2016

PATTERN RECOGNITION IN CLASS IMBALANCED DATASETS

Nahian A. Siddique
Virginia Commonwealth University

PATTERN RECOGNITION IN CLASS IMBALANCED DATASETS

A Dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical & Computer Engineering at Virginia Commonwealth University.

by

NAHIAN ALAM SIDDIQUE
Bachelor of Science in Electrical & Electronic Engineering, BUET, 2011

Director: Yuichi Motai, Ph.D.
Associate Professor, Department of Electrical and Computer Engineering.

Virginia Commonwealth University
Richmond, Virginia
August, 2016.

Acknowledgement

First of all, I would like to express my deep gratitude to my MS supervisor Dr. Yuichi Motai. Lessons that I have learnt from him as mentee, will act as a guideline for my future career. I would also like to express my sincere appreciation to Dr. Ruixin Niu and Dr. Preetam Ghosh, members of my dissertation committee, for offering their time, experience and advice.

I would like to express my appreciation to Seonyeong Park, Emrah Benli and Ammar Osama, my fellow graduate students at the Sensory Intelligence Lab. I express my sincere appreciation to Ron Volpicella, my mentor at Commonwealth Center for Advanced Manufacturing, Virginia. I would also like to thank all the staff at the Department of Electrical and Computer Engineering.

This journey would never have been possible without the support of my family: my parents, Dr Md Nurul Alam Siddique and Anjuman Ara, my brother and his wife, Nafiul Alam Siddique and Tanjia Arif, my niece, Wania Izma Siddique, and my wife, Tasnia Subrin. I can never appreciate them enough. Last but not the least, I express my sincerest gratitude to the Almighty Allah.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ACT	Adaptive Feature-Space Conformal Transformation
ANN	Artificial Neural Network
AUC	Area Under the Curve
CPU	Central Processing Unit
CTC	Computed Tomographic Colonography
CUDA	Compute Unified Device Architecture
DSAE	Distributed Storage Access Emphasizes
ELM	Extreme Learning Machine
EM	Expectation-Maximum
FN	False Negative
FP	False Positive
GD	Gradient Descent
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
GRBF	Gaussian Radial Basis Function
GSVM-RU	Granular SVM-Repetitive Under-Sampling
KBA	Kernel Boundary Adjustment
KFD	Kernel Fisher Discriminant
KKT	Karush–Kuhn–Tucker
k-NN	*k* Nearest Neighbors
LHCG	Large Hadron Collider Grid

LM	Local Model
MLE	Maximum Likelihood Estimation
NN	Neural Network
RBF	Radial Basis Function
RKHS	Reproducing Kernel Hilbert Space
ROC	Receiver Operating Characteristics
SDC	SMOTE with Different Costs
SLFN	Single Hidden Layer Feed Forward Network
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
UCI	University of California, Irvine
WNN	Weighted Neural Network
WNN-UID	Weighted Neural Network for Under-Sampled Imbalanced Dataset

Abstract

PATTERN RECOGNITION IN CLASS IMBALANCED DATASETS

A Dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical and Computer Engineering at Virginia Commonwealth University.

by

Nahian Alam Siddique

Bachelor of Science in Electrical & Electronic Engineering, BUET, 2011

Director: Yuichi Motai, Ph.D.

Associate Professor, Department of Electrical and Computer Engineering.

Class imbalanced datasets constitute a significant portion of the machine learning problems of interest, where recognizing the ‘rare class’ is the primary objective for most applications. Traditional linear machine learning algorithms are often not effective in recognizing the rare class. In this research work, a specifically optimized feed-forward artificial neural network (ANN) is proposed and developed to train from moderate to highly imbalanced datasets.

The proposed methodology deals with the difficulty in classification task in multiple stages—by optimizing the training dataset, modifying kernel function to generate the gram matrix and optimizing the NN structure. First, the training dataset is extracted from the available sample set through an iterative process of selective under-sampling. Then, the proposed artificial NN comprises of a kernel function optimizer to specifically enhance class boundaries for imbalanced datasets by conformally transforming the kernel functions. Finally, a single hidden layer weighted neural network structure is proposed to train models from the imbalanced dataset. The proposed NN architecture is derived to effectively classify any binary dataset with even very high imbalance ratio with appropriate parameter tuning and sufficient number of processing elements.

Effectiveness of the proposed method is tested on accuracy based performance metrics, achieving close to and above 90%, with several imbalanced datasets of generic nature and compared with state of the art methods. The proposed model is also used for classification of a 25GB computed tomographic colonography database to test its applicability for big data. Also the effectiveness of under-sampling, kernel optimization for training of the NN model from the modified kernel gram matrix representing the imbalanced data distribution is analyzed experimentally. Computation time analysis shows the feasibility of the system for practical purposes. This report is concluded with discussion of prospect of the developed model and suggestion for further development works in this direction.

1 INTRODUCTION

Class imbalance [1] occurs frequently in datasets from many real-world applications, *i.e.* anomaly detection [2], intrusion detection [3], fraudulent detection [4], medical diagnosis [5] and web mining [6] etc.. In fact, majority of classification applications involve the class of interest to be a rare or at least a minority class concept [7]. Machine learning problems applicable in the medical field mostly involve recognizing a rare class, *i.e.* recognizing cancer cells, anomalous growth, tissue etc. [2], [5], [8]. For web mining and data mining, where data samples occur as stream—available at different times, pattern classification learning involves concept drift, target variable changing over time in unforeseen ways as well as highly imbalanced class distribution [9], [10]. Commerce related applications require machine learning algorithms to detect anomaly, suspicious transactions and error [11]–[13] both novel and previously encountered in nature. Thus, imbalance class datasets has grown into a very interesting topic of research in the present day machine learning community. A large number of studies have been conducted for investigating the impact of class imbalance on supervised machine learning, on the basis of training a classifier directly from a static set of training data [7]. In recent time, classification of imbalanced datasets is appearing in new applications and hence has emerged as one of the most critical and important topic of research in the machine learning community.

The contribution of this article is three-fold. Firstly, an under-sampling method that works independently and in conjunction with high dimensional conformal kernel feature transformation is proposed. Secondly, the high dimensional conformal kernel feature space, where the dataset in consideration is expected to be linearly separable, is modified to accommodate the linear learning machine without losing conformality and iterative transformation retraining. Thirdly, a single hidden layer feed-forward network with additive neurons for marginal performance improvement is proposed with fast convergence time as one of the goals. We present our work on implementing a new method of kernel modification method for a weighted neural network (NN) optimized for under-sampled imbalanced dataset (WNN-UID). We developed a segmentation method by creating subset of the sample problems and optimize the support vectors on that subset. Incorporation of kernel modification in weighted network model allows us to effectively classify non-linearly separable unevenly distributed datasets and transform the kernel space according to the dataset characteristics. Generally kernels and non-linear tools have the tendency to favor the majority class. We addressed that issue by adjusting kernel matrices to skew the hyper plane toward the majority class. With an intermediate objective to "rebalance" the data set under consideration, proposed methodology can improve classification performance by modifying the underlying data distribution in two aspects: extraction of informative samples that are essential for classification, and elimination of a large amount of redundant, or even noisy, samples. The developed method is compared against state of the art works at latter chapters.

1.1 ORGANIZATION OF THE THESIS

The remaining of this report is organized as follows. In chapter 2, a brief literature review as theoretical foundation for the proposed algorithm is presented. Chapter 3 discusses the overview of the proposed algorithm. Chapter 4 and 5 develop the proposed concept and theoretically analyze the merits and limitations of it. Chapter 6 contains the experimental results and the concluding discussions are presented in Chapter 7. For lengthy chapters, details of how the chapter is organized is presented in the introductory section of corresponding chapter. The conclusion is followed by a list of references.

2 LITERATURE REVIEW

Imbalance in data can be handled intuitively from two aspects of the problem [6], [7]. According to [1] class imbalance in dataset can be intrinsic and caused by the nature of the data itself or be extrinsic and caused by misrepresentation of one class for some foreign/unknown reason. Whatever causes the imbalance, it is often impossible to avoid class imbalance data and detection of the minority class is usually the case that is of interesting application. In [1], [7], [14] we see that learning from imbalanced datasets are particularly difficult because of the inherent linear nature of the fundamental machine learning algorithms that are used in the community. In the first section, we discuss about the state of research in machine learning community in dealing with class imbalance in training dataset. In the next following two sections, we take a closer look at some of these works organized according their respective fields of contribution. As a conclusion to the literature review, in section 2.4, a comparative analysis of some state of the arts methods is presented.

Table 2-1: List of Reviewed Works

Contribution Type	List of Works
Review & Survey	[1], [2], [3], [7], [14], [15], [18], [19], [20], [31]
Dataset Class Imbalance Reduction	[6], [17], [21], [23], [25], [26]
Algorithmic Optimization	[8], [12], [16], [27], [28], [34] [35], [38
Hybrid	[9], [13], [17]

2.1 TRAINING FROM CLASS IMBALANCED DATA

Chawla *et al* in [15] discussed about the emergence of class imbalanced data as an important and useful topic of research in machine learning community. They present the limitation of existing traditional learning machines for class imbalance data and discuss ways to solve these. The most potential solution being, reduce the existing imbalance present in training data. Zhou and Liu in [16] review on using state of the art methods in solving class imbalance difficulty in machine learning by modifying the underlying linear learning machine, i.e. artificial neural network, support vector machine, principal component analysis etc. Seiffert *et al* in [17] discussed effectiveness of hybrid ensemble methods, data boosting approaches and algorithm optimization for learning from class imbalanced data. In [18] Galar *et al* reviews several hybrid approaches on solving class imbalance learning.

In their survey paper [7], He and Garcia summarizes several approaches and demonstrate that, there are three types approaches in handling the difficulty in class imbalanced data. The first type is to modify the actual dataset for emphasizing the information needed to train the classifier. The second type transports the original dataset to a different space (using kernel trick) where both the classes become more evenly distributed in concept of a Euclidean space and choose that kernel space to be such that, it accommodates for accurate learning through a linear learning machine. The latter becomes a problem of finding an appropriate kernel feature space—to be more specific modifying any base kernel space to achieve the desired features. The third type involves with hybrid approaches and/or ensemble methods that are applicable to specific datasets. Here, we devise a strategy to solve the difficulty in learning from class imbalanced data. We refer to the first type of approaches as *training dataset class imbalance reduction* since these methods mostly modify the training dataset by respectively adding or subtracting samples into or from the original sample set to reduce class imbalance in the training set. We refer to the second type of approaches as *algorithmic optimization for class imbalanced training* since these mostly deal with modifying the kernel space where the linear machine can perform. Since we employ a variant of neural network as our base classifier we focus mostly on kernel modification for neural networks. In the following three sections we present some state of the art approaches on solving learning from class imbalanced data. Table 2-1 presents a summary of reviewed relevant state of the art manuscripts.

2.2 Training Dataset Class Imbalance Reduction

Training dataset class imbalance reduction methods attempts to reduce the ratio of number of majority class samples to number of minority class samples in training dataset by somehow changing the effective number of either the majority class samples or minority class samples or both in order to create a balanced class distribution of the training dataset. Typical approaches of training dataset optimization for class imbalanced learning applications consist of oversampling of the minority class [21], under-sampling [22] of the majority class or modification (data cleaning [23]) of an imbalanced data set by some mechanisms in order to provide a balanced distribution. Performance speed of the classification algorithm depend on the amount of critically considered data samples and effective computational complexity. For example, in kernel based SVM, $K(\mathbf{X}, \mathbf{X}_k)$, is regarded as a measure of similarity (/distance) between the new sample $\mathbf{X}$ and $\mathbf{X}_k$. This kernel function is calculated for each of the support vectors $\mathbf{X}_k$ for every new sample $\mathbf{X}$. Then, it is classified using the sum of these kernel values and a bias. One method to speed up the SVM classification is by decreasing the number of support vectors, which represent the critical data samples. Studies have shown that for several base classifiers (*i.e.* ANN, SVM, kNN etc.), a balanced data set provides improved overall classification performance compared to an imbalanced data set [7]. For most imbalanced data sets, the application of sampling techniques does indeed aid in improved classifier accuracy.

