang symbol.

The target space could be considered as a universe in the Yin-Yang theory. To a specified hypothesis $h \in H$, all examples in the universe *TS* are divided into two primal groups positive and negative data, which matches Yang and Yin. The positive data is the subset of all correctly classified examples where negative data is the rest. An example could be positive or negative. The negative data does not mean the data is wrong or corrupt. What negative data can be known is that a hypothesis cannot make it well-separated. Negative data strongly depends on the hypothesis. Whether an example is positive or negative is relative. To a specific example, hypothesis A classifies it to be negative while hypothesis B may classify it to be positive. Furthermore, even for the same hypothesis, an example probably belongs to positive or negative in terms of the different parameters α of a hypothesis $h(x)=f(x, \alpha)$.

Yin-Yang Theory claims that Yin and Yang together form a universe. Yin and Yang are opposite group, and each can be always found in the opposite within the other. This is the foundation philosophy of negative data mining. The Yin-Yang theory indicates that negative data contains the positive information. The more information, the

higher accuracy of learn machine can be gotten. By mining negative data, the accuracy of machine learning will be enhanced[21].

There are two ways of improving the performance of classification. One improves the learning algorithm or method to reduce approximation and estimation error by choosing a suitable learning algorithm or invent a new algorithm. Here we only consider the other way to mine training data to increase the accuracy of hypothesis such as boosting and bagging[22-25].

1.6 Introduction to Negative Data Driven Compensating Hypothesis Approach (NDDCHA)

For a specific model to a specific learning problem, there are several ways to improve the model or hypothesis if misclassified examples exist. The first way makes hypothesis space larger than the target space. The second is to reduce estimation error. The third is to make the size of examples large. The last is the training data mining in which a sequence of learning algorithms takes advantage of the distribution of data to create a combination of algorithm. A good example is the SVM ensemble powered by the bagging and boosting approach [22-25].

In bagging, each base learning algorithm is trained independently by using randomly chosen training examples via a bootstrap technique. In boosting, the base learning algorithm is trained using training data examples chosen according to the examples' distribution. The boosting approach calls a weak base learning algorithm more than one time. The base learning algorithm could be any algorithms such as ANN or SVM. Each time weak algorithm is fed with a different subset of the training examples

and generates a new weak prediction rule. After many rounds, the boosting algorithm combines these weak rules into a single prediction rule to produce more accurate rule. The problems of how each distribution should be chosen in each round, and how the weak rules should be combined into a single rule is to maintain set of weights over the training set. Therefore, the boosting approach is combining a series of hyper-surfaces into a single hyper-surface where each hyper-surface is independent. Chang [26] proposed a boosting SVM classifier with logistic regression for imbalanced training data by using clustering technique. Kim *et al.* [27] proposed bagging and boosting SVM approach and tested majority voting, least squares estimation based weighting, and double-layer hierarchical combination aggregating methods. Vapnik in his book [5] gave a detail explanation on how to use SVM ensemble powered by Schapire's AdaBoost algorithm [25]. The main drawbacks of bagging and boosting are time consuming and the performance largely depends on the training data of probability distribution and aggregation methods. The Boosting could be considered as a negative mining algorithm which emphasizes learning on misclassified data or negative data.

The negative data driven compensating hypothesis approach (NDDCHA) driven by the negative data information is proposed in this paper. This approach looks similar to the SVM ensemble, which is learning technique where multiple SVMs are trained to solve the same problem [5, 28-30]. The SVM ensemble is to generate a sequence of SVMs by using Bagging or Boosting approaches and then combining their predicting. The difference is that the ensemble approach is combining the results of SVMs and each SVM is independent, while NDDCHA is compensating the labels of base SVM by a sequence of patching SVMs and making training examples well separated by using AR metric

(1-8). The NDDCHA works on the negative data and the size of negative training data is reduced in each pass and therefore it converges quickly in the rate of exponentially. In our practice, the number of passes is not greater than 3.

The main idea of the approach proposed here is to maximize the $yh(x)$ for every example x in the training phase by using a series of hypotheses $h^{(i)}(x)$ $i=0...k,$ whereas testing data find appropriate $h^{(i)}(x)$ by using vector similarity technique to predict the example in the predicting phase. The approach is to improve the capacity of generalization and reduce the approximation error by extending the traditional learning method like SVMs in two aspects. The first is to compensate hypothesis by making use of the examples from training data S with high training error due to $\mathscr{H}<TS$ and not well-separated examples. The second is data cleaning and data enhancing by utilizing the negative data which has high predicting error or not well-separated in the phase of training and testing.

The rest of this paper is organized as follows. In Chapter 2, the related work including boosting, k-nearest neighbor algorithm (KNN) and SVM are introduced; the concepts of NDDCHA and principle of generalization theory are also introduced. In the Chapter 3, the concept of negative examples is introduced. In Chapter 4, the algorithm of NDDCHA is studied in detail. In Chapter 5, the statistical negative example learning is studied. Finally in Chapter 6, the main contribution of this paper is summarized.

CHAPTER 2

RELATED WORK

There are two general strategies in improving an algorithm. One is modification of algorithm structure, and the other is modification training data. The first one includes changing the objective function of optimization, for examples, approach of support vector machine to decision tree[31]. The second one includes the bagging and boosting. The approaches in this dissertation focus on the second strategy. The related works are briefly introduced in this chapter, including boosting and bagging, locally weighted regression, kernel and support vector machines.

2.1 Boosting and Bagging

Breimans's bagging[24] and Freund and Schapire's boosting[23, 25, 32] [33] both form a set of classifiers or hypotheses that are combined by voting. Bagging generates replicated bootstrap examples of the data and boosting adjusts the weights of training examples. Two approaches are based on theoretical analyses of the behavior of the composite classifiers. Bagging can be applied for the situation where a small agitating the training data set will result in significant changes in the classifier built. Boosting strengthens the base or weak learn algorithm.

Boosting based on PAC learning[34] causes the learner to focus on those misclassified examples then it generates new classifiers by adjusting the weight of examples. The high weight of example indicates the high influence on the classifier

constructed. Boosting learns examples many times. In every time of learning, the weight of examples is adjusted to reflect the accuracy of classifier built on previous iteration. Obviously, the misclassified example will be assigned high weight on the next iteration. In the testing phase multiple classifiers are combined by majority voting strategy to form a composite classifier. Boosting uses different voting strength in terms of the accuracy of component classifier in the training phase. How to determine the weight of examples is key point of boosting. One implementation of boosting is AdaBoost[33] shown in the following:

Given: Training data set S defined on the (1-2).

Initialize the weight distribution: $D_1 = \frac{1}{l}$

For t=1 to T

Train weak learner using distribution D_t

Get week hypothesis $h_t : X \to \{-1,+1\}$ with error $\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$

Choose $a_t = \frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t})$

Update: $D_{t+1}(i) = \frac{D_i(i)}{Z_t}\begin{cases} e^{-a_t} & if\ h_t(x_i) = y_i \\ e^{a_t} & if\ h_t(x_i) \neq y_i \end{cases} = \frac{D_t(i)\exp(-a_t y_i h_t(x_i))}{Z_t}$ where Z_t is a normalization factor

The output the composite classifier or hypothesis:

$$H(x) = sign(\sum_{t=1}^{T} a_t h_t(x))$$

Determining the number of T is stop criterion that uses two ways:

- if $\varepsilon_t > 0.5$

- $h_t(x)$ correctly classified all examples

Other boosting implementations include Brownboost[35] and Logitboost[36]. If weight zero was assigned to the correctly classified examples in the extremely condition, then the next iteration of learning will only use the negative data. The side effect in this case is that the size of next generation training data may be imbalanced. The assumption of bagging and boosting is that a small change of examples on a given distribution will cause significant changes on the classifier built. As long as the accuracy of every component classifier is greater than 50%, Freund and Schapire proved that accuracy of the composite classifier on the given training data set increases in the rate of exponentially quickly as the number of iterations increasing. However, the composite classifier cannot guarantee the generalization performance. And boosting also produces severe degradation on some datasets [37]. Most existing boosting algorithms are limited to combine only a finite number of hypotheses, and the generated ensemble is usually sparse. Lin *et al.* proposed infinite ensembles may surpass finite and/or sparse ensembles in learning performance and robustness[38]. Bagging and boosting requires that the learning system should not be stable, and then the small changes to the training examples should have considerable changes in the hypothesis[37, 39].

2.2 Kernel Methods

Kernel methods[40] provide an alternative solution to non-linear system by projecting the data into a high dimensional feature space where data can be solved by linear system. The successful applications of kernel based algorithm have been found in different areas, for examples, pattern recognition[41, 42], time series prediction[43], text

categorization[44], gene expression profile analysis[45], DNA and protein analysis[46] and etc.

Suppose a vector x in the *input space* $\boldsymbol{X}$ projects into $\phi(x)$ in the *feature space* $\mathcal{F}$.

$$x = (x_1, x_2, ..., x_n) \mapsto \phi(x) = (\phi_1(x_1), \phi_2(x_2), ..., \phi_m(x_m)),$$

$$x \in X, \phi(x) \in F = \{\phi(x) \mid x \in X\} \tag{2-1}$$

The n-dimensional vector x has n coordinates in the input space, the coordinates are called *attributes*. And the coordinates in the feature space is called *features*. If $m<n$, this is known as dimensionality reduction. If $m>>n$, this is known as curse of dimensionality. Using to large number of features may lead to the overfitting problem[1]. In the mean time, the large number of features increases the computational cost.

A kernel is a function K, such that $\forall x, z \in X$

$$K(x, z) = \langle \phi(x) \cdot \phi(z) \rangle \tag{2-2}$$

where function $\phi(x)$ is a non-linear mapping function from $\boldsymbol{X}$ to an inner product feature space $\mathcal{F}$. A kernel function calculates an inner product which expresses a degree of similarity of two vectors. Kernel function is symmetric, but not all of symmetric functions over $X \times X$ are kernels. Kernel function has to be positive definite according to Mercer's theorem[1]. Many researches extended kernel function in practical application. Ong *et al*. proposed methods to learn non-positive kernel [47], which has been promising in empirical applications. Kernel function plays a key role in determining the performance of SVM. S. Armari and S. Wu proposed a method of modifying a kernel function to improve the performance of SVM based on the Riemannian geometrical structure[48]. Srivastava *et al*. proposed a method of mixture density Mercer Kernels

which learn kernel directly from data[49]. The following are some most common used nonlinear kernel functions

Polynomial Kernel $$K(x,z) = (a * \langle x | z \rangle + c)^d \tag{2-3}$$

Radial Basis Kernel $$K(x,z) = \exp(-\gamma \| x - z \|^2) \tag{2-4}$$

Sigmoid Kernel $$K(x,z) = \tanh(\gamma * \langle x | z \rangle + c) \tag{2-5}$$

2.2.1 Distance in the Feature Space

Kernel expresses domain knowledge about the pattern being constructed, encoded as a similarity metric between two vectors [50, 51]. Let K be a kernel over $X \times X$, then a distance d of two vectors $\boldsymbol{x}$ and $\boldsymbol{z}$ in the feature space defined as [52]:

$$d(x,z) = \|\phi(x) - \phi(z)\| = \sqrt{K(x,x) - 2K(x,z) + K(z,z)} \tag{2-6}$$

Radial basis kernel (2-4) function has a close relation between kernel and distance. $\| x - z \|^2$ can be substituted by any metric that calculates the distance between $\boldsymbol{x}$ and $\boldsymbol{z}$. The angle θ between two vectors $\boldsymbol{x}$ and $\boldsymbol{z}$ in the feature space satisfies

$$\cos\theta = \frac{\langle \phi(x) | \phi(z) \rangle}{\|\phi(x)\| \cdot \|\phi(z)\|} = \frac{K(x,z)}{\sqrt{K(x,x) \cdot K(z,z)}}, \ \theta \in [0, 2\pi) \tag{2-7}$$

Suppose $\theta 1$ is the angle between vectors $\boldsymbol{x}1$ and $\boldsymbol{z}$ and $\theta 2$ is the angle between vectors $\boldsymbol{x}2$ and $\boldsymbol{z}$. The follow equation can be gotten:

$$\eta = \frac{\cos\theta_2}{\cos\theta_1} = \frac{K(x_2,z)\sqrt{K(x_1,x_1)K(z,z)}}{K(x_1,z)\sqrt{K(x_2,x_2)K(z,z)}} = \frac{K(x_2,z)\sqrt{K(x_1,x_1)}}{K(x_1,z)\sqrt{K(x_2,x_2)}} \tag{2-8}$$

Theorem 2.1 *Given that the feature space defined by kernel function* $K(x,z) = \langle \phi(x) | \phi(z) \rangle$ *is linear space, and then the hypothesis* $h(x)$ *in the feature space is*

linear which can be expressed $h(x) = \langle \phi(x) \cdot w \rangle + b$ *, where* w *is a constant vector to define hyperplane,* b *is a bias.*

- *The vector* $\boldsymbol{x}_1$ *is closer to vector* $\boldsymbol{z}$ *than vector* $\boldsymbol{x}_2$, $\|\phi(x_1) - \phi(z)\| \le \|\phi(x_2) - \phi(z)\|$
- $|\cos\theta_1| \le |\cos\theta_2|$, *where angles* $\theta 1$, $\theta_2 \in [0, \pi]$ *defined on the equation (2-7).* $\theta 1$ *is the angles between* $\phi(x_1) - \phi(z)$ *and* $\boldsymbol{w}$*; and* $\theta 2$ *is the angles between* $\phi(x_2) - \phi(z)$ *and* $\boldsymbol{w}$*, as shown below.*

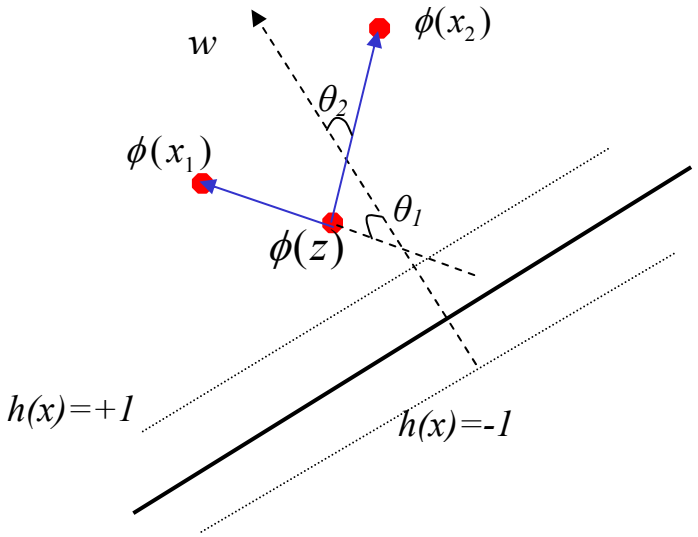

Figure 2.1 *h(x)* is the hyper-plane in the feature space. Points x_1, x_2, and z in the input space are mapped into feature space. w is the normal vector of hyper-plane *h(x)*=0.

Then:

$$|h(x_1) - h(z)| \le |h(x_2) - h(z)| \tag{2-9}$$

Proof:

$\because \|\phi(x_1) - \phi(z)\| \le \|\phi(x_2) - \phi(z)\|$, both sides multiply $\|w\| \cdot |\cos\theta_1| \cdot |\cos\theta_2|$

$\Rightarrow \|\phi(x_1) - \phi(z)\| \cdot \|w\| \cdot |\cos\theta_1| \cdot |\cos\theta_2| \le \|\phi(x_2) - \phi(z)\| \cdot \|w\| \cdot |\cos\theta_2| \cdot |\cos\theta_1|$

$\Rightarrow \left|\langle \phi(x_1) - \phi(z) | w \rangle \cos\theta_2\right| \le \left|\langle \phi(x_2) - \phi(z) | w \rangle \cos\theta_1\right|$ and $\because \langle x | z \rangle = \|x\| \cdot \|z\| \cos\theta$

$\Rightarrow \left|\langle \phi(x_1) | w \rangle - \langle \phi(z) | w \rangle \cos\theta_2\right| \le \left|\langle \phi(x_2) | w \rangle - \langle \phi(z) | w \rangle \cos\theta_1\right|$ since inner product is distributive.

$\Rightarrow |h(x_1) - h(z)| \cdot |\cos\theta_2| \le |h(x_2) - h(z)| \cdot |\cos\theta_1|$ and $\because |\cos\theta_1| \le |\cos\theta_2|$

$\Rightarrow |h(x_1) - h(z)| \le |h(x_2) - h(z)|$, then the theorem is proved.

The theorem **2.1** shows that predicating label is much similar if two vectors in the feature space are closer, and the angle between vectors is smaller. It also shows that the predicting label depends on hyperplane because the angle is relative to w.

Suppose that vector z is an instance from testing data set; examples (x_1, y_1) and (x_2, y_2) are from training data set. If $d(x_1, z) \le d(x_2, z)$ in equation (2-6) and $\eta^2 \ge 1$ in equation (2-7), the conclusion is the value of predicting label $h(z)$ of instance z is closer to y_1 .than y_2. If θ 1=0, that means vector $\phi(x_1) - \phi(z)$ parallels to hyperplane, and no matter what θ 2 is, vector $\phi(x_2)$ is further to $\phi(z)$ than vector $\phi(x_1)$. The theorem **2.1** is the foundation of vector similarity in the feature space.

2.2.2 Polynomial kernel

The polynomial kernel is defined:

$$K(x, z) = (\langle x | z \rangle + c)^d \text{, d>1} \qquad (2\text{-}10)$$

For example d=2, and vector x or z has n dimensions,

$$K(x,z) = (\langle x|z\rangle + c)^2 = (\sum_{i=1}^{n} x_i z_i + c)(\sum_{j=1}^{n} x_j z_j + c)$$
$$= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j z_i z_j + \sum_{i=1}^{n} 2c x_i z_i + c^2 \qquad (2\text{-}11)$$
$$= \sum_{(i,j)=(1,1)}^{(n,n)} (x_i x_j)(z_i z_j) + \sum_{i=1}^{n} (\sqrt{2c}x_i)(\sqrt{2c}z_i) + c^2$$

Therefore, the total feature of degree 2 polynomial kernel is $\binom{n+1}{n} + n + 1 = \binom{n+2}{n}$. In general, the total feature of degree *d* polynomial kernel has $\binom{n+d}{n}$ number of features.

2.3 Support Vector Machines (SVMs)

Support Vector Machines (SVMs) [1, 5, 53-55] is a learning algorithm for classification, regression and density estimation. SVM has been used successfully in many areas including bioinformatics[56]. For example, the SVM can be used to learn polynomial, radial basis function (RBF) and multi-layer perceptron (MLP) classifiers. SVMs are based on the structural risk minimization (SRM) principle, which incorporates capacity control to reduce overfitting.

2.3.1 The Maximal Margin Classifier

The basic SVM is a linear classifier to separate the training data $\boldsymbol{S}$ into two classes. The separator or hyperplane is $\boldsymbol{w}^T\boldsymbol{x}+b=0$, where $\boldsymbol{w}$ is the weight vector and b is the bias term. Suppose the training data are separable, the optimal hyperplane satisfies conditions in the following by maximize the margin of separator which is the width of separation between classes,

$$\begin{aligned} &\underset{\mathbf{w},b}{\text{Minimize}} \quad \Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{subject to} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, i = 1,\ldots,l \end{aligned} \tag{2-12}$$

The risk functional is the $\Phi(\mathbf{w})$. Lagrange multipliers are introduced $\lambda_i \geq 0, i = 1,...,l$ for each constraint in (2-12). The following Lagrangian is gotten:

$$L(\mathbf{w},b,\Lambda) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l}\lambda_i\left[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\right] \tag{2-13}$$

where $\Lambda = (\lambda_1,\ldots,\lambda_l)^T$ are the Lagrange multipliers, one for each example. The task is to minimize (2-13) with respect to ***w***, *b*, and maximize it with respect to Λ. Differentiating with respect to ***w*** and *b* and setting the derivatives equal to 0 to get optimal point

$$\frac{\partial L(\mathbf{w},b,\Lambda)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{l}\lambda_i y_i \mathbf{x}_i = 0 \tag{2-14}$$

and

$$\frac{\partial L(\mathbf{w},b,\Lambda)}{\partial b} = -\sum_{i=1}^{l}\lambda_i y_i = 0 \tag{2-15}$$

The optimal w* is

$$\begin{aligned} &\mathbf{w}^* = \sum_{i=1}^{l}\lambda_i^* y_i \mathbf{x}_i \\ &\text{subject to} \sum_{i=1}^{l}\lambda_i y_i = 0 \end{aligned} \tag{2-16}$$

Substituting (2-13) by (2-16):

$$F(\Lambda) = \sum_{i=1}^{l}\lambda_i - \frac{1}{2}\|\mathbf{w}\|^2 = \sum_{i=1}^{l}\lambda_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\lambda_i\lambda_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \tag{2-17}$$

We get dual problem of primal problem (2-12):

$$\begin{aligned} &\text{Maximize } F(\Lambda) = \Lambda^T I - \frac{1}{2}\Lambda^T D\Lambda \\ &\text{subject to } \Lambda \geq 0, \Lambda^T y = 0 \end{aligned} \tag{2-18}$$

where $y = (y_1, \ldots, y_l)^T$, I is an identity matrix and D is a symmetric $l \times l$ matrix with elements $D_{ij} = y_i y_j \mathbf{x}_i^T \mathbf{x}_j$. To solve the above convex quadratic programming (QP) problem, we get the classifier $f(\mathbf{x})$,

$$\begin{aligned} &h(\mathbf{x}) = \sum_{i=1}^{l} y_i \lambda_i^* \mathbf{x}^T \mathbf{x}_i + b^*, \quad b^* = y_i - \mathbf{w}^{*T} \mathbf{x}_i \\ &f(\mathbf{x}) = \text{sgn } h(\mathbf{x}) \end{aligned} \tag{2-19}$$

where λ*i>0. If λ*i≠0, the i[th] vector x_i is a ***support vector***. To a nonlinear separable problem, an *n*-dimensional input vector ***x*** is projected into a high *m*-dimensional space using nonlinear function $\phi(\mathbf{x}) : \mathcal{R}^n \to \mathcal{R}^m$, and then the output is linear. Then (2-19) becomes

$$f(x) = \text{sgn } h(\mathbf{x}) = \text{sgn}(\phi(\mathbf{x})^{\mathrm{T}} \mathbf{w}^* + b^*) = \text{sgn}\left(\sum_{i=1}^{l} y_i \lambda_i^* \phi(\mathbf{x})^{\mathrm{T}} \phi(\mathbf{x}_i) + b^* \right) \tag{2-20}$$

The function (2-20) is in the form of inner products $\phi(\mathbf{x})^T \phi(\mathbf{z})$, which can be represented by a *kernel function* $K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z}) = \langle \phi(\mathbf{x}) | \phi(\mathbf{z}) \rangle$. Using kernel functions makes it not necessary to find the mapping function. Therefore the kernel function is a way to construct non-linear hyper-surface. For instance, if a polynomial kernel is used, then hypothesis *h(x)* can be represented by a continuous hyper-surface with polynomial *h(x)* in the input space whereas *h(x)* is also a hyperplane in the high-dimensional feature space through mapping function $\phi(\mathbf{x})$. The hyperplane *h(x)* is determined by the training examples x_i *i=1...l.* If the training examples are changed the shape of the hyper-surface will also be changed. The hyper-surface is usually vectors sensitive. In the SVM, the

hyperplane is determined by the support vectors that are only part of example subset of training set. Only the support vectors will affect the hyperplane. The overfitting is much serious if the number of the support vectors is close to the size of training data set. By using kernel function, the decision function (2-19) becomes

$$f(\mathbf{x}) = \operatorname{sgn} h(x) = \operatorname{sgn}\left(\sum_{i=1}^{l} y_i \lambda_i^* K(\mathbf{x}, \mathbf{x}_i) + b^* \right) \tag{2-21}$$

where the bias is given by

$$b^* = y_i - \mathbf{w}^{*T} \phi(\mathbf{x}_i) = y_i - \sum_{j=1}^{l} y_j \lambda_i^* K(\mathbf{x}_j, \mathbf{x}_i) \tag{2-22}$$

for any support vector $\mathbf{x}_i$.

2.3.2 The Soft Margin Optimization

SVM introduced a vector of slack variables $\Xi = (\xi_1, \xi_2, ..., \xi_l)^T$ when the hypothesis is found to be inconsistent with any single example as show on Figure 2.2. It does not completely eliminate a hypothesis if an inconsistent example is found.

$$\begin{aligned} &\underset{\mathbf{w},\mathrm{b},\Xi}{\text{Minimize}} \quad \Phi(\mathbf{w}, b, \Xi) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{l} \xi_i^k \\ &\text{subject to} \quad y_i (\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \xi_i \geq 0, i = 1, \ldots, l \end{aligned} \tag{2-23}$$

where C is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the training error term; k is integer k>0. If C is too small, insufficient stress will be placed on fitting the training data. If C is too large, the algorithm leads to overfitting. The slack variable ξ^k is related to noise sensitivity. The optimization hypothesis *h(x)* of learning task (2-23) has a similar form to the equation (2-19).

$$L(\mathbf{w}, b, \Lambda, \Xi, \Gamma) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{l} \lambda_i [y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b) - 1 + \xi_i] - \sum_{i=1}^{l} \gamma_i \xi_i + C \sum_{i=1}^{l} \xi_i \quad (2\text{-}24)$$

where $\Lambda = (\lambda_1, \ldots, \lambda_l)^T$ as before, and $\Gamma = (\gamma_1, \ldots, \gamma_l)^T$ are the Lagrange multipliers corresponding to the positive of the slack variables. Differentiating with respect to $\mathbf{w}$, b and Ξ and setting the results equal to zero to obtain

$$\begin{aligned} \frac{\partial L(\mathbf{w}, b, \Lambda, \Xi, \Gamma)}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^{l} \lambda_i y_i \phi(\mathbf{x}_i) = 0 \\ \frac{\partial L(\mathbf{w}, b, \Lambda, \Xi, \Gamma)}{\partial b} &= -\sum_{i=1}^{l} \lambda_i y_i = 0 \\ \frac{\partial L(\mathbf{w}, b, \Lambda, \Xi, \Gamma)}{\partial \xi_i} &= C - \lambda_i - \gamma_i = 0. \end{aligned} \quad (2\text{-}25)$$

The dual problem of soft margin is:

$$\begin{aligned} &\text{Maximize} \quad F(\Lambda) = \Lambda^T I - \frac{1}{2} \Lambda^T D \Lambda \\ &\text{subject to} \quad 0 \le \Lambda \le C, \Lambda^T y = 0 \end{aligned} \quad (2\text{-}26)$$

where $y = (y_1, \ldots, y_l)^T$, I is an identity matrix and D is a symmetric $l \times l$ matrix with elements $D_{ij} = y_i y_j \mathbf{x}_i^T \mathbf{x}_j$. The decision function implemented is exactly as before in (2-19). The bias term b^* is given by (2-18) where a support vector $\mathbf{x}_i$ is for which $0 < \lambda_i < C$. Hsu et al paper [57] gives a general guide to choose a good value of C. The paper[58] also discussed how to tune parameter automatically. An alternative algorithm was presented to get maximum margin between training examples and decision boundary[59]. To compare different SVMs, C is not easy to use. v-Support Vector Machine(v-SVM) is introduced in[60]. In v-SVM, C is replaced by v limited on interval (0, 1]. The parameter v is asymptotically with an upper bound on the number of margin errors and a lower bound on the number of support vectors.

