High Speed Algorithms for Outlier Detection and Classification over Huge-size Network Data Streams

Knowledge Engineering & Discovery Research Institute (KEDRI), Auckland University of Technology, New Zealand.

(FY2008)
ANNUAL ACHIEVEMENT REPORT

Date: March 31, 2009

To: Hiroshi IKEGAWA, Vice President, Member of the Board of Directors
   National Institute of Information and Communications Technology (NICT)

From: Dr. Paul S. Pang
   Senior Research Fellow, Director of Center for Adaptive Pattern Recognition Systems,
   Knowledge Engineering & Discovery Research Institute, Auckland University of Technology, New Zealand

The above party wishes to inform you that the Commissioned Research titled “High Speed Algorithms for Outlier Detection and Classification over Huge-size Network Data Streams” under the Commissioned Research Agreement dated August 5, 2008 has been completed and hereby submits this report pursuant to Article 9 and 23 of the agreement.

1. Summary of the Commissioned Research

   (1) Title of the research and development
       Research and Development of Existing Methods for Outlier Detection and Classification

   (2) Commencement date and completion date of the Commissioned Research
       Commencement date: Aug. 5, 2008
       Completion date: March 31, 2009

   (3) Names of researchers engaged in the Commissioned Research and sections they belong to
       Dr. Paul S. Pang, Center for Adaptive Pattern Recognition Systems, KEDRI
       Prof. Nik Kasabov, KEDRI
       Gary Chen, Center for Adaptive Pattern Recognition Systems, KEDRI
       Kshitij Dhoble, Center for Adaptive Pattern Recognition Systems, KEDRI
       Zbynek Michlovsky, Center for Adaptive Pattern Recognition Systems, KEDRI
2. Achievement of the Commissioned Research
   As the Accomplishment Report in Paper 1 shows.

   (1) Experimental environment setup

   A. Software Computing Environmental setup
   Metlab ver 7.7 R2008b was installed in KEDRI as algorithm experimenting and
   system prototyping environment.

   GCC 4.3.3 was installed on Windows XP/Vista working as a platform from
   prototype to software system implementation.

   GCC is convenient for us to implement the developed intrusion detection system
   in a Linux based server system to satisfy the speed requirement of real time
   intrusion detection. Additionally, GCC is GPL (general public license) software.
   It includes compilers for C, C++, Objective-C, Fortran, Java, and Ada, as well as
   libraries for these languages, and supports both Windows and Linux operation
   system. Furthermore, there are also a lot of GCC supportable toolboxes available,
   such as valgrind (http://en.wikipedia.org/wiki/Valgrind), a toolbox on memory
   debugging, and the OpenCV for pattern recognition and computer vision. This
   will speed up the procedure of transforming from a Matlab algorithm prototype
   to network application software system.

   B. Project Database

   ❖ Remote maintained internet security datasets: (Server: 221.186.68.81)
     Ida Two-class Benchmarking
     KDD2008
     Malware Traces
     Object Classification
     Simulated Zeroday Attack Dataset
     WebSpamDetection

   ❖ KEDRI maintained datasets:
     UCI Datasets
     KDDCUP99 dataset
     Enron-spam dataset
     Ling-spam dataset
     PU1 and PU123 dataset

   (2) Network Traffic Analyzer based on String Kernel
For network intrusion and virus detection, ordinary methods detect malicious network traffic and viruses by examining packets, flow logs or content of memory for any signatures of the attack. This implies that it will be problematical to detect attacks until their signatures are created. Addressing unknown attack detection, we developed in this paper a novel network traffic and memory analyzer, by a new string kernel based SVM (support vector machine) supervised machine learning. The proposed method detects a network attack without earlier determined attack signatures. When testing the correctness of our method implementation and SVM optimal configuration, we used string data from Reuters datasets and compared the experimental results with previous work. As the next step work, we will apply our approach to other network security problems such us testing memory dump and network traffic data.