Fig. 1 demonstrates effect of a simple data modification on class separation boundary obtained using linear machine. Fig. 1.(a) depicts the original dataset along with the ideal separation boundary. In Fig. 1.(b), it is shown that, minority class samples are too few and are underrepresented compared to majority class. In Fig. 1.(c) the majority class is undersampled and the learned boundary approximates the target boundary more closely when compared to (b). From the working principle of SVM we can verify that (c) represents a preferable configuration (*i.e.* at least for a linear kernel) compared to (b). The classifier in Fig. 1.(c) uses a undersampled dataset for the negative class. The data imbalance ratio affects the classification performance heavily. For Fig. 1.(b), a large portion of the positive (minority) class training data is misclassified and is regarded as a poor performance because of low sensitivity (true positive rate). On the other hand,

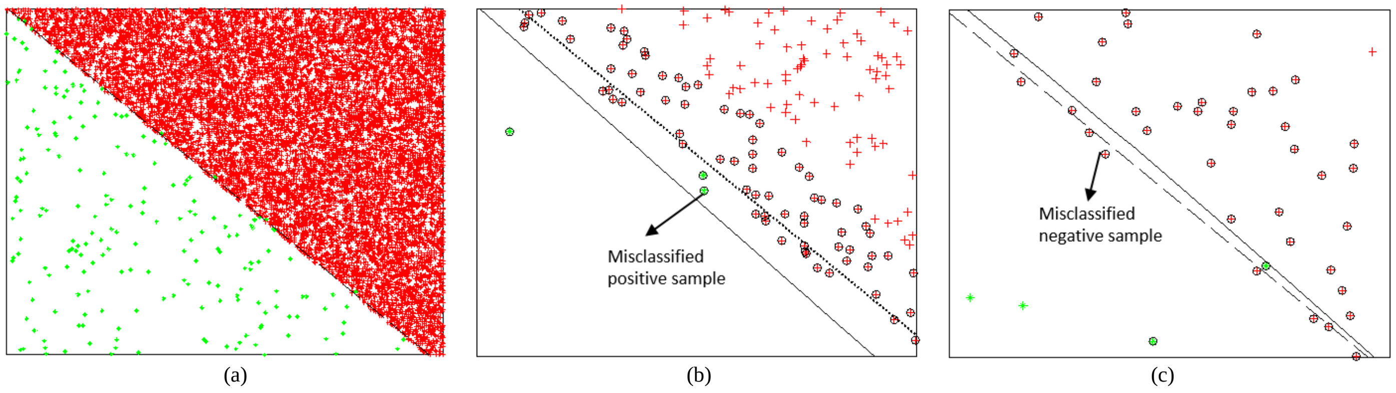

Figure 2-1: Effect of Data Modification. Increasing the ratio between number of minority class (green star) samples and number of majority class (red plus) samples increase the classifier performance. The broken line and solid line represent the generating (target) class separation line and learned separation line respectively. (a) Original dataset (green to red ratio=1:50), (a) Magnified view of a portion of the original dataset and class separation lines (c) Same magnification view of a portion of the dataset with udersampled majority class (green to red ratio=1:10) to reduce imbalance between two classes along with class separation lines. The gaps between the target and learned separation lines are much higher for highly imbalanced dataset (b).

for Figure 2-1.(c), even though some of the negative (majority) class samples are misclassified but it is considered a better classifier than that of Figure 2-1.(b), because of higher sensitivity. This justifies the necessity of modifying the training dataset to achieve improved performance.

Random under sampling removes randomly selected majority class samples from the original data set, S. In particular, we randomly select a subset of majority class examples from the entire majority class samples set, S_{maj}, and remove these samples from S so that $|S|=|S_{min}|+|S_{maj}|-|E|$, where E is the set of randomly selected majority class samples. Consequently, under sampling readily gives us a simple method for adjusting the balance of the original data set S, without any consideration of how the samples are distributed in input space or feature space.

Random under-sampling [22] is one of the most naive approach to curing difficultiy in calassifying imbalanced data, and often works surprisingly good considering the randomness of the method. But more often than not, it fails to provide very good result, especially when compared to other existing more complex approaches. The subset of majority class, generated randomly, fails to represent the actual class. A preferred alternative is informed under sampling, one example of which considers the K-nearest neighbors (KNN) of the corresponding sample to accomplish under sampling. Based on the characteristics of the given data distribution, four KNN under-sampling methods were proposed in [24]: NearMiss-1, NearMiss-2, NearMiss-3, and the “most distant” method. Experimental results suggest that these KNN based methods often achieve significant performance improvement over random sampling for imbalanced learning.

Tomek links [14] have been effectively applied to remove the overlapping that is introduced by sampling methods. Tomek links [14] can be defined as a data cleaning approach searching for pairs of minimally distanced nearest neighbors of opposite classes. One very popular, albeit computationally expensive, scheme for selective oversampling is the synthetic sampling: the synthetic minority oversampling technique (SMOTE) [25]. SMOTE is a powerful method that has shown a great deal of success in various applications. The SMOTE algorithm generate new artificial data samples between existing minority-class samples based on similarity measures. This essentially can be considered an interpolation technique [26] based on various metric, typically the Euclidean distance in input space.

There have been several published works in the community that apply general sampling and ensemble techniques to the linear learning framework. Some examples include the SMOTE with Different Costs (SDCs) method [27] and the ensembles of over/under sampled SVMs. In our work we were motivated by the excellent performance of under sampling done based on SVM by Tang *et al* [28]. In this algorithm representatives of the minority class are selected from the subset of support vectors, and thus the SVM itself works as the mechanism for under sampling.

2.3 ALGORITHMIC OPTIMIZATION FOR CLASS IMBALANCED TRAINING

Algorithmic optimization approaches modify standard linear learning algorithms to more effectively handle the class imbalance present in training dataset. Popular approaches include inter-class hyperplane optimization in the kernel feature space, uneven misclassification cost, kernel space optimization etc. Any linear machine learning algorithm can use the kernel mapping [29] to map the data from a Euclidean input space, $\mathcal{I}$, to a high-dimensional Hilbert feature space $\mathcal{H}$, in which a classification or regression problem becomes linear. Mercer's Kernel trick [30] allows to transform the data to an infinite dimensional Reproducing Kernel Hilbert Space (RKHS), $\mathcal{H}$, and use the advantage of higher dimension in pattern learning without actually dealing with the infinite dimension. The transformation from $\mathcal{I}$, to $\mathcal{H}$ is given by, k: $\mathcal{R}^d$x$\mathcal{R}^d$→$\mathcal{H}$. If k is a positive semi-definite symmetric function, gram matrix, K is defined by its component, as (1).

$$k_{ij} = \langle \Phi(x_i), \Phi(x_j) \rangle \qquad (1)$$

The mapped data reside on the high dimensional surface $\mathcal{S}$ in $\mathcal{H}$. The degree of freedom constrained by $\mathcal{S}$ is same as the dimensionality of the input space $\mathcal{I}$. The shape of $\mathcal{S}$ is determined by the associated mapping Φ. For a Φ with all derivatives continuous and defined, as in the case of any RBF function, the surface $\mathcal{S}$ in $\mathcal{H}$ is smooth, and thus can be considered as a Riemannian manifold. This is the sufficient condition to define a Riemannian metric, g_{ij}, for $\mathcal{S}$. A conformal transformation [31], also called a conformal mapping, is a transformation, T, which maps the elements $\Phi(\boldsymbol{X})$ in kernel space $\mathcal{H}$, to elements $\Phi(\boldsymbol{Y})(=T(\boldsymbol{X}))$ in new conformal kernel space $T(\mathcal{H})$ while preserving the local angles between the elements after mapping, where $\mathcal{H}$ is a domain in which the elements $\Phi(\boldsymbol{X})$ reside [32]. Usually, an analytic function is conformal at any point where it has a nonzero derivative. Some commonly used conformal functions are: $\boldsymbol{X}^2$, $e^{-\boldsymbol{X}}$, and $e^{-\boldsymbol{X}^2}$.

The distance between two points on $\mathcal{S}$ can be measured using two concepts: the distance between two points along a straight line in $\mathcal{H}$, which is the so-called Euclidean distance, and the distance between two points along a path on $\mathcal{S}$, to be specific the shortest distance along $\mathcal{S}$ computed by integration. This distance, called the Riemannian distance [33], is computed by a metric induced on $\mathcal{S}$. This metric is known as the Riemannian metric, denoted by g_{ij}—which is the computation unit for the distance between any two points, corresponding to axis i and j, on $\mathcal{S}$. The components of a Riemannian metric can be viewed as coefficients which are multiplied with the differential displacements d_{xi} in $\mathcal{I}$ to compute the distance ds in $\mathcal{H}$ in a generalized Pythagorean theorem,

$$ds^2 = \sum_{i,j} g_{ij} dx_i dx_j \qquad (2)$$

This distance can be calculated using Mercer's kernel trick as shown in (3).

$$g_{ij}(\mathbf{x}) = \left(\frac{\partial^2 K(\mathbf{x}, \mathbf{x}')}{\partial x_i \partial x_j} \right)_{\mathbf{x}'=\mathbf{x}} \qquad (3)$$

Kernels allow to, compute and utilize the pattern in Hilbert space through an inner product space. There are several standard functions for utilizing the kernel trick. Each type of kernel function demonstrate certain characteristics that can be beneficial to certain application. Table 2-2 lists some popular kernels and their parameters. Success of any classification method largely depends on the choice of kernel function. There is no such kernel function that is universally best for all application [31]. The choice of kernel for a specific application depends on characteristics of the data, mechanism of the employed algorithm, and target application. Often the most suitable kernel is found empirically [31]. The experiments in this research work employ following types of

Table 2-2: Popular Kernel Functions

Kernel Name	Kernel Function, $k(x,x')$
Linear	$x^T x'$
Polynomial	$(a \cdot x^T x' + 1)^b$
Gaussian RBF	$\exp\left(-\gamma \, \| x - x' \|^2\right)$
Laplacian	$\exp\left(- \| x - x' \| \big/ 2\sigma^2\right)$
Sigmoid	$\tanh(a \cdot x^T x' + \mathrm{b})$
Circular	$\cos^{-1}(x_1) - x_1\sqrt{1 - x_1^2}, \text{for } x_1 = \| x - x' \| \big/ \sigma < 1$

kernels: i) RBF Kernel, ii) Gaussian Kernel, iii) Polynomial Kernel, iv) Sigmoid Kernel and v) Linear Kernel.

Next, in section 2.4 we summarize some of the state of the art methods specifically developed to classify imbalanced datasets. We also point out their limitations and hence our proposal of an alternate to those methods as a mean of learning from imbalanced data.

2.4 COMPARISON AMONG THE RELEVANT STUDIES

In table 2-3, we briefly present some of the recent research works and compare our method with those in order to see where the proposed method lies compared to these. The SVM-WEIGHT [34] implements cost sensitive learning to handle imbalance in the representation of two classes. Cao's ESPO and INOS [21] computation requirement grows exponentially for very sparsely distributed datasets and becomes very slow with large databases. In [25] Chawla et al demonstrate effectiveness of SMOTE algorithm by synthetically oversampling the minority class and thus reduce representative imbalance between two classes. Akbani's modification of SVM-SMOTE [27] is not suitable for handling big data due to its tendency to inflate number of training samples. Wu's KBA [35] uses conformal transformation of the kernel function and thus optimizing the hyperplane that differentiates between the two classes. Extreme learning machines (ELM) [36], [37] are a major category of kernel NN learning methods. Zong's weighted ELM [38] is specifically developed for classification of imbalanced datasets. Now all of these work well if the imbalance is moderate. Table 2-3 outlines their principals and limitations.

The proposed method is devised to overcome these limitations by reducing the imbalance between representation of the two classes and also optimizing the kernel function. Compared to SVM-WEIGHT, as the penalty optimization is not relevant here as SVM will be used for under-sampling.