2.3.3 Karush-Kuhn-Tucker condition (KKT)

Karush-Kuhn-Tucker conditions (KKT), are[1]

$$\begin{aligned} \lambda_i[y_i(w^T\phi(x_i)+b)-1+\xi_i]=0 \\ \xi_i(\lambda_i - C)=0 \end{aligned} \tag{2-27}$$

The KKT conditions imply that non-zero slack variables ξ_i can be occur in $\lambda_i \neq C$. Then the following can be obtained [61].

$$\begin{aligned} \lambda_i = 0 \quad &\Rightarrow \; y_i h(x_i) \geq 1 \; and \; \xi_i = 0 \\ 0 < \lambda_i < C \quad &\Rightarrow \; y_i h(x_i) = 1 \; and \; \xi_i = 0 \\ \lambda_i = C \quad &\Rightarrow \; y_i h(x_i) < 1 \; and \; \xi_i > 0 \end{aligned} \tag{2-28}$$

If $y_i h(x_i) \geq 1$, then x_i is correctly classified and well separated. Otherwise, x_i is support vector. If $y_i h(x_i) \leq 0$, then x_i is misclassified. If $0 < y_i h(x_i) < 1$, then x_i is correctly classified but its confidence is small.

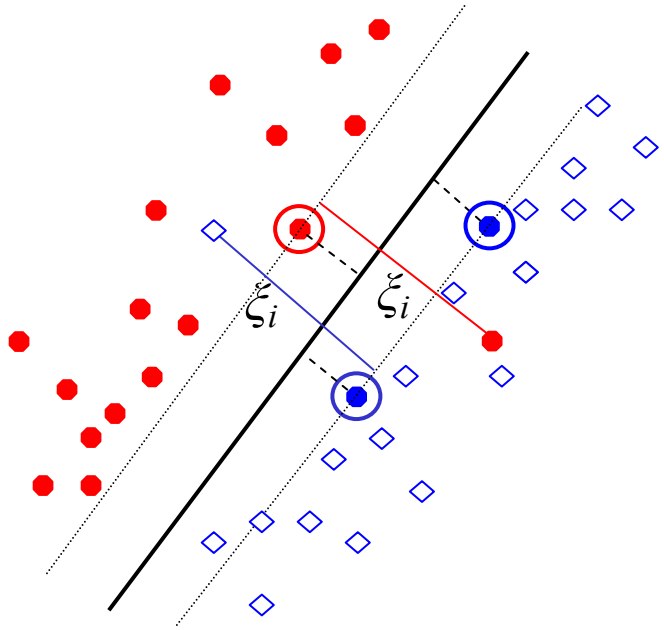

Figure 2.2 Maximal Margin, Support vectors and noisy examples

The geometrical interpretation of support vector classification is that the SVM searches the optimal separating surface in the hypotheses space. This optimal separating hyperplane has many nice statistical properties. The value $h(x)/\|w\|$ is the geometric distance between vector *x* to the hyperplane.

According to KTT condition (2-27) and for all $y_i h(x_i) \le 1$, we get $\lambda_i \le C$ and

$$y_i(w^T \phi(x_i) + b) - 1 + \xi_i = 0 \tag{2-29}$$

$$\Rightarrow y_i h(x_i) = 1 - \xi_i \tag{2-30}$$

$h(x_i)$ is the predicting label of instance x_i. Maximizing AR metric is exactly minimizing the 1-norm empirical error in the soft margin SVM if $h(x_i)$ is set to 1 when $h(x_i)$ >1. The reason is below:

$$AR = \frac{1}{l}\sum_{i=1}^{l} y_i h(x_i) = \frac{1}{l}\sum_{i=1}^{l}(1 - \xi_i) = 1 - \frac{1}{l}\sum_{i=1}^{l}\xi_i \tag{2-31}$$

In SVM application, one only needs to determine the kernel function, and the regulation parameter to control trade off between margin and empirical error. This characteristic is convenient because of less parameter user needs to decide. However, this is also drawback because it provides less control in a complex application. NDDCHA gives much control than standard SVM, like Boosting provides much control than base learning algorithm.

2.4 k-NEAREST NEIGHBOR (KNN) and Knowledge Representation

The instance-based approaches can construct a different hypothesis for each distinct testing vector where the hypothesis is created from a subset of training data. Aha, Kibler and Albert described three experiments in Instance-based learning (IBL)[62]. In the first

experiment (IB1), to learn a concept or knowledge simply required the program to store every example. When an unclassified example was presented to be classified, it used a simple Euclidean distance measure as vector similarity method to determine the nearest neighbor of the object and the class given to it was the class of the neighbor. This scheme has capability to tolerate some degree of noise in the data. The disadvantage is that it requires a large amount of storage memory. IB1 is actually an instance ($k=1$) of k-nearest neighbor method under the condition of all possible examples are known.

In the second experiment, it extended the performance of IB1 and reduced the storage by classifying new example. Examples correctly classified were ignored and only incorrectly classified examples were stored to be part of concept. The knowledge of correctly classified examples (positive data) is included in the classifier or hypothesis. This scheme used less memory and was less noise tolerant than IB1.

The third experiment (IB3) used the scheme of IB2 and maintained a record of the number of correct and incorrect classification attempts for each saved examples. This record summarized an example's classification performance. IB3 uses a significance test to determine which examples are good classifiers and which ones are believed to be noisy. The latter are discarded from the concept description. This method strengthens noise tolerance while keeping storage requirements down. In the IBL, it is a naïve approach to only store and search those incorrectly classified examples (negative data).

Euclidean distance between two vectors could be a metric of vector similarity. This technique is widely used in the instance-based learning[63, 64] such as k-nearest neighbor and locally weighted regression[2]. However the distance is calculated on the Euclidean space which is not suitable for the feature space defined by kernel function.

Kernel function also describes a similarity of vectors, for example, the basic kernel function $K(x,z)=\langle x|z\rangle=\|x\|\cdot\|z\|\cos(\theta)$. Therefore, the kernel function is a probably way to tell the similarity of vectors in the feature space because the feature space is defined by kernel methods. When the distance metric is applied, the distance between two vectors is calculated based on all attributes of vector. This metric may lead to performance degradation if irrelevant attributes are present, which is a type of the *curse of dimensionality*. As the number of attributes increasing, the computational cost and generalization capacity can be degraded which is a phenomenon also belongs to the category of curse of dimensionality.

Binary classification *k*-NEAREST NEIGHBOR algorithm[2, 65] is in the following:

Training algorithm:

For each training example $(x_i, y_i), i=1..l$, add the examples to the training set $\boldsymbol{S}$.

Classification algorithm:

Given a query instances x_t to be classified.

Let $x_1, x_2, ..., x_k$, denote the k instances from S that are nearest to x_t,, $y_1, y_2, ..., y_k$ are labels of these instances. The hypothesis returns:

$$h(x_t)=\arg\max_{v\in\{-1,+1\}}\sum_{i=1}^{k}\delta(v,y_i) \tag{2-32}$$

where δ is an indicator function, $\delta(x_1,x_2)=1$ if $x_1=x_2$ and where $\delta(x_1,x_2)=0$ otherwise.

When target function is a continuous real value, the *k*-NEAREST NEIGHBOR algorithm will be the same as binary classification on above except the equitation (2-32) is replaced by

$$h(x_t) = \frac{1}{k}\sum_{i=1}^{k} y_i \tag{2-33}$$

The distance-weighted NEAREST NEIGHBOR algorithm uses equation (2-32) to be replaced by

$$h(x_t) = \frac{\sum_{i=1}^{k} w_i y_i}{\sum_{i=1}^{k} w_i} \tag{2-34}$$

where weight w_i is the reciprocal value of Euclidean distance between x_t and x_i.

$$w_i = \frac{1}{\sqrt{\|x_t - x_i\|^2}} = \frac{1}{\sqrt{\langle x_t | x_t \rangle - 2\langle x_t | x_i \rangle + \langle x_i | x_i \rangle}} \tag{2-35}$$

The performance of the KNN depends on a locally constant posteriori probability assumption. This assumption, however, becomes problematic in high dimensional spaces due to the curse of dimensionality and the noise of data. Wang el at proposed an adaptive nearest neighbor algorithm for classification by considering the size of influence sphere and confidence level of example instead of Euclidean distance[66]. KNN could be considered a presentation of knowledge. KNN provides a method of vector similarity which could be used in the feature space defined by kernel function.

CHAPTER 3

METHODOLOGY

The computational cost of learning method and accuracy and intelligibility issues are concerned in the empirical machine learning processing. In the rapid increasing computational capability of computer, the performance/cost ratio has not been emphasized in most applications, especially in the batch machine learning which is trained off-line. Accuracy is a main concern in all applications of learning and relatively easy to be measured compared to the intelligibility. The approach proposed mainly focuses on the accuracy by mining negative data.

3.1 Concepts of Negative Data

3.1.1 Introduction of negative and positive data

The training data set is partitioned into two disjoint subsets, misclassified, and correctly classified examples in terms of a hypothesis $h(x)$. The misclassified examples are ***negative data***. The correctly classified examples are ***positive data***. Positive data is consisted of ***not well-separated*** and ***well-separated data***. Not well-separated data has small confidence, say its confidence $|h(\boldsymbol{x})| < \mu$ and threshold μ is an arbitrary number great than zero, to claim that they are positive while well separated data has high confidence to be positive. Negative and not well-separated examples together are called μ***-extended negative data*** whereas the well-separated examples are called μ***-shrunk positive data***. Negative data and extended negative data are exchangeable to be used in

the not confused environment. For the same reason, positive data and shrunk positive data are also exchangeable. ***Positive*** or ***negative data*** are exactly 0-***shrunk positive*** or 0-***extended negative data*** respectively. To ground our discussion of concepts above, consider the example task of learning which has 5 training examples and hypothesis *h(**x**)*:

$$S = \{(x_1,+1),(x_2,-1),(x_3,+1),(x_4,-1),(x_5,-1)\}$$

Suppose the predicted values $h(x_1)=0.3$, $h(x_2)= -1.2$, $h(x_3)= 1.0$, $h(x_4)= -0.5$, $h(x_5)= 0.8$, and the threshold $\mu=0.6$. Then misclassified data or negative data is $\{(x_5,-1)\}$, and classified data is $\{(x_1,+1),(x_2,-1),(x_3,+1),(x_4,-1)\}$. The *0.6*-extended negative data is $\{(x_1,+1),(x_4,-1),(x_5,-1)\}$ and *0.6*-shrunk positive data is $\{(x_2,-1),(x_3,+1)\}$.

An example could be positive or negative. The negative data does not mean the data is wrong or corrupt. What negative data can be known is that a hypothesis can not make it well-separated. Negative data depends on the hypothesis. Whether an example is positive or negative is relative. To a specific example, hypothesis A classifies it to be negative while hypothesis B may classify it to be positive. Furthermore, even for the same hypothesis, an example probably belongs to positive or negative in terms of the different confidence threshold μ.

The parts of the hyper-surface classifying negative examples need to be repaired in order to improve performance. As shown on Figure 1.2, the rectangle with thick solid color needs to be repaired. The other parts of the hyper-surface classify the positive data sets as well-separated and they have high generalization capacity.

3.1.2 Separator and partitioner

A binary classification of SVM is a linear classifier in the feature space. The data set must be mapped into feature space from input space if the classifier is non-linear. The

mapping mechanism is finished by so-called kernel function. Then it is much suitable to discuss data separable concepts on the linear space if SVM is a base learning algorithm. Without loss of generality, we discuss concepts on the linear space where the hyper-surface becomes a hyperplane since a non-linear space can be mapped into linear space by a mapping function. If there is a hyperplane *h(x)* that correctly classifies all training data set S, we say that the data set are separable. If no such hyperplane exists the data set are said to be non-separable. In general, if the data set has noise or non-optimal hypothesis is used, the data set cannot be separated. As shown in Figure 3.1, the subset consists of misclassified examples which are points with the solid color. If an example *(x,y)* is correctly classified according to the classifier or separator *h(x)=0*, *(x,y)* is said to be in the consistent subset $(x,y) \in CS \subseteq S$, otherwise *(x,y)* is in the inconsistent subset $IS=S-CS$, $(x,y) \in IS$. The testing data set T is the union of all correct and in correct classified data set $T=TP+TF+FP+FN$, then consistent subset is $CS=TP+TN$, and inconsistent subset $IS=FP+FN$ as seen on Table 1.1.

A data subset of *CS* is said to be not well-separated, denoted to *NWSS,* if the points of these examples are much close to the hyperplane *h(x)=0.* The data subset $WSS=CS - NWSS$ is said to be well separated. Examples in the *IS* and together with *NWSS* is called in the ***extended negative data subset*** $N=IS+NWSS$. The metric of "much close to" is given by a ***partitioner*** *p(h,x,y)* which is a fuzzy word depending on the hyperplane *h(x)* and data set ***S***. The return value of *p(h,x,y)* is the logical value either *true* or *false*.

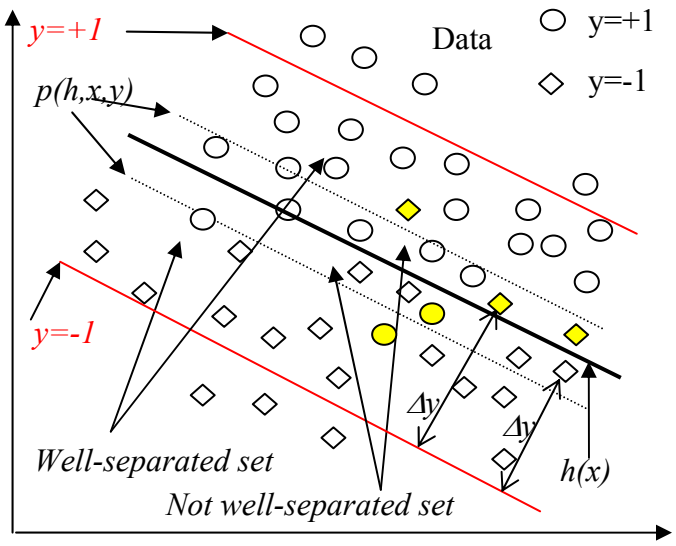

Figure 3.1 Well-separated data and not well-separated data are in the different area. The points with solid pattern are misclassified.

One simple example of partitioner in the classification is the crisp boundary $p(h, x, y)$: $|y-h(x)| \leq \mu$, $\mu=0.5$ then not-well-separated data set is

$$NWSS=\{(x, y) | (x, y) \in CS, ||y-h(x)|| \leq \mu, \mu=0.5\}.$$

The second example is the fuzzy partitioner, and then *WSS* and *NSWW* are fuzzy set. The boundary of partitioner is a range. In the third example, a non-symmetric linear partitioner $p(h,x,y)$: $y-h(x) \leq -\mu_1$, $y-h(x) \geq \mu_2$, $\mu_1, \mu_2 \in [0,0.5]$ is defined on both sides of hyperplane,

$$NWSS=\{(x,y) | (x,y) \in CS, y-h(x) \leq -\mu_1, y-h(x) \geq \mu_2 , \mu_1, \mu_2 \in [0,0.5]\}.$$

How to define a partitioner and how to choose the parameter for the partitioner depend on the real application which could reference to the ratio of number of support vectors and training examples, VC Dimensions, the size of training set, cross-validation and etc. Usually, the cross validation is an efficient method to determine the partitioner. Note that

the well separated data set *WSS* is correctly classified. And it is also called ***shrunk positive data subset P***, briefly called positive data. The boundary between positive data set and negative data set is called ***border***. In general, the relationships of the subset mentioned above are

$$\begin{gathered} S = CS+IS, \\ CS = \{(x,y)|\ (x,y)\in S,\ y*h(x)>0\}, \\ IS = \{(x,y)|\ (x,y)\in S,\ y*h(x)\leq 0\}, \\ CS = WSS+ NWSS, \\ NWSS\ = \{(x,y)\ |\ (x,y)\in CS,\ p(h, x, y)=true\}, \\ WSS = \{(x,y)|(x,y)\ \in CS,\ p(h, x, y)=false\}, \\ P = WSS, \\ N = NWSS+IS. \end{gathered} \tag{3-1}$$

We can say the positive and negative data set are divided by both separator *h(x)* and partitioner *p(h,x,y)*. Let *d(h,x,y)* = [*p(h(x), x, y)=true or y*h(x)≤0*]. Then *N= {(x,y) | (x,y)∈S, d(h,x,y)}* and *d(h,x,y)* is denoted a ***divider*** to divide the training data set into positive and negative data set. In this dissertation, only linear partitioner is considered then the precise definition of negative data is given in the section.

3.1.3 μ-Negative data

Definition 3.1 (Data Type): *suppose h(x) is a hypothesis learned from a training data set*

$$S = \{(x_1, y_1), (x_2, y_2),..(x_l, y_l)\}.\ i = 1..l \tag{3-2}$$

and a vector of variables

$$\Xi = (\xi_1, \xi_2,..., \xi_l)^T, \xi_i \in \mathcal{R}, i = 1,\ldots, l \tag{3-3}$$

to let it satisfy the following equality:

$$\xi_i = 1 - y_i h(x_i) \tag{3-4}$$

An example x_i is ***negative data*** *if $\xi_i > 1$, whereas it is* ***positive data*** *if $\xi_i < 1$. An example x_i is* ***μ-negative data*** *if $\xi_i > 1-\mu$, $0 < \mu < 1$, which is* ***extended negative data*** *whereas it is* ***μ -positive data*** *if $\xi_i < 1-\mu$ which is* ***shrunk positive data****. The μ-negative data is denoted by*

$$N(S, h, \mu) \tag{3-5}$$

The μ-positive data is denoted by

$$P(S, h, \mu) \tag{3-6}$$

The ratio of the μ-negative data and μ-positive data is denoted by

$$c(S, h, \mu) = \frac{N(S, h, \mu)}{P(S, h, \mu)} \tag{3-7}$$

which is a measure of degree of unbalancing in terms of training data, hypothesis and threshold μ.

The misclassified examples are $N(S, h, 0)$ according to definition of μ-negative data while correctly classified examples are $P(S, h, 0)$. The accuracy of hypothesis *h(x)* is least 50%, then we get $c(S, h, 0) < 1$. However, $c(S, h, \mu)$ is not always less than 1 which depends on the number of support vectors. The threshold ***μ*** plays a ***divider*** role in the training data set. The concept of negative data can be extended to the testing data set. The terms of negative data, extended negative data or ***μ***-negative data are exchangeable to be used in the not confused environment. It is the same reason to exchange the terms of positive data, shrunk positive data or ***μ***-positive data.

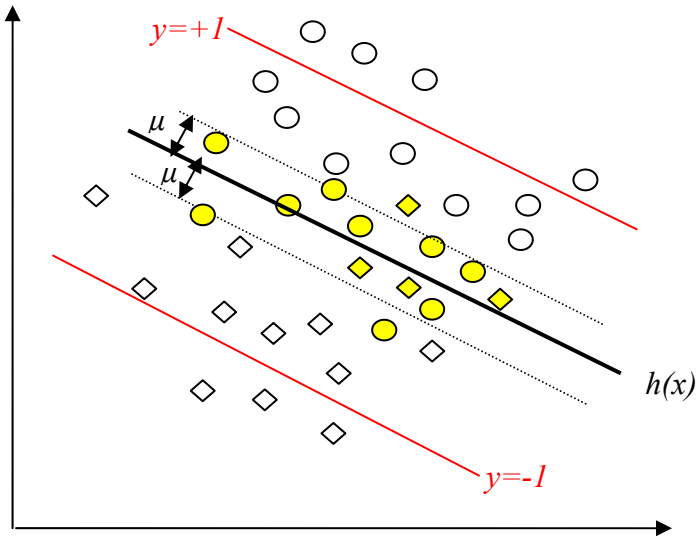

Figure 3.2 μ-negative examples are defined in the SVM feature space, which are points marked with solid pattern.

Theorem 3.1**(Negative Support Vectors)** *all negative examples of training data in SVM are support vectors.*

Proof:

For all negative examples, the maximum of threshold $\boldsymbol{\mu}$ is less than 1, $0 < \mu < 1$**,** according to definition of negative data.

Since $\xi_i > 1 - \mu$, we get $\xi_i > 0$.

According to the equality of $y_i h(x_i) = 1 - \xi_i$, all negative examples satisfies the inequality $y_i h(x_i) < 1$ for $i = 1..l$.

According to the KKT conditions $\xi_i(\lambda_i - C) = 0$ in the (2-27), we can get $\lambda_i - C = 0 \Rightarrow \lambda_i = C \neq 0$

All examples with non-zero Lagrange multipliers are support vectors according to the definition of support vector. Therefore, the theorem is proved.

The misclassified examples x_i meet the condition $y_i h(x_i) < 0$, thereby all misclassified examples are support vectors. According to the Theorem 3.1, all misclassified training examples and examples located on the region between SVM margins are support vectors. It is known there is negative data if examples are not separable, because there are existed misclassified examples.

Theorem 3.1 shows that the number of negative examples is related to number of support vectors. Negative examples are caused from non-separatable data. Decreasing number of negative examples obviously enhances generalization capability of SVM because that means less classification error. Based on this reason, reducing the number of support vectors can improve the performance of SVM. Reduced SVM actually demonstrates this idea[67, 68].

The hypothesis of SVM is determined by kernel function and support vectors, therefore the negative data dominate the performance of classifier. One kind of negative example x_i is named outlier where $\xi_i >> 1$. Thereby, outliers will degrade largely performance of hypothesis. Outliers are assumed to be removed from original *training* data in the data preprocessing phase in this dissertation.

3.2 Motivation

The classifier or hypothesis in the classification is a hyper-surface in multidimensional space. A low accuracy hypothesis indicates that some areas of hyper-surface are not exactly or close to the underlying function. Intuitively, repairing those lower accuracy areas will improve the hypothesis accuracy. Then compensating a

hypothesis has the same meaning as repairing a hyper-surface. The hyper-surface in the input space is mapping to a hyperplane in the feature space by a kernel function in the SVM [1]. The hyper-surface is smooth in the SVM since feature space is a linear space and commonly used kernel functions are continuous in the space. A hyper-surface is comprised of a set of small pieces of sub-hyper-surface in the fuzzy control because of the nature of fuzzy membership function segments [69]. Therefore, the hyper-surface could have more than one outlines either a single large surface or a series of small surfaces. As shown in Figure 1.4, the single hyper-surface $h_1(x)$ is not enough for the high predicting accuracy because any improvements in training accuracy will be prone to overfitting, such as in the high degree of polynomial hyper-surface. To reduce the possibility of overfitting, the low degree of hyper-surface $h_2(x)$ is preferred, here $h_2(x)$ is curve line. However, $h_2(x)$ will lead to low predicting accuracy or underfitting. Therefore, in order to improve the predicting accuracy or generalization capacity, the hyper-surface $h_1(x)$ needs to be repaired so that it approximates to the underlying function $f(x)$. There are five possible ways in which one can repair.

In the first method, a spline hyper-surface is a set of piecewise sub-hyper-surface which is applied to a collection of subsets of input space $\boldsymbol{X}$ by using clustering technique to segment the input space $\boldsymbol{X}$ into sub-spaces. The method can be thought in this way using a pile of mosaics to lay tiles to cover the whole surface of terrain. Each sub-hyper-surface predicts the sub input space. The advantage is that each sub-hyper-surface matches specific sub input space well; hence, the machine gives output with low error. A sub-hyper-surface only has relationship to neighbor sub-hyper-surface. This feature makes good improvement capability by decreasing the overfitting. The property of

locality of spline hyperplane makes the re-training machine in a partial area of hyper-surface rather than the whole system, which reduces re-training time and also makes the system robust. The limitation is to resolve the issue of connecting the sub-hyper-surface smoothly in the boundary to combine together as a whole hyper-surface and also to divide into sub-space. The criteria of clustering are still not really clear and large size of training data is needed. This method also requires a reasonable size of training data set. Vladimir Vapnik gave a method of kernel generating spline [5]. Spline kernel is powerful and B-spline kernel SVM can be interpreted the CMAC network introduced by Horvath [70].

In the second method, a spline kernel [71] in SVMs is chosen to repair hyper-surface. In this approach, the continuity is guaranteed but the locality is lost because *h(x)* is not a spline function. Another limitation of spline kernel method is that the order *k* of spline kernel cannot be high, usually, $k \leq 4$ [3].

The third and fourth methods are tentative ideas and may not be practical. In the third method, cut-paste approach keeps the most area of *h(x)*. It replaces the partial area with high error in *h(x)* by a new small size of hyper-surface *h'(x),* like patching a hole of *h(x)* by a new small hyper-surface *h'(x)*. This procedure assumes that there are a lot of holes needed to patch. In this method the issues, like how to determine the holes and how to make the boundary smooth between patches and hyper-surface, have not been addressed to. The fourth method pre-stress is to find the high error regions in the hyper-surface and further to give the opposite regulation on those regions. This method stretches and compresses the hyper-surface by the force according to negative data set. The advantage is to keep the hyper-surface smooth, whereas, the disadvantage is the

difficulty to control force and the neighbor areas of repaired parts in the hyper-surface are affected.

In the last method, overlapping approach uses a base hyper-surface, which approximates the main outline of underlying function, which is created by a base learning algorithm. Further, a number of patching hyper-surfaces are overlapped onto base hyper-surface to form a new hyper-surface. This approach is adopted in this work.

So far, we know why, where, and how to repair a hyper-surface. Now the question is, how we could ascertain whether the compensated hyper-surface is enhanced or not. In the generalization theory, the structural risk minimization (SRM) model is to maximize the hyperplane margin measures in the feature space while minimizing the empirical error and hence prevent overfitting by controlling [5]. The empirical risk minimization model (ERM) only minimizes the empirical risk which may lead to low capacity of generalization. The ERM principle is intended to use in the large size of examples. Suppose the empirical risk is fixed, the hypothesis with large margin has higher capacity of generalization than those hypotheses with small margins. The SRM principle defines a trade off between the quality of the approximation of given examples and complexity of the approximation function[3, 5].

In fact, the SVM is based on the SRM to expand the capacity of generalization and then reducing the overfitting. A hypothesis h with large value $h(x)$ for a given vector x in the binary classification makes examples well-separated in the data set; while h with small margin makes examples not well-separated. A hypothesis making examples well-separated has high generalization capacity because the margin of these examples is large. If $h(x)<0$, it is said that x is classified into class -1 whereas if $h(x)>0$, x is in the class +1.

The $h(x)=0$ is a separator. For instance, the two hypotheses h1 and h2 both make all examples separated. Here, $h_1(x)>0$ means that x is in the class +1 while $h_1(x)<0$ means that x is in the class -1, where x is an example. If $|h_1(x)| > |h_2(x)|$, then h1 has higher generalization capacity x on than h_2. Therefore, the accuracy of training is not a precise measure because of the fact that both hypotheses with 100% accuracy have different generalization abilities. The formula (1-8) is more suitable to be a metric for generalization ability. We expect the hypothesis to output a real number $h(x) \in R$. Therefore, a learner like the decision tree is not considered as base learning approach here. This is because the hypothesis of decision tree is an indicator function. The SVM will be used as a learner and predictor. Yet another reason for using SVM is that the estimation error is well controlled [1, 3, 4].