**Research Output:**


(3) Active learning Method

Datasets acquired in real practices are often vast, inaccurate, inconsistent and unlabelled. It becomes laborious, costly and time consuming process to label the data for reasons such as classification or knowledge discovery, which the traditional machine learning algorithms are incapable of achieving. This is where 'active learning' comes in play. 'Active learning' is a class of learning algorithms which aim to create an accurate classifier by iteratively selecting essentially important unlabelled data points by means of adaptive querying and training the classifiers on those data points that are potentially useful for the targeted learning task. Nonparametric Discriminant Analysis (NDA) is discrimination based selective sampling technique which reduces the effects of outliers and works well on anomalous datasets thereby minimizing the computational cost. The NDA is one of the methods used in the proposed active learning model. This proposal presents a research on a discrimination-based active learning in which Nonparametric Discriminant Analysis (NDA) is extended for fast discrimination analysis and data sampling. A base classifier Support Vector Machine (SVM) is applied to discover and merge the knowledge from the newly acquired data. Performance tests are carried out on benchmark datasets from UCI.
Research Output:


(4) Multi-label classification methods

As network intrusions are usually associated with multiple labels in a hierarchy structure, the classification of network intrusions naturally possess a hierarchical multi-label classification (HMC) problem, in which every instance may belong to more than one class. Addressing HMC problem, in this paper we propose a novel approach of hierarchy minimum enclosing balls (HMEB), where a label hierarchy is modeled as data HyperSphere hierarchies, which are a set of data composed into smallest enclosing hypersphere, called minimum enclosing ball (MEB). The MEBs are separating, encompassing and overlap with each other in an organized tree-like structure. Given an unlabeled sample, HMEB seeks a MEB that encloses the sample, and labels the sample according to the MEB's position in the hierarchy. The proposed method has been validated on the hierarchy multi-label problem from KDD'99 and Yeast dataset, respectively. The HMEB improves the classification accuracy of U2R from 13.2% to 82.7% and R2L from 8.4% to 45.9%, as compared to the winner of KDD'99. Also, the efficiency of HMEB is demonstrated, as the computational time stays steady while the size of training data exponentially manifolds in another experiment of Yeast data.

Research Output:


(5) Divide-and-conquer method study

Addressing the capability of a given dataset that could be better modeled by a modular method for classification, this paper proposed a new type of SVM aggregating method, called gridding SVM Classification Tree (gSVMT). The proposed gSVMT achieves to decompose a problem without loss of the original discriminant information, discover
subregions contained SVM knowledge in the data through a recursive SVM-supervised data part, and for each obtained subregion, an individual SVM is allocated for approximating the subregion knowledge, and a group of such SVMs are aggregated along the grid in an order that they are explored. Experiments with a Gaussian synthetic data, 13 benchmark machine learning datasets, have highlighted the utility of gSVMs on its very competitive classification capability. In particular, gSVMs is demonstrated absolutely better than single SVM for those datasets with high sparseness and/or class-imbalance. This has been further demonstrated with its very successful real application on face membership authentication.

Research Output:

Shaoning Pang, and Nik Kasabov, “Spanning SVM Classification Tree along the Grid of Data Sparseness”, KEDRI-NICT project report, March 2009. Page 49-71

Shaoning Pang, Tao Ban, Youki Kadobayashi, and Nik Kasabov, gSVMT: Aggregating SVMs over a Dynamic Grid Learned from Data, Proceedings of International Workshop on Data Mining and Artificial Intelligence (DMAI’08), pp. 72-79, 24 December, 2008, Khulna, Bangladesh. (Appendix: B)

Best Paper Award, IEEE international Workshop on Data Mining and Artificial Intelligence, DMAI2008, in conjunction with IEEE international Conference on Computer and Information Technology (Appendix: A).

An extended version of the above conference paper has been submitted to Information System, Elsevier.

(6) ICONIP2008 Special issue

This thematic issue of “Adaptive Soft Computing Techniques and Applications” It is noticeable that new challenge occurs recently upon the explosive growth of mobile and network communications. The adaptive soft computing is confronting a number of new difficult tasks, such as spam filtering, internet intrusion detection, and malicious software attack prevention, etc. To deal with these problems, a successful method should pay special attention to the following aspects: huge amount of information presented as data streams, real time system response demanded by applications, dynamic property of the data sources, severe sampling bias in the training data, and inequality of misclassification costs.