Table 2-3: Comparison of Similar Methods

Work	Principal	Limitation(s)
SVM-WEIGHT [14]	Cost sensitive learning. Larger penalty for false negatives than for false positives	No mechanism for penalty optimization
ESPO [15]	Structure preserving oversampling	Optimizes feature space, not kernel space
Weighted ELM [17]	ELM modified to incorporate weight reflecting class distribution.	Achieves good performance for moderately imbalanced class distribution.
SMOTE [25]	Synthetic over-sampling of minority class. Synthetic samples are placed on linear connections in the input plane.	Synthetic sampling is done in the input feature space
SVM-SMOTE Akbani [21]	SVM modified with SMOTE variant for class imbalanced training	Very slow convergence with larger datasets
KBA [35]	Optimization of class separating hyper-plane to adjust for the imbalanced class ratio.	No explicit mechanism for controlling varying class imbalance ratio. Only cost based optimization
WNN-UID	Majority class is repeatedly under-sampled using segmented SVM. Finally remaining samples are used with a conformal transformation of kernels.	~

Also, as SVM is used for the under-sampling, the selection of informative samples are done in the higher order kernel space instead of the input feature space as in SMOTE. And as the kernel boundary is adjusted after the under-sampling is completed, the computation cost is much less than KBA. Weighted ELM is a modification of the original ELM theory [36] to work with imbalanced datasets. The limitation of weighted ELM is it linearly reacts to increase in imbalance ratio and hence is not suitable for highly imbalanced datasets. More justification on these claims about the proposed algorithm are presented in the following sections.

3 PROPOSED ARCHITECTURE: OVERVIEW

A brief overview of the proposed classifier architecture is presented in this chapter. The complete process of developing an artificial NN optimized for training using a class imbalanced database is discussed as a sequence of two consecutive methods. In Figure 3-1 an outline of the proposed methodology is shown.

In chapter 4, we discuss the dataset modification method—which is the first stage of the proposed methodology. Chapter 4 is comprised of two sections. The dataset modification method is primarily a technique of subsampling the majority class samples with minimum information loss. In section 4.1, we present the proposed method of repetitive under-sampling of the majority class

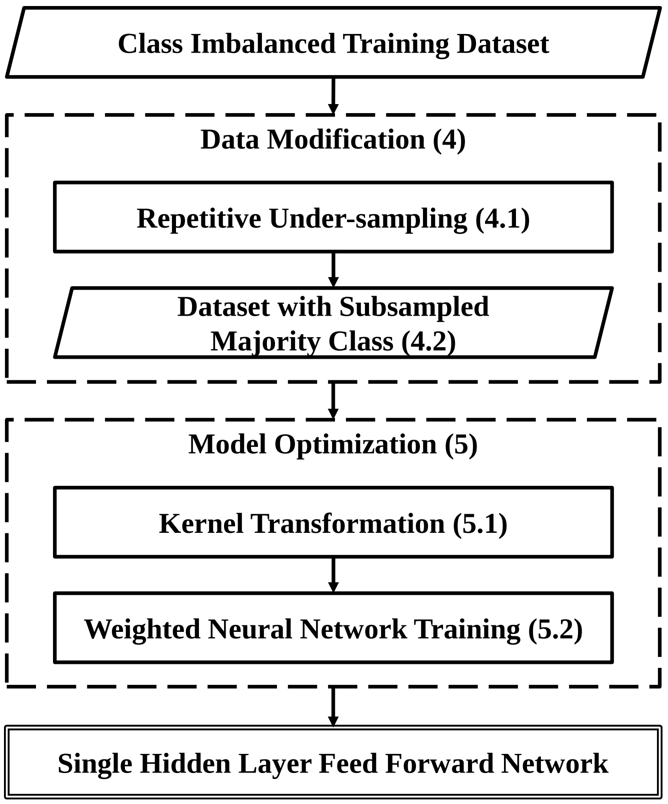
Class Imbalanced Training Dataset
Data Modification (4)
Repetitive Under-sampling (4.1)
Dataset with Subsampled Majority Class (4.2)
Model Optimization (5)
Kernel Transformation (5.1)
Weighted Neural Network Training (5.2)
Single Hidden Layer Feed Forward Network

Figure 3-1: An outline of the proposed methodology.

samples and in section 4.2, we present the information loss associated with the sub-sampling process.

In chapter 5, we discuss the method for developing the artificial NN model. Chapter 5 is also comprised of two sections. Before designing the NN, the data must be transformed to a high dimensional feature space, namely Hilbert space, to allow for considering the inherent non-linear differentiation present in between the two classes of the dataset. Section 5.1 discusses a kernel transformation method, again optimized for class imbalanced data, and section 5.2 discusses the training of the proposed WNN.

4 DATASET MODIFICATION: LOSS-OPTIMIZED UNDER-SAMPLING

Objective of the data modification stage is to reduce the existing imbalance between two classes by subsampling the majority class. To address the relative imbalance, in 4.1 the majority class sample is first randomly partitioned into smaller segments and then each segment is processed with the entire minority sample. Each of the smaller subset is then modeled and only the informative samples are identified. Only these informative majority class samples are considered in the next step of the algorithm and thus reducing the imbalance between representations of the two classes. In 4.2 we justify this approach by computing information loss and compare with random under-sampling and Synthetic Minority Over-sampling Technique (SMOTE). Tang *et al* in [22] measure granularity of the dataset and under-sample randomly to reduce the class imbalance present in the data. Several of recent research works on learning imbalanced data reduces inherent imbalance by similar under-sampling techniques. The dataset modification stage of the proposed methodology employs the same underlying principal for dataset modification for imbalanced datasets that, both principal components and support vectors tend to recognize the most important samples for learning. We under-sample the majority class samples to reduce inter-class imbalance and create a more favorable functioning environment for the main learning algorithm. The subsequent stages of the proposed methodology takes advantage of the reduced imbalanced data, and employs kernel optimization and neural network weighting to deal with class imbalance. The proposed methodology only partially depends on the data modification stage (we present a section in the experiments section with results demonstrating performance of the proposed method in absence of the data modification stage).

4.1 REPETITIVE UNDER-SAMPLING

Repetitive under-sampling is the first stage of the proposed methodology, as can be seen from Figure 4-1. Objective of this stage is to reduce, not eliminate, the class imbalance present in the training dataset. We describe the under-sampling technique in three steps: Random Segmentation, Local Modeling and Merging. In step 1, the negative samples (or the majority class samples) are randomly segmented into several parts. In step 2, each segment of negative samples and the entire set of positive class samples are passed to a kernel based support vector machine training algorithm which then identifies the informative samples as support vectors or principal components. In step 3, these separately selected informative samples are aggregated and the remaining is passed to the

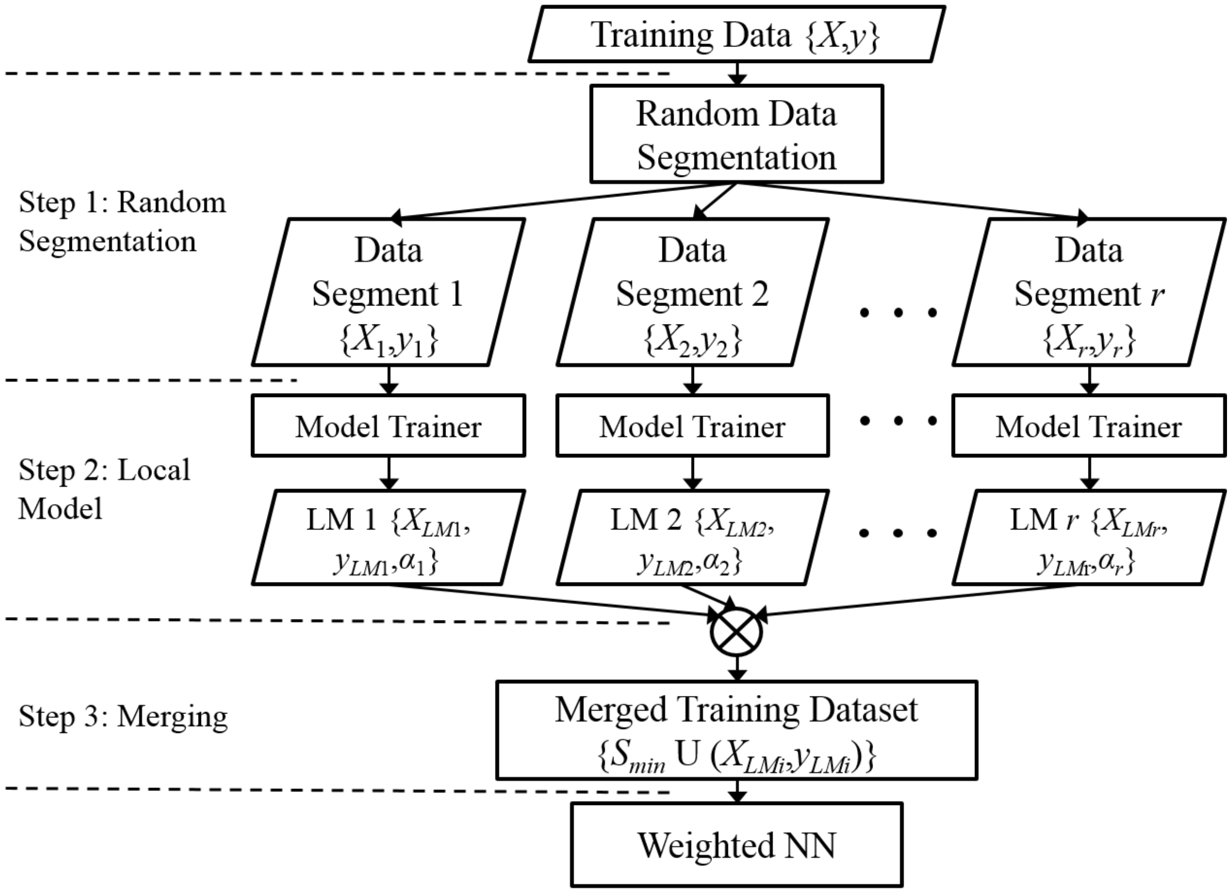

Figure 4-1: The three steps of Repetitive Under-sampling. The majority class is subsampled repeatedly by identifying informative samples as in LM.

step 1 for another iteration of the process until certain conditions are fulfilled. Figure 4-1 shows these steps and in the following paragraphs, these three steps are described in details.

4.1.1 Random Segmentation

Step 1 is the random segmentation, where the class imbalanced data is segmented into smaller sets with intention to reduce class-imbalance. Here, the training dataset is denoted by S, which contains large number of negative samples and small number of positive samples. Each sample in S can be described as (x_i,y_i) tuple, where $x_i \in \mathscr{R}^d$ is a sample described in d dimensional Cartesian space and $y_i \in \{+1,-1\}$ is the class label. The set of all negative (also referred to as majority) class samples is S_{maj}, and the set of all positive (also referred to as minority) class samples, S_{min}. In order to deal with the imbalanced class representation, S_{maj} is divided randomly into several negative class segments. Number of negative class segments depends on the class imbalance ratio and the actual dataset characteristics and is determined empirically. Each of these negative class segments are then used to construct a data segment. The i^{th} data segment is represented as $S_i{:=}\{X_i,Y_i\}$, where X_i is the set of sample descriptions and Y_i denotes the set of corresponding class labels. X_i contains all of the positive class samples as well as the i^{th} negative class segment. If the number of extracted data segments is r, we can write,

$$\begin{gathered}\bigcup_{i=1}^{r} S_i = S \\ S_i \cap S_j = S_{\min}\end{gathered} \qquad (4)$$

Here, in the second equation, $i{\neq}j$ and $i,j \in \{1,2,\ldots,r\}$.