3.3 Characteristics of SVMs

A training data set $\boldsymbol{S}$ with number of examples $\boldsymbol{l}=|\boldsymbol{S}|$ is generated in independently and identically (*i.i.d*) according to a fixed and unknown distribution $\boldsymbol{D}$. The examples are drawn from the distribution $P(\boldsymbol{x})$ while response y is from distribution $P(y|x)$ and an example (x_i,y_i) drawn according to $D = P(x,y) = P(x)P(y|x)$. A learner $\boldsymbol{L}$ consists of some classes of functions $h(\boldsymbol{x}, \boldsymbol{\alpha})$ defined over $\boldsymbol{X}$, which are the subset of hypotheses $\boldsymbol{H}$, $h(\boldsymbol{x}, \boldsymbol{\alpha}) \subseteq \boldsymbol{H}$, $\boldsymbol{\alpha} \in \Delta$, $\boldsymbol{\alpha}$ is an adjustable parameter which is generated by the learner according to the training set. For example, $\boldsymbol{\alpha}$ is corresponding to the weights and biases in the neural network with fixed architecture. $h(\boldsymbol{x}, \boldsymbol{\alpha})$ is written by $h(\boldsymbol{x})$ shortly once $\boldsymbol{\alpha}$ is determined. Each h in $\boldsymbol{H}$ is a function of the form $h:\boldsymbol{X}\rightarrow\boldsymbol{Y}$. To choose a best approximation to the underlying functional relationship based on the training set, one must consider the loss $L(y, h(\boldsymbol{x}, \boldsymbol{\alpha}))$ between response y from training set to a given input $\boldsymbol{x}$ and the response $h(\boldsymbol{x},$

α) provided by the learner. The expected value of the loss (3-8) given by the risk functional [72]:

$$R(\alpha) = \int L(y, h(x, \alpha)) dP(x, y) . \tag{3-8}$$

The goal is to minimize the risk functional to find the $h_0=h(\boldsymbol{x}, \boldsymbol{\alpha}_0)$ a maximum likelihood hypothesis. *P(x, y)* is unknown but its information is covered by the training set. The empirical regression model, empirical risk minimization (ERM), minimizes the following loss function (3-9):

$$L(y, f(x, \alpha)) = (y - f(x, \alpha))^2 \tag{3-9}$$

The task of learning algorithm is to minimize the risk functional (3-1) with loss function (3-9) where distribution *P(x, y)* is unknown and fixed and training data is given. The general model of learning problem can be described as follows [73]. The risk function:

$$R(\alpha) = \int Q(z, \alpha) dP(z), \alpha \in \Lambda, z \in S \tag{3-10}$$

where *Q(z, **α**)* is a specific loss function and *z* is a pair of *(**x**, y)* in the training examples. The empirical risk functional is

$$R_{emp}(\alpha) = \frac{1}{l} \sum_{i=1}^{l} Q(z, \alpha) . \tag{3-11}$$

There are two theorems from Vapnik [11]:

Theorem 3.2: *A hypothesis space **H** has VC dimension **d**. For any probability distribution P(x,y) on binary classification with probability* $1 - \delta$ *over random training sets **S**, any hypothesis h□**H** that makes **k** errors on **S** has error no more than*

$$\text{err}(h) \leq \frac{k}{l} + \frac{2}{l}\left(d \log \frac{2el}{d} + \log \frac{4}{\delta} \right) \tag{3-12}$$

provided d ≤ l. The VC dimension of a hypothesis space is a measure of the number of different classifications ability. The value of k/l is true error.

The inequality (3-12) shows that the capacity of generalization depends not only on the empirical error but also on the hypothesis space. It provides several ways to low error bound:

- Reduce VC dimension *d*
- Minimize number of training error *k*
- Increase size of data set

Theorem 3.3: *If a size l of training set **S** is separated by the maximal margin hyperplane, then the expectation of the probability of test error is bounded by the expectation of the minimum of three values: the ratio n/l, where n is the number of support vectors, the ratio* $(R||w||^2)/l$, *where* $R=max(||\boldsymbol{x}||)$, $\boldsymbol{x} \in S$ *and* $||w||^2$ *is the value of margin, and the ratio m/l. where m is the dimensionality of the input space **X**.*

$$err \le \min(\frac{n}{l}, \frac{R\|w\|^2}{l}, \frac{m}{l}) \tag{3-13}$$

Inequality (3-13) gives four ways to improve the generalization ability

- increase the size of training set
- make margin as large as possible
- reduce the dimensionality
- reduce support vectors

Reducing support vectors can also speed up the classification of SVM; Quang-Anh Tran et al proposed a method by using *k*-mean clustering. The *k* central vectors from *k* groups of clusters form new data set to reduce the size of data set. According to [74], the

support data can be extracted and then to reduce support vector. The tradeoff between speed and performance is controlled by *k* value[75]. ERM principle succeeds in dealing with large size of training data. It can be justified by considering the inequality (3-12). When *k/l* is large, *err* is small. A small value of *k/l* cannot guarantee a small value of the actual risk. Classical approach ignores the last three ways; only relies on the first one. The NDDCHA relies on the first one by mining negative data, the second one by making training data well-separated and third one by implicitly calculating the vector similarity in the feature space through kernel function.

SVM assumes training examples is i.i.d.; and costs of misclassification into different classes are the same. When these assumptions are violated, the standard SVM does not work properly[76].

3.4 VC Dimension

VC dimension of a set of indicator functions $Q(z,\alpha)$. $\alpha \in \Lambda$, is the maximum number *d* of examples that can be separated examples $z_1, z_2, ..., z_d$ into two classes in all 2^d possible ways using functions of the set, if for any *n* there exists a set of *n* examples that can be shattered by the set $Q(z,\alpha)$. $\alpha \in \Lambda$, then the VC dimension is equal to infinity. VC dimension can be estimated by [73]:

$$d \geq R^2 \|w\|^2 \tag{3-14}$$

where R is the radius of the smallest sphere that contains the all vectors in the feature spaces, and $\|w\|$ is the norm of the weights in the SVM. VC dimension is determined by kernel and training examples.

Gaussian kernel is employing an infinite dimensional feature space.

3.5 Vector Similarity

The vector-similarity is a metric to describe similar degree of two vectors. The vector-similarity depends on the learning problem greatly.

Definition 3.1: given 2 vectors $\boldsymbol{x}1$ and $\boldsymbol{x}2$, $\{\boldsymbol{x}1, \boldsymbol{x}2\} \subset X$. The similar degree is defined by function $\text{vs}(\boldsymbol{x}1, \boldsymbol{x}2) \in [0,1]$.

$$x_1 \propto x_2 = vs(x_1, x_2) \quad \begin{cases} = 0 & \text{if } x_1 \neq x_2 \\ = 1 & \text{if } x_1 = x_2 \\ \in (0,1) & \text{if } x_1 \approx x_2 \end{cases} \tag{3-15}$$

The symbol $\propto$ is borrowed as similar operator here. The Euclidean distance of vector x_1 and x_2, the cosine of the angle between x_1 and x_2, correlation coefficient [77, 78], and the sum of errors of all attributes of two vectors can be used as measures of the vector-similarity. Bandemer [79] gives a batch of vector similarity functions on fuzzy set. *vs* is not related to label y and it can be defined in many ways, e.g.

$$\begin{aligned} x_1 \propto x_2 &= sig \bmod(\|x_1 - x_2\|) \\ x_1 \propto x_2 &= \frac{\langle x_1 | x_2 \rangle}{\|x_1\| \cdot \|x_2\|} \end{aligned} \tag{3-16}$$

Data set similarity describes overall similar degree of two different data sets by comparing vector similarity of every vector in two data sets. In the binary classification, each vector $\boldsymbol{x}$ associated with a label y. If $y1$ and $y2$ are in the same class +1 or -1, $\text{vs}(\boldsymbol{x}1, \boldsymbol{x}2)$ keeps unchanged, otherwise, if $y1$ and $y2$ are in different class, we let $\text{vs}(\boldsymbol{x}1, \boldsymbol{x}2) < 0$. Therefore, the extended similarity defined by:

$$vs(x_1, x_2) = (x_1 \propto x_2)\, y_1 y_2 \tag{3-17}$$

Definition 3.2 (Vector Similarity): *given 2 data set A and B data sets, where* $A \subset X$ *and* $B \subset X$ *, and a constant number* $\upsilon \in [0,1]$.

$$\forall\ \mathrm{x}_{\mathrm{A}} \in A \text{ and } \exists\ \mathrm{x}_{\mathrm{B}} \in B$$

$$\text{such that: } \left| vs(\mathrm{x}_{\mathrm{A}}, x_B) \right| \geq \upsilon$$

where vs is the function of vector similarity defined in the (3-17) . Then we can say data sets A and B is similar in the degree of υ*, which is denoted by:*

$$A \propto B \geq \upsilon \tag{3-18}$$

For example, $A \propto B \geq 0.7$ is the lowest similar degree of $\boldsymbol{x}$A and $\boldsymbol{x}$B greater than 0.7 for any vector $\boldsymbol{x}$A in A and $\boldsymbol{x}$B in B. To positive data *P* and negative data *N* in terms of hypothesis *h(x)*, it is desired $P \propto N$ has lower value. Otherwise, an instance $\boldsymbol{x}$ from test data set *T* is hard to be recognized as similar to *P* or *N* in the KNN. Therefore, the following conditions are desired:

$$\begin{aligned} P \propto N &\leq \upsilon_1 \\ N \propto T &\geq \upsilon_2 \end{aligned} \tag{3-19}$$

where υ_1 and υ_2 are decided by application. In contrast with similar degree of two data sets, dissimilar degree of two data sets can be defined as below:

$$1 - A \propto B \tag{3-20}$$

When Euclidean distance is used as similar metric, vector similarity is exactly 1-NEAREST NEIGHBOR algorithm. In terms of the concept of KNN, vector similarity can be extended to 1-k vector similarity from 1-1 vector similarity which compares only two vectors.

3.6 Theoretical Analysis on NDDCHA

A feature space is a linear space determined by a kernel function implicitly. The dimensionality in the feature space is very large although the dimensionality in the input

space is not large. And the dimensionality increases abruptly as the complexity of kernel increasing. For example, feature spaces F_3 and F_4 constructed by a polynomial kernel in degree 3 and degree 4 respectively, the dimensionality of F_3 is $\dim_3 = \binom{m+3}{m}$ and the dimensionality of F_4 is $\dim_4 = \binom{m+4}{m}$, where m is dimensionality in the input space. Supposed m=50, this number is reasonable large for most applications, then the radio of d_4 and d_3:

$$\frac{\dim_4}{\dim_3} = \frac{m(m-1)(m-2)(m-3)3!}{m(m-1)(m-2)4!} = \frac{m-3}{4} = \frac{50-3}{4} \approx 11 \tag{3-21}$$

The sparse algebraic polynomial $P_m(x,\alpha)$, $\alpha \in R^m$, is a set of polynomials of arbitrary degree that contains m terms. For example, $P_2(x,\alpha) = \alpha_1 x^{d_1} + \alpha_2 x^{d_2}, (\alpha_1, \alpha_2) \in R^2$ with two nonzero terms. To estimate VC dimension with the set of loss of functions

$$Q(z,\alpha) = (y - P_m(x,\alpha))^2 \tag{3-22}$$

Karpinskyi and Werther[80] showed that the bound of VC dimension d for the sparse algebraic polynomials is 2d*,

$$3m \le d^* \le 4m + 3 \tag{3-23}$$

If $P_m(x,\alpha)$ is a hypothesis, which is determined by a kernel function of SVM, then formula (3-23) indicates that a high degree of polynomial has a large VC dimension and a low degree of polynomial kernel has a small VC dimension. To avoid overfitting or to get a small confidence interval, one has to construct machines with small VC dimension[3], such as low degree of polynomial kernel.

The VC dimension in the above example on (3-21) will increase 33 times according the formula (3-23). F_3 has $\dim_3$ terms and F_4, has dim 4 terms. The VC dimension d_3 in the feature space is limited in the range of formula (3-23).

$$\begin{aligned} 3\dim_3 \le d_3^* \le 4\dim_3 + 3 \\ 3\dim_4 \le d_4^* \le 4\dim_4 + 3 \end{aligned} \tag{3-24}$$

Then the ratio of VC dimensions in the feature space F_4 and F_3 is about 3*11=33. VC dimension is a metric to evaluate the complexity of a hypothesis. A hypothesis with high VC dimension will be prone to overfitting easily according to inequality (3-12)[3].

The above example of polynomial kernel tells that using lower degree polynomial as much as possible can get the high generalization capacity. On the other hand, if the set of functions has small VC dimension, then it is difficult to approximate the training data. To other types of kernel, the similar conclusion can be also gotten. For example, the exponential kernel could be approximated by polynomial kernel. However, there exists a tradeoff between overfitting and poor approximation. Therefore, the approach NDDCHA proposed here uses a lower complexity of kernel as a base learning algorithm to reduce the chance of overfitting, and uses compensated hypotheses to increase the accuracy of approximation.

3.7 The Patterns of Examples Distribution in the Feature Space

Two factors of the value of the empirical risk and the value of the confidence interval have been considered to minimize the risk in a given set of functions in implementing the SRM inductive principle. If examples are linear separable, then no extra work needs to do since basic SVM can solve this kind of applications perfectly as shown on Figure 3.5. In practice, data and hypotheses are not perfect. To the point of

view of data, the examples from input space include noise, outlier; distribution of examples is not i.i.d.; the number of examples in class +1 and -1 are imbalanced. To the point of view of hypothesis, the size of hypothesis space is limited. The optimization method also restricts the learning machine to get global optimal solution in most learning algorithm except SVM. These issues are considered in NDDCHA.

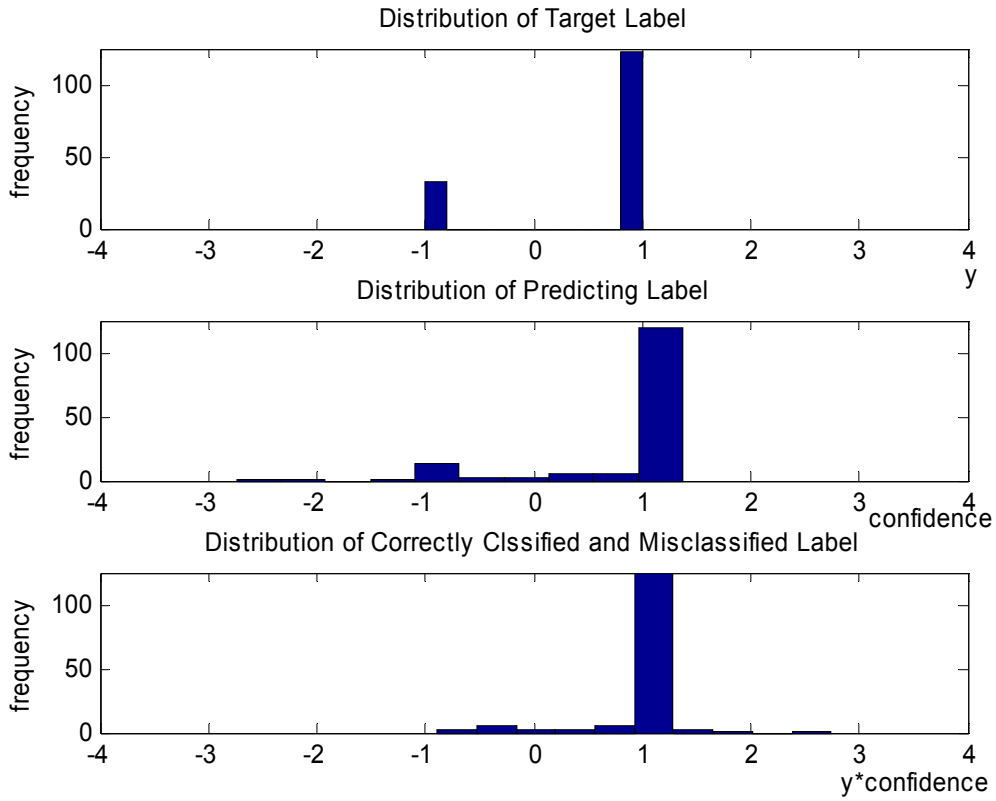

Figure 3.3 Distribution of target labels and predicating label on the *hepatitis*[81]

The figure 3.3 and 3.4 show the distribution of target labels and predicating label in the data sets of *hepatitis* and *musk2*. To predicting labels, degree 3 of polynomial kernel with C=0.0001 are used on the *hepatitis*. And degree 3 of polynomial kernel with C=0.0050 are used on the *musk2*. The top box shows the distribution of target label. Two data sets illustrate that datasets are strongly imbalanced. The middle box shows the distribution of

predicting labels. The instances which predicting value is within -1 and +1 are support vectors. The bottom box shows that all positives are correctly classified instances.

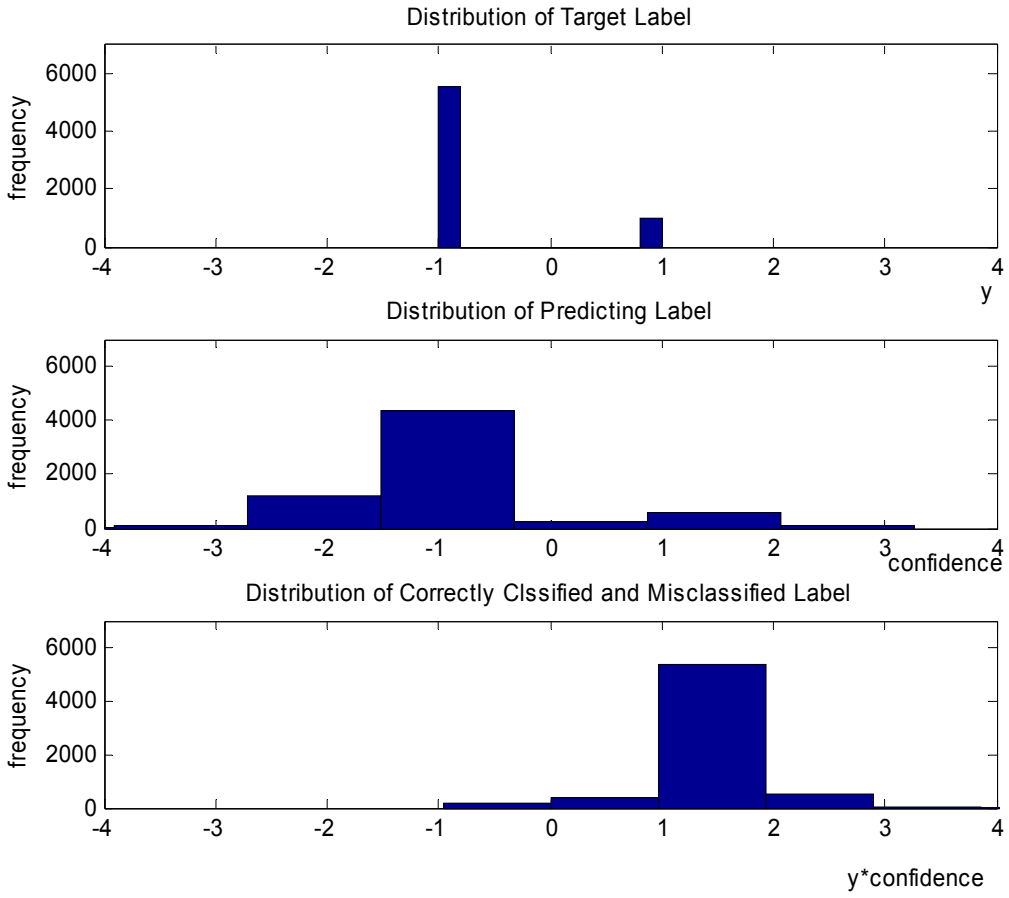

Figure 3.4 Distribution of target labels and predicating label on the *musk2*

3.7.1 Small size or imbalanced training data

The example size *l* is said to be small when ratio of *l/d* is small, say *l/d*<20, where *d* is the VC dimension of hypotheses space[5]. E.g. data set *Hepatitis* [82]has 155 examples with 19 input attributes. 123 examples are in the class +1 and 32 examples in the class -1. The estimated VC dimension of *Hepatitis* is about d=3253.84 according to

inequality (3-14) when polynomial kernel $(\langle x|z\rangle+1)^3$ is used in SVM^{light}[83, 84]. The ratio of *l/d* is 0.0496 << 20. The number of support vector is 80 and training error is zero. When polynomial kernel $(\langle x|z\rangle+1)^2$ is used, VC dimension is about d=328.72 with 2.38% training error. The ratio of *l/d* is 0.4715<<20. Therefore, Theorem 3.2 of predicating error bound cannot be applied in the small size training data set application.

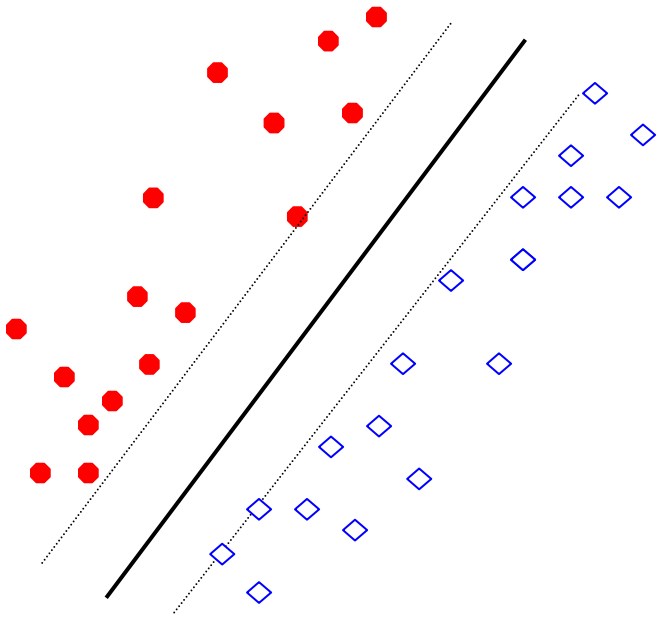

Figure 3.5 Linear separable examples

3.7.2 Noise, outlier and missing value example

An outlier is an example that lies outside the overall pattern of a distribution[85]. Usually, the presence of an outlier indicates some sort of problem. This can be a case which does not fit the hypothesis constructed, or an error in measurement as shown on Figure 3.6. In NDDCHA, the outliers are negative examples having long distances to the

hyperplane which could be simply deleted from data set. Noisy examples are also negative data which are either random or systematic. The random noise examples can be eliminated in the SVM. One of systematic noises could be caused from the small size of hypothesis space or non-optimal parameters α of specified hypothesis $h(x,\alpha), \alpha \in \Lambda$. Because SVM depends on a small set of support vectors comparing to training data set, the hypothesis may be sensitive to noises and outliers. According to Theorem 3.1, outliers are support vectors. Sometimes, a few of attributes in an example are missing value because of several reasons including: the value is hard to get or the value is still going to get. Then the value of kernel is decreased if all values of attributes are normalized to interval [0, 1]. The symptom is exactly like noisy examples because we can think the missing value is resulted from noises.

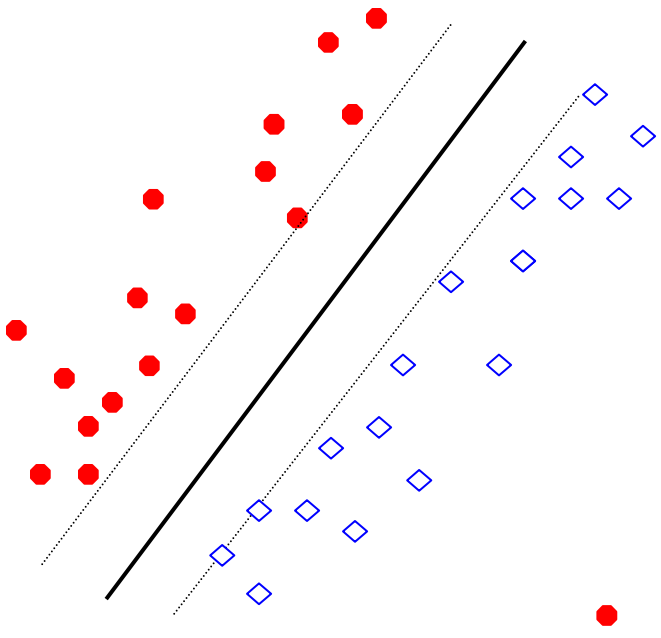

Figure 3.6 An example of outlier, the red circle on the right-bottom is an outlier which is far from other examples.

Chun-fu Lin and Sheng-de Wang proposed fuzzy support vector machines (FSVMs) providing a method to classify data with noises and outliers by associated with a fuzzy membership value to each training example[86]. The knowledge of membership is acquired from strategies of kernel-target[87] and KNN, which find the modified hyperplane by FSVMs in the feature space. FSVM provides a way to deal with the noises and outliers although the computational cost is high.

3.7.3 Compensatable negative examples

The pattern of compensatable negative data shown on Figure 3.7 is compensatable directly by patching a single hypothesis. Some examples of class +1 appeared circle points on area B are misclassified. However, these examples are not interviewed with class -1 appeared rectangle points. SVM tries to minimize total risk for all examples. To achieve that, examples on area B is obviously not on the margin according to the risk functional of soft margin SVM defined on Equation (2-23). The regulation parameter C is preferred to choose a small value then insufficient stress will be placed on fitting the training data, otherwise SVM will trend to overfit the training data. The parameter k on the equation (2-23) also contributes to the negative data. If $k = 0$ then second term $C\sum_{i=1}^{l} \xi_i^k$ counts the number of training errors, therefore, the lowest value k is $k = 1$. The value $k = 2$ is also used although this is more sensitive to outliers in the data. If we choose $k = 2$ then we are performing regularized least squares, i.e. the assumption is that the noise in examples is normally distributed. No matter what parameters are chosen on SVM, the pattern of Figure 3.7 cannot be eliminated since pattern is that result of that examples is not well distributed or i.i.d..