- Incremental learning; Multi-task learning; Lifelong learning
- Evolving connectionist system learning in dynamic environments
- Applications to real time intelligent system
• Advanced applications to detect cyber terrorist attacks like intrusion, phishing sites, spam emails, etc.
• Other applications that call for incremental learning or learning from very large database.
• Issues relevant to above mentioned or related fields

**Research Output:**

The proposal of special issue has been accepted by Journal of Memetic Computing, Springer, Please refer to the attached CFP in Appendix: C, and Springer website at, http://www.springer.com/engineering/journal/12293

3. Statement of expenses
   The Cost Report is attached.

4. Acquired assets
   The ledger of the Acquired Assets is attached.
Network Traffic and Memory Analyzer based on String Kernels

Zbynek Michlovsky, Tao Ban, Shaoning Pang, and Nik Kasabov

Abstract

For network intrusion and virus detection, ordinary methods detect malicious network traffic and viruses by examining packets, flow logs or content of memory for any signatures of the attack. This implies that it will be problematical to detect attacks until their signatures are created. Addressing unknown attack detection, we developed in this paper a novel network traffic and memory analyzer, by a string kernel based SVM (support vector machine) supervised machine learning. The proposed method detects a network attack without earlier determined attack signatures. When testing the correctness of our method implementation and SVM optimal configuration, we used string data from Reuters datasets and compared the experimental results with work in [4]. As the next step work, we will apply our approach to other network security problems such us testing memory dump and network traffic data..

Index Terms

String Kernel, Support Vector Machine, Intrusion Detection System,

I. INTRODUCTION

Upon computers and Internet being more and more integrated into our common life, higher security requirements have been imposed on our computer and network system. One of many ways that we can take for increasing the security is to use the Intrusion Detection System (IDS). Intrusion detection (system) is a process of monitoring events occurring in a computer system or network and analyzing them on signs of possible incidents.
The whole problematic definition of IDS is described in [15]. In general, IDS can be grouped into two categories: statistical anomaly based IDS and signature based IDS. The idea of statistical anomaly IDS is to detect intrusions by comparing traffic with normal traffic model, looking for deviations. Due to the diversity of network traffic, it is difficult to model normal traffic, as we know that a normal email relaying or peer-to-peer queries may also show like with some intrusion traffic characteristics. Moreover, even for abnormal traffic, it does not in fact constitute an intrusion/attack. Hence, anomaly detection often has a high false alarm rate, thus is seldom used in practice. For signature based IDS, network traffic is examined for predetermined attack patterns known as signatures. A signature consists of a string of characters (or bytes). Nowadays many intrusion detection systems also support regular expressions and even behavioral fingerprints [18]. The difficulty of signature based IDS system is that only intrusions whose signatures are known can be detected and it is necessary to constantly update a collection of these signatures to mitigate emerging threats [19].

Most of attack signatures are created by security experts who analyze network traffic and host logs after intrusions have occurred, whereas sifting through thousands lines of log files and looking for characteristics that uniquely identify as an intrusion is a vast and error prone undertaking. To overcome this shortcoming and detect unknown attack (i.e. signature is not determined), we researched machine learning on string content recognition techniques. The motivation is to train a classifier to distinguish between malicious and harmless flows or memory dump and utilize the trained classifier to classify real network flow and memory dump.

As support vector machine (SVM) is one of the most successful classification algorithms in the data mining area, we used SVM in this work for string data machine learning, thus different string kernelled SVM are developed. In spite of the limitation of the SVM on training efficiency, the advantage of SVM used in IDS is in needless of knowledge about attacks signatures, as they are learned automatically by the SVM during the training procedure.

In our experiments, we used SVM machine - libSVM [1] and we have added four implementations of string kernels algorithms: n-gram kernel, substring kernel (gap-weighted subsequence kernel), edit distance kernel and bag of words kernel, respectively. We incorporate our kernel functions implementations with libSVM [1] in two ways, in first case we added a library with functions to libsvm-string which is special implementation for string input data [14], and in second case we precomputed our kernel matrix and then we add it into the original libsvm
implementation. The goal of the work is to compare performance each string kernel in network traffic and memory dump classification and examine suitability of this approach in real operations. We have put the importance to find best parameters of SVM and kernels function with following improving of efficiency string kernels algorithms.

This paper is organized as follows. Section 2 presents kernels functions which we used with theoretical background, Section 3 discusses our implementation and testing of our system, Section 4 describes experiments and results. And finally the conclusion and exploration of future work is described in Section 5.