4.1.2 Local Model

Step 2 of the under-sampling stage is construction of a *local model*, which can be constructed over a kernel based statistical machine learning process. This is the core step of the under-sampling stage. Each of the data segment, S_i—generated in step 1, is modeled using either kernel Fisher discriminant (KFD) or support vector machine (SVM). KFD can be used to extract the informative negative samples. The principal components in KFD are generated as a linear combination of training samples. The training samples with higher Eigen value are more informative for the classification task [8]. For the i^{th} data segment, S_i, let the output coefficient vector of KFD is given by α_i, which is a vector of length $|S_i|$. Here, $|\cdot|$ operator is used to denote the number of elements of the enclosed set. The values in α_i correspond to the weight of the respective sample towards constructing a classifier. On the other hand for SVM, usefulness of a sample in computing the

```
Begin
        Initialize
    $X_{LMi} \leftarrow \{\}$
    $y_{LMi} \leftarrow \{\}$
  End Initialize
  KFD or SVM optimization $\rightarrow \alpha_i$
  Normalize $\alpha_i$
  For $idx \leftarrow$ 1 to $|S_i|$
    If $|\alpha_i(idx)| \geq \theta_\alpha$
      Include $X_i(idx)$ in $X_{LMi}$
      Include $Y_i(idx)$ in $y_{LMi}$
    End if
  End for
End
```

Figure 4-2: Algorithm for extracting Local Model.

class boundary is determined its Lagrange coefficient. The Lagrangian in the SVM model can be compared to the α_i for the KFD. A higher coefficient in α_i corresponds to a more useful sample for the purpose of reconstruction of the database in consideration. Figure 4-2 show the algorithm for step 2: Local Model (LM) for a Data Segment.

4.1.3 Merging

Step 3 is merging local models extracted in step 2 to construct a unified dataset for the WNN. To reduce the imbalance in representations of the two classes, all samples from S_{min} is included in the unified dataset without repetition, while only the important samples, as determined in step 2, are included in the final training set. A kernel based piecewise linear classifier should be able to perform at minimum as good as the weakest of the local models used to construct the merged dataset. For simplicity and efficiency, we assume that, all minority samples are informative, hence the merged dataset should include entire of the minority samples. Construction of the merged dataset can be expressed by,

$$S_U := S_{\min} \bigcup_{i=1}^{r} \{\mathrm{X}_{LMi}, \mathrm{y}_{LMi}\} \qquad (5)$$

—where the merged dataset, S_u, is a superset of the set of all samples from minority class and the informative negative samples as extracted from step 2, local modeling. If in step 2 all informative negative samples are extracted using local models, S_u contains all of them and hence can be regarded as sufficient for achieving optimum classification performance.

4.2 Dataset with Sub-sampled Majority Class: Associated Information Loss

The sub-sampling process mentioned in section 4.2 suffers from the difficulty in determining optimum number of data segments, r. Optimum value of r depends on several characteristics of the training dataset. Intuitively it seems that, extraction of more data segments to reduce information loss would result in the best result. A very high value of *r* would result in very little reduction in majority class samples for the machine learning algorithm—which is the first problem we want to solve in this work. Moreover, information contributed by two different segments may be contradictory to each other in some cases (*i.e.* when each of the segments are largely insufficient and represent different concept of the classes). Hence, lesser number of data segments may be preferred over many data segments.

Concept of entropy is used to characterize the (im)purity of an arbitrary collection of examples. Information Gain is the expected reduction in entropy caused by partitioning the examples according to a given attribute. The entropy of an arbitrary set, with *r* different subsets in it, is given by:

$$I(S) = -\sum_{i=1}^{r} P(S_i)\log_2(P(S_i)) \qquad (6)$$

Here, $P(S_i)$ denotes the probability density function corresponding to the sample set under consideration. And for a specific division, D_r, the gain ratio is given by,

$$G(S, D_r) = -\sum_{v\in\{1,2,\cdots,r\}} \frac{S_v}{|S|} I(S_v), \quad \forall S_v \in \left\{\bigcup S_i\right\} \qquad (7)$$

To ensure that each of the local models are optimum, step 1 through 2 are repeated as long as the local models are equally good. To construct the concept of goodness of a model, the weakest performance of any classifier is identified as that of a naïve classifier, which classifies all sample to be from the majority class. That is why the simple set theory approach to merge the local models in an aggregative manner without any repetition is adopted. The gain ratio associated with a naïve classifier is given by,

$$G(S,D_N) = -\sum_v \frac{|S_v|}{|S|} I(S_v),\ \forall S_v \in D_N := \left\{S_{maj}, S_{\min}\right\} \tag{8}$$

After every iteration, $G(S,D_r)$ is computed and the under-sampling continues as long as the decrease in magnitude of $G(S,D_r)$ becomes insignificant, and $G(S,D_r) > G(S,D_N)$. The iteration is stopped when one of the following conditions are met: a preset number of iterations are completed or $G(S,D_r)$ starts to decrease. This, minimizes information loss associated with the applied subsampling process.

The information loss associated with the segmentation process can be parametrically measured using (8) and hence can be numerically optimized throughout the proposed methodology. The entropies corresponding to sub-sections in 4.1 are shown in rest of this section. It will be evident from the entropy calculation that, the proposed under-sampling method successfully reduces class imbalance but still retain the necessary information. In the repetitive under-sampling process, the initial entropy of the original un-segmented dataset is given by,

$$\begin{aligned} I(S) = &-\frac{|\mathrm{S}_{maj}|}{|\mathrm{S}_{maj}|+|\mathrm{S}_{\min}|}\log\left(\frac{|\mathrm{S}_{maj}|}{|\mathrm{S}_{maj}|+|\mathrm{S}_{\min}|}\right) \\ &-\frac{|\mathrm{S}_{\min}|}{|\mathrm{S}_{maj}|+|\mathrm{S}_{\min}|}\log\left(\frac{|\mathrm{S}_{\min}|}{|\mathrm{S}_{maj}|+|\mathrm{S}_{\min}|}\right) \end{aligned} \tag{9}$$

4.2.1 Random Segmentation

After the j^{th} iteration of step 1, the random under-sampling, the new entropy of the system is given by,

$$\begin{aligned} I_j(S) = &-\frac{|\mathrm{S}_{\min}|}{|\mathrm{S}_{maj}|+|\mathrm{S}_{\min}|}\log\left(\frac{|\mathrm{S}_{\min}|}{|\mathrm{S}_{maj}|+|\mathrm{S}_{\min}|}\right) \\ &-\sum_i \frac{|X_i|}{|X_i|+|\mathrm{S}_{\min}|}\log\left(\frac{|X_i|}{|X_i|+|\mathrm{S}_{\min}|}\right) \end{aligned} \tag{10}$$

Here, *i* denotes the index of the random segment and the summation is computed over all the segments. There exist only one term that corresponds to the minority class, since minority class is not segmented. Equation (10) represents a specific case of (8), where the summation parameter *i* corresponds to each segment in *j*th iteration. Either of (8) or (10) can be used to find the optimum segmentation. After several iteration of the method described in IV.*A*, the random under-sampling process, $I_j(S)$ starts to saturate and the segmentation iteration is stopped.

4.2.2 Local Model

The entropy of the k^{th} local model is given by,

$$I_{LMk} = -\frac{|X_{pk}|}{|X_{LMk}|}\log\left(\frac{|X_{pk}|}{|X_{LMk}|}\right) - \frac{|X_{nk}|}{|X_{LMk}|}\log\left(\frac{|X_{nk}|}{|X_{LMk}|}\right) \qquad (11)$$

Here X_{pk} and X_{nk} denotes the set of positive support vectors and the set of negative support vectors of the k^{th} local model respectively. And the associated imbalance ratio of the k^{th} model is much lower as expected.

4.2.3 Merging

After the 3rd step, merging, is completed—the entropy of the under-sampled dataset is given as the summation of all the I_{LMk}s. Now, for the entropy,

$$|X_{nk}| < |S_{maj}| \Rightarrow \sum_k \frac{|S_{\min}|}{|X_{nk}|} > \frac{|S_{\min}|}{|S_{maj}|} \qquad (12)$$

Hence, (12) justifies utility of the segmentation step of the proposed algorithm for a dataset with imbalanced class. In the next chapter, our proposed model optimization technique is described.

5 MODEL OPTIMIZATION FOR CLASS IMBALANCED DATASET

Model optimization encompasses the entire process that develops a suitable model for capturing the characteristics of the target classes based on available input features. To deal with the inherent non-linearity of the target classes, in V.*A* a kernel optimization technique is presented. This proposed kernel optimization technique accept the moderately imbalanced input data, and transform it into sample set residing in a Hilbert space optimized for a learning machine. All the computation required for the classification must be valid after the optimization is completed in an inner product space. The dataset in consideration is then trained and classified using a kernel based weighted neural network (WNN) training algorithm. So, in V.*B* our proposed method of optimizing a universal single hidden layer feed-forward network is described.

5.1 KERNEL TRANSFORMATION

The kernel function is used to map the data from input space to high dimensional Hilbert feature space. In order to address the imbalance between the two classes' representation in the training dataset, the initially chosen kernel is modified to enhance the high dimensional hyperplane region that separates the two classes. In the following two sub-sections the mathematical foundation of modifying the kernel function and derive an expression to approximate the advantage of kernel transformation are described respectively.

5.1.1 Mathematical Foundation

To optimize an ordinary kernel function to gradually be transformed into a kernel function that better suits the class imbalance problem, a conformal function is associated with the original kernel function as a multiplication factor. As long as the conformal function is positive semi-definite for practical purposes and is continuous, the modified function is a valid kernel function. Let, the original kernel function from (1) be modified as:

$$\tilde{K}(\mathbf{x},\hat{\mathbf{x}}) = D(\mathbf{x})D(\hat{\mathbf{x}})K(\mathbf{x},\hat{\mathbf{x}}) \qquad (13)$$

The most suitable conformal function, $D(\cdot)$, is found to be very much data dependent just like the kernel function itself. As long as, $D(\cdot)$ is positive continuous function, $\tilde{K}_{us}(\cdot,\cdot)$ is a valid kernel function which is the only requirement to apply Mercer's kernel trick.

To find a suitable function for $D(\cdot)$, the Riemannian manifold distance associated with the RKHS is examined. The incremental distance, *ds*, along the margin is needed to be magnified in order to aid the classification of the imbalanced dataset. Look into the distance as defined in the RKHS:

$$ds^2 = \| d\mathbf{z} \|^2 = \| \Phi(\mathbf{x}+d\mathbf{x}) - \Phi(\mathbf{x}) \|^2 \qquad (14)$$

Here, ***dz*** denotes the boundary in the RKHS and ||·|| operator is used to denote the L-norm of the enclosed vector quantity. Applying the Taylor's expansion for vector spaces and ignoring higher orders of the differential terms, we can write for all practical purposes:

$$\Phi(\mathbf{x}+d\mathbf{x}) = \Phi(\mathbf{x}) + \Phi'(\mathbf{x})d\mathbf{x} \qquad (15)$$

Here, $\Phi'(\mathbf{x}) = \frac{d}{d\mathbf{x}}\Phi(\mathbf{x})$. Substituting (15) in (14), get,

$$\begin{aligned} ds^2 &= \| \Phi'(\mathbf{x})d\mathbf{x} \|^2 \\ &= d\mathbf{x}^T \Phi'(\mathbf{x})^T \Phi'(\mathbf{x})d\mathbf{x} \\ \therefore ds^2 &= \sum_{k,l} \left. \frac{\partial k(\mathbf{x},\hat{\mathbf{x}})}{\partial x_k \partial \hat{x}_l} \right|_{\hat{\mathbf{x}}=\mathbf{x}} dx_k dx_l \end{aligned} \qquad (16)$$

Substituting the kernel function *k*() in (16) by (13) gives,

$$ds^2 = \sum_{k,l} \left. \frac{\partial D(\mathbf{x})D(\hat{\mathbf{x}})k(\mathbf{x},\hat{\mathbf{x}})}{\partial x_k \partial \hat{x}_l} \right|_{\hat{\mathbf{x}}=\mathbf{x}} dx_k dx_l \qquad (17)$$

Let, $$g_{kl}(\mathbf{x}) = \left. \frac{\partial D(\mathbf{x}) D(\hat{\mathbf{x}}) k(\mathbf{x}, \hat{\mathbf{x}})}{\partial x_k \partial \hat{x}_l} \right|_{\hat{\mathbf{x}}=\mathbf{x}}$$

And, $$D(\mathbf{x}) = \sum_m \exp\left(-\frac{\| \mathbf{x} - \mathbf{x}_m \|_n^n}{\theta} \right) \tag{18}$$

Where, $\mathbf{x}_m$ is one of the informative samples as selected in section 4.1, and θ is a normalization factor chosen by the system designer. Number of samples near the boundary is a good choice for θ. Equation (17) and (18) together completes the kernel optimization for the imbalanced dataset. The derived kernel is used to formulate the gram matrix and passed on to the proposed weighted neural network for classification.