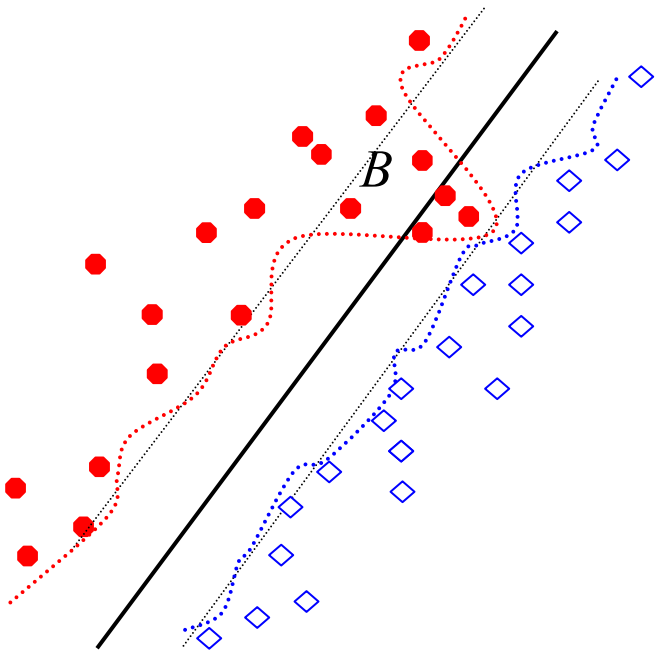

Figure 3.7 Single side negative examples

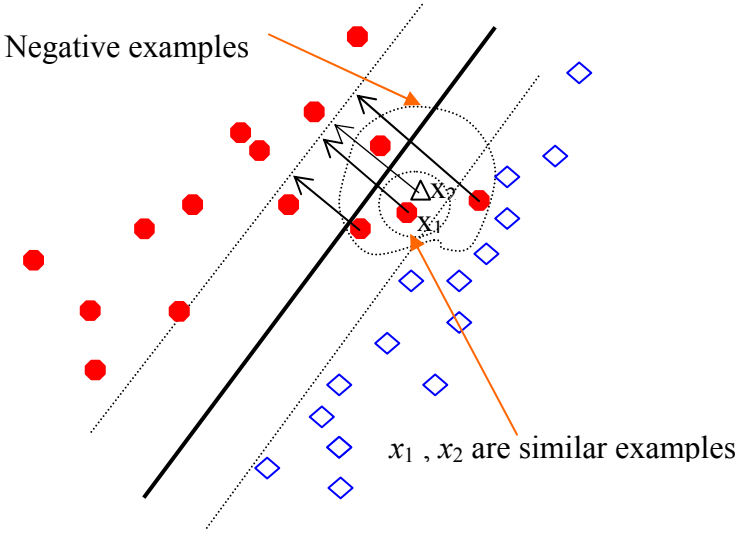

Figure 3.8 Patching a testing example in the directly compensatable pattern. Circle points are in class+1; rectangle points are class -1; triangle point is test point.

3.7.4 Not compensatable negative examples

The pattern of compensatable negative data shown on Figure 3.9 is not compensatable directly by patching a hypothesis. Positive and negative examples are interwoven on the area A where examples cannot be linear separated in the feature space. One idea is to re-use SVM again to only those examples in the area A. Since feature space is implicated by kernel function, examples cannot be gotten directly. Furthermore, the dimensionality of vector in the feature space is very high based on the analysis of applying a polynomial kernel on the section of Theoretical Analyzing on NDDCHA. The approach proposed here is still patching hypothesis. However, the difficulty is how to select a desired patching hypothesis in the testing phase.

Applying which patching hypothesis to testing examples is based on the vector similarity between a testing example and negative data set. As shown on Figure3.8, vector x_1 is in the negative data subset $\boldsymbol{N_i}$ which is similar with testing vector x_2, then x_2 will apply for the patching hypothesis used for $\boldsymbol{N_i}$. If any similar vector of x_2 cannot found, x_2 cannot not be compensated.

Sometimes it is very difficult to find a similar vector in the negative data subset as shown on Figure 3.10, x_2 is similar with $x_1 \in \{+1\}$ and $x_3 \in \{-1\}$. The direction of compensating for x_2 is opposite. Suppose real $x_2 \in \{+1\}$, if we choose x_1 as similar vector x_2, then result is desired. Otherwise, we choose x_3 as similar vector x_2, and then the result is even worse. To attach whether x_2 is in the class {+1} or {-1}, *k*-nearest neighbor method are used, which method is more precise than vector similarity by comparing a vector x_2 to a group of neighbor vectors in the negative examples.

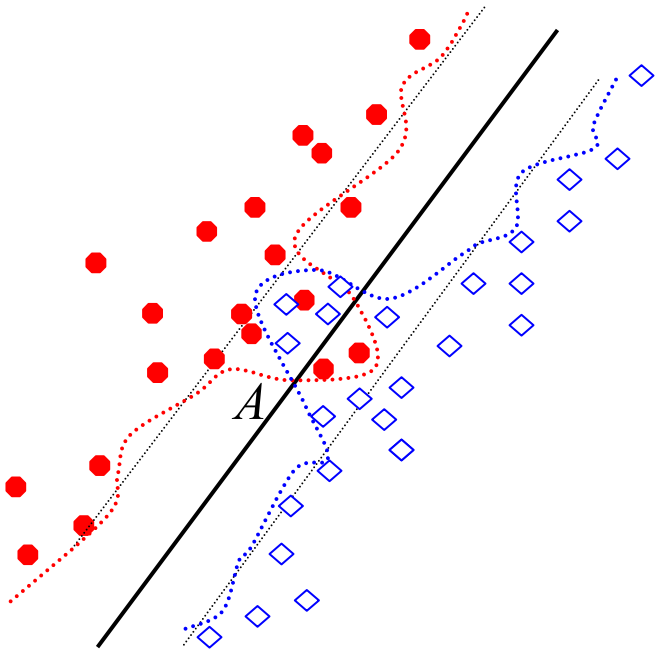

Figure 3.9 Interweaved positive and negative examples

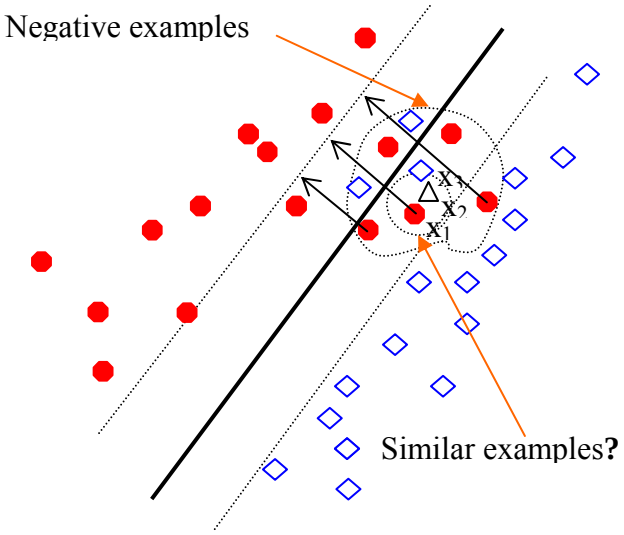

Figure 3.10 Patching a testing example in the non-directly compensatable pattern. Circle points are in class+1; rectangle points are class -1; triangle point is test point.

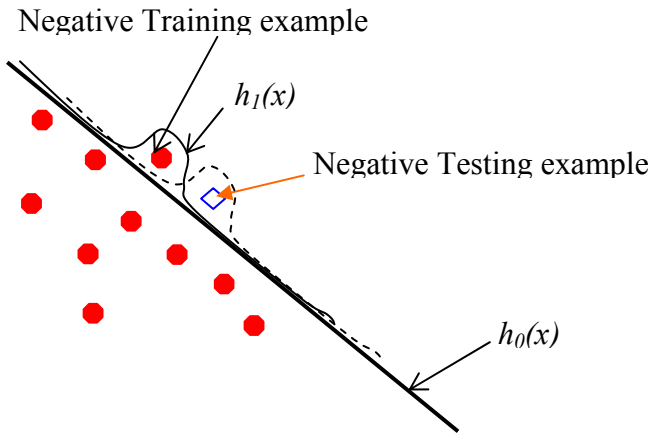

Figure 3.11 The negative training example is compensated by $h_1(x)$ when in the training phase, but negative testing example can not be compensated by $h_1(x)$.

3.7.5 Imbalanced examples

SVM minimizes the risk functional for all examples. It does not try to minimize distinctively the risk of class +1 examples or the risk of class -1 examples. This is because SVM adopts the assumption of examples which is i.i.d., and number of two classes {−1,+1} of examples is balanced. Because of this inherent insufficiency of SVM, SVM has less power on imbalanced examples than on well balanced examples. Figure 3.12 shows imbalanced examples, the number of class +1 examples is great than number of class -1. The hyperplane created by SVM will be prone to the side that has more examples than other side. Therefore, the predictive negative value is closed to zero if minority class is class -1 because SVM optimizes accuracy. In this case ROC metric can

let SVM focus on the minority class. In certain conditions 2-norm soft margin SVMs can maximize AUC[15].

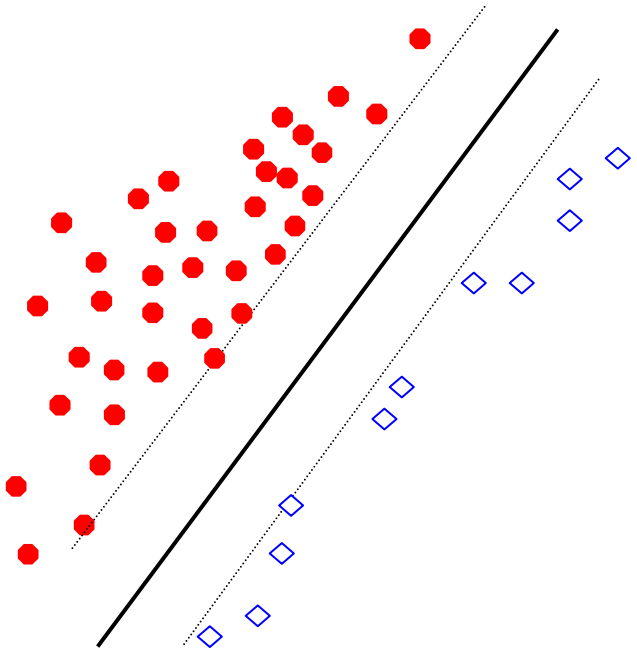

Figure 3.12 Imbalanced examples

To deal with imbalanced class problem, one could modify data distribution by using over-sampling and under-sampling[88]. Over-sampling replicates the minority class while under-sampling removes partial majority class. Both sampling makes training dataset balanced. The drawback of under-sampling is to lose some useful data. To overcome that, the support vectors can be kept and non-support vectors can be removed in the SVM because SVM is determined by support vectors. Over-sampling may bring overfitting. In this dissertation, under-sampling is adopted as shown on Figure 3.13. Under-sampling training data keeps all supports vectors and partial non-support vectors which depend on the number of negative data. Non-support vectors may not be included when support vectors area has more positive data.

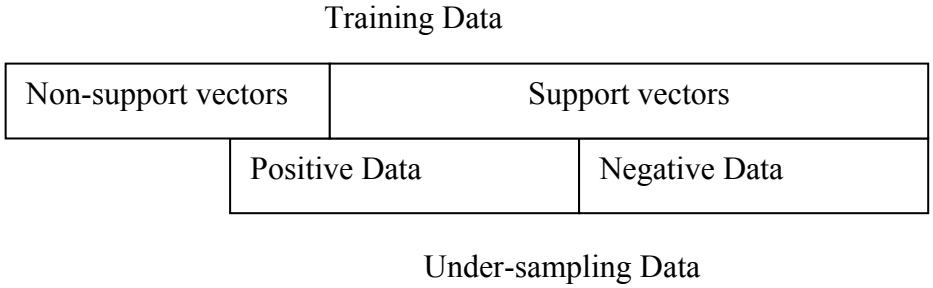

Figure 3.13 Under-sampling strategy

Yan et al proposed SVM ensemble to deal with imbalanced examples by combining small class examples with a piece of large class examples to form new *k* data sets and use *k* SVMs [89]. Each $h_i(x), i = 1..k$ is a decision function. Since majority voting and probability based combination assumes all classifiers are equal weights, the proposed strategy is creating a hierarchical SVM. The final classifier or hypothesis for an instance *x* is $h(x) = h(h_1(x), h_2(x), ..., h_k(x))$ as shown on Figure 3.14.

The bias *b* could be used as a regulation parameter to control the position of hyperplane. So far, the problem, how to recognize these cases above, is still remained. Imbalanced examples, outlier, and linear separatable examples can be detected intuitively. Our work focuses on the compensatable negative examples and interweaved positive and negative examples. Compenstable examples can be compensated by moving patching hyper-surface to the desired direction. Interweaved examples can be distinguished by *k*-nearest neighbor in the feature space. Vector similarity can be thought as 1-nearest neighbor.

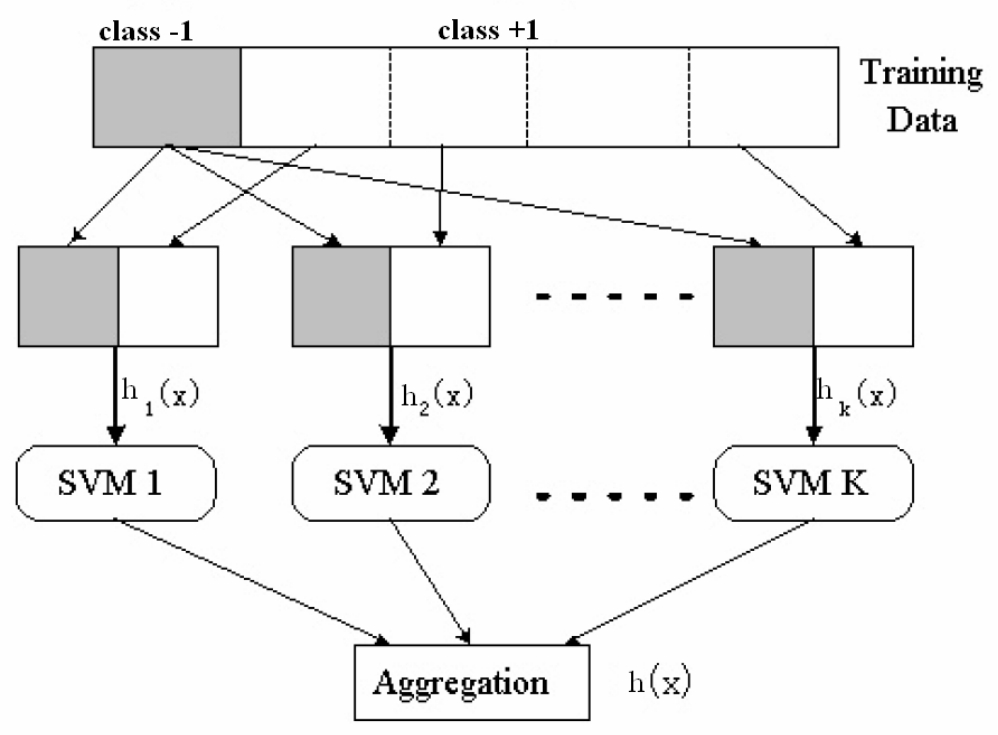

Figure 3.14 Architecture of Yan et al SVM ensembles

3.8 Compensating Hypothesis Approach

Compensating hypothesis approach proposed in this paper for the binary classification enhances the useful data information by mining negative data. This approach is based on the Support Vector Machines with 1+*k* times learning, where the base learning hypothesis is iteratively compensated *k* times. This approach produces a new hypothesis on the new data set in which, each label is a transformation of the label from the negative data set, further producing the child positive and negative data subsets in subsequent iterations. This procedure refines the model created by the base learning algorithm, creating *k* number of hypotheses over *k* iterations, as shown on Figure 3.15. *h(x,i-1)* is the $(i\text{-}1)^{th}$ patched hypothesis, i=1..k. A patching hypothesis $h^{(i)}(x)$ is patching hypothesis to compensate the base hypothesis. Area A belongs to not well distributed examples. If

h(x,i-1) fits the area A, *h(x,i-1)* will be overfitting. The compensated hypothesis is $h(x,i)=h(x,i-1)+ h^{(i)}(x)$. Note that *h(x,0)* is base hypothesis created by binary SVM.

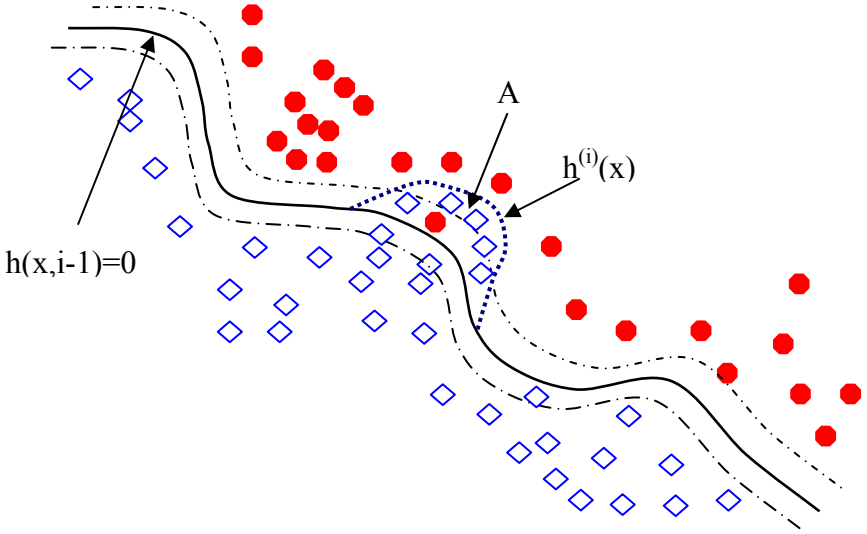

Figure 3.15 Compensating hypothesis approach

This approach is similarly applying for the Divide and Conquer principle. It perfectly accord with the philosophy of attacking the main issues first and then minor issues. The main issue is attacked by creating a main framework, which is a base hypothesis, to fit most of i.i.d. examples in the training data set. The minor issues are attacked by compensating hypotheses.

CHAPTER 4

ERROR DRIVEN COMPENSATING HYPOTHESIS APRROACH

It is impossible to select a perfect model for a practical problem without approximation error in a learning algorithm. Imagining that underlying function *f(x)* is a fluctuant terrain, it is hard to fit the terrain by using a huge size of carpet *h(x)*. The reason is that only training set and limited priori knowledge is available. The main idea to reduce the approximation error is to compensate the parts of not well-fit huge carpet by a sequence of small size of carpets $h^{(i)}(x)$ which is driven by the negative data subset of training data.

4.1 Negative Data

Let training data set $S_0=S$ in the definition (1-1). It can be partitioned into two subsets according to a divider *d(h,x,y)*, $(x,y) \in S_0$, where *h(x)* is produced by a base learning algorithm. One subset is positive subset $S_1^{\#}$, which is a set of well-separated examples from S_0 by the hypothesis $h^{(0)}(x)=h(x)$. The remaining of S_0 is the negative subset S_1 satisfying $S_0 = S_1^{\#} + S_1$. The negative data does not mean the data is wrong or corrupt. What negative data can be known is that the hypothesis cannot make it well-separated. The negative data is not limited to those incorrect data alone, but may also contain a few correctly classified data as well. The definition of negative data is given on 3.1. Boosting uses an equation $a_t = \frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t})$, where ε is the error in the last iteration, to weight those negative examples. In NDDCHA, the divider is strongly related to the

number of support vectors, size of training examples, VC dimension, and radius of the sphere containing all examples, which can be determined by cross- validation method. Not all negative examples contain useful information. The outliers obviously are junk. Thereby, the negative examples with confidence grater than 1.0 will not be considered in the negative patching learning.

To the μ-negative data, the criterion to judge which example needs to compensate is below:

$$y_i h(x_i) \le \mu \ , i = 1..l \tag{4-1}$$

because the misclassified examples meet $y_i h(x_i) \le 0$ and not well separated examples meet $0 \le y_i h(x_i) \le \mu$. The compensation value is

$$y_i - h(x_i) \ , i = 1..l \tag{4-2}$$

E.g. assume $\mu = 0.4$ predicting value of an instance $x_1$is $h(x_1)$ = -0.3 and target label is y_1=-1, $y_1 h(x_1) = -1 * (-0.3) = 0.3 < \mu$, then instance x_1 needs to compensate. The compensation value of the instance x_1 is $y_1 - h(x_1) = -1 - (-0.3) = -0.7$.

4.2 Training Phase

Let ***X*** be a collection of input vectors from training set ***S***, and ***Y*** be a vector consisting of all labels of training set

$$\mathrm{X} = \{\mathrm{x} \mid (\mathrm{x}, \mathrm{y}) \in \mathrm{S}\} \quad \text{and} \quad \mathrm{Y} = \{\mathrm{y} \mid (\mathrm{x}, \mathrm{y}) \in \mathrm{S}\} \ . \tag{4-3}$$

Let $h^{(i)}(x)$ are the patching negative models working on the training negative subset S_i and $d^{(i)}(h^{(i)},x,y)$ be dividers. And S_i is the negative subset of S_{i-1} according to $d^{(i-1)}(h^{(i-1)},x,y)$.

Here *h(x,i)* is the comprehensive patching model. And *h(x,k)* is the final model providing to testing procedure.

$$\begin{cases} h(x,0) = h^{(0)}(x) \\ h(x,i) = h(x,i-1) + h^{(i)}(x) \end{cases} \quad for\ i = 1..k\ . \tag{4-4}$$

The sign + in above expression (4-4) provides an overlap operation for two models. Therefore the hypothesis h(x) of the NDDCHA approach is $h(x,k) = \sum_{i=0}^{k} h^{(i)}(x),\ for\ k > 0$.

The training data sets are defined as follows,

$$\begin{cases} S_0 = S \\ S_i = \{(x, \Delta y_i) \in S_{i-1} | x \in X, d^{(i-1)}(h^{(i-1)}, x, \Delta y_{i-1}), \\ \qquad \Delta y_i = \Delta y_{i-1} - h(x, i-1), and\ \Delta y_0 \in Y\} \\ S_i^{\#} = S_{i-1} - S_i \qquad for\ i = 1..k \end{cases} . \tag{4-5}$$

The labels on the training subset S_i are the differences or residuals of predicated labels and expected labels. For instance, a vector *(x,y)=(0.9,+1)* has one attribute 0.9 and label +1 from training data. The predicting label y' of x is 0.01. Although (x,y) is correctly classified because y'=0.01>0 is in the class +1, (x,y) is not well separated. On the next pass learning, the vector *(x,y)* becomes (0.9, +1-0.01)=(0.9, 0.99) as the new training data. The hypothesis is produced by training on the residual data since the idea of NDDCHA is to compensate the base hypothesis each time. Since above algorithm is iterated over k times, it has to be regression learning algorithm. There are a total of *1+k* passes in this algorithm. $S_i^{\#}$ are the positive data subsets and do not change during the training. The algorithm has two phases, training and testing as shown in Figure 4.1 and 4.2. The final training output is $\bigcup_{i=1}^{k+1} S_i^{\#}$. In the training phase, the hierarchy of training is a

chain.

4.3 Learning Termination Criteria

In every step, training is driven by negative data and it produces a series of hypotheses $h^{(i)}(x)$. The key point in the training is the learning termination criteria *TC(k,S)* and depends on how large the value of *k* is. There are many possible ways to determine *k*, out of which, three are mentioned here. In the first case, *k* is taken as the number

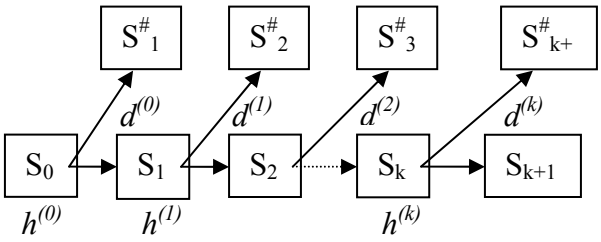

Figure 4.1 Training phase: S_i is negative data subset, $S^{\#}_i$ is the positive data subset, $h^{(i)}(x)$ is the patching model or hyper-surface, and *d(i)* are dividers, for i=1…k

of iterations when the size of S_k is less than the number of input attributes $|S_k| \leq |x|$. In the second case, *k* is taken as the number of iterations when the difference size between S_{k-1} and S_k ,i.e., $|S_{k-1}|-|S_k| \leq \mu$ is small enough, here μ is a positive integer. In the third case, *k* is taken as the number of iterations when *h(x, k)* gives the expected output with the specified accuracy.

4.4 Testing Phase

In the testing phase, the hypotheses from training are used to create patching data to compensate the base hypothesis. The key point in the testing phase is to determine the suitable patching hypothesis. The function of vector set similarity VS accepts two data sets S_i and T_{i-1}, one from the training data set and the other from the testing data set, to generate a subset of T_i from T_{i-1}. As a result, each vector $x_1 \in T_{i-1}$ becomes similar to at least one vector $x_2 \in S_i$, denoted to $vs(x_1, x_2) \geq \delta$, where $\delta \in [0,1]$ is the degree of similarity. For example, when $x_1 = x_2$, then $vs(x_1, x_2) = 1$, and when $x_1 \neq x_2$, then $vs(x_1, x_2) = 0$. Here P_i predicts labels on negative data set T_i

$$\begin{cases} T_0 = T \\ T_i = VS(S_i, T_{i-1}),\ if\ \ \forall x_1 \in T_{i-1},\ \exists x_2 \in S_i, vs(x_1, x_2) \geq \delta,\ \delta \in [0,1],\ for\ i = 1..k \end{cases} \tag{4-6}$$

$$\begin{cases} P_0 = \{(x, y) \mid x \in T_0, y = h^{(0)}(x)\} \\ P_i = \{(x_1, y) \mid x_1 \in T_{i-1}, x_2 \in S_i, vs(x_1, x_2) \geq \delta, y = h(x_1, i)\},\ for\ i = 1..k \end{cases} \tag{4-7}$$

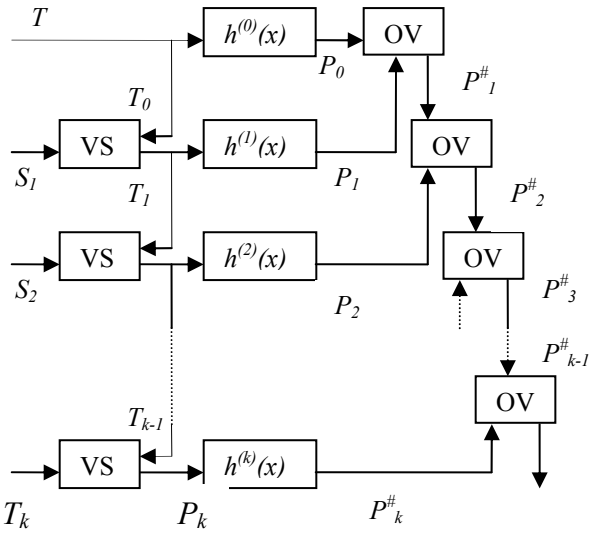

Figure 4.2 Testing phase.

VS is the module of vector-similarity, *OV* is the module of overlapping. S_i is negative data from training phase. $T_0=T$ is testing data set. T_i has the similar vectors or elements between S_i and T_{i-1} P_i is the patching hyper-surface. $P^{\#}_i$ is compensated testing outputs. $i=1..k$. $P^{\#}_k$ is final testing output.