II. SVM AND KERNELS THEORY

Support vector machines (SVM) are groups of supervised learning methods that can be applied to classification or regression. SVM maps the data points into a high dimensional feature space, where a linear learning machine is used to find a maximal margin separation [12], [13]. One of the main statistical properties of the maximal margin solution is that its performance does not depend on the dimensionality of the space where the separation takes place. In this way, it is possible to work in very high dimensional spaces, such as those induced by kernels, without over fitting.

Kernels provide support vector machines with the capability of implicitly mapping non-linearly separable data points into a different dimension, where they are linearly separable. This method is also called Kernel trick [12].

Kernel function $K(x, y)$ can be expressed as a dot product in a high dimensional space. If the arguments to the kernel are in a measurable space $X$, and if the kernel is positive semi-definite for any finite subset $\{x_1, ..., x_n\}$ of $X$ and subset $\{c_1, ..., c_n\}$ of objects

$$\sum_{i,j} K(x_i, x_j)c_ic_j \geq 0$$

then there exist a function $\phi(x)$ whose range is in an inner product space of possibly high dimension, such that

$$K(x, y) = \phi(x)\phi(y).$$

The kernel method allows for a linear algorithm to be transformed into a non-linear algorithm. This non-linear algorithm is equivalent to the linear algorithm operating in the range space of $\phi$. However, because kernels are used, the $\phi$ function is never explicitly computed. The kernel
representation of data amounts to a nonlinear projection of data into a high-dimensional space where it is easier to separate into classes [17]. Most popular kernels suitable for SVM are e.g. Polynomial Kernel, Gaussian Radial Basis Kernel, Hyperbolic Tangent Kernel [16]. All of these kernels operate with numerical data. For our purpose is necessary to use string kernels which are described in following section.

III. STRING KERNELS USED IN SVM

Regular kernels for SVM work merely on numerical data, which is unsuitable for internet security where huge amount of string data is presented. Towards extending SVM for string data processing, we implemented the following string kernels algorithms in our experiments.

A. Gap-Weighted Subsequence Kernel

The theory about this kernel is described in the book Kernel Methods for Pattern Analysis [2]. The main idea behind the gap-weighted subsequence kernel is to compare strings by means of the subsequences they contain - the more subsequences and less gaps they contain the more similar they are. For reducing dimensionality of the feature space we consider non-contiguous substrings that have fixed length \( p \). The feature space of gap-weighted subsequence kernel is defined as

\[
\phi^p_u(s) = \sum_{i: u = s(i)} \lambda^{l(i)}, u \in \Sigma^p,
\]

where \( \lambda \in (0, 1) \) is decay factor, \( i \) is index the occurrence of subsequence \( u = s(i) \) in string \( s \) and \( l(i) \) is length of the string in \( s \). We weight the occurrence of \( u \) with the exponentially decaying factor \( \lambda^{l(i)} \). The associated kernel is defined as

\[
\kappa(s, t) = \langle \phi^p(s), \phi^p(t) \rangle = \sum_{u \in \Sigma^p} \phi^p_u(s)\phi^p_u(t).
\]

We consider computing an intermediate dynamic programing table \( DP_p \) whose entries are:

\[
DP_p(k, l) = \sum_{i=1}^{k} \sum_{j=1}^{l} \lambda^{k-i+l-j} \kappa_{p-1}^S(s(1:i), t(1:j))
\]

than we can evaluate the kernel by

\[
\kappa_p^S(sa, tb) = \begin{cases} 
\lambda^2 DP_p(|s|, |t|) & \text{if } a = b; \\
0 & \text{otherwise}
\end{cases}
\]
The complexity of computation requires to compute the table $DP_p$ for single value of $p$ is clearly $O(|s| |t|)$ as is the complexity of computing $\kappa_p^S$ from $DP_p$ making overall complexity of computing the kernel $\kappa_p(s, t)$ equal to $O(p |s| |t|)$.

---

**Algorithm 1** Gap-weighted subsequence kernel pseudocode [1]

**Require:** Input strings $s$ and $t$ of lengths $n$ and $m$, $p$ is the length of subsequence and $\lambda$ is weight parameter.