5.1.2 Entropy Analysis

Let, for the under-sampled dataset, S_{us}, the modified gram matrix be denoted by $\widetilde{K}_{us}$ and for dataset, S, the modified gram matrix be denoted by $\widetilde{K}$. Also let, the sorted list of Eigen values corresponding to the gram matrix, $\widetilde{K}$, be: $\Lambda=\{\lambda_1, \lambda_2, \lambda_3, \dots ,\lambda_n\}$ and the sorted list of Eigen values corresponding to the gram matrix $\widetilde{K}_{us}$ be: $\Lambda_{us}=\{\lambda'_1, \lambda'_2, \lambda'_3, \dots ,\lambda'_{ru}\}$. When, S_{us}, is a representative subset of S,

$$\Lambda_{us} \mathrel{\tilde{\subset}} \Lambda \tag{19}$$

The information loss of repetitive sub-sampling in high dimensional feature space representation is given by,

$$SSE_{Dr} = \sum_{i=1}^{|S_{us}|} (\lambda_i - \lambda'_i)^2 + \sum_{i=|S_{us}|+1}^{|S|} \lambda_i^2 \tag{20}$$

The experimental objective of the methodology is to minimize SSE_{Dr}. Now, to compute the information loss when the general kernel is used, let, for S_{us}, the gram matrix be denoted by K_{us} and for S, the gram matrix be denoted by K. Also let, the sorted list of Eigen values corresponding to K, be: $\Lambda^*=\{\lambda^*_1, \lambda^*_2, \lambda^*_3, \dots , \lambda^*_n\}$ and the sorted list of Eigen values corresponding to K_{us}, be:

$\Lambda^*_{us}=\{\lambda'^*_1, \lambda'^*_2, \lambda'^*_3, \ldots, \lambda'^*_{ru}\}$. Here, the information loss of repetitive sub-sampling in high dimensional feature space representation is given by,

$$SSE^*_{Dr} = \sum_{i=1}^{|S_{us}|} (\lambda^*_i - \lambda'^*_i)^2 + \sum_{i=|S_{us}|+1}^{|S|} \lambda^{*2}_i \tag{21}$$

The second terms in (20) and (21) are already very small since the under-sampled training set is assumed to contain most of the necessary information for separating the two classes. So (20) and (21) respectively becomes,

$$SSE_{Dr} \approx \sum_{i=1}^{|S_{us}|} (\lambda_i - \lambda'_i)^2 \tag{22}$$

$$SSE^*_{Dr} \approx \sum_{i=1}^{|S_{us}|} (\lambda^*_i - \lambda'^*_i)^2 \tag{23}$$

Comparing (20) and (21), the first terms correspond to the repetitive under-sampling process. The kernel transformation, described in (13), increases the distance of the training samples from the hyper-plane in average. This shifts the magnitude of (20) or (21) towards the first term from the second term, and hence we can write:

$$\frac{SSE_{Dr}}{SSE^*_{Dr}} \approx \frac{\sum_{i=1}^{|S_{us}|} (\lambda_i - \lambda'_i)^2}{\sum_{i=1}^{|S_{us}|} (\lambda^*_i - \lambda'^*_i)^2} > 1 \tag{24}$$

So, the information loss for the kernel modification is less than that for the original kernel method. This demonstrates the utility for the kernel transformation.

5.2 WEIGHTED NEURAL NETWORK

The proposed NN is a single hidden layer feed forward network (SLFN). An interesting characteristics of artificial NN is that, they can be optimized in different ways to approximate the

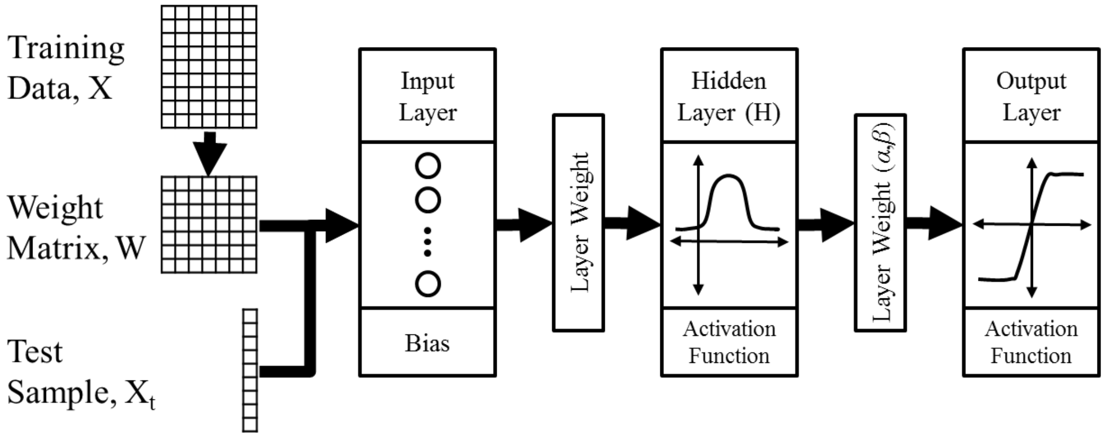

Figure 5-1: Functional outline of WNN. The weight matrix, *W*, is an accessory to the training of the SLFN and is not used for normal operation.

same target system [37]. In 2006 Huang *et al* showed a single hidden layer NN with randomly generated hidden layer weights can approximate any target function, given that it contains sufficient number of neurons, by optimizing only the output layer weights [36], whereas the nodes remain random. We discuss the proposed NN in following paragraphs.

5.2.1 Overview of the Weighted Network Architecture

To optimize the training process of an ordinary SLFN for the class imbalanced training dataset, an optimized weight matrix *W* is associated with the input data. The conceptual diagram of the network is shown in Figure 5-1. The training data, *X*, along with the label vector is coupled with a diagonal weight matrix, *W*. The input layer associates it with corresponding weights and pass as input to the randomly selected hidden layer. The hidden layer passes the data through respective activation function of the corresponding processing element and the output of hidden layer is transferred to the output layer activation function through another layer of weight matrix.

The output of the output layer is the network output which, in ideal scenario, matches with the input labels for training dataset. Notice, the dimension of input data as well as the number of hidden layer neurons are variable. The weight matrix affects the way that network treats individual sample of the input data only in training stage and inherently is not part of the network. So, the weight matrix is a component of training the network, not a component of the trained network that classifies unknown samples. Hence, while the network is not learning/training the weight matrix W has no effect on its operation and can be treated as a general SLFN.

5.2.2 Mathematical Model

In order to present the proposed WNN, first the notations are defined. For a given training dataset $\{X,Y\}$, let $X=[x_1,x_2,\ldots,x_N]$ and $Y=[y_1,y_2,\ldots,y_N]^T$. y_i denotes the class label while x_i is the sample description. An NxN diagonal matrix W is associated with every training sample x_i. Intuitively for x_i from the minority class (positive class), the associated weight W_{ii} is relatively larger than for x_i from majority class (negative class). To maximize the marginal distance and to minimize the weighted cumulative error with respect to each sample, we have an optimization problem:

Minimize: $\| H\beta - Y \|^2$ and $\| \beta \|$

Subject to: $H\beta = Y + Z$

Where, H is the output of the hidden layer and β is the weight vector connecting hidden layer to the output layer. And the error matrix $Z=[\zeta_1, \zeta_2,\ldots, \zeta_N]^T$.

So, the optimization problem can be written as:

Minimize, $L_P := \| \beta \|^2 + W \sum_{i=1}^{N} \|\zeta_i\|^2$ (25)

Where, $\zeta_i = y_i - h(x_i)\beta$, and, $H=[\, h(x_1); h(x_2);\ldots; h(x_N)]$

Equation (25) is the formulation of the optimization problem addressed in the preceding paragraph. The constraints β and ζ_i are defined as above and (25) can be minimized equating first derivatives with respect to β, W and ζ_i to zero. Determining the optimal weight matrix, W, is critical for the performance of the algorithm for highly imbalanced datasets. We set W as a diagonal matrix, where diagonal element corresponding to sample $\mathbf{x}_i$ is given by, $w_{ii}=1/n_s$. Here, n_s corresponds to the effective number of training samples of the class, that is represented by x_i.

If we transform the data to a RKHS, $\mathcal{H}$, the transformation from input space, $\mathcal{I}$, to $\mathcal{H}$ is given by, $k:\mathcal{R}^d \times \mathcal{R}^d \rightarrow \mathcal{H}$. If K is the gram matrix, where $k_{ij}=\langle\Phi(\mathbf{x}_i),\Phi(\mathbf{x}_j)\rangle$, then K is a Mercer kernel matrix. According to Karush–Kuhn–Tucker (KKT) theorem, the equivalent dual optimization problem becomes minimization of:

$$L_D := \|\beta\|^2 + CW\sum_{i=1}^{N}\|\zeta_i\|^2 - \sum_{i=1}^{N}\alpha_i(\mathrm{h}(\mathrm{x}_i)\beta - y_i + \zeta_i) \quad (26)$$

—where α_i is the i^{th} Lagrangian coefficient of the hidden layer and C is an arbitrary scalar which is constrained only by (27) and (28). Introduction of C ensures the numerical solution in (27) and (28) converges and to minimize the effect of introduction of an arbitrary constant, C should be as large as the numerical solution allows. The only constrain on β and ζ are imposed by their definitions, as mentioned prior to (25). Hence, each element of β and ζ is a finite real number. Inclusion of the third added term in the problem enables the network to accommodate an ability to emphasize specific training samples, *i.e.* the minority class samples as well as the flexibility to sustain occasional error in training samples. Equation (26) can be solved under two different conditions, for $N \leq L$ and for $L<N$, as shown in (27) and (28). How to obtain (27) and (28), is shown in next paragraph.

When, $N \leq L$, (26) can be solved to,

$$\beta = H^T \left(I/C + WHH^T \right)^{-1} WY \tag{27}$$

And, for $N > L$, (26) can be solved to,

$$\beta = \left(I/C + H^T WH \right)^{-1} H^T WY \tag{28}$$

Equation (27) and (28) show that this proposed architecture does not require iterative parameter updates of β. Rather, H, W, and Y can directly calculate the parameter β. Equation (27) (28) are derived as shown in Proof below.

Proof:

Let, α=[α_1, α_2,…, α_N].

By taking partial derivatives of (26) with respect to β and equaling to zero, we get,

$$\frac{\partial L_D}{\partial \beta} = 0 \rightarrow \beta - \sum_{i=1}^{N} \alpha_i h(x_i) = 0$$
$$\therefore \beta = (\alpha H^T)^T = H\alpha^T \tag{29}$$

Again, taking partial derivatives of (26) with respect to ζ_i and equaling to zero, we get,

$$\frac{\partial L_D}{\partial \zeta_i} = 0 \rightarrow CW\sum_{i=1}^{N} \zeta_i - \sum_{i=1}^{N} \alpha_i = 0$$
$$\therefore \alpha_i = CW\zeta_i \ , \quad \forall i \in \{1, 2, \cdots, N\} \tag{30}$$

And again, taking partial derivatives of (26) with respect to α_i and equaling to zero, we get,

$$\frac{\partial L_D}{\partial \alpha_i} = 0 \rightarrow -\sum_{i=1}^{N} (\mathrm{h}(\mathrm{x}_i)\beta - y_i + \zeta_i) = 0$$
$$\therefore y_i - \zeta_i = \mathrm{h}(\mathrm{x}_i)\beta \ , \quad \forall i \in \{1, 2, \cdots, N\} \tag{31}$$

—which is the defining equation for ζ_i. While, (29), (30) and (31) constitute the KKT optimality condition. Substituting ζ_i in (30) using (31) and then substituting α in (29) gives us β. Depending on the rank of the matrix H, we employ inverse or pseudo-inverse to solve for the system.

When, $N \leq L$, right pseudo inverse is computationally less expensive and we determine the solution as (27). And, when $L < N$, left pseudo inverse is computationally less expensive and we determine the solution as (28).

■

This proves (27) and (28)—which constitute the functional basis of the proposed NN structure. The identity matrix I in the solutions are included to avoid the singularity incidence that often occur in practical databases. For a very high value of C, C^{-1} approaches zero, hence the effect of including the extra term in solution affects negligibly to the numerical solution as long as differentiable by the numerical computation precision. For example, with computation environment with 64 bit signed floating point numbers with 51 precision bits, the boundary condition is given by, $C < 10^{2^{11}}$.