In above expressions, δ is the regulating parameter to control the degree of two vectors' similarity. It can be seen that, T_i is similar to S_i, so that $h^{(i)}(x)$ can be used for testing T_i .to generate the values of P_i . These values are overlapped on to compensate labels as $P^{\#}_i=OV(P^{\#}_i,P^{\#}_{i-1})$. The final ouputs $P^{\#}_k$ are the predicting labels. For instance, Euclidean distance $||x_1-x_2||$ can be used as vector similarity function, where x_1 is in the testing set T_{i-1} and x_2 is in the negative training data subset S_i. The vector similarity function $vs(x_1,x_2)$ is $1.0-||x_1-x_2||$ if x_1 and x_2 are normalized. The output labels which are compensated value are given as follows.

$$\begin{cases} P_0^{\#} = P_0 \\ P_i^{\#} = OV\,(P_{i-1}^{\#}, P_i) \end{cases} \qquad i = 1..k\ . \tag{4-8}$$

It can be seen that in the training phase the learner uses the hypotheses $h(x)=h(x,i)$ together with the partitioner function $p^{(i)}(h^{(i)},x,y)$ as divider, generating a positive group $S^{\#}(k)=\bigcup_{i=1}^{k+1} S_i^{\#}$ and a negative group $S_{k+1}=S-S^{\#}(k)$. We find a subset of testing data T_i, which is similar to Si and use the hypotheses produced on Si for testing T_i. The '+' operation is one case of OV function, and hence the final testing result is treated as a summation of overlapping function $P_k^{\#} = \sum_{i=1}^{k} P_i^{\#}$.

4.5 Discussion of Vector Similarity in the Feature Space

Compensating an example strongly depends on the function of vector similarity. There are four possibilities of compensating an example:

1. expected to compensate, the result compensates
2. expected to compensate, but not compensate
3. expected not to compensate, the result compensates
4. expected not to compensate, and not compensate

Case 1 improves accuracy of base hypothesis. Case 2 and 4 are harmless. In worst case, they keep the accuracy of base hypothesis. Case 3 is dangerous case and need further study.

Those testing examples which meet the conditions of Theorem 2.1 will be compensated. The vector-similarity plays extremely important role in NDDCHA learning. To apply the repairing hyper-surface, the first thing is to find out which vectors in testing data set need to be compensated. The vector-similarity is used to find the relationship of vectors in the negative data subset S_i and testing data subset T_{i-1}. Only those vectors in T_{i-1} with high similarity to those in S_i need to be compensated, that is $S_i \propto T_{i-1} \geq \upsilon$, v is the degree of similarity.

The evaluation of vector-similarity in the classification application in the SVMs cannot be obtained directly because of the fact that the similarity in the feature space is different with in the input space $\boldsymbol{X}$. The connection between input space and feature space was discussed to find a vector in the input space by given a vector in the feature space[90]. The main idea is that the comparison of two vectors x_1 and x_2 are computed on

the feature space. There are three criteria of similarity to be considered to determine the similar degree of x_1 and x_2 if:

1. x_1 and x_2 are located on the same side of the hyperplane on either class +1 side or class -1 side.
2. Distances of x_1 and x_2 to the hyperplane are close, then, x_1 and x_2 have similar separable capability. The difference of two distances is less than δ_1
3. $||x_1 - x_2||$ is small enough, which is less than δ_2.

To compute the vector-similarity in the feature space $h(x,i)$, suppose $h(x,i)=\boldsymbol{w}^T\boldsymbol{x}+\boldsymbol{b}$. The condition 1 is equivalent to $h(x_1,i)*h(x_2,i) >0$; and condition 2 is

$$
\begin{aligned}
&\left|dist(x_1,i) - dist(x_2,i)\right| \\
&= \left|\frac{h(x_1,i)}{\|w\|} - \frac{h(x_2,i)}{\|w\|}\right| \\
&= \frac{1}{\|w\|}\left|h(x_1,i) - h(x_2,i)\right| \\
&\le \delta_1 \\
&\Rightarrow \left|h(x_1,i) - h(x_2,i)\right| \le \delta_1\|w\| = \theta
\end{aligned}
$$

Note that θ is constant. The difference of $\left|h(x_1,i) - h(x_2,i)\right|$ is bound on $||w||$ that means the limitation is not tight if $||w||$ is large. This phenomenon shows again that SVM maximizing the $||w||2/2$ is correct. Condition 3 is $d_i(x_1,x_2) = \sqrt{K_i(x_1,x_1) - 2K_i(x_1,x_2) + K_i(x_2,x_2)} \le \delta_2$ based on the equation (2-6), where K_i is the kernel function used in the ith training.

The vector similarity algorithm in the feature space of SVM is

function vector_similarity(x_1, x_2, i)

if $h(x_1,i)*h(x_2,i) <0$ **then**

return $+\infty$

else

return 1-max(sigmoid(h(x_1,i) - h(x_2,i)), sigmoid(d_i(x_1,x_2)))

The difference of confidence *h(x_1,i) - h(x_2,i)* and distance $d_i(x_1,x_2)$ need to normalize so that the output of function vector_similarity has comparability among examples. One example of normalization method is a sigmoid function which outputs a value between 0 and 1.

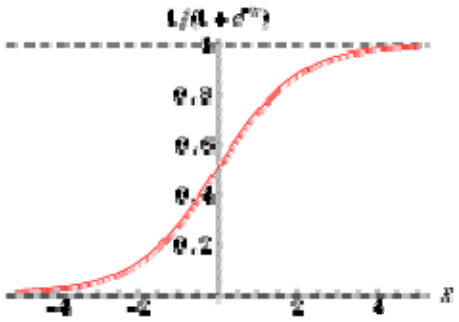

Figure 4.3 Sigmoid function

$$sigmoid\ (x) = \frac{1}{1+e^{-x}}. \tag{4-9}$$

Algorithm *k*-nearest neighbor is an instance learning method which learns hypothesis only upon a new instance querying. The significant advantage of instance-based learning is local approximation. The disadvantage of that is that the cost of classifying a new example can be high. When vector-similarity function *vs* is called, *k*-nearest neighbor algorithm retrieves *k* number of similar related examples from negative data to determine how strong relationship between x_1 and x_2.

In the NDDCHA, suppose $x_t \in T_i, (x_n, y_n) \in S_i$, where T_i is testing subset and S_i is negative subset in the training data. The function **vs**(x_t,x_n, *i*) >0 if x_n is in the class +1, otherwise **vs**(x_t,x_n, *i*) <0 if x_n is in the class -1.

vs(x_t, x_n, i)

if y_n <0

then return – vector_similarity(x_t, x_n, i)

else

return + vector_similarity(x_t, x_n, i)

To enhance to power of vector similarity function, k-NEAREST NEIGHBOR algorithm is introduced. A testing example is $x_T \in T_{i-1}$ and k examples nearest to x_T are $x_1, x_2, ...x_k \in S_i$. The distances between x_T and $x_1, x_2, ...x_k$ are $d_1, d_2, ..., d_k$. The distance weighted k-NEAREST NEIGHBOR gives estimated value of vector similarity Δy_T to x_T:

$$\Delta y_T = \frac{\sum_{j=1}^{k} d_j \Delta y_j}{\sum_{j=1}^{k} d_j}. \tag{4-10}$$

where Δy_j is the compensated value in the i[th] training phase. To normalize the vector similarity value $vs \in [-1,+1]$, an extended sigmoid function is introduced:

$$extended\ \_sigmoid\ (x) = 2sigmoid\ (x) - 1 = \frac{2}{1+e^{-x}} - 1. \tag{4-11}$$

4.6 Algorithm of NDDCHA

The procedure of NDDCHA has three parameters, S_0 is the training data set; T_0 is the testing data set; and δ is the degree of vector similarity. The return value of the algorithm is the predictive labels of testing data set. Six subroutines are invoked:

$h^{(i)}(x)$=LEARN(S_i)

P_i=PREDICT(T_i, $h^{(i)}(x)$)

$S^{\#}_{i+1} \cup S_{i+1}$= DIVIDER($S_i$, $h^{(i)}(x)$)

T_i = VS(S_i, T_{i-1},δ)

$P^{\#}_i$ = OV($P^{\#}_{i-1}$,P_i)

TC(k,S)

LEARN denotes training routine to get the model or hypothesis; PREDICT routine predicts the labels of given data set and model. DIVIDER is the routine to divide training data set into positive and negative data subset by given the hypothesis and the function partitioner *d(h,x,y)*. In each pass, the function VS and DIVDER could be different. The algorithm is described below as pseudo-code. The code given below follows the sequence or procedure developed and shown above in this section.

NDDCHA (S0, T0, δ)

▷Learning phase

S[0] ← S0

h[0] ← LEARN(S[0])

i ← 0

repeat

  i ←i+1

  ($S^{\#}$[i], S[i]) ← DIVIDER(S[i-1], h[i-1])

  h[i] ← LEARN(S[i])

until TC(i,S)

k ← i ▷ the number of iteration in repeat loop

▷ Testing phase

T[0] ← T0

P[0] ← PREDICT (T, h[0])

$P^{\#}[0] \leftarrow P[0]$

for i←1 **to** k **do**

(T[i],V[i]) ← VS(S[i],T[i-1], δ)

if T[i] ≠ Φ ▷ T[i] is not empty set

then P[i] ← PREDICT(T[i], h[i])

P#[i] ← OV(P#[i-1], P[i], V[i])

return P#[k]

DIVIDER(S[i-1], h[i-1])

X ← Δ**Y**←Φ ▷ initialize to empty set

foreach (x,y) **in** S[i-1] **do** ▷ let *(X,ΔY) be S[i-1]*

X ← **X** ∪ {x}

Δ**Y**← Δ**Y** ∪ {y}

S[i] ← Φ

foreach (x, Δy[i-1]) **in** (X,ΔY**)** **do**

Δy[i] ← PREDICT(x, h[i-1])

if d(h[i-1], x, Δy[i-1])

then S[i] ← S[i]∪{(x, Δy[i])}

Δy[i] ← Δy[i-1]- Δy[i] ▷ *update ΔY*

$S^{\#} \leftarrow S[i-1] - S[i]$

return (S#[i], S[i])

VS(S[i],T[i-1], δ)

$T[i] \leftarrow \Phi$

$V[i] \leftarrow \Phi$

foreach x1 **in** T[i-1] **do**

  foreach x2 **in** S[i] **do**

    if $|vs(x1,x2, i)| \geq \delta$

    then $T[i] \leftarrow T[i] \cup \{x1\}$

      V$[i] \leftarrow V[i] \cup \{sign(vs(x1,x2, i))\}$

      break

return (T[i], V[i])

To apply the NDDCHA learning algorithm, it is required that the partitioner function *d(h,x,y)*, terminate criteria function *TC(k, S)* and vector similarity $vs(x_1,x_2, i)$ to be provided. The performance of NDDCHA very much depends on the selection of partitioner and vector-similarity function which needs a priori knowledge of learning task.

4.7 NDDCHA Algorithm Simulation

NDDCHA algorithm is implemented by Perl which uses a slight modified SVM^{light} [83, 84] as the base learning algorithm including learning and classifying modules. Three binary classification case studies, on *musk2*[91], *breast cancer*, and *cement*, have been analyzed. Before the cases were studied, the three functions, partitioner function *d(h,x,y)*,

terminate criteria function *TC(k,S)* and vector similarity $vs(x_1,x_2)$ needed to be defined. To simplify the complexity of computation, the partitioner was defined on the feature space by *d(h, x, y)*= **iff**(h(x) < ε,**true**, **false**), ε□[0,0.5]. And *TC(k,S)* was defined by *TC(i, S[i])=**iff**(|S[i]|≤|x|, **true**, **false**)*. Vector similarity Euclidean distance method was used for *musk2*; and feature space method was used for *breast cancer* and *cement*. Feature space vector similarity approach is adopted in our work because of the fact that

TABLE 4.1 Comparison of three data sets

	musk2	*cancer*	*cement*
SVM Average accuracy%	90.30	89.92	70.92
NDDCHA average accuracy%	94.92	90.86	72.56
Increase accuracy%	4.62	0.94	1.64

the data in two cases of three has missing value. The euclidean method is obviously not suitable for this situation since this method will make a vector with missing value in a high degree of similarity. The threshold δ of vector similarity is 0.7 in the dataset *musk2*, and δ =0.6 is used for both datasets *breast cancer* and *cement*.

The data sets used in case studies *musk2* and *breast cancer* are from UCI Knowledge Discovery in Databases (KDD) Archive [82], and the data set used in *cement* is from Wangchang cement company. The *n*-fold cross-validation is performed in each case. The original data is randomly divided into *n* groups; each group has the same or

approximate size. Each group does not have the similar distribution of classes. One group is used as testing data and the remaining n-1 groups are grouped as training data. The validation procedure runs n times and each time the testing data is from the i^{th} group, *i=1...n*. The average result of *n*-fold is the final accuracy given as shown in TABLE 4.1. It should be mentioned that the parameters of SVM are not well tuned because the cases are used to show the NDDCHA has better performance than the base learning algorithm.

TABLE 4.2 Simulation on the data set *musk2*

No	TP	FN	FP	TN	M	C	A%	TP	FN	FP	TN	M	C	A%	I%
1	39	63	0	558	63	597	90.45	83	19	14	544	33	627	95.00	5.03
2	41	61	0	558	61	599	90.76	83	19	13	545	32	628	95.15	4.84
3	38	64	0	558	64	596	90.30	84	18	14	544	32	628	95.15	5.37
4	36	66	0	558	66	594	90.00	87	15	22	536	37	623	94.39	4.88
5	39	63	0	558	63	597	90.45	93	9	15	543	24	636	96.36	6.54
6	37	65	0	558	65	595	90.15	88	14	14	544	28	632	95.76	6.22
7	39	63	0	558	63	597	90.45	81	21	19	539	40	620	93.94	3.86
8	37	64	0	559	64	596	90.30	86	15	19	540	34	626	94.85	5.04
9	35	66	0	558	66	593	89.98	78	23	15	543	38	621	94.23	4.73
10	36	65	0	558	65	594	90.14	81	20	17	541	37	622	94.39	4.71
Aver	38	64	0	558	64	596	90.30	84.4	17	16	542	34	626	94.92	5.12

The table has left and right sides. The left side is the result of regular SVM. The right side is the result of NDDCHA. TP = true positives, FN = false negatives, FP = false positives, TN = true negatives, M = misclassified = FN+FP, C=correct classified, A%=accuracy = C/(C+M)*100% and I%= the accuracy improved= (right A% - left A%)/left A% *100%

*Musk*2 has 6,598 examples with 168 attributes and no missing value. 10-fold cross-validation shows that the accuracy of prediction is 90.3% by using SVM RBF model with parameter γ=0.5. Before SVM is used to predict the test data, all data

including training set and test set are normalized by *unit normal scaling* approach [92]. The operation of normalization improves the average accuracy improvement from *72%* to *90.3%*. The unit normal scaling approach is applied on all data sets. The number of support vectors is very large near the size of training set, the ratio of support vectors to examples *r=88%,* which means that there are lots of noise involved in the data set, or the hypothesis space is less than the target space. In a high noise problem, many of the slack variables become non-zero, and the corresponding examples become support vectors [93]. The model $h^{(0)}(x)$ classify training set and positive data *P* and negative data *N* are divided by the partitioner *d(h, x, y)*= **iff**(h(0)(x) < 0.3,**true**, **false**), which implies all data

TABLE 4.3 Simulation on the data set *Cancer*

No	TP	FN	FP	TN	M	C	A%	TP	FN	FP	TN	M	C	A%	I%
1	28	12	0	74	12	102	89.47	29	11	0	74	11	103	90.35	0.98
2	22	16	0	76	16	98	85.96	24	14	0	76	14	100	87.72	2.05
3	41	9	0	64	9	105	92.11	43	7	0	64	7	107	93.86	1.90
4	32	10	0	72	10	104	91.23	36	6	0	72	6	108	94.74	3.84
5	26	16	0	71	16	97	85.84	28	14	0	71	14	99	87.61	2.06
Aver	30	13	0	71	13	101	88.92	32	10	0	71.4	10	103	90.86	2.17

The labels have the same meaning as TABLE 4.2

predicting output in (-0.3, 0.3) are negative subset. The number of compensating is one. The size of *N* is 612, data set *N* is trained as hypothesis $h^{(1)}(x)$ and then $h^{(1)}(x)$ classifies the testing data set again. The accuracy is as low as 15.7% and uses only 2 support vectors. The reason is the size of training set *N* is too small. However, $h^{(1)}(x)$ with *0.5%* ratio of number of support vectors and the number of training data gives 100% accuracy to classify group *N*. It is interesting to note that only two support vectors alone can give such a high accuracy on the negative training data set *N*. Dietterich et al. [94] proposed this algorithm, iterated-discrim APR, in their paper. Compared to their other seven algorithms described in different papers, the iterated-discrim APR demonstrated the best

performance, resulting in the correct prediction of 89.2% with confidence interval [83.2%-95.2%] by using 10-fold cross-validation with 102 examples on Musk data 2. The NDDCHA using 10-fold cross-validation with 612 examples gave the high predicating confidence interval with [94.23, 95.76] as shown in Table 4.2.

TABLE 4.4 Simulation on the data set *Cement*

No.	TP	FN	FP	TN	M	C	A%	TP	FN	FP	TN	M	C	A%	I%
1	46	53	14	132	67	178	72.65	55	44	21	125	65	180	73.47	1.13
2	33	50	10	151	60	184	75.41	43	40	19	142	59	185	75.82	0.54
3	36	60	10	138	70	174	71.31	45	51	16	132	67	177	72.54	1.73
4	23	67	12	142	79	165	67.62	36	54	18	136	72	172	70.49	4.25
5	35	69	10	130	79	165	67.62	48	56	16	124	72	172	70.49	4.25
Aver	34.6	59.8	11.2	138.6	71	173	70.92	45.4	49	18	131.8	67	177.2	72.56	2.38

The labels have the same meaning as TABLE 4.2.

Cancer has 569 examples with 32 attributes and missing value. 5-fold cross-validation shows that the accuracy of prediction has been improved from 88.92% to 90.86% by using two learning approaches as shown in Table 4.3. Since there are missing values in this data set, the vector similarity works on the feature space. The kernel function of SVM is mapping x from input space into feature space. Note that the i^{th} attribute of vector x in the feature space is the combination of attributes of x in the input space, and then it overcomes the effects due to missing value. The same vector similarity method is used for *cement* data set for the same reason as *Cancer*. The *cement* has 1221 examples with 11 attributes and missing data. The partitioner in the *cancer* and *cement* is also $d(h, x, y)$= **iff**(h(0)(x) < 0.3,**true**, **false**). The similarity degree is $vs(x_1,x_2) \geq 0.99$. The 5-fold cross-validation shows that the testing accuracy is increased from 72.56% to 74.94%. From Table 4.2,4.3 and 4.4, it is observed that NDDCHA can always improve

the testing accuracy in every fold testing. The time cost on *Musk2* is around 30 minutes in the M-Pentium 2.0 GHz, Windows XP PRO computer. It is less than 5 minutes in the other two data sets.

It is shown that the vector similarity function is sensitive to learning problems in the cases studied. The final performance is dependent on the degree of vector similarity. The Euclidean distance method works on input space which means that the predicting value should be similar if input vectors are similar. However, Euclidean distance method treats every attribute in the vector with equal contribution. This is not true for the real world problem because some attributes are more significant. Therefore, all attributes must be at least normalized before data is fed into the learning machine. When missing value exists in the data set, the distances between the vector with missing attributes and normal vector will be small. As a result, these two vectors seem to be similar, but, in fact, they are not. In general, the Euclidean method is suitable for large size of examples without missing data such as in data set *musk2*. Conversely, the data set *cancer* has small size examples and missing attributes. The good vector similarity approach is to compare the similarity of two vectors in the feature space. The reason is that *n*-dimensional input vector $\boldsymbol{x}$ is projected into a high *m*-dimensional space using nonlinear function $\phi(\mathbf{x}) : R^n \rightarrow R^m$, and then missing attributes in the input space do not appear to be missing on the feature space. The feature space is determined by kernel functions, so the similarity is related to learning model.

CHAPTER 5

STATISTICAL NEGATIVE EXAMPLES LEARNING APPROACH

Statistical negative examples learning approach (SNELA) has two or three stages of learning including the base learning, the negative learning and the boosting learning for binary classification as shown in Figure 5.1.

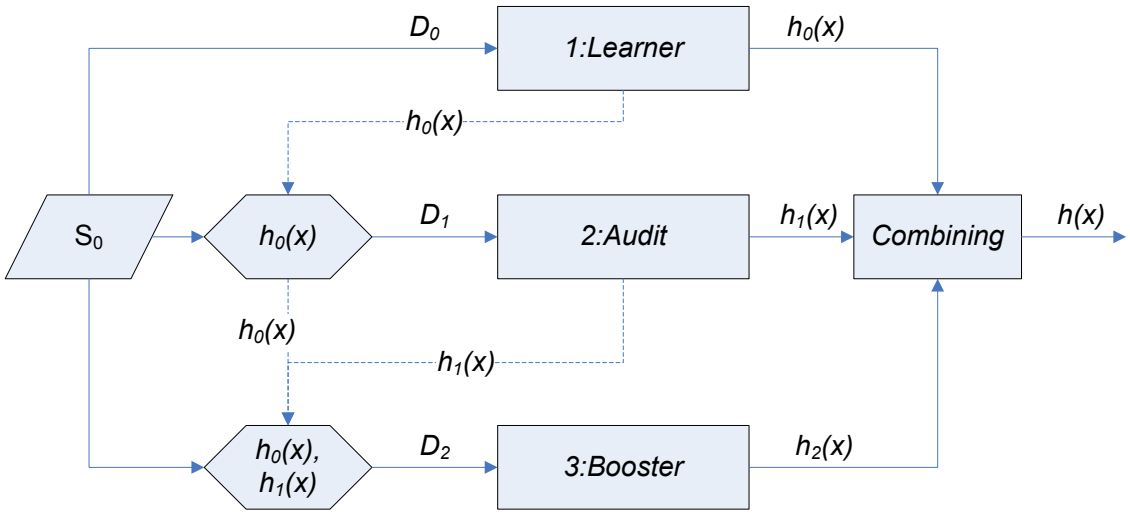

Figure 5.1 Scheme of SNELA

The base learning which is named *learner*, employs a regular support vector machine to classify main examples and recognize which examples are negative. The negative learning which is named *audit*, judges the predicting results of *learner*, works on the negative training data which is strongly imbalanced to predict which instance could be negative based on *learner*. When an instance is predicted by *audit* as negative, this instance is claimed to be misclassified by *learner*. The next step is compensation, where

we move this instance into opposite class either from class +1 to class -1 or vice versa if the instance is negative. Furthermore, boosting learning *booster* is applied when audit does not have enough accuracy to judge *learner* correctly. *Booster* works on the training data subset with which *learner* and *audit* do not agree. The classifier for testing is the combination of learner, audit and booster. The *classifier* for testing a specific instance returns the *learner*'s result if *audit* acknowledges *learner*'s result and *learner* agrees with *audit*'s judgment, otherwise returns the *booster*'s result. If *audit* has enough accuracy, boosting learning may be skipped.

5.1 Concept of True Error

The notation $\Pr_{x\in\mathscr{D}}[\pi(x)]$ means the probability of Boolean expression $\pi(x)$ holding on instance x drawn from input space X according to distribution $\mathscr{D}$. In general, the equation below is held[2]:

$$\Pr_{x\in\mathscr{D}}[\pi(x)] = \sum_{x\in X} D(x)\Pr[\pi(x)] \tag{5-1}$$

where *D(x)* is the probability of instance x chosen under distribution D.

Concept c is a Boolean function on some spaces of instances. The probability $\Pr_{x\in\mathscr{D}}[h(x)\neq c(x)]$ is called the *error* of hypothesis h on concept c under the distribution $\mathscr{D}$. The instance x is drawn from input space X according to the distribution $\mathscr{D}$. If the error is equal to or less than ε, then h is called ε-close to the target concept c under $\mathscr{D}$. In the binary classification $\Pr_{x\in\mathscr{D}}[h(x)\neq c(x)]$ is equivalent to $\Pr_{x\in\mathscr{D}}[yh(x)<0]$ where y is target label of instance x.

The true error concept is introduced here to demonstrate the compensation condition in the worst case of negative learning. The error is related to the distribution of

examples and hypothesis specified. The following is the definition of the true error.

Definition 5.1: The **true error** $err_{\mathcal{D}}(h)$ of hypothesis *h* with respect to target label *y* and distribution $\mathcal{D}$ is the probability that a randomly generated instance *x* drawn from $\mathcal{D}$ is misclassified.

$$err_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}}[h(x) \neq y] = \varepsilon \tag{5-2}$$

where the notation $\Pr_{x \in \mathcal{D}}$ is denoted by the probability taken over the instance distribution $\mathcal{D}$ and the pair (*x, y*) is an underlying labeled example.

The training error $err_S(h)$ is the ratio of the number of misclassified examples over the total number of examples on the training data set S in terms of a hypothesis h(x) that is defined below.

$$err_S(h) \equiv \frac{1}{|S|} \sum_{(x,y) \in S} 1_{h(x) \neq y} = \varepsilon_S \tag{5-3}$$

The expression 1_π is defined to be 1 if the predicate π holds and 0 otherwise. |S| is the size of data set S. Suppose training examples *S* are chosen i.i.d. according to the distribution $\mathcal{D}$. If there exists an algorithm *A* learning on *S* to output a hypothesis *h* that probability at least $1-\delta$ is ε-close to the target concept *c* under $\mathcal{D}$ for given parameter $\delta \in [0,0.5)$ and $\varepsilon \in [0,0.5)$. It is denoted by

$$h^A(S, \varepsilon, \delta) \equiv \Pr_{x \in \mathcal{D}}\{err_S(h) > \varepsilon\} < \delta \tag{5-4}$$

The training error could be zero. For example, we choose a high VC dimension kernel and a large number of regularation parameter in the soft margin of SVM. The predicting error is the ratio of the number of misclassified examples over the total number of examples on the testing data set.

The testing error or predicting error $err_T(h)$ is the ratio of the number of misclassified examples over total number of instance on the testing data set T in terms of a hypothesis *h(x)* which is defined below:

$$err_T(h) \equiv \frac{1}{|T|}\sum_{x \in T} 1_{h(x) \neq y} = \varepsilon_T \tag{5-5}$$

The expression 1_π is defined to be 1 if the predicate π holds and 0 otherwise. |T| is the size of testing data set T, the label *y* is the underlying target value of instance *x*.

A data set S is i.i.d. drawn from D divided into four parts as *TP*, *TN*, *FP* and *FN* in terms of a fixed binary hypothesis h(x) by which the true error is great than training error $\varepsilon_S = err_S(h)$, such that $S = TP \bigcup TN \bigcup FP \bigcup FN$, $P = TP \bigcup TN$, $N = FP \bigcup FN$ and $err_{\mathcal{D}}(h) \geq \varepsilon_S$ because the *S* is a sub- set of the whole space with distribution D.

$$TP \cup TN = \{(x, y) \mid y_i h(x) > 0, i = 1..l\} \tag{5-6}$$

$$FP \cup FN = \{(x, y) \mid y_i h(x) < 0, i = 1..l\} \tag{5-7}$$

The true error is not training error. The true error is a real metric of generalization capacity. A learner is consistent if it outputs hypothesis that perfectly fits the training examples. |*P*| is the number of examples in the positive set whereas |*N*| is the number of examples in the negative data set. For any classification algorithm, the result is meaningless if |*P*| is less than |*N*|. The random learner can give error of 0.5. Since $|P| > |N|$, then $\varepsilon \in [0, 0.5)$ where ε is true error of hypothesis *h(x)*. The true accuracy is $1 - \varepsilon$.