**Ensure:** Kernel evaluation $\kappa(s, t) = \text{Kern}(p)$.

```plaintext
1: DPS(1 : n, 1 : m) = 0;
2: for $i = 1 : n$ do
3:     for $j = 1 : m$ do
4:         if $s_i = t_j$ then
5:             DPS($i, j$) = $\lambda^2$;
6:         end if
7:     end for
8: end for
9: DP(0, 0 : m) = 0;
10: DP(1 : n, 0) = 0;
11: for $l = 2 : p$ do
12:     Kern($l$) = 0;
13:     for $i = 1 : n - 1$ do
14:         for $j = 1 : m - 1$ do
15:             $DP(i, j) = \text{DPS}(i, j) + \lambda DP(i - 1, j) + \lambda DP(i, j - 1) - \lambda^2 DP(i - j, j - 1)$;
16:             if $s_i = t_j$ then
17:                 $\text{DPS}(i, j) = \lambda^2 DP(i - 1, j - 1)$;
18:             end if
19:         end for
20:     end for
21: end for
22: end for
```
B. Levenshtein distance

Levenshtein (or edit) distance [8] counts differences between two strings. The distance is the number of substitutions, deletions or insertions required to transform string \( s \) with length \( n \) to string \( t \) with length \( m \).

The time complexity of the computation is \( O(|s| |t|) \). If the strings \( s \) and \( t \) have the same length, the complexity equals to \( O(n^2) \).

C. Bag of words kernel

In Bag of words kernel is text represented as an unordered collection of words, disregarding grammar and word order. Words are any sequences of letters from the basic alphabet separated by punctuation or spaces. We can represent a bag as a vector in a space in which each dimension is associated with one term from the dictionary

\[
\phi : d \mapsto \phi(d) = (tf(t_1, d), tf(t_2, d), ..., tf(t_N, d)) \in \mathbb{R}^N,
\]

where \( tf(t_i, d) \) is the frequency of the term \( t_i \) in the document \( d \). Hence, a document is mapped into a space of dimensionality \( N \) being the size of the dictionary, typically a very large number [2].

D. N-gram kernel

N-grams transform documents into high dimensional feature vectors where each feature corresponds to a contiguous substring [4]. The feature space associated with n-gram kernel is defined as

\[
\kappa(s, t) = (\phi^n(s), \phi^n(t)) = \sum_{u \in \Sigma^n} \phi^n_u(s) \phi^n_u(t)
\]

where

\[
\phi^n_u(s) = |\{(v_1, v_2) : s = v_1 uv_2\}|, \quad u \in \Sigma^n.
\]

We have used for computing n-gram kernel naive approach therefore the time complexity is \( O(n |s| |t|) \).
Algorithm 2 Pseudocode of Levenshtein distance.

Require: Input strings $s$ and $t$ of lengths $n$ and $m$.

Ensure: Edit distance of the strings $s$ and $t$. \{Two dimensional m $M$ is used to hold the edit distance values.\}

\begin{align*}
1: & \text{if } n = 0 \text{ then} \\
   & \quad \text{return } m \\
2: & \text{end if} \\
3: & \text{if } m = 0 \text{ then} \\
   & \quad \text{return } n \\
4: & \text{end if} \\
5: & \text{for } i = 0 : n + 1 \text{ do} \\
   & \quad M[i][0] = i \\
7: & \text{end for} \\
8: & \text{for } j = 0 : j = m + 1 \text{ do} \\
9: & \quad M[0][j] = j \\
10: & \text{end for} \\
11: & \text{for } i = 1 : n + 1 \text{ do} \\
12: & \quad \text{for } j = 1 : m + 1 \text{ do} \\
13: & \quad \text{if } s[i - 1] = t[j - 1] \text{ then} \\
14: & \quad \quad \text{cost}=0; \\
15: & \quad \text{else} \\
16: & \quad \quad \text{cost}=1; \\
17: & \quad \text{end if} \\
18: & \quad M[i, j] = \min(M[i - 1, j] + 1, M[i, j - 1] + 1, M[i - 1, j - 1] + \text{cost}); \\
19: & \quad \text{end for} \\
20: & \text{return } M[n][m];
\end{align*}
IV. Experiments and Discussions

A. Datasets and Experimental Setup

We have used the labeled and reformatted string data from Reuters-21578 dataset for testing our kernel functions. Only relevant information from tags topic, title, body, text was extracted, some unknown tags i.e. & $ was removed and finally all characters have been changed to lowercases. Each string has been normalized to the length of longest string by repeating its content. All text lines have been added class labels corresponding to the label file of Reuters dataset.