As concluding remarks, the proposed network architecture is fast—since it is a single layer network and does not depend on iterative procedures for convergence. The computation complexity of the network is also reduced by a factor of S_N compared to ordinary approach as will be shown in section 6.8. Next in Chapter 6, we extend our discussion to experimental setup and results of the proposed algorithm for several class imbalanced datasets.

6 Experimental Results and Analysis

To conduct successful experiments we evaluate the performance of the classification algorithm with appropriate metrics and compare our results with other related works. Hence, we organize remaining of this chapter as follows: in section 6.1 we present a description of the datasets used for experimentation; in section 6.2 we describe the performance metrices used to evaluate the proposed algorithm; in section 6.3 we present experimental classification performance of the proposed algorithm on representative datasets; in section 6.4 we analyze receiver operating characteristics of the proposed algorithm; in section 6.5 the data modification stage is critically investigated; in section 6.6 the kernel optimization stage is critically investigated with experimental results; in section 6.7 the proposed method is compared with state of the art methods in terms of performance; and in section 6.8 computational time for training of the proposed method is analyzed.

6.1 Data

To compare the proposed method with other relevant works [25], [28], [34], [35] we used imbalanced datasets widely used throughout the community for evaluating classification algorithms. Following paragraphs present a short description of each of the datasets.

All datasets, except CTC [8], we used for testing and comparing the performance of the proposed methodology, were extracted from the UCI learning repository [39]. We used several datasets from this source. Table 6-1 presents a summary of these datasets. The CTC is a compiled collection of Computed Tomographic Colonography images from 8 different hospitals.

Table 6-1 presents information about 23 datasets used for the experimental procedure of the proposed method. The “#Attributes” denotes the number of features available in the source dataset. For some of the datasets (*i.e.* Wisconsin Breast Cancer), this number includes an ID of the sample

Table 6-1: Datasets Used for Evaluation of the Proposed Algorithm

Dataset	# Attributes	Ratio (P/N*100%)	# Samples
Wisconsin(Breast Canc.)	32	59.4	569
ISOLET B	617	4	7797
ISOLET D	617	4	7797
ISOLET A	617	4	7797
ISOLET E	617	4	7797
ISOLET I	617	4	7797
ISOLET O	617	4	7797
ISOLET U	617	4	7797
ISOLET M	617	4	7797
ISOLET S	617	4	7797
ISOLET R	617	4	7797
Gene Sequence(Splice Junc.)	61	33.3	3190
Segmentation (seg1)	19	16.67	210
Glass (g7)	10	15.7	214
Euthyroid (euth1)	24	13.5	2000
Landsat Satellite (sat)	36	9.73	6435
Abalone$_1$ (9 vs 18)	8	5.75	731
Oil (oil)	49	4.38	937
Car (car3)	6	4.16	1728
Yeast (yeast5)	8	3.56	1484
Mammography	6	2.32	11183
Abalone$_2$ (ab19)	8	0.772	4177
*CTC**	884736	10.1	3576

All datasets are from UCI repository [39] except
* CTC [8], which is Computed Tomographic Colonography images compiled from 8 hospitals

which is not used for the final classification process. Also, for some of the datasets the #Attribute includes the class label. The P/N ratio is defined as the ratio of number of positive samples and number of negative samples. Intentionally datasets that cover a diverse range of P/N ratio was selected to investigate the performance of the proposed algorithm at different levels of imbalance. One should consider that, table 6-1 provides information about the source dataset, not the training/test data—which are subset of the source datasets. In the next section the metrics for evaluating performance of the proposed algorithm are presented.

6.2 Performance Metrics

With highly skewed data distribution, the overall accuracy(=(TP+TN)/(P+N)) metric is not sufficient. Here, TP, TN, P and N stands for number of true(correctly detected) positives, true (correctly detected) negatives, positives (in the whole set) and negatives (in the whole set) respectively. For example, a naïve classifier that predicts all samples as negative has high accuracy—while practically, it is totally useless in detecting rare positive samples.

To deal with class imbalance, two kinds of metrics have been proposed: G-mean and F-measure. G-Mean is the geometric mean of two accuracies, namely sensitivity(=TP/P) and specificity(=TN/N). G-mean is defined as:

$$\text{G-mean} = \sqrt{{}^{TP}\!/_{P} * {}^{TN}\!/_{N}} = \sqrt{sensitivity * specificity} \qquad (32)$$

F-Measure, concentrates on high detection accuracy of one class while very moderately incorporating the detection accuracy of the other class and can be computed from precision(=TP/(TP+FP)) and recall(=sensitivity).

$$\text{F-Measure} = \frac{2 * Precision * Recall}{Precision + Recall} \qquad (33)$$

Another common metric is the area under the curve (AUC) of the receiver operating characteristics (ROC— sensitivity vs FP/(TP+FP) curve). AUC is more suitable for comparing several algorithms on same dataset. The choice of the metric is application dependent. For moderately to highly imbalanced classification problems, G-Mean provides good measure of effectiveness. Very highly imbalanced classification problems usually employ both G-Mean and Accuracy to investigate the trade-off between design constrains and application requirement. In this article, both will be used for analysis of the proposed algorithm. Also for comparison with relevant methods we use overall accuracy, G-Mean and computation time for training.

6.3 Classification Results

The proposed algorithm was developed and tested in MATLAB executed on a 64bit Microsoft windows OS. All the experimental results are obtained using 5-fold to 10-fold cross-validation.

Table 6-2: Accuracy Performance on Representative Datasets

Dataset	Accuracy	Sensitivity	G-mean	F-measure
Wisconsin	98.1(0.8)	95.8(2.3)	97.6(1.6)	97.4(1.7)
Isolet B	84.7(1.8)	86.3(9.2)	85.5(10.9)	30.3(12.1)
Isolet D	79.5(1.6)	74.7(9.1)	69.6(7.8)	61.7(8.1)
Isolet A	89.7(1.7)	71(11.5)	80.1(8.4)	34.6(9.6)
Isolet E	91.6(1.5)	85.7(5.8)	88.7(4.2)	43.9(4.9)
Isolet I	79.5(1.9)	43.3(8.3)	59.24.0)	14(4.3)
Isolet O	84(1.6)	49.3(9.4)	28.5(8.8)	44.3(9.5)
Isolet U	85(1.5)	96.7(8.7)	90.4(6.4)	33.1(7.2)
Isolet M	79.8(1.5)	44.7(5.3)	60.2(3.3)	14.5(3.8)
Isolet S	80.9(2.0)	52.7(6.2)	65.7(3.6)	17.5(5.0)
Isolet R	81.8(1.8)	52.3(4.0)	65.9(2.3)	18.1(2.7)
GeneSeq	87.4(1.2)	85.9(8.9)	86.9(6.1)	77.3(6.7)
Segmentation	97.6 (1.4)	80(6.4)	89.7(4.0)	90.6(4.6)
Glass	92.1 (3.2)	82.8(9.1)	88(6.1)	73.8(6.3)
Euthyroid	92.4(1.3)	89.5(8.2)	91.1(7.9)	73.7(10.6)
Satel.	90.1(1.9)	82.1(11.0)	86.4(7.6)	59.6(8.2)
Abalone$_1$	92.3 (2.6)	77.5(9.1)	85(6.4)	52.5(7.0)
Oil	91.9 (2.3)	76.9(5.9)	84.4(3.5)	44.1(3.8)
Car	98.4(0.9)	75.4(6.4)	86.5(4.6)	78.8(4.7)
Yeast	78.4(2.6)	74.5(2.0)	76.5(2.3)	19.1(2.5)
Mammography	94.6(1.5)	82.7(7.3)	88.6(5.8)	41(6.1)
Abalone$_2$	79.4(1.9)	75(6.7)	77.2(7.2)	25.3(7.5)
CTC	91.3(0.9)	86.4(3.1)	89.7(2.4)	86.9(2.9)
Mean	87.9	70.0	77.0	44.1

For cross validation we allowed up to 25% overlap to accommodate for highly imbalanced data and scarce positive samples. The experimental results for performance metrics are shown in Table 6-2.

Table 6-2 shows four accuracy related metrics *i.e.* Accuracy, Sensitivity (Recall), G-Mean and F-measure for all 23 datasets along with standard deviation of corresponding metric along the recorded runs for each dataset. The last row shows the mean for all 23 datasets. Notice the accuracy varies between 78% and 98%—a fairly acceptable range considering the highly imbalanced data distribution. The standard deviation portrays consistency of performance for the proposed algorithm for respective dataset. For any imbalanced data distribution, Sensitivity is one of the most important performance metric. Notice the high magnitude of parenthesized standard deviation values, which denote the performance varied highly within different runs. This signifies the high imbalance ratio in training set, as some of the runs did not contain sufficient positive class samples to correctly represent the minority class— a typical problem for highly class imbalanced data. As can be seen from table 6-2, proposed WNN-UID demonstrated acceptable sensitivity performance. Overall G-mean and F-Measure portrays more complete evaluation of the performance of any classification algorithm. From table 6-2 it is evident that for most of the datasets, the performance of WNN-UID is acceptable regarding the imbalance nature of the concerning datasets.

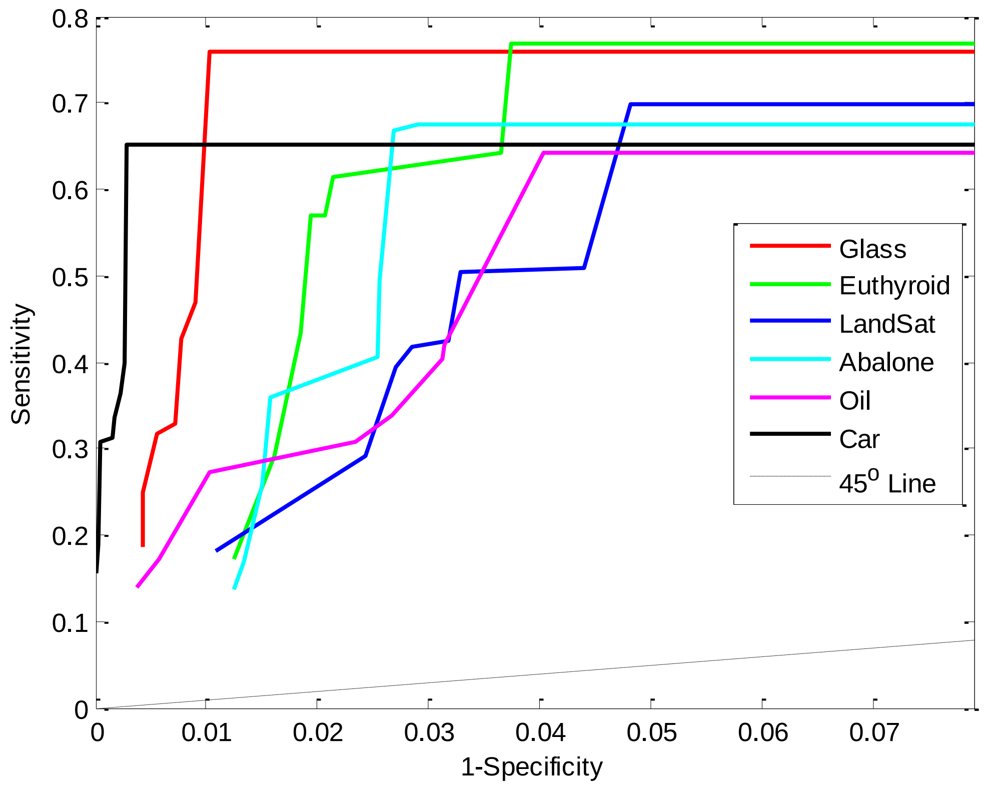

Figure 6-1: Receiver Operating Characteristics Curve for varying the Neural Network Parameter β, corresponding to six representative UCI Datasets.