To deal with imbalanced class problem, one could modify data distribution by using over-sampling and under-sampling[88]. Over-sampling replicates the minority class while under-sampling removes partial majority class. Both sampling makes training

dataset balanced. The drawback of under-sampling is to lose some useful data. Over-sampling may bring overfitting. Given an imbalanced data set $D_0 = P_0 \cup N_0$, where P_0 is the subset of class +1 and N_0 is the subset of class -1, the size of P_0 is greatly larger than the size of N_0, that is $|P_0| >> |N_0|$. We have three methods to alleviate the unbalancing including under-sampling, over-sampling and the combination of above two sampling which is named hybrid-sampling.

The under-sampling technique is shown on the Figure 5.2. The result of under sampling is $D_1 = P_1 \cup N_1$. The data subset N_1 is simply duplicated from N_0, while P_1 is extracted randomly from P_0.

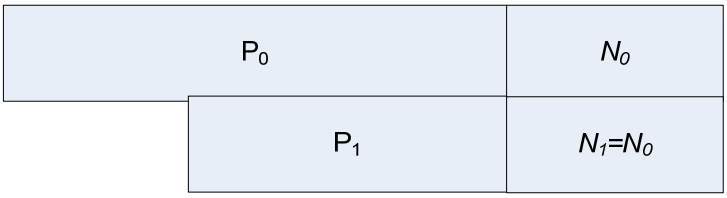

Figure 5.2 Under-sampling strategy

The under-sampling coefficient and the over-sampling coefficient are defined as below:

$$\gamma = \frac{|P_1|}{|P_0|} \tag{5-8}$$

$$\rho = \frac{|N_1|}{|N_0|} \tag{5-9}$$

where $\gamma \in (0,1]$ and $\rho \in [1,+\infty)$. In under-sampling ρ=1 while in the S is a sub- set of the whole space with distribution over-sampling $\gamma = 1$. The parameter c is the ratio of negative examples N over positive examples P which describes the degree of unbalancing. Parameter r is the reciprocal of c.

$$c = \frac{|N_1|}{|P_1|} \qquad r = \frac{1}{c} = \frac{|P_1|}{|N_1|} \tag{5-10}$$

Where $c \in (0,1]$, means that the sampling technique cannot make N_1 larger than P_1.

The over-sampling technique is shown on Figure 5.3. The data subset P_1 is simply duplicated from P_0, while N_1 is created by copying all examples in N_0 and duplicating some examples in N_0.

Figure 5.3 Over-sampling strategy

The hybrid-sampling combines the under-sampling and over-sampling strategies as shown on Figure 5.4. We get $|P_1| = \gamma |P_0|, \gamma < 1$ and $|N_1| = \rho |N_0|, \rho > 1$.

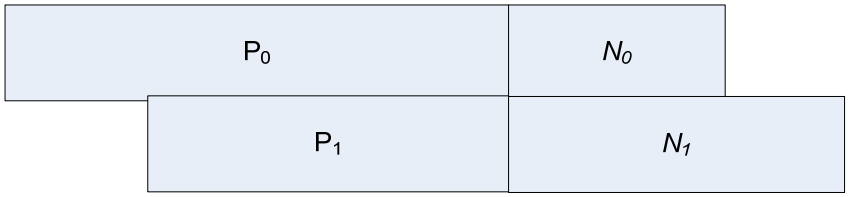

Figure 5.4 Hybrid-sampling strategy

Suppose $D_0(x)$ and $D_1(x)$ are the possibilities of instance x being chosen under D_0 and D_1 respectively as shown on Figure 5.5. P_0 and P_1 are in class $+1$ whereas N_0 and N_1 are in class -1. P_1 and N_0 is separated by $h_0^{SVM}(S_0, \varepsilon_0, \delta_0)$. $D_1 = P_1 + N_1$ is the result of sampling from D_0. The hypothesis $h_1(x) = h_1^{SVM}(D_1, \varepsilon_1, \delta_1)$ if SVM learning algorithm is used here.

Let $p_i = \Pr[h_i(x) \neq y]$ be the chance for instance x is misclassified by hypothesis $h_i(x)$. The true error of hypothesis $h_0(x)$ is ε_0 learned from data set S_0. P_0 is in class +1

correctly classified by $h_0(x)$ while N_0 is in class -1. The hypothesis $h_1(x)$ with error ε_1 is learned from D_1 which is the output of sampling from D_0. The sampling strategy used here is randomly extracting or duplicating the examples from D_0. When domain knowledge is used, for example, only support vectors are extracted during sampling, it is not in this case. Then the following equation is obtained[95]:

$$D_1(x) = \frac{D_0(x)}{\gamma + \rho}[\gamma \times \frac{1 - p_0(x)}{1 - \varepsilon_0} + \rho \times \frac{p_0(x)}{\varepsilon_0}] \tag{5-11}$$

The hypothesis $h_1(x)$ with error ε_1 is gotten by training on S_1. We have the following equation:

$$\begin{aligned} 1 - \varepsilon_1 &= \sum_{x \in X} D_1(x)[1 - p_1(x)] \\ &= \frac{1}{\gamma + \rho}\{\frac{\gamma}{1 - \varepsilon_0}\sum_{x \in X} D_0(x)[1 - p_0(x)][1 - p_1(x)] + \frac{\rho}{\varepsilon_0}\sum_{x \in X} D_0(x)p_0(x)[1 - p_1(x)]\} \\ &= \frac{1}{\gamma + \rho}[\frac{\gamma}{1 - \varepsilon_0} \times t + \frac{\rho}{\varepsilon_0} \times u] \end{aligned} \tag{5-12}$$

Then we get

$$t = \frac{(1 - \varepsilon_0)[\varepsilon_0(1 - \varepsilon_1)(\gamma + \rho) - u\rho]}{\gamma\varepsilon_0} \tag{5-13}$$

The data set sampling is to construct a new data set S from another data set D according to the controlling parameter α and sampling technique *sampling*.

$$S \equiv sampling(D, \alpha), \text{such that } |S| = \alpha|D| \tag{5-14}$$

The data set S could be the subset of D if under-sampling technique is used $P_1 \equiv UnderSampling(P_0, \gamma)$. Or data set D is a subset of S on the over-sampling $N_1 \equiv OverSampling(N_0, \rho)$.

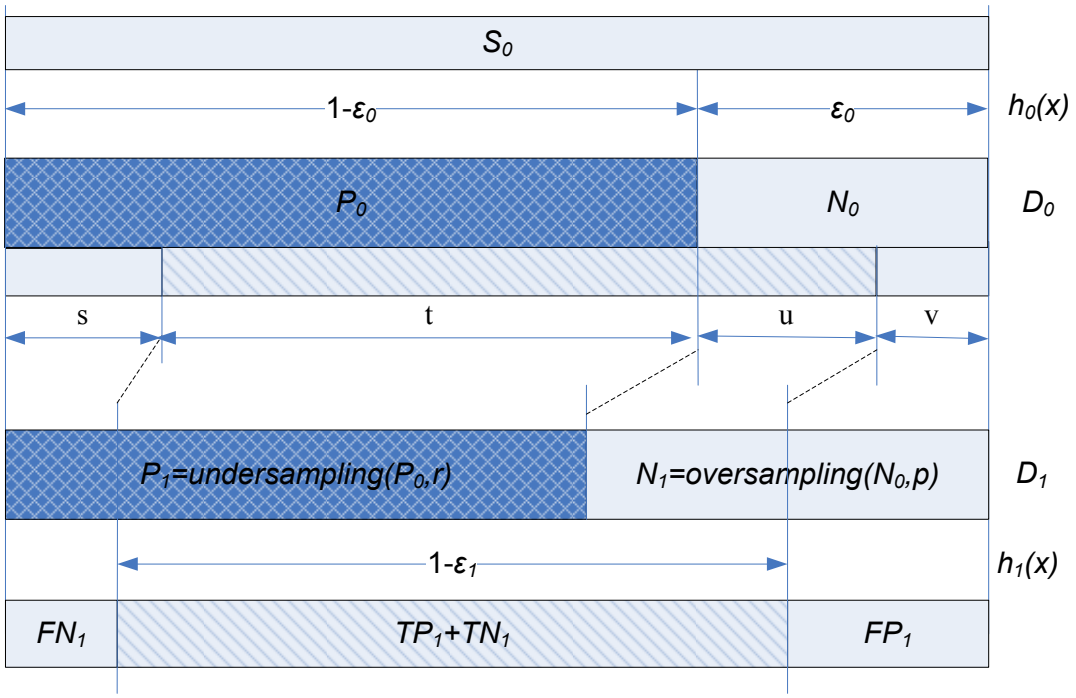

Figure 5.5 Possibility of sampling data

5.2 Introduction to Statistical Negative Examples Learning Approach

The hypothesis of the naïve learning of SNELA is a decision function, which considers an instance either in class +1 or class -1. The extended learning of SNELA considers the confidence which is associated to an example. The scheme of two stages SNELA is shown on Figure 5.6. The figure shows a scheme of negative example learning. S_0 is the training data set. The learner gets the hypothesis $h_0(x) = h_0^{SVM}(S_0, \varepsilon_0, \delta_0)$ by base learning algorithm SVM. The data sets P_0 and N_0 are positive and negative data of S_0 respectively separated by $h_0(x)$. S_1 is the training data set for *audit*, which is under sampling from positive data P_0 and over sampling from negative

data N_0. The negative hypothesis $h_1(x)$ is *audit* $h_1(x) = h_1^{SVM}(S_1, \varepsilon_1, \delta_1)$. TP_1 and TN_1 can be used to correct testing error created by base hypothesis.

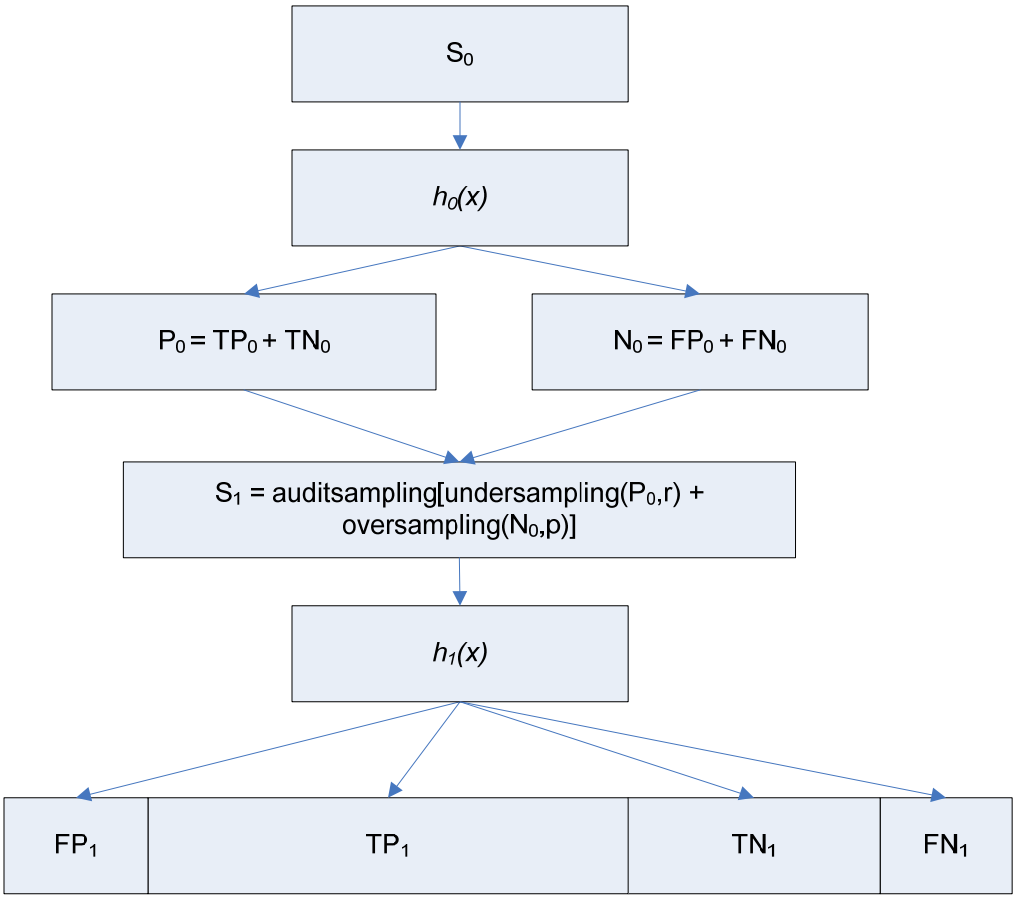

Figure 5.6 The scheme of two stages learning including base and negative learning.

Since $|P_0| >> |N_0|$, construction compensated data set S_1 for negative learning is to use under sampling on P_0 whereas over sampling on N_0. The data set S_1 is not simplified from the union of the results of under and over sampling. It was translated into compensated data set format for training by the function auditsampling() defined in (5-16).

The section of SNELA considers that the hypothesis outputs a confidence value. Given a training data set $S_0 = \{(x_i, y_i)\}, i = 1..l$, testing data set $T_0 = \{x_i\}, i = 1..m$ and a parameter $1 > \mu > 0$, an SVM is trained on S_0 to output hypothesis $h_0(x)$. When S_0 is not separable, negative data subset N_0 of S_0 is not empty, as shown on the figure below.

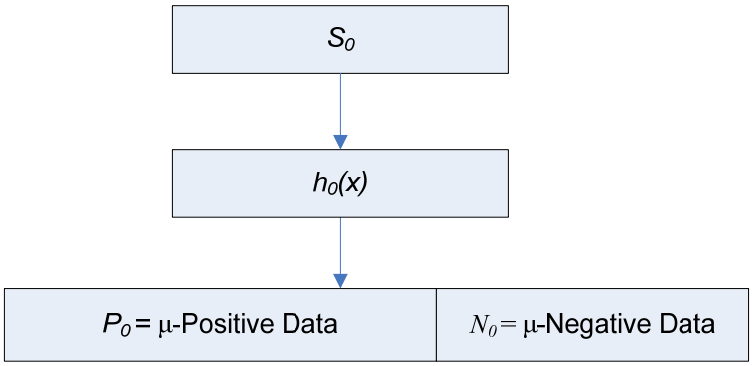

Figure 5.7 The scheme of base learning. $S_0 = P_0 \bigcup N_0$

The testing data set T_0 is predicted by $h_0(x)$ to output correctly classified instances T_P and misclassified instances T_N since $h_0(x)$ is not capable to separate all examples as shown on the figure below.

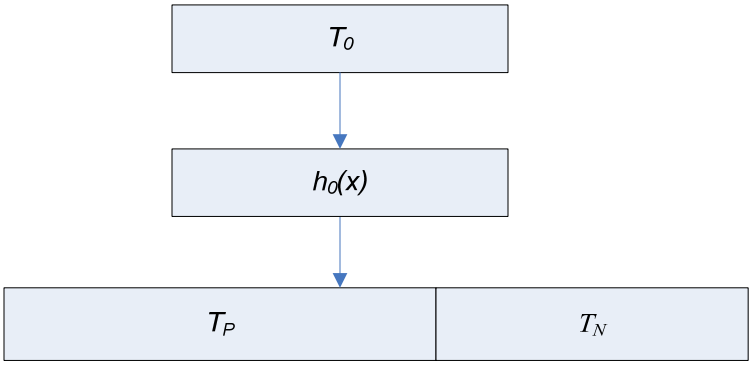

Figure 5.8 The scheme of base testing. T_P is the correctly predicted instances and T_N is incorrectly predicted instances. In this stage, T_P and T_N are unknown, where $T_0 = T_P \bigcup T_N$.

If some of instances in the T_N were known in advance, those instances could be inversed into the opposite class. For example, the target label of an instance x_5 is class +1. The instance x_5 is a misclassified instance in the T_0 by using hypothesis $h_0(x)$, so instance x_5 is predicted to be class -1. In this stage of practical application, whether the instance x_5 is correctly classified or misclassified is unknown if we do not have known target label. If x_5 can be predicted to be misclassified instance by another hypothesis $h_1(x)$ with a certain confidence, say 70% possibility, x_5 can be corrected from class -1 into class +1. For that reason, the key point is how to construct the hypothesis $h_1(x)$.

In order to construct the hypothesis $h_1(x)$, a training data set S_1, named *compensated data set*, is required to be constructed. The information including training data set S_0, hypothesis $h_0(x)$, positive examples P_0 and negative examples N_0 are available before $h_1(x)$ is constructed. The main idea is to extract partial examples from P_0 and all examples from N_0 because $|P_0|>>|N_0|$. Another reason to include all negative examples is because the goal of $h_1(x)$ is to predict misclassified instances which are strong related to the negative N_0 example.

Figure 5.9 Construction of compensated training data S_1 for $h_1(x)$ using under-sampling strategy

The compensated training data set S_1 is defined below:

$$\text{Let } P_1 = undersampling(P_0, \gamma) \text{ and } N_1 = N_0 \tag{5-15}$$

$$auditsampling(S_1) \equiv \\ S_1 = \{(x \mid h_0(x), +1) \mid (x, y) \in P_1\} \cup \{(x \mid h_0(x), -1) \mid (x, y) \in N_1\} \tag{5-16}$$

The hypothesis $h_1(x)$ is learned from training data set S_1. If over-sampling is used, let $N_1 = oversampling(N_0, \rho)$. Testing data set T_0 is predicted by both hypotheses $h_0(x)$ and $h_1(x)$.

$$T_1 = \{(x, y) \mid x \in T_0, y = h_0(x)\} \\ T_2 = \{(x, y) \mid x \in T_0, y = h_1(x)\} \tag{5-17}$$

Assume the error of hypotheses $h_0(x)$ and $h_1(x)$ are ε_0 and ε_1 respectively. If ε_0 and ε_1 meet some criterions, which will be discussed in the sections 5.4, $h_1(x)$ can improve the predicting of $h_0(x)$ on the testing data set T_0. If $h_0(x)$ has less confidence on predicting instance x to be y while $h_1(x)$ has large confidence to say x is negative, then the predicted label y by $h_0(x)$ should be inversed; let y be $-y$.

5.3 Analysis of Two Stages Learning

Theoretically, true error ε is less than predicting error ε_T, $\varepsilon \le \varepsilon_T$, while predicting error ε_T is greater than training error ε_S practically, $\varepsilon_S \le \varepsilon_T$, if the hypothesis is fixed. This is resulted from the approximation and estimation error of learning machine. The true error is hard to get without exact data distribution information. Therefore, in most cases, predicting error is the substitution of true error as approximate estimation because the true error is impossible to get in almost all applications. When we work on the worst case of negative learning, training error is used to replace predicting error or true error. The reason is if a hypothesis with small training error cannot make final predicting get desired accuracy, the true error must be greater than the desired error. Therefore, the

following discussion assumes true error, training error and predicting error are not discriminated.

- The training data set S_0 has l examples, $l=|S_0|$.
- The true errors $\varepsilon_0 \in [0,\ 0.5), \varepsilon_1 \in [0,\ 0.5)$ are the error of hypotheses of $h_0(x)$ and $h_1(x)$. Then, $|P_0|>|N_0|$ and $|P_1|>|N_1|$ for the base training data sets $S_0 = P_0 \bigcup N_0$ and compensated training data set $S_1 = P_1 \bigcup N_1$.

The parameter c is the ratio of negative examples N over positive examples P. Parameter r is the reciprocal of c.

$$c = \frac{|N_1|}{|P_1|} \qquad r = \frac{1}{c} \tag{5-18}$$

5.3.1 Under-Sampling

In under-sampling, P_1 is a subset of P_0 where all examples in P_1 are extracted from P_0 randomly. N_1 is small data set comparing to P_1. Therefore, N_1 keeps all examples from N_0 except outliers:

$$\begin{aligned} &P_1 \subseteq P_0 \\ &N_1 = N_0 \\ &|P_1| = r|N_1| \end{aligned} \tag{5-19}$$

The minimum of c and maximum of r in the training phase of base leaning in terms of hypothesis $h_0(x)$ are also defined below. $|N_0|$ could be considered as fixed value.

$$\begin{aligned} \mathrm{c}_{\min} &= \frac{|N_1|_{\min}}{|P_1|_{\max}} = \frac{|N_0|}{|P_0|} = \frac{l\varepsilon_0}{l(1-\varepsilon_0)} = \frac{\varepsilon_0}{1-\varepsilon_0} \quad \Rightarrow \ \mathrm{c} \in [0,1] \\ r_{\max} &= \frac{1}{\mathrm{c}_{\min}} = \frac{1-\varepsilon_0}{\varepsilon_0} = \frac{1}{\varepsilon_0} - 1 \quad \Rightarrow \ \mathrm{r} \in [1,+\infty) \end{aligned} \tag{5-20}$$

In the negative learning stage, under-sampling strategy is adopted. So let $|N_1| = |N_0| = l\varepsilon_0$ because $|S_0| = l$ and $|N_0| << |P_0|$. For example, $|N_0| = 0.25|P_0|$ if the accuracy of $h_0(x)$ is 80%. The training data of negative learning is $S_1 = P_1 \bigcup N_1$. The size of S_1 and the predicted data subsets in terms of hypothesis $h_1(x)$ are listed below:

$$|P_1| = r|N_1| = lr\varepsilon_0 \qquad |N_1| = l\varepsilon_0 \tag{5-21}$$

$$\begin{aligned} |FP_1| &= lr\varepsilon_0\varepsilon_1 & |TP_1| &= lr\varepsilon_0(1-\varepsilon_1) \\ |TN_1| &= l\varepsilon_0(1-\varepsilon_1) & |FN_1| &= l\varepsilon_0\varepsilon_1 \end{aligned} \tag{5-22}$$

To make negative learning useful, the number of correctly classified examples must be greater than the number of misclassified examples, including misclassified negative examples and positive examples. Then, the following inequality needs be met.

$$\begin{aligned} & |TN_1| > |FP_1| + |FN_1| \\ \Rightarrow\ & l\varepsilon_0(1-\varepsilon_1) > lr\varepsilon_0\varepsilon_1 + l\varepsilon_0\varepsilon_1 \\ \Rightarrow\ & 1-\varepsilon_1 > r\varepsilon_1 + \varepsilon_1 \\ \Rightarrow\ & \varepsilon_1 < \frac{1}{2+r} \end{aligned} \tag{5-23}$$

The explanation is shown in Figure **Error! Reference source not found.**.

The strong condition $|TN_1| > |FP_1| + |FN_1|$ needs to be met in order to improve the accuracy of base learning. The weak condition $|TN_1| > |FN_1|$ must be met too. The $|FP_1|$ examples are misclassified and still are not judged correctly in negative learning. This tells us that the upper bound of true error ε_1 in negative learning stage is $\frac{1}{2+r}$. If the training error of negative learning is greater than the upper bound, negative learning should not be processed. As shown in the Figure 5.11, the relationship of true error of negative learning and the ratio of the size of positive and negative examples in

base learning indicates that goal is not hard to achieve, because the area under curve is small comparing to the area over curve.

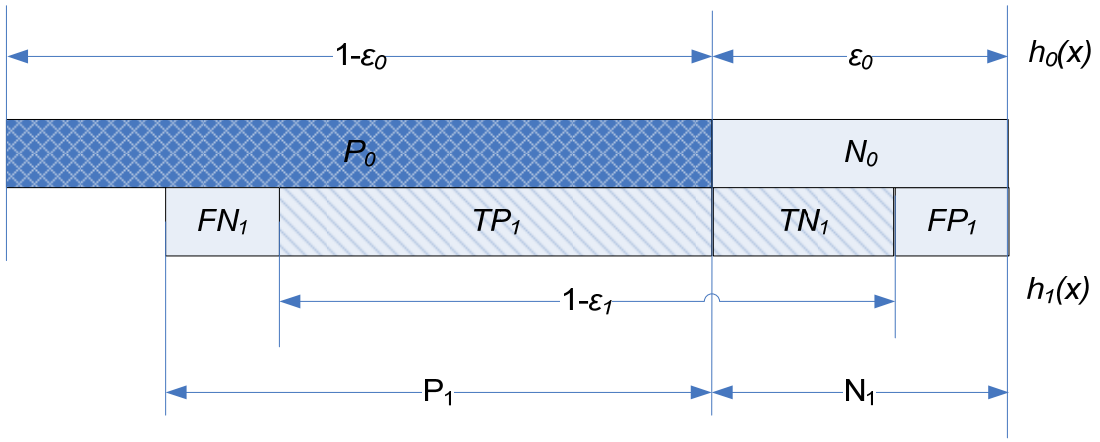

Figure 5.10 The number of correctly judged examples in the negative leaning is $|TN_1|$. The $|FN_1|$ examples are correctly classified in base learning but not be judged correctly.

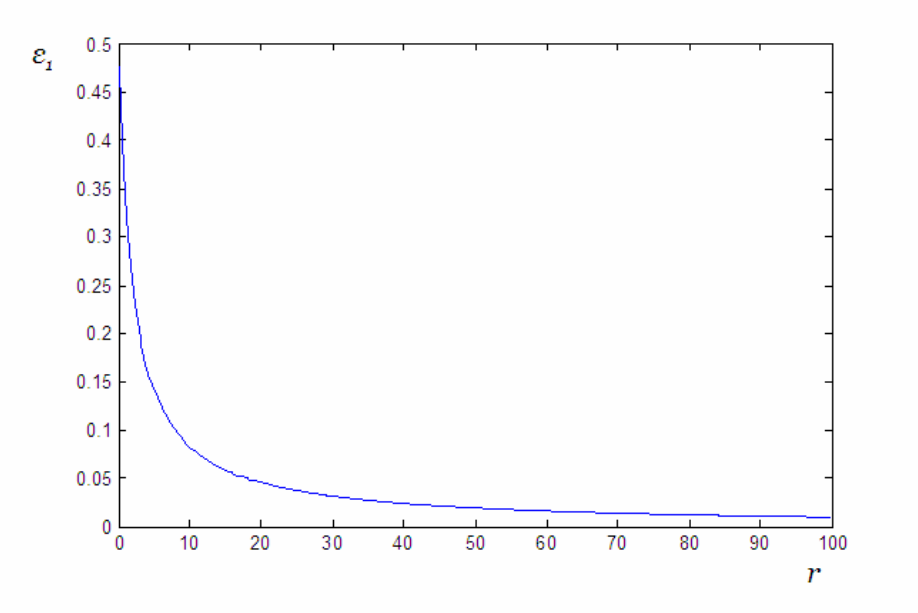

Figure 5.11 $r-\varepsilon_1$ relationship diagram

In the Figure 5.11, the x-axis is r which is the ratio of the size of positive and negative examples in base learning stage; y-axis is ε_1 which is the true error of hypothesis in negative learning stage. When ε_1 falls into the area under the curve, the improvement is made in the predicting negative examples.

The under-sampling strategy shows how to select the parameter r is more critical. In the worst case, r is equal to r_{max}. In that case, the following inequality is gotten.

$$\varepsilon_{1,\max} < \frac{1}{2+r_{\max}} = \frac{1}{2+\frac{1}{\varepsilon_0}-1} = \frac{\varepsilon_0}{1+\varepsilon_0} \tag{5-24}$$

The relationship between $\varepsilon_0 - \varepsilon_{1,\max}$ is shown below.