New dataset: the same as a previous: all strings which has been in more than

In testing our kernel functions, we refer to similar works presented in [4]. We incorporate our kernel functions implementations (KFI) with libSVM [1] in two ways, in first case we added a library with KFI to libsvm-string which is special implementation for string input data [14]. In second case, we pre-computed our kernel matrix and then inserted it into the original libsvm implementation. We set the training sample from the Reuters datasets to 380 (90) documents for training (testing) with topic earn. Using default settings of libsvm we had accuracy about 75% what looks incomparable with results in the [4]. To improve our results was necessary to investigate the optimal configuration of SVM. For that reason the 5-fold validation grid has been set to search two parameters - substring length and C (cost). For 5-fold cross validation we used 368 documents from training and 92 documents from testing Reuters datasets. Each split of the data had the following sizes and numbers of positive examples in training and test sets: numbers of positive examples in training (testing) set out of 368 (92): earn 152 (38); acquisition 108 (27); crude 72 (18); corn 36 (9). The searching is in progress now.

B. Results

For each kernel function the theoretical time complexity has been compared with real time complexity measured by program Gprof - what is presented in table I for two 4k length strings and variant length of substrings.

The table II presents results from text classification with Edit Distance and BOW precomputed kernels and varying parameter C. Results from text classification with Subsequence precomputed
<table>
<thead>
<tr>
<th>Kernel function</th>
<th>Substring length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td>substring kernel</td>
<td>0.60s</td>
</tr>
<tr>
<td>n-gram kernel</td>
<td></td>
</tr>
<tr>
<td>edit distance kernel*</td>
<td></td>
</tr>
<tr>
<td>bow kernel*</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE I: Measured time computation of Substring kernel functions**

<table>
<thead>
<tr>
<th>Kernel function</th>
<th>C parameter of C-SVC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Edit Distance Kernel</td>
<td>41.3</td>
</tr>
<tr>
<td>BOW Kernel</td>
<td>34.6</td>
</tr>
</tbody>
</table>

**TABLE II: Classification accuracy [%] of Edit Distance Kernel and BOW Kernel functions with varying parameter C. Computed with precomputed kernel.**

Kernels with varying parameter C and subsequence length are presented in the table III and classification using N-gram is shown in IV. Every value is average from 5-fold cross validation.

V. CONCLUSIONS AND FUTURE WORK

In our future work we will apply our approach to security issues like testing memory dump and network traffic data. After finding the optimal parameter we will compare accuracy and real time complexity with common IDS like Snort. Future work will be also focused on improvements

<table>
<thead>
<tr>
<th>Subsequence length</th>
<th>C parameter of C-SVC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>32.8</td>
</tr>
<tr>
<td>8</td>
<td>29.8</td>
</tr>
<tr>
<td>12</td>
<td>26.7</td>
</tr>
<tr>
<td>16</td>
<td>28.9</td>
</tr>
<tr>
<td>20</td>
<td>29.1</td>
</tr>
</tbody>
</table>

**TABLE III: Classification accuracy [%] of Subsequence Kernel function with varying parameter C and subsequence length Computed with precomputed kernel. NOT COMPLETE**
TABLE IV: Classification accuracy [%] of N-gram Kernel function with varying parameter C and substring length. Computed with precomputed kernel. NOT COMPLETE

of kernel algorithms and possibilities of classifying the hidden application network traffic.

REFERENCES


aIncNDA: Active Mode Incremental
Nonparametric Discriminant Analysis Learning

Shaoning Pang, Kshitij Dhoble, Gary Chen, and Nik Kasabov

Abstract

This paper presents a novel active mode incremental nonparametric discriminant analysis (aIncNDA) learning method, in which previous passive incremental NDA is extended with data selective sampling, and performs active online discrimination analysis. Given an incoming instance \( y \), the proposed aIncNDA computes a discrimination residue ratio between within-class and between-class \( \nu \), in which the residue is calculated using the \( k \)th regional nearest neighbor to class mean vector \( \| N_N_k(x, C) - \mu_C \| \). The proposed aIncNDA is capable of estimating the discriminant contribution for every newly presented instance, because \( \nu \) imitates the fundamental NDA \( tr(S_b^{-1}S_w) \) criterion for a maximum separation between classes and minimum separation within classes. In the experiment, we described how the discriminative instances can be significantly selected based on discrimination residue with, at most, minor sacrifices in learning rate and classification accuracy. The experimental results show that the proposed aIncNDA performs gracefully