6.4 ROC Analysis

Fig 6-1 shows ROC for selected databases corresponding to tuning the network bias, β, from section 5.2. For all datasets, the algorithm saturates after certain sensitivity is achieved and no more improvement in sensitivity can be achieved. This characteristics is attributed mostly to rarity of the positive class samples and limitation of ability of the machine learning method used to learn the pattern. This may be eliminated by using entirely different activation function or by drastically changing the number of hidden nodes or by changing the kernel function used. Six different colors represent the ROC curve for six different UCI datasets from table 6-1. Also the algorithm responds differently to different datasets—which indicates the usefulness of the algorithm for the corresponding dataset.

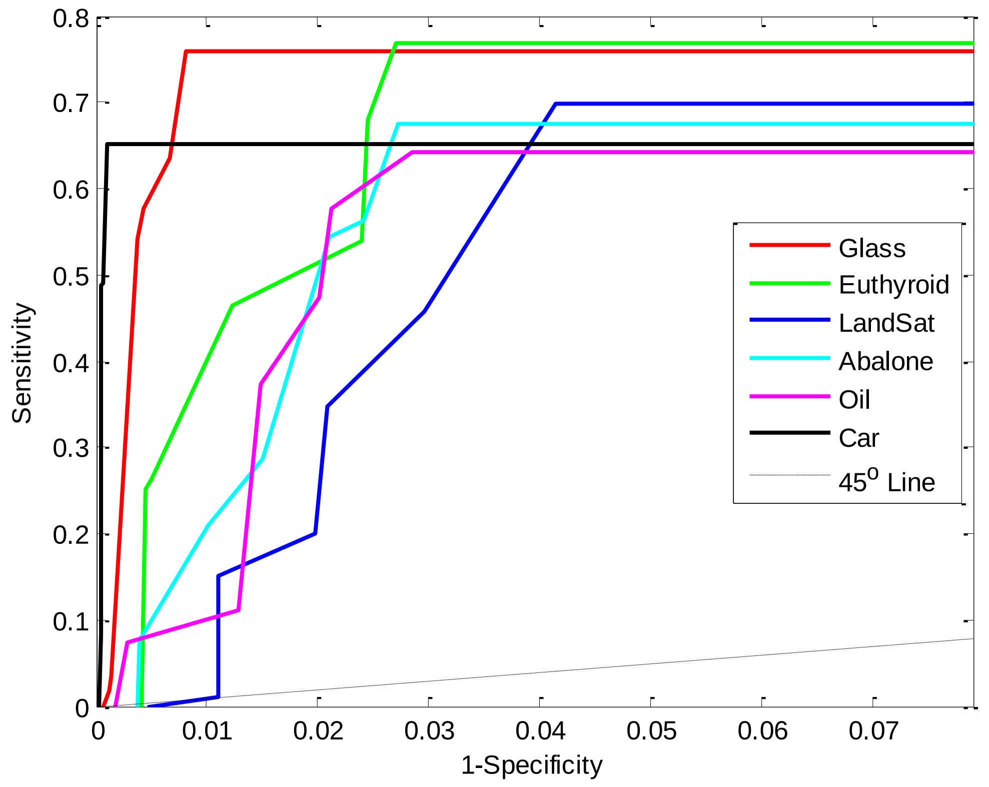

Figure 6-2: Receiver Operating Characteristics Curve generated by controlling the conformal transformation of the kernels, corresponding to six representative UCI Datasets.

Figure 6-2 shows ROC for selected databases corresponding to tuning the maximum number of boundary points for kernel transformation, from section 5.2. Similar to Figure 6-1 the ROC curve is shown for six UCI datasets. Without any kernel tuning a linear machine learning mechanism, *i.e.* a SLFN, exhibits very poor sensitivity. Also notice, the sensitivity rises very quickly with introduction of kernel optimization. Again, the impact of high imbalance ratio cannot be entirely eliminated with introduction of kernel optimization and the impact is training dataset dependent. With rare samples of the positive class, achieving a 100% sensitivity is practically not feasible. If the results of Figure 6-1 and Figure 6-2 are compared, it is easily perceptible that, the highest sensitivity for a pre-specified dataset is fixed—which imply the proposed algorithm to be robust.

6.5 Analysis of Data Modification

Proposed WNN-UID method consists of two step improvement, as mentioned in chapter 3, compared to traditional classification methods: dataset modification (chapter 4) and model optimization (chapter 5). In this section we present a comparative analysis of accuracy

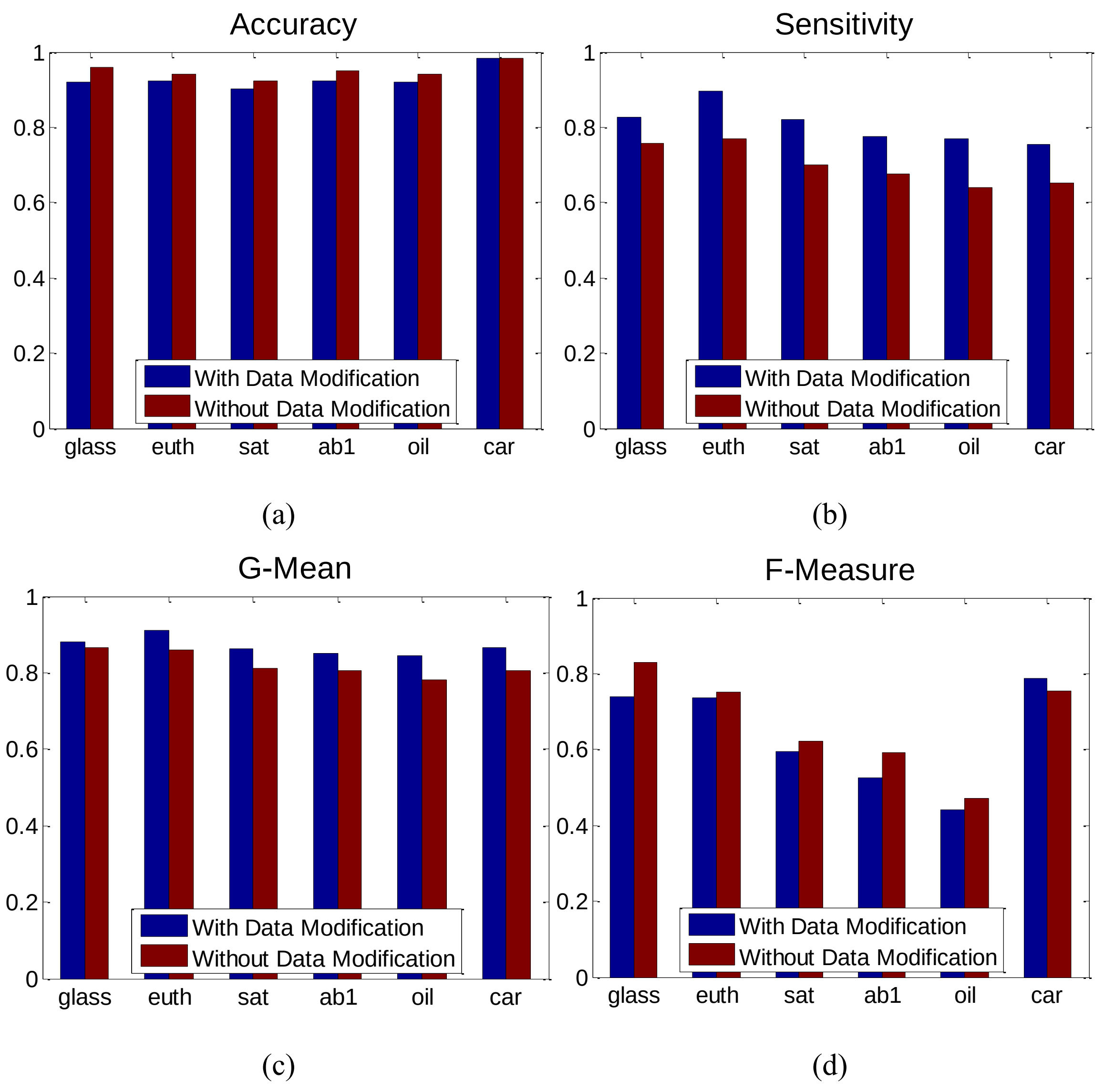

Figure 6-3: Comparison of effect of Data Modification on (a) Accuracy, (b) Sensitivity, (c) G-Mean and (d) F-Measure, for six representative datasets. The blue bars correspond to performance metric of the WNN-UID algorithm and the brown bars correspond to performance metric of the same WNN algorithm without any data modification. G-mean.

performance between the proposed WNN-UID approach and a modified version of the proposed WNN, without any dataset modification.

Figure 6-3 shows (a) Accuracy, (b) Sensitivity, (c) G-Mean and (d) F-Measure in presence of the data modification stage of the proposed WNN-UID and without the data modification stage for selected six datasets. Figure 6-3.(a) indicates that accuracy declines with addition of the data modification stage. But Figure 6-3.(b) indicates that sensitivity increases with addition of a data modification stage. For imbalanced class distribution, accuracy often portrays a very incomplete perception of the performance of the classifier. Our experiment on all 22 datasets revealed an average of about 2.5% increase in accuracy with removal of the data modification stage and an average of about 11% increase in sensitivity with addition of the data modification stage. Figure 6-3.(c) indicates that G-Mean increases with addition of the data modification stage. But Figure 6-3.(d) indicates that F-Measure decreases with addition of a data modification stage. For imbalanced class distribution, G-mean is often considered the most critical performance metric. Our experiment on all 22 datasets revealed an average of about 5% increase in accuracy with addition of the data modification stage and an average of about 1.5% increase in F-Measure with removal of the data modification stage. The results presented in Figure 6-3 demonstrate that the introduced data modification increase all four performance metrics and hence is worthy to be included in the learning process. Especially the increase in sensitivity imply a net effectiveness of the learning algorithm. The comparatively moderate increment of accuracy is a mere reflection of high imbalance ratio present in the datasets under consideration.

The data modification stage is a critical stage of the proposed WNN-UID. It leverages the performance of the classifier by reducing imbalance ratio even in presence of a WNN, and the advantage is clearly perceptible by the increase in sensitivity and G-Mean metrics.

Table 6-3: Comparison of Effect of Kernel Transformation on the Proposed WNN-UID.

Metric	Without Kernel Modification	With Kernel Modification
Accuracy	0.829	0.877
Sensitivity	0.563	0.693
G-Mean	0.691	0.764
F-Measure	0.374	0.421

6.6 ANALYSIS OF KERNEL OPTIMIZATION

Now we analyze the effectiveness of the kernel modification step of the proposed method. The purpose of introducing a kernel modification stage was to increase effectiveness of the classification algorithm. Note that, for highly imbalanced datasets, achieving a high accuracy is not the utmost priority. Rather achieving higher sensitivity and specificity is of higher priority. Which is why we chose to use the sensitivity, G-Mean and F-Measure metrics as evaluator of performance in the first place.

Table 6-3 Presents mean of all the performance metrics for all datasets. The two sets of results, above and below, correspond to implementation of the WNN-UID algorithm with kernel optimization step and without kernel optimization step respectively. Notice the change in each of the performance metrics with introduction of the kernel optimization step. The increase for accuracy is comparatively small—which is an expected result, as we mentioned earlier in this section. For sensitivity, G-mean and F-measure the increase in performance is significant. This justifies the necessity of the conformal transformation of kernel step, as discussed in section 5.1. The sensitivity increased by a fair amount of ~17% in average for the corresponding databases. Both G-mean and F-Measure increase by about 25% with introduction of the kernel optimization

step—which is a good achievement. In summary, it can be safely claimed that the kernel optimization step is an effective addition to the proposed WNN-UID algorithm.

6.7 Comparison among the relevant studies

In Table 6-4 we compare our results with those of SMOTE (with SVM) [25], GSVM-RU [28] and KBA [35]. As already listed in section 2.4, these are state of the art representative methods that address imbalance in training datasets.