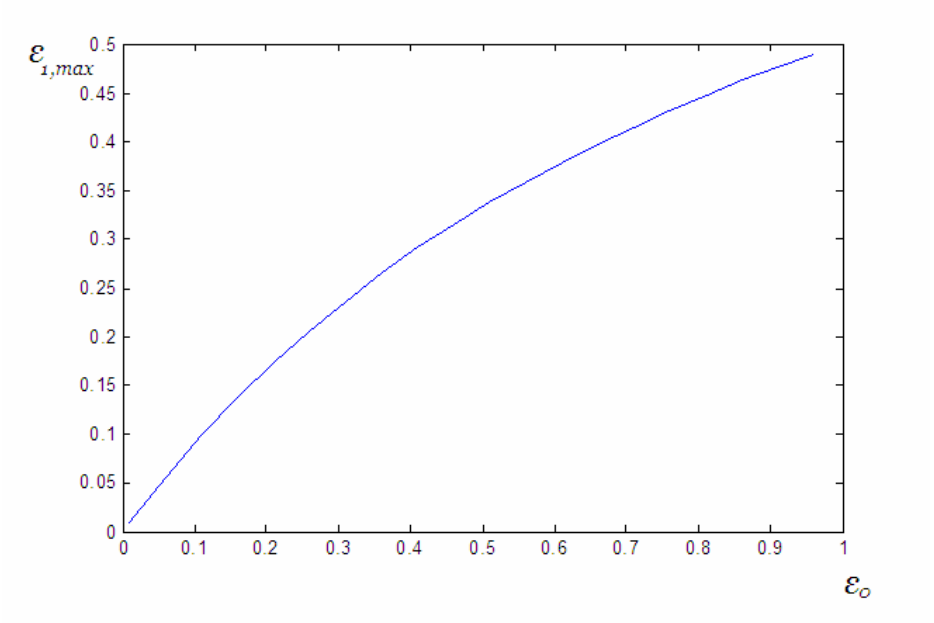

Figure 5.12 $\varepsilon_0 - \varepsilon_{1,\max}$ relationship diagram, the performance is improved in the predicting negative examples when ε_1 falls into the area under the curve

To understand the given analysis, here are a couple examples:

(1) ε_0=0.20, r=2, |T|=200

$$\Rightarrow r_{\max} = \frac{1-0.20}{0.20} = 4$$

$$\Rightarrow \varepsilon_{1,\max} < \frac{0.20}{1+0.20} = 0.1667 < \varepsilon_0$$

$$\Rightarrow \varepsilon_1 < \frac{1}{2+r} = \frac{1}{2+2} = 0.25$$

In base learning, $|N_0| = \varepsilon_0 |T| = 0.20 * 200 = 40$ examples are probably misclassified and $|P_0| = (1-\varepsilon_0)|T| = 0.8 * 200 = 160$ are correctly classified. In negative learning, $|FN_1| = \varepsilon_1 |N_0| = 40 * 0.25 = 10$ examples in N_0 are not compensated while $|TN_1| = (1-\varepsilon_1)|N_0| = 40 * 0.75 = 30$ examples in N_0 are compensated. In the meantime, $|FP_1| = \varepsilon_1 r |N_0| = 80 * 0.25 = 20$ examples in P_0 are correctly classified in base learning while they are compensated wrongly in negative learning. So the total misclassified examples are $|FP_1| + |FN_1|$ =30. The negative learning makes $|TN_1|$ examples correctly classified and $|FP_1|$ examples misclassified. The inequality | $|TN_1| > |FP_1| + |FN_1|$ has to be met; otherwise, the negative learning is useless. This example shows that it is possible for SNELA to improve final performance provided true error of negative learning is less than 0.1667 no matter what the ratio r is. On the other hand, SNELA cannot improve performance when true error is greater than 0.1667 if no positive examples are suppressed. The data set is close to be balanced when r=2. In this case, true error ε_1=0.33 is allowed to be larger than $\varepsilon_{1,\max}$ =0.1667.

(2) ε_0=0.10, r=2,

$$\Rightarrow \varepsilon_{1,\max} < \frac{0.10}{1+0.10} = 0.091$$

$$\Rightarrow \varepsilon_1 < \frac{1}{2+2} = 0.25$$

5.3.2 Over-Sampling and Hybrid Sampling

The over-sampling and hybrid-sampling are illustrated on the figures as follows.

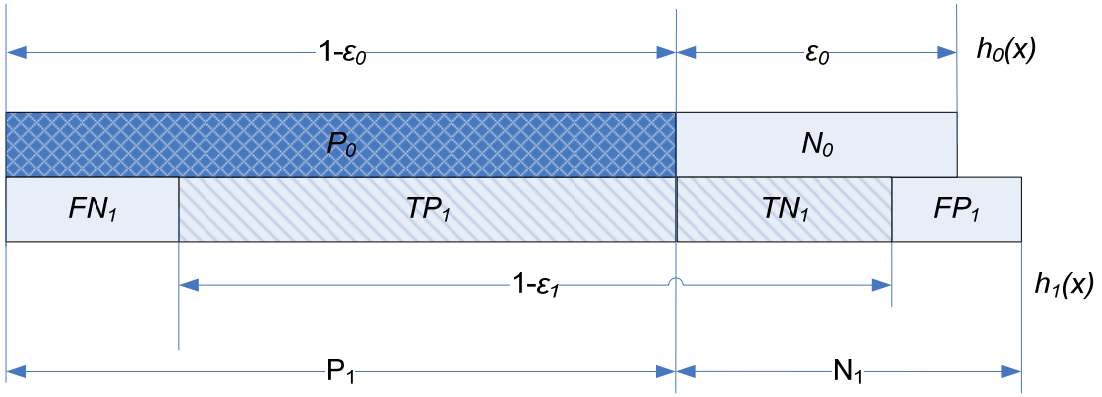

Figure 5.13 over-sampling strategy

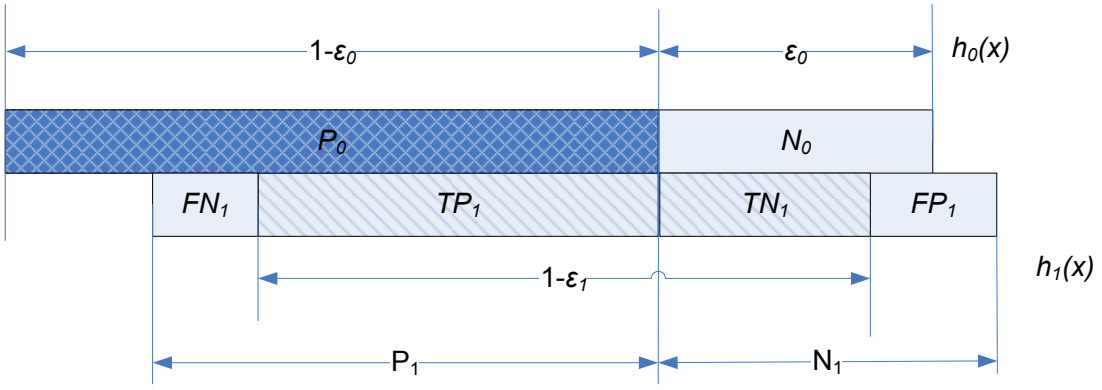

Figure 5.14 under-sampling and over-sampling could be considered as the special case of hybrid-sampling.

In hybrid-sampling as shown on above figure, let $P_1 = undersampling(P_0, \gamma)$ and $N_1 = oversampling(N_0, \rho)$. Then the following equations can be gotten:

$$\begin{aligned} |P_1| &= \gamma |P_0| \\ |N_1| &= \rho |N_0| \\ |P_1| &= r |N_1| \end{aligned} \tag{5-25}$$

$$\Rightarrow r = \frac{|P_1|}{|N_1|} = \frac{\gamma |P_0|}{\rho |N_0|} = \frac{\gamma (1-\varepsilon_0)}{\rho \varepsilon_0} \tag{5-26}$$

In over-sampling γ = 1, ρ >1 is used whereas γ < 1, ρ =1 in under-sampling. The strong condition still maintains the same in different sampling techniques as it can be seen on the inequality (5-27).

$$\begin{aligned} & |TN_1| > |FP_1| + |FN_1| \\ \Rightarrow & l\varepsilon_0 \rho (1-\varepsilon_1) > l\rho \varepsilon_0 \varepsilon_1 + l\gamma (1-\varepsilon_0)\varepsilon_1 \\ \Rightarrow & 1-\varepsilon_1 > \varepsilon_1 + \frac{\gamma (1-\varepsilon_0)\varepsilon_1}{\rho \varepsilon_0} = \varepsilon_1 + r\varepsilon_1 \\ \Rightarrow & \varepsilon_1 < \frac{1}{2+r} \end{aligned} \tag{5-27}$$

Then the discussion in the last section is still true here. In order to balance the data, the condition $|P_1| \geq |N_1|$ should be met.

$$\begin{aligned} & |P_1| \geq |N_1| \\ \Rightarrow & \rho |P_0 \cup N_0| \varepsilon_0 \leq \gamma |P_0 \cup N_0| (1-\varepsilon_0) \\ \Rightarrow & (\frac{\rho}{\gamma})_{\max} = \frac{1-\varepsilon_0}{\varepsilon_0} \\ \Rightarrow & r_{\max} = \frac{1-\varepsilon_0}{\varepsilon_0} \end{aligned} \tag{5-28}$$

5.4 Algorithm of Two-stage Learning

Given a training data set S_0 and corresponding hypothesis $h_0(x)$, positive examples P_0 and negative examples N_0 can be gotten by $h_0(x)$ such that $P_0 \bigcup N_0 \xleftarrow{h_0(x)} S_0$. Under

sampling algorithm or over-sampling can be used for the imbalanced data set, where the size of negative data is far less than that of positive $|N| << |P|$. To construct a new training data set S_1, which is called compensated data set, for negative learning; all positive examples P_1 of S_1 are labeled as class +1 whereas the rest of negative examples N_1 are labeled by class -1. Removing some positive examples is controlled by under-sampling regularization parameter γ that controls the size of positive examples P_1. The over- sampling parameter ρ controls the possibility to duplicate the negative example *(x, y)* from N_0 assigning the weight of negative example *(x, y)*, and then we get $|N_1| = \rho|N_0|$ where $1 \le \rho$. Function rand () generates random value between 0 and 1. The function construct_audit_dataset returns a data set S_1 for audit learning. Below is the algorithm to construct compensated training data for audit:

construct_audit_dataset (S_0, h_0, P_0, N_0, γ, ρ)

i ← 0

k ← 0

foreach (x, y) **in** S_0 **do**

▷ Predict the label of instance in the testing data set S_0

$y_1 \leftarrow h_0(x)$

▷ Append feedback from last predicting label

x[|x|+1] ← y_1;

▷ Keep all negative examples because of unbalance examples

if $(x, y) \in N_0$ ▷ μ negative example

▷ Let negative be class -1

if rand() $\leq \rho$

$S_1[k{+}{+}] \leftarrow (x, -1)$

$p \leftarrow \rho$

while $p \geq 1$

if rand() $\leq$ p-1 ▷ Duplicate the negative example if $|N_1|$ is too small

$S_1[k{+}{+}] \leftarrow (x, -1)$

$p \leftarrow p\text{-}1$

end while

▷ $(x, y) \in P_0$ Keep partial positive examples, under sampling

else if rand() $\leq \gamma$

▷ let positive example be class +1

$S_1[k{+}{+}] \leftarrow (x, +1)$

$i \leftarrow i+1$

end foreach

return S_1

Two-stage learning algorithm including training and testing algorithms:

SNELA2 (S_0, T_0, μ, γ, ρ, δ_1, δ_2)

▷Learning phase

$(h_0, \varepsilon_0) \leftarrow$ LEARN (S_0) ▷ ε_0 is training error

▷ S_0 is divided into positive and negative data set in terms of μ

$(P_0, N_0) \leftarrow$ DIVIDE (S_0, h_0, μ)

▷ Form a compensated training set for negative learning

$S_1 \leftarrow$ construct_audit_dataset (S_0, h_0, P_0, N_0, γ, ρ)

$(h_1, \varepsilon_1) \leftarrow$ LEARN (S_1)

▷ Testing phase

$T_1 \leftarrow$ PREDICT (T_0, h_0)

▷ if the criterion is not met, return base predicting result.

if $\varepsilon_1 \geq \frac{1}{2+r}$

then

return T_1

$T_2 \leftarrow$ PREDICT (T_0, h_1)

for i=1 to $|T_0|$ **do**

▷ Positive constants δ_1, δ_2 are confidence threshold.

if $T_1[i].y < \delta_1$ **and** $T_2[i].y < -\delta_2$

then

$T_3[i].y \leftarrow - T_1[i].y$ ▷ reverse predicting class

else

$T_3[i].y \leftarrow T_1[i].y$ ▷ keep class unchanged

end for

return T_3

5.5 Three-stage Learning of SNELA

The limitation of two-stage negative learning is that the strong condition or upper bound is related to the ratio of positive and negative examples. And it assumes a small size of examples for the audit. Comparing to original training data set for the learner, it can still reflect the distribution of data space. The condition $\varepsilon_1 < \frac{1}{2+r}$ is not easy to meet because learning on the imbalanced data is much more difficult than that on the balanced data. The audit has strong capability on correcting the predicting results of the learner by reversing possibility of sampling data the result from class +1 into -1 or vice versa. That requires the audit has a higher accuracy than the respective learner's. Two-stage learning overcomes those shortcomings by applying the confidence of SVM, which assumes a lower confidence of instance and then has lower possibility to predict correctly. Three-stage learning extends two-stage learning by adding an extra learning stage to learn on the hardest negative data. The extra learning algorithm is called booster and its behavior is close to the boosting algorithm. The booster's goal is to enhance the audit's accuracy by mining those negative examples that are disagreed by the learner and the audit.

Three-stage learning learns on three different distributions D_0, D_1 and D_2; and output a combined hypothesis. Three learning methods are *learner*, *audit* and *booster* respectively. In this dissertation these methods are SVMs. The *learner* is trained on original distribution of data in the normal way to output hypothesis $h_0(x)$ with error $\varepsilon_0 = \Pr_{x \in D_0}[yh_0(x) < 0]$ as shown on the top of Figure 5.15. The *learner* has found some examples that are misclassified on the original distribution. To enhance the capability of *leaner*, the audit works on the distribution D_1 that includes harder parts of the distribution

to judge the *learner*'s ability of prediction. The instances chosen according to the distribution D_1 are sampled from positive and negative examples classified by h_0. The *audit* outputs hypothesis $h_1(x)$ with error $\varepsilon_1 = \Pr_{x \in D_1}[yh_1(x) < 0]$. Finally, D_3 is constructed by removing from D_0 the examples on which h_1 agree with h_0. The *booster* produces hypothesis $h_2(x)$. The combinational hypothesis $h(x)$ is: given an instance x, if $h_1(x) = +1$ then let $h(x) = h_0(x)$; otherwise if $h_2(x) = +1$ then let $h(x) = h_0(x)$ else $h(x) = -h_0(x)$.

The area $P_s \cup N_u$ with distribution D_1 cannot be classified by hypothesis $h_1(x)$ because $h_1(x)$ outputs class -1 in this area. Therefore, the hypothesis $h_2(x)$ learned from $P_s \cup N_u$ enhances prediction as majority vote[96] strategy.

The possibility of s, t, u, v is defined as follows:

- The possibility $s = \Pr_{x \in D_0}[yh_0(x) > 0 \wedge h_1(x) = -1]$, the area that is NOT supposed to be compensated is compensated. This is a bad situation.
- The possibility $t = \Pr_{x \in D_0}[yh_0(x) > 0 \wedge h_1(x) = +1]$, the area that is NOT supposed to be compensated is not compensated. This is a good situation.
- The possibility $u = \Pr_{x \in D_0}[yh_0(x) < 0 \wedge h_1(x) = -1]$, the area that is supposed to be compensated is compensated. This is a good situation.
- The possibility $v = \Pr_{x \in D_0}[yh_0(x) < 0 \wedge h_1(x) = +1]$, the area that is supposed to be compensated is NOT compensated. This is a bad situation.

The possibility $s + u$ is the area expected NOT to be compensated, while $u + v$ is the area expected to be compensated. In summary, the four possibility areas are listed in the table below.

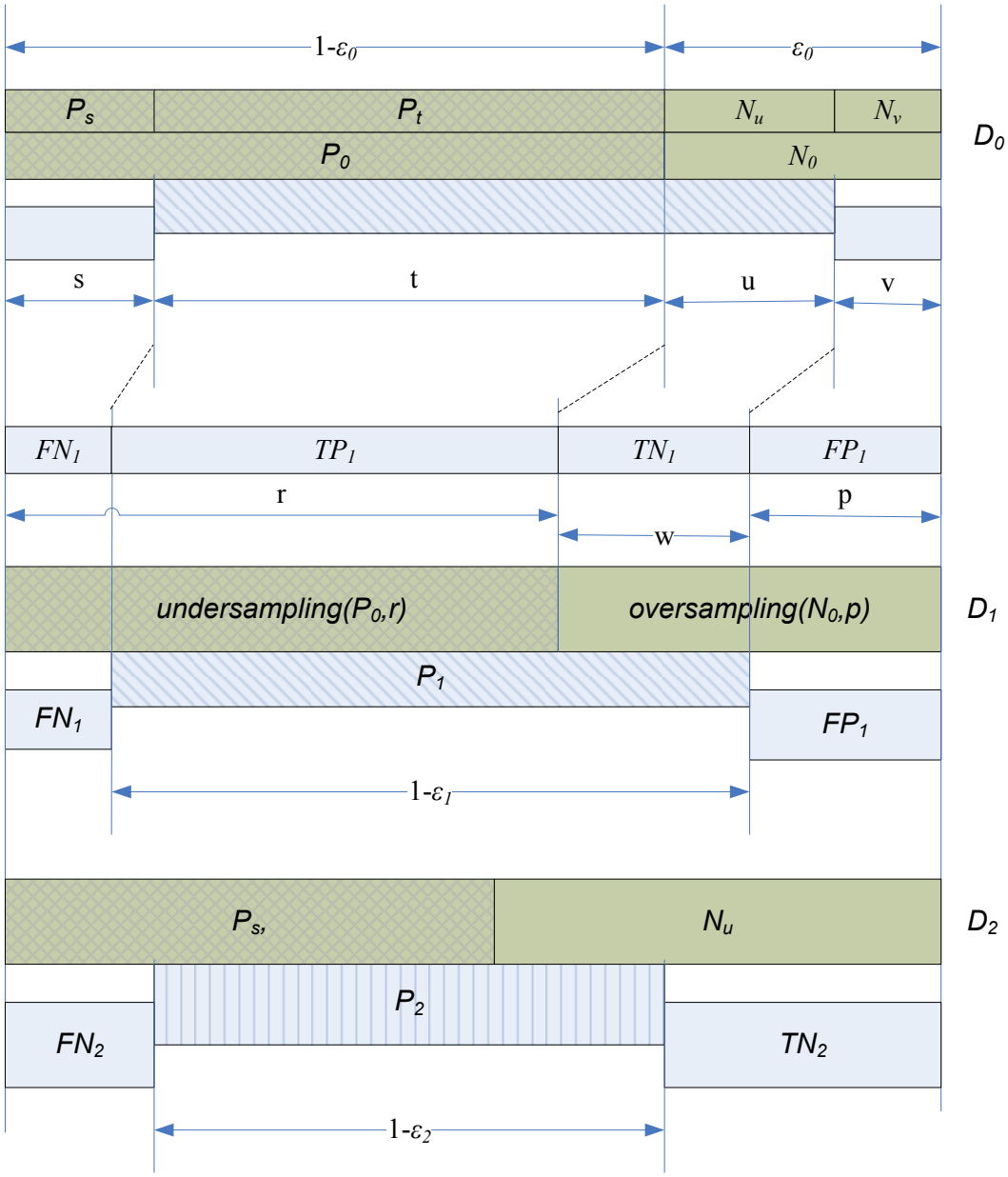

Figure 5.15 The distribution D_0, D_1 and D_2 on the three-stage learning. $P_0 = P_s \cup P_t$, $N_0 = N_u \cup N_v$. $P_1 = TP_1 \cup TN_1$ and $N_1 = FP_1 \cup FN_1$

TABLE 5.1 The possibility of four areas

Area	Target Label	Learner $h_0(x)$	Audit $h_1(x)$	audit
s	+1	+1	-1	Detect wrong
	-1	-1	-1	
t	+1	+1	+1	Correctly detect
	-1	-1	+1	
u	+1	-1	-1	Correctly detect
	-1	+1	-1	
v	+1	-1	+1	Not detect
	-1	+1	+1	

$$s+t=\Pr_{x\in D_0}[yh_0(x)>0]=1-\varepsilon_0 \tag{5-29}$$

$$u+v=\Pr_{x\in D_0}[yh_0(x)<0]=\varepsilon_0 \tag{5-30}$$

Combining equation (5-13), (5-29) and (5-30), we know the value of s and v can be solved:

$$\begin{aligned} s &= 1-\varepsilon_0-t \\ &= 1-\varepsilon_0-\frac{(1-\varepsilon_0)[\varepsilon_0(1-\varepsilon_1)(\gamma+\rho)-u\rho]}{\gamma\varepsilon_0} \\ &= \frac{1-\varepsilon_0}{\gamma\varepsilon_0}[\gamma\varepsilon_0+\varepsilon_0(\varepsilon_1-1)(\gamma+\rho)+u\rho] \\ &= \frac{1-\varepsilon_0}{\gamma\varepsilon_0}[\varepsilon_0\varepsilon_1(\gamma+\rho)-\rho\varepsilon_0+u\rho] \\ &= a+bu \end{aligned} \tag{5-31}$$

$$v=\varepsilon_0-u \tag{5-32}$$

where $a=\frac{1-\varepsilon_0}{\gamma}[\varepsilon_1(\gamma+\rho)-\rho]$ and $b=\frac{\rho(1-\varepsilon_0)}{\gamma\varepsilon_0}$

The probability of instance x being chosen under D_2 is defined:

$$D_2(x) = \frac{D_0(x)\Pr[h_1(x) = -1]}{s+u} \tag{5-33}$$

Then, we get the error of combinational hypothesis *h(x)* as follows.

$$\begin{aligned}
&\Pr_{x\in D}[yh(x) < 0] \\
&= \Pr_{x\in D}[yh_0(x) < 0 \wedge h_1(x) = +1] \vee \Pr_{x\in D}[yh_2(x) < 0 \wedge h_1(x) = -1] \\
&= v + \sum_{x\in X} D_0(x)\Pr[h_1(x) = -1]p_2(x)
\end{aligned}$$

$$\begin{aligned}
&= v + \varepsilon_2(s+u) \\
&= \varepsilon_0 - u + \varepsilon_2(a + bu + u) \\
&= \varepsilon_0 + a\varepsilon_2 + (\varepsilon_2 b + \varepsilon_2 - 1)u \\
&= a\varepsilon_2 + \varepsilon_0 + [\frac{\rho(1-\varepsilon_0)}{\gamma\varepsilon_0}\varepsilon_2 + \varepsilon_2 - 1]u
\end{aligned} \tag{5-34}$$

$\because u \le \varepsilon_0$

$$\begin{aligned}
&\Pr_{x\in D}[yh(x) < 0] \\
&\le a\varepsilon_2 + \varepsilon_0 + \frac{1}{\gamma}[\rho\varepsilon_2 - \gamma\varepsilon_0 + (\gamma-\rho)\varepsilon_0\varepsilon_2] \\
&= a\varepsilon_2 + \frac{1}{\gamma}[\rho\varepsilon_2 + (\gamma-\rho)\varepsilon_0\varepsilon_2] \\
&= a\varepsilon_2 + \frac{\rho}{\gamma}\varepsilon_2 + \frac{(\gamma-\rho)\varepsilon_0\varepsilon_2}{\gamma}
\end{aligned} \tag{5-35}$$

We know $\varepsilon_i < 0.5, i = 0,1,2$, and then the rightmost item above is less than 0. The following equality can be gotten:

$$
\begin{aligned}
&\Pr_{x\in D}[yh(x)<0] \\
&\le a\varepsilon_2+\frac{\rho}{\gamma}\varepsilon_2+\frac{(\gamma-\rho)\varepsilon_0\varepsilon_2}{\gamma} \\
&=\frac{1-\varepsilon_0}{\gamma}[\varepsilon_1(\gamma+\rho)-\rho]\varepsilon_2+\frac{\rho}{\gamma}\varepsilon_2+\frac{(\gamma-\rho)\varepsilon_0\varepsilon_2}{\gamma} \\
&=\frac{\varepsilon_2}{\gamma}[\rho\varepsilon_0+\varepsilon_1(\gamma+\rho)(1-\varepsilon_0)]+\frac{(\gamma-\rho)\varepsilon_0\varepsilon_2}{\gamma} \\
&=\frac{\varepsilon_2}{\gamma}[\gamma\varepsilon_0+\varepsilon_1(\gamma+\rho)(1-\varepsilon_0)] \\
&=\frac{\varepsilon_2}{\gamma}[\gamma\varepsilon_0+\varepsilon_1(\gamma+\rho)-\varepsilon_0\varepsilon_1(\gamma+\rho)]
\end{aligned}
\tag{5-36}
$$

If we do not consider the prior knowledge of the difference among the errors $\varepsilon_0,\varepsilon_1,\varepsilon_2$, then we assume $\varepsilon_i<\varepsilon, i=0,1,2$

$$\Pr_{x\in D}[yh(x)<0]\le\frac{1}{\gamma}[(2\gamma+\rho)\varepsilon^2-(\gamma+\rho)\varepsilon^3] \tag{5-37}$$

We get the error bound of three-stage learning *h(x)* below:

$$\Pr_{x\in D}[yh(x)<0]=O(\varepsilon^2)<O(\varepsilon) \tag{5-38}$$

The error bound (5-37) indicates that the error bound is small when ε is small. In conclusion, for $\forall\,\varepsilon_i<\varepsilon, i=0,1,2$ in the three-stage learning, the error of hypothesis returned is bounded $O(\varepsilon^2)$. That significantly reduces the error of original hypothesis *learner* which is bound to $O(\varepsilon)$. The error bound is exactly the same as the result of Boosting algorithm[95] when $\gamma=\rho=0.5$, that is $\Pr_{x\in D}[yh(x)<0]\le 3\varepsilon^2-2\varepsilon^3$. The inequality (5-36) of the error bound of *h(x)* is rewritten in below format:

$$\Pr_{x\in D}[yh(x)<0]\le\varepsilon_2[(\varepsilon_0+\varepsilon_1-\varepsilon_0\varepsilon_1)+\frac{\rho}{\gamma}(\varepsilon_1-\varepsilon_0\varepsilon_1)] \tag{5-39}$$

This indicates that a small value ρ/γ is preferred to have small error bound. However, a small γ and large ρ are usually chosen in the practical application because the audit shows low error ε_1 in the balanced training examples D_1. Therefore, there exists a tradeoff between ε_1 and ρ/γ.

We know $\varepsilon_0 \le \varepsilon_1 \le \varepsilon_2$; and we choose $0 \le \gamma \le 1$, $1 \le \rho$ and $\frac{\rho}{\gamma} \le \frac{1-\varepsilon_0}{\varepsilon_0}$ in the practice because the small size of training examples leads to bad performance of hypothesis. Thus from inequality (5-36) and equation (5-26) the error bound of *h(x)* is equivalent to the following:

$$\begin{aligned} &\Pr_{x\in D}[yh(x)<0] \\ &\le \varepsilon_2[\varepsilon_0+\varepsilon_1(1-\varepsilon_0)(1+\frac{\rho}{\gamma})] \\ &= \mathrm{c}+\mathrm{d}\varepsilon_1(1+\frac{\rho}{\gamma}) \end{aligned} \tag{5-40}$$

$$\Pr_{x\in D}[yh(x)<0] \le \mathrm{c}+\mathrm{d}\varepsilon_1(1+\frac{1-\varepsilon_0}{r\varepsilon_0}) \tag{5-41}$$

Where c= $\varepsilon_0\varepsilon_2$ and d= $\varepsilon_2(1-\varepsilon_0)$ are constants as long as $\varepsilon_0, \varepsilon_2$ are fixed. If over-sampling strategy decreases the value $\varepsilon_1(1+\frac{\rho}{\gamma})$ or $\varepsilon_1(1+\frac{1-\varepsilon_0}{r\varepsilon_0})$, then the over-sampling strategy can be used. The algorithm of three-stage learning SNELA3 is shown below:

SNELA3 (S_0, T_0, μ, γ, ρ)

▹Learning phase

(h_0, ε_0)← LEARN (S_0) ▹ ε_0 is training error

▹ S_0 is divided into positive and negative data set in terms of μ

$(P_0, N_0) \leftarrow$ DIVIDE (S_0, h_0, μ)

▷ Form a compensated training set for audit in the negative learning

$S_1 \leftarrow$ construct_audit_dataset $(S_0, h_0, P_0, N_0, \gamma, \rho)$

$(h_1, \varepsilon_1) \leftarrow$ LEARN (S_1)

▷ Form a compensated training set for booster in the third stage learning

$S_2 \leftarrow$ construct_booster_dataset (S_0, h_0, h_1)

$(h_2, \varepsilon_2) \leftarrow$ LEARN (S_2)

▷ Testing phase

Foreach x **in** T_0 **do**

if $h_1(x) = +1$

$T_1[i] \leftarrow h_0(x)$

else if $h_2(x) = +1$

$T_1[i] \leftarrow h_0(x)$

else

$T_1[i] \leftarrow -h_0(x)$

end foreach

return T_1

The function of construct data set S2 for booster in the third stage learning is in the following.

construct_booster_dataset (S_0, h_0, h_1)

k ← 0

foreach (x, y) **in** S_0 **do**

▷ Predict the label of instance in the testing data set S_0

$y_1 \leftarrow h_0(x)$

▷ Append feedback from last predicting label

x[|x|+1] ← y_1;

▷ $x \in s \cup u$, h_1 does not agree with h_0

if $h_1(x) < 0$

if $y*h_0(x)<0$

▷ $x \in u$ learner misclassified instance x, let negative be class -1

S_2[k++]← (x, -1)

else if $y*h_0(x)>0$

▷ x ∈s, let positive example be class +1

S_2[k++]← (x, +1)

end foreach

return S_2

Comparing to two-stage learning, three-stage learning is guaranteed to have good performance although $\varepsilon_0 \le \varepsilon_1 \le \varepsilon_2$. The lower accuracy hypothesis can improve the base learning accuracy by choosing appropriated parameter of ρ / γ which is independent with

the error bound $O(\varepsilon^2)$. The third stage learning is a booster which can be run many times to boost the base learning algorithm. Note that the error bound is only related to the ratio of the size of positive and negative examples instead of the total number of examples. This indicates that the balance of positive and negative is sensitive to the final performance.

The algorithm run in the SVM can be modified in this way:

- $h_i(x) = +1$ is equivalent to $h_i(x) > \mu_i$, i=0,1,2
- $h_i(x) = -1$ is equivalent to $h_i(x) < \mu_i$, i=0,1,2

The parameter μ can be verified by AUC on the equation (1-9) because AUC is independent on the bias *b*. The default value of μ is zero in the SVMs as shown on the figure below. Sometimes, SVM gives a low accuracy whereas AUC shows high value in imbalanced examples prediction. The reason is that bias value is not correct. The bias can be moved around to maximize the accuracy of the hypothesis. The parameter μ is computed in the equation (5-42).

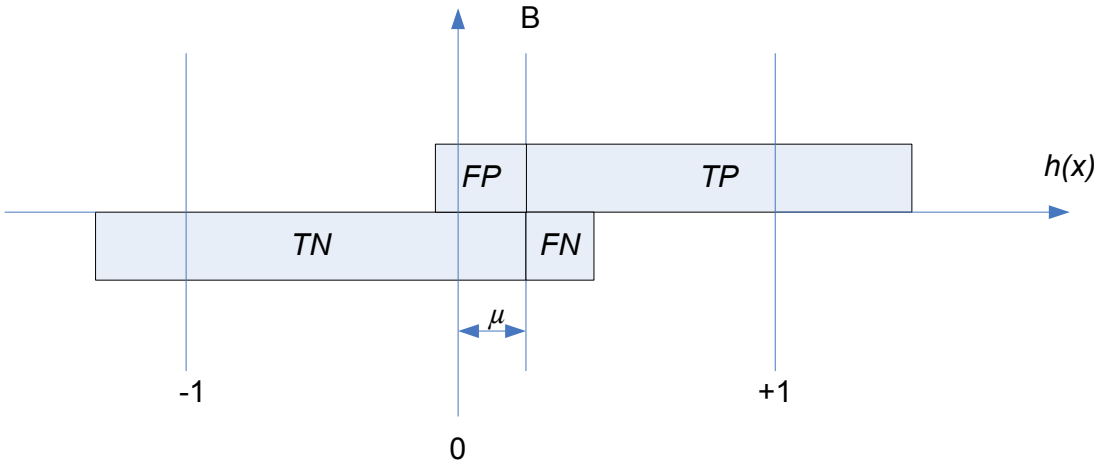

Figure 5.16 The parameter μ is determined by moving around the line B to minimize the size of $FP \cup FN$

$$\mu_k = \arg\max_{(x,y)\in S_k} \frac{\sum_{i=1}^{|S_k|} 1_{y_i(h_k(x_i)-\mu_k)>0}}{|S_k|}, k = 0,1,2 \tag{5-42}$$

where 1_π is defined to be 1 if the predicate π holds and 0 otherwise.

5.6 Simulation

Six data sets were studied in this simulation. *Breast-cancer-Wisconsin*[97-99] with 699 examples and 10 attributes is binary classification problem. The class +1 examples are Benign 458 (65.5%) and class -1 examples are Malignant 241 (34.5%). Zhang,J. used 369 examples of them and the best accuracy obtained is 93.7% by 1-nearest neighbor[100]. *Ionosphere* is the data set with size of 351 examples and 34 attributes. David Aha used nearest neighbor to attain an accuracy of 92.1%, and Ross Quinlan's C4 algorithm attains 94.0% (no windowing), and that IB3attained 96.7% [101].

TABLE 5.2 Overview of negative learning performance

Accuracy%	Other	SVM	NDDCHA	SNELA2	SNELA3
Breast-cancer-Wisconsin	**93.7**	89.6	90.3	92.3	92.5
Ionosphere	**96.7**	84.3	84.3	84.3	83.4
Liver-disorder	-	69.0	68.3	72.5	75.3
Lung-cancer	77.0	74.0	72.0	70.8	**77.2**
Pima Indians Diabetes	**76.0**	72.3	72.4	73.2	74.0

Liver-disorder has 345 examples with 6 attributes without missing values. *Lung-cancer* has 32 examples with 34 attributes. It is 3 classes classification, the number of three classes are 9,13,10 respectively[102]. The best bias accuracy is 77%. *Pima Indians*

Diabetes has 768 examples with 8 attributes, 500 positive examples and 268 negative examples. Using 576 training instances, the sensitivity and specificity of their algorithm was 76% on the remaining 192 instances[103].

In the Table 5.3, the column other is the best accuracy found in other approaches. The columns SVM, NDDCHA, SNELA2 and SNELA3 are the accuracy values by using their methods. The results show that SNELA3 has better performance than SNELA3 in most cases. SNELA2 has high risk in predicting negative data because it depends on the confidence value of SVM. It tells us the value of *h(x)* is not exact confidence of instance *x* in the feature space defined by *h(x)*. SNELA does give the best performance and it does improve the single SVM method. That is the goal of negative data mining to improve the accuracy of original algorithm by mining negative data.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Summary

In the supervised machine learning field, the generalization capability is archived by training on a set of known examples from the input spaces. The predicting error is inevitable. One source of error is from the low powerful learning algorithm which either has approximation error or estimation error or both. Another source is from not-well-distributed data which is not i.i.d.. Therefore, the negative examples exist widely in all known learning algorithms. Two negative data mining approaches are proposed in this dissertation by understanding two types of errors. The basic philosophy of two approaches is that firstly negative examples contains positive information, and secondly the approaches make improvement on the generalization capacity of base learning algorithm as much as possible, and does not make any changes on the base learning machine if the improvement cannot be made. Therefore, the challenges include learning on the imbalanced positive and negative examples, and the determination of improvement criteria on the base learning algorithm.

The proposed NDDCHA improves the learning algorithm performance through compensating the base hypothesis by utilizing the negative data set. Useful information content in the negative data is mined to benefit the model of an application. This approach expands the hypotheses space to close the target space so that the

approximation error will be reduced. A hypothesis with high VC dimension has high capability to classify the examples. Its drawback is that it is prone to overfitting. NDDCHA can use low VC dimension hypothesis as base algorithm and use high VC dimension of hypothesis to compensate the base one to reduce overfitting. NDDCHA separates every example in the hyperplane as much as possible which matches the principle of Vapnik's generalization theory of maximizing the margins. The cases show that the NDDCHA does increase the performance.

A tutor usually can predict a student performance based on judging student's work. If the tutor's judgment on student's work has high confidence, then it is helpful to the student. Otherwise, a mentor may be needed to confirm which one is correct between the student and the tutor in order to improve student's performance. SNELA is two or three stages learning including *learner*, *audit* and *booster* like the scenario cooperated by student, tutor and mentor. SNELA based on the theoretical analysis improves the performance of base learning algorithm *learner* by creating one or two additional hypothesis *audit* and *booster* to mine the negative examples. The error ε of *learner* is proved to decrease from $O(\varepsilon)$ to $O(\varepsilon^2)$.

6.2 Future Work

The NDDCHA offers the flexibility of using other non-SVM algorithms as base and patching learning algorithm. Hence, as a future work, we will study the above said cases with other learning algorithms like ANN as base learning algorithm. The methods of vector similarity and partitioner functions are chosen by the user based on the understanding of the properties of the training data. The future work also includes

investigating the relationship between the vector similarity function and the data distribution, using the fuzzy partitioner.

SNELA assumes the data used is i.i.d. and a hypothesis learned on a small size of examples is still capable to predict other instances. This is not true in the practical application, especially, in the imbalanced training data. The compensated data set and boost data set are transformed from original data set which is not exactly under the control of original data distribution. Then, further work will investigate this phenomenon. Future work of SNELA also includes the run time and memory space complexity analysis.

BIBLIOGRAPHY

[1] J. S.-T. Nello Cristianini, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*: Cambridge University Press, 2000.

[2] T. M. Mitchell, *Machine Learning*, 1 ed: McGraw-Hill, 1997.

[3] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[4] M. P. a. A. Shilton, "Adaptive Support Vector Machines for Regression," *Proc. of the 9th International Conference on Neural Information processing, Singapore*, 2002.

[5] V. Vapnik, *Statistical Learning Theory*: John Wiley & Sons, 1998.

[6] M. Palaniswami, Shilton, A., Ralph, D., and Owen B., "Machine learning using support vector machines," presented at International conference on Artificial Intelligence in Science and Technology, Hobart, Australia,, 2000.

[7] B. Scholkopf, "Statistical Learning and Kernel Methods," *http://citeseer.ist.psu.edu/507312.html*, 2000.

[8] M. Fuchs, "Instance-based learning by searching," 1997.

[9] T. R. Payne, P. Edwards, and C. L. Green, "Experience with rule induction and k-nearest neighbor methods for interface agents that learn," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 9, pp. 329-335, 1997.

[10] J. Egan, *Signal detection theory and ROC analysis*: Academic Press, 1975.

[11] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, pp. 1145-1159, 1997.

[12] Lara and J. v. Schalkwyk, "The magnificent ROC," *http://www.anaesthetist.com/mnm/stats/roc/*, 2000.

[13] D. M. Green and J. A. Swets., *Signal detection theory and psychophysics*. New York: Wiley, 1966.

[14] C. C. a. M. Mohri, "AUC optimization vs. error rate minimization," *http://citeseer.ist.psu.edu/cortes03auc.html*, 2004.

[15] A. Rakotomamonjy, "Optimizing Area Under Roc Curve with SVMs," *http://citeseer.ist.psu.edu/731744.html*.

[16] S. R. Gunn, "Support Vector Machine for Classification and Regression," Engineering, Science and Math. School of Electronics and Computer Science,University of Southhampton 1998.

[17] S. Huang, K. K. Tan, and K. Z. Tang, *Neural network control : theory and applications*. Hertfordshire, England Williston, VT: Research Studies Press; Distribution North America, AIDC, 2004.

[18] R. L. Harvey, *Neural network principles*. Englewood Cliffs, NJ: Prentice Hall, 1994.

[19] R. Bharath and J. Drosen, *Neural network computing*. New York: Windcrest/McGraw-Hill, 1994.

[20] O. Omidvar, Elliott, and L. David, *Neural systems for control*. San Diego: Academic Press, 1997.

[21] F. Jiang, A. P. Preethy, and Y.-Q. Zhang, "Compensating Hypothesis by Negative Data," presented at IEEE: International Conference on Neural Networks and Brain, Beijing,China, 2005.

[22] R. E. Schapire, "The Boosting Approach to Machine Learning An Overview," *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

[23] R. E. Schapire, "A Brief Introduction to Boosting," presented at Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, 1999.

[24] L. Breiman, "Bagging predictors," *Machine Learning* vol. 24, pp. 123-140, 1996.

[25] R. E. Schapire, "The Boosting Approach to Machine Learning An Overview," presented at MSRI Workshop on Nonlinear Estimation and Classification, Berkeley, CA, 2001.

[26] Y.-c. I. Chang, "Boosting SVM Classifiers with Logistic, Regression,http://www.stat.sinica.edu.tw/library/c_tec_rep/2003-03.pdf."

[27] S. P. Hyun-Chul Kim, Hong-Mo Je , Daijin Kim , Sung Yang Bang "Pattern classification using support vector machine ensemble," presented at 16th International Conference on Pattern Recognition, 2002.

[28] A. J. Sharkey, *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*, 1st ed. New York: Springer-Verlag, 1999.

[29] S. P. Hyun-Chul Kim, Hong-Mo Je, Daijin Kim, Sung Yang Bang: , "Support Vector Machine Ensemble with Bagging," presented at Pattern Recognition with Support Vector Machines, First International Workshop, SVM 2002, Niagara Falls, Canada, 2002.

[30] S. Pang, D. Kim, and S. Y. Bang, "Membership authentication in the dynamic group by face classification using SVM ensemble," *Pattern Recognition Letters* vol. 24, pp. 215-225, 2003.

[31] J. B. Kristin P. Bennett , R.P.I, "A Support Vector Machine Approach to Decision Trees " Rensselaer Polytechnic Institute, Troy, NY 1997.

[32] R. E. Schapire, Y. Freund, P. L. Bartlett, and W. S. Lee, "Boosting the margin: a new explanation for the effectiveness of voting methods," presented at 14th Int. Conf. Machine Learning, San Mateo, CA, 1997.

[33] Y. Freund and R. E. Schapire, "A Short Introduction to Boosting," *Journal of Japanese Society for Artificial Intelligence*, vol. 14, pp. 771-780, 1999.

[34] L. G. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. 27, pp. 1134-1142, 1984.

[35] Y. Freund, "An Adaptive Version of the Boost By Majority Algorithm," presented at "{COLT}: Proceedings of the Workshop on Computational Learning Theory, 1999.

[36] J. H. Friedman, T. Hastie, and R. Ribshirani, "Additive Logistic Regression: A Statistical View of Boosting," *The Annals of Statistics*, vol. 28, pp. 337-374, 2000.

[37] J. R. Quinlan, "Bagging, Boosting, and C4.5," *{AAAI}/{IAAI}* vol. 1, pp. 725-730, 1996.

[38] H.-T. Lin and L. Li, "Novel Distance-Based SVM Kernels for Infinite Ensemble Learning," presented at The 12th International Conference on Neural Information Processing, Taipei, Taiwan, 2005.

[39] Y. Freund and R. E. Schapire, "Experiments with a New Boosting Algorithm " presented at International Conference on Machine Learning, 1996.

[40] A. J. Smola and B. Schölkopf, "On a kernel-based method for pattern recognition, regression, approximation and operator inversion," *Algorithmica*, vol. 22, pp. 211-231, 1998.

[41] Y. A. LeCun, L. D. Jackel, L. Bottou, A. Brunot, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Müller, E. Säckinger, P. Y. Simard, and V. N. Vapnik, "Learning algorithms for classification: A comparison on handwritten digit recognition,," *Neural Networks*, pp. 261–276, 1995.

[42] C. J. C. Burges and B. Schölkopf, *Improving the accuracy and speed of support vector learning machines*: Cambridge, MA: MIT Press, 1997.

[43] K.-R. Müller, A. J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. N. Vapnik, "Predicting time series with support vector machines," presented at Artificial Neural Networks—ICANN'97, Berlin, Germany, 1997.

[44] S. Dumais, J. Platt, D. Heckerman, and M. Sahami, "Inductive learning algorithms and representations for text categorization," presented at 7th international conference Information Knowledge Management, 1998.

[45] T. Furey, N. Cristianini, N. Duffy, D. Bednarski, M. Schummer, and D. Haussler, "Support vector machine classification and validation of cancer tissue samples using microarray expression data," *Bioinformatics*, vol. 16, pp. 906-914, 2000.

[46] A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller, "Engineering support vector machine kernels that recognize translation initiation sites in DNA," vol. 16, pp. 799-807, 2000.

[47] C. S. Ong, X. Mary, S. Canu, and A. J. Smola, "Learning with Non-Positive Kernels," presented at Proceedings of the 21st International Conference on Machine Learning, Banff,Canada, 2004.

[48] S.-i. Amari and S. Wu, "Improving Support Vector Machine Classifiers by Modifying Kernel Functions," *Neural Networks*, pp. 783--789, 1999.

[49] A. N. Srivastava, J. Schumann, and B. Fischer, "An Ensemble Approach to Building Mercer Kernels with Prior Information," presented at IEEE Systems Manand Cybernetics Conference Workshop on Ensemble Methods in Extreme Environments,, 2005.

[50] B. Vanschoenwinkel and B. Manderick, "Substitution Matrix based Kernel Functions for Protein Secondary Structure," *http://citeseer.ist.psu.edu/723055.html*.

[51] B. Scholkopf, "The Kernel Trick for Distances," *NIPS*, pp. 301-307, 2000.

[52] B. Vanschoenwinkel and B. Manderick, "A weighted polynomial information gain kernel for resolving pp attachment ambiguities with support vector machines," presented at The Eighteenth International Joint Conferences on Artificial Intelligence(IJCAI-03) 2003.

[53] R. F. E. Osuna, and F. Girosi. , "Support vector machines: training and applications.," in *AI Memo*: MIT, 1997, pp. 1602.

[54] C. J. C. Burges, "A Tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, pp. 121-167, 1998.

[55] C. C. Kristin P. Bennett, "Support Vector Machines: Hype or Hallelujah?," *SIGKDD Explorations*, vol. 2, 2000.

[56] I. G. Bernhard Schölkopf, Jason Weston, "Statistical Learning and Kernel Methods in Bioinformatics."

[57] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A Practical Guide to Support Vector Classification " *citeseer.ist.psu.edu/689242.html*.

[58] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing Multiple Parameters for Support Vector Machines," *Machine Learning*, vol. 46, pp. 131-159, 2002.

[59] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers " *Computational Learing Theory*, pp. 144-152, 1992.

[60] P.-H. Chen, C.-J. Lin, and B. Schölkopf, "A Tutorial on v-Support Vector Machines," *www.csie.ntu.edu.tw/~cjlin/papers/nusvmtutorial.pdf*, 2002.

[61] V. N. Vapnik, *Estimation of Dependences Based on Empirical Data*: Springer-Verlag, 1982.

[62] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37-66, 1991.

[63] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally Weighted Learning," *Artificial Intelligence Review*, vol. 11, pp. 11-73, 1997.

[64] C. G. A. A. Moore, and S.A. Schaal, "Locally Weighted Learning For Control," *AI Review*, vol. 11, pp. 75-113, 1997.

[65] J. Zhan, C. LiWu, and S. Matwin, "Building k-nearest neighbor classifiers on vertically partitioned private data," 2005.

[66] W. Ji-Gang, P. Neskovic, and L. N. Cooper, "An Adaptive Nearest Neighbor Algorithm for Classification," 2005.

[67] L. Kuan-Ming and L. Chih-Jen, "A study on reduced support vector machines," *Neural Networks, IEEE Transactions on*, vol. 14, pp. 1449-1459, 2003.

[68] W. Fangfang and Z. Yinliang, "A Novel Multi-Reduced Support Vector Machine," 2005.

[69] C.-T. S. Jyh-Shing Roger Jang, Eiji Mizutani *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, 1st ed: Pearson Education, 1996.

[70] G. Horvath, "CMAC neural network as an SVM with B-spline kernel functions," 2003.

[71] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Transactions on Neural Networks*, vol. 10, pp. 988-999, 1999.

[72] A. Gammerman, *Computational learning and probabilistic reasoning*. Chichester ; New York: Wiley in association with UNICOM, 1996.

[73] V. N. Vapnik, *The nature of statistical learning theory*, 2 ed: Springer, 2000.

[74] B. Schölkopf, C. Burges, and V. Vapnik, "Extracting Support Data for a Given Task," presented at First International Conference on Knowledge Discovery & Data Mining, Menlo Park, 1995.

[75] Q.-L. Z. Quang-Anh Tran, Xing Li, "Reduce the number of support vectors by using clustering techniques," presented at Machine Learning and Cybernetics, 2003 International Conference on, 2003.

[76] Y. Lin, Y. Lee, and G. Wahba, "Support Vector Machines for Classification in Nonstandard Situations," *Machine Learning*, vol. 46, pp. 191 - 202 2002.

[77] D. H. a. C. K. J. S. Breese, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering,," *Proceeding of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, 1998.

[78] Z. W. Kai Yu, Xiaowei Xu, Martin Ester, "Feature Weighting and Instance Selection for Collaborative Filtering," *Proc. 2nd International Workshop on Management of Information on the Web - Web Data and Text Mining (MIW□01)*, 2001.

[79] H. Bandemer and W. Nather, *Fuzzy data analysis*. Dordrecht, Netherlands ; Boston: Kluwer Academic Publishers, 1992.

[80] M. Karpinski and T. Werther, "VC Dimension and Uniform Learnability of Sparse Polynomials and Rational Functions," *SIAM J Computing*, vol. 22, pp. 1276 - 1285, 1993.

[81] G. Cestnik, I. Konenenko, and I. Bratko, "A Knowledge-Elicitation Tool for Sophisticated Users," presented at Progress in Machine Learning, 1987.

[82] S. a. B. Hettich, S. D., "The UCI KDD Archive ": Irvine, CA: University of California, Department of Information and Computer Science. , 1999.

[83] T. Joachims, *Learning to Classify Text Using Support Vector Machines*: Kluwer Academic Publishers, 2002.

[84] T. Joachims, *Making large-Scale SVM Learning Practical*: MIT Press, 1999.

[85] D. S. Moore and G. P. McCabe, *Introduction to the Practice of Statistics* 4ed: W.H. Freeman & Company, 2002.

[86] C.-f. L. S.-d. Wang, "Training algorithms for fuzzy support vector machines with noisy data," presented at Neural Networks for Signal Processing,NNSP'03. 2003 IEEE 13th Workshop on, 2003.

[87] N. Cristianini, o. Shawe-Taylor, and A. Elisseeff, "On Kernel-Target Alignment," *Advnaces in Neural Infromation Processing System*, vol. 14, 2001.

[88] G. Weiss and F. Provost, "The effect of class distribution on classifier learning," *Technical Report ML-TR 43, Department of Computer Science, Rutgers University*, 2001.

[89] R. Yan, A. Hauptmann, R. Jin, and Y. Liu, "On Predicting Rare Class with SVM Ensemble in Scene Classification " presented at IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'03), 2003.

[90] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. J. Smola, "Input Space vs. Feature Space in Kernel-Based Methods," *IEEE Transactions on Neural Networks*, vol. 10, pp. 1000-1017, 1999.

[91] T. G. Dietterich, A. Jain, R. Lathrop, and T. Lozano-Perez, *A comparison of dynamic reposing and tangent distance for drug activity prediction*, vol. 6. San Mateo,CA Morgan Kaufmann. . 1994.

[92] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to linear regression analysis*, 3rd ed. New York: Wiley, 2001.

[93] E. F. Osuna, R. Girosi, F. , "An improved training algorithm for support vector machines," presented at Proceedings of the IEEE Workshop Neural Networks for Signal Processing, 1997.

[94] R. H. L. T. G Dietterich, and T. Lozano-Perez. , "Solving the multiple instance problem with axis-parallel rectangles," *Artificial Intelligence in Engineering*, vol. 89, pp. 31--71, 1997.

[95] R. E. Schapire, "The Strength of Weak Learnability," *Machine Learning*, vol. 5, pp. 197-227, 1990.

[96] Freund, "Boosting a Weak Learning Algorithm by Majority," presented at {COLT}: Proceedings of the Workshop on Computational Learning Theory, 1990.

[97] O. L. Mangasarian and W. H. Wolberg, "Cancer diagnosis via linear programming," *SIAM News*, vol. 23, pp. 1&18, 1990.

[98] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology," presented at Proceedings of the National Academy of Sciences, USA, 1990.

[99] K. P. Bennett and O. L. Mangasarian, "Robust linear programming discrimination of two linearly inseparable sets," *Optimization Methods and Software* pp. 23-34, 1992.

[100] J. Zhang, "Selecting typical instances in instance-based learning," presented at Proceedings of the Ninth International Machine Learning Conference, Aberdeen, Scotland 1992.

[101] V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker, "Classification of radar returns from the ionosphere using neural networks," *Johns Hopkins APL Technical Digest*, vol. 10, pp. 262-266., 1989.

[102] Z. Q. Hong and J. Y. Yang, "Optimal Discriminant Plane for a Small Number of Samples and Design Method of Classifier on the Plane," *Pattern Recognition*, vol. 24, pp. 317-324, 1991.

[103] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes, "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus," presented at Proceedings of the Symposium on Computer Applications and Medical Care, 1988.