Table 6-4 compares performance of the proposed WNN-UID with that of recent state of the art methods. WNN-UID achieves similar or better accuracy if compared to the SVM based SMOTE

Table 6-4: Comparison of Performance with other Methods for Representative UCI Datasets

Dataset	Performance Metric (relevant methods)	WNN-UID (Accuracy)	WNN-UID (G-Mean)
Seg.	98.1 [25], 98.1 [35]	97.6	89.7
Glass	91.8 [25] , 93.7 [35]	92.1	88
Euth.	92.4 [25] , 94.6 [35]	92.4	91.1
Satel.	89.9 [28]	90.1	86.4
Abalone$_1$	86.5 [28]	92.3	85
Oil	84.9 [28]	91.9	84.4
Car	99.0 [25] , 99.9 [35]	98.4	86.5
Yeast	69.9 [25] , 82.2 [35] , 87.8 [28]	78.4	76.5
Mamm.	89.0 [28]	94.6	88.6
Abalone$_2$	0 [25] , 57.8 [35] , 81.1 [28]	79.4	77.2

algorithm. The KBA algorithm demonstrates somewhat mixed—on some datasets better, and on some datasets worse—performance when the accuracies are compared with the proposed WNN-UID. But because of unavailability of any other metric than accuracy for the KBA algorithm the comparison is not very conclusive. On the other hand the GSVM-RU method demonstrates very similar performance in terms of G-Means as the proposed WNN-UID. Since G-Means is considered as a very reliable performance metric for classification of class imbalanced datasets, WNN-UID can be considered a well alternate of GSVM-RU. Since GSVM-RU is known to be a highly iterative process with slower convergence time, WNN-UID can be considered as a preferred alternative.

6.8 Computational Time

The kernel boundary adaptation employed is an operation with complexity in $O(T'_{max}N^2)$ where N and T'_{max} are number of samples and number of iterations respectively. It inherently contains $T'_{max}+1$ SVM training operations. Employing the segmented SVM, we reduce the N by a factor s_N and the new complexity becomes: $O(T'_{max}N^2/s_N)$. The SMOTE increases the number training samples by a factor of $n_i>1$ and hence the SVM training complexity becomes: $O(T'_{max}N^2\ n_i)$ and the total complexity is given by: $O(T'_{max}N^2\ n_i) + O(N^2\ n_i)$. The latter corresponds to the oversampling technique. The KBA however is an iterative process that requires long time to converge in absence of undersampling methods. The complexity is given by $O(T'_{maxKBA}N^2)$, where T'_{maxKBA} is number of iterations for the KBA to converge (or the preset to stop iteration). The complexity of the GSVM-RU can be simplified to as summation of $O(T'_{max}N^2/s_i)$, where s_i denotes the reduction factor for majority samples in i^{th} iteration.

Table 6-5 shows the comparison of computation time for training of the proposed algorithm on UCI datasets with other methods from [25] and [35]. The proposed algorithm and the other comparative algorithms were tested on MATLAB executing on a 64bit Microsoft windows OS. The MATLAB 2013a ran on a system with 4th generation Intel i7 Haswell processor with 3.4GHz clock speed and 16GB DDR3 memory. The time required for training on different datasets vary with sample size, number of attributes and the complexity of the kernel boundary.

Table 6-5, compares time (in seconds) required for training the corresponding model for SMOTE with SVM, KBA and WNN-UID for representative UIC datasets. The time required for training on different datasets vary with sample size, number of attributes and the complexity of the kernel boundary. As can be seen from the table, WNN-UID requires slightly more time, about 25% on average, to train if compared to SMOTE. This is an expected outcome since, SMOTE comprises of data modification only and do not include any kernel adjustment. Also, for most datasets WNN-UID achieves higher accuracy than SMOTE. WNN-UID requires much less time for training when

Table 6-5: Comparison of Computation Time (second per training sample) on representative UCI Datasets

Dataset	SMOTE [25]	KBA [35]	WNN-UID
Seg.	1.0	4.3	1.4
Glass	1.1	4.3	1.5
Euth.	1.7	17.1	1.9
Car	1.3	6.4	1.8
Yeast	1.1	5.7	1.6
Abalone$_2$	1.5	11.9	1.7
Mean	1.28	8.28	1.68

compared with the KBA. On representative datasets, as in table 6-5, WNN-UID shows 3-fold to 8-fold improvement in training computation time compared to KBA. Even though the accuracy of both methods are very similar, WNN-UID requiring much less time, on average about 20% of the time required by KBA, makes it a more preferable method for most applications.

7 CONCLUSION

A new algorithm for under-sampling class imbalanced data and fast training weighted neural network was developed. The proposed WNN-UID takes advantage of, data modification (chapter 4) for imbalanced class distribution, as well as, model optimization (chapter 5) by kernel modification. First, WNN-UID extracted the data modified from the available sample set through an iterative process of selective under-sampling. Then, the initial kernel function is incrementally optimized to specifically enhance class boundaries for imbalanced datasets by conformally transforming the kernel functions. WNN-UID demonstrated promising results of accuracy, close to 90% and above as well as 80% and above for G-mean. The analysis of ROC curves and comparison tables demonstrated that the data modification and model optimization both contributed to increase the sensitivity from 15% to 25%, and from 15% to 35%, respectively. The proposed algorithm showed 3-fold to 8-fold improvement in time required for training computation when compared to some of the state of the art methods. We plan to extend this algorithm for distributed datasets, medical datasets and cloud environment.

.

References

[1] N. V. Chawla, “Data Mining for Imbalanced Datasets: An Overview,” in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds. Springer US, 2005, pp. 853–867.

[2] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly Detection: A Survey,” *ACM Comput Surv*, vol. 41, no. 3, p. 15:1–15:58, Jul. 2009.

[3] B. R. Raghunath and S. N. Mahadeo, “Network Intrusion Detection System (NIDS),” in *First International Conference on Emerging Trends in Engineering and Technology, 2008. ICETET ’08*, 2008, pp. 1272–1277.

[4] M. Behdad, L. Barone, M. Bennamoun, and T. French, “Nature-Inspired Techniques in the Context of Fraud Detection,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 42, no. 6, pp. 1273–1290, Nov. 2012.

[5] M. Zieba, “Service-Oriented Medical System for Supporting Decisions With Missing and Imbalanced Data,” *IEEE J. Biomed. Health Inform.*, vol. 18, no. 5, pp. 1533–1540, Sep. 2014.

[6] S. Garcia and F. Herrera, “Evolutionary Training Set Selection to Optimize C4.5 in Imbalanced Problems,” in *Eighth International Conference on Hybrid Intelligent Systems, 2008. HIS ’08*, 2008, pp. 567–572.

[7] H. He and E. A. Garcia, “Learning from Imbalanced Data,” *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[8] Y. Motai and H. Yoshida, "Principal Composite Kernel Feature Analysis: Data-Dependent Kernel Approach," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 8, pp. 1863–1875, Aug. 2013.

[9] L. Rutkowski, M. Jaworski, L. Pietruczuk, and P. Duda, "Decision Trees for Mining Data Streams Based on the Gaussian Approximation," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 108–119, Jan. 2014.

[10] J. B. Gomes, M. M. Gaber, P. A. C. Sousa, and E. Menasalvas, "Mining Recurring Concepts in a Dynamic Feature Space," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 95–110, Jan. 2014.

[11] C.-H. Yun and M.-S. Chen, "Mining Mobile Sequential Patterns in a Mobile Commerce Environment," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 37, no. 2, pp. 278–295, Mar. 2007.

[12] T. Minegishi and A. Niimi, "Detection of fraud use of credit card by extended VFDT," in *2011 World Congress on Internet Security (WorldCIS)*, 2011, pp. 152–159.

[13] W. Song, J. Jia, and Y. Peng, "Error Detection by Redundant Transaction in Transactional Memory System," in *2011 6th IEEE International Conference on Networking, Architecture and Storage (NAS)*, 2011, pp. 220–224.

[14] S. Kotsiantis, D. Kanellopoulos, P. Pintelas, and others, "Handling imbalanced datasets: A review," *GESTS Int. Trans. Comput. Sci. Eng.*, vol. 30, no. 1, pp. 25–36, 2006.

[15] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Editorial: Special Issue on Learning from Imbalanced Data Sets," *SIGKDD Explor Newsl*, vol. 6, no. 1, pp. 1–6, Jun. 2004.

[16] Z.-H. Zhou and X.-Y. Liu, “Training cost-sensitive neural networks with methods addressing the class imbalance problem,” *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 1, pp. 63–77, Jan. 2006.

[17] C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, and A. Napolitano, “RUSBoost: A Hybrid Approach to Alleviating Class Imbalance,” *IEEE Trans. Syst. Man Cybern. - Part Syst. Hum.*, vol. 40, no. 1, pp. 185–197, Jan. 2010.

[18] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, “A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 42, no. 4, pp. 463–484, Jul. 2012.

[19] F. Provost, “Machine learning from imbalanced data sets 101,” in *Proceedings of the AAAI’2000 workshop on imbalanced data sets*, 2000, pp. 1–3.

[20] Y. SUN, A. K. C. WONG, and M. S. KAMEL, “CLASSIFICATION OF IMBALANCED DATA: A REVIEW,” *Int. J. Pattern Recognit. Artif. Intell.*, vol. 23, no. 4, pp. 687–719, 2009.

[21] H. Cao, X.-L. Li, D. Y.-K. Woon, and S.-K. Ng, “Integrated Oversampling for Imbalanced Time Series Classification,” *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 12, pp. 2809–2822, Dec. 2013.

[22] D. L. Donoho and J. Tanner, “Precise Undersampling Theorems,” *Proc. IEEE*, vol. 98, no. 6, pp. 913–924, Jun. 2010.

[23] H. Hj Mohamed, T. L. Kheng, C. Collin, and O. S. Lee, “E-Clean: A Data Cleaning Framework for Patient Data,” in *2011 First International Conference on Informatics and Computational Intelligence (ICI)*, 2011, pp. 63–68.

[24] H. He, *Self-adaptive systems for machine intelligence*. John Wiley & Sons, 2011.

[25] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” *J Artif Int Res*, vol. 16, no. 1, pp. 321–357, Jun. 2002.

[26] J. Luengo, A. Fernández, S. García, and F. Herrera, “Addressing data complexity for imbalanced data sets: analysis of SMOTE-based oversampling and evolutionary undersampling,” *Soft Comput.*, vol. 15, no. 10, pp. 1909–1936, 2011.

[27] R. Akbani, S. Kwek, and N. Japkowicz, “Applying support vector machines to imbalanced datasets,” in *Machine Learning: ECML 2004*, Springer, 2004, pp. 39–50.

[28] Y. Tang, Y.-Q. Zhang, N. V. Chawla, and S. Krasser, “SVMs Modeling for Highly Imbalanced Classification,” *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 39, no. 1, pp. 281–288, Feb. 2009.

[29] S. Bergman, *The kernel function and conformal mapping*, vol. 5. American Mathematical Soc., 1970.

[30] J. Mercer, “Functions of Positive and Negative Type, and their Connection with the Theory of Integral Equations,” *Philos. Trans. R. Soc. Lond. Ser. Contain. Pap. Math. Phys. Character*, vol. 209, pp. 415–446, Jan. 1909.

[31] Y. Motai, “Kernel Association for Classification and Prediction: A Survey,” 2014.

[32] R.-J. Yang, “Conformal transformation in f (T) theories,” *EPL Europhys. Lett.*, vol. 93, no. 6, p. 60001, 2011.

[33] J. L. Barbosa, M. Do Carmo, and J. Eschenburg, *Stability of hypersurfaces of constant mean curvature in Riemannian manifolds*. Springer, 2012.

[34] X. Yang, Q. Song, and A. Cao, “Weighted support vector machine for data classification,” in *2005 IEEE International Joint Conference on Neural Networks, 2005. IJCNN ’05. Proceedings*, 2005, vol. 2, pp. 859–864 vol. 2.

[35] G. Wu and E. Y. Chang, “KBA: kernel boundary alignment considering imbalanced data distribution,” *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 786–795, Jun. 2005.

[36] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: Theory and applications,” *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, Dec. 2006.

[37] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: a new learning scheme of feedforward neural networks,” in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, 2004, vol. 2, pp. 985–990.

[38] W. Zong, G.-B. Huang, and Y. Chen, “Weighted extreme learning machine for imbalance learning,” *Neurocomputing*, vol. 101, pp. 229–242, Feb. 2013.

[39] M. Lichman, “UCI Machine Learning Repository.” University of California, Irvine, School of Information and Computer Sciences, 2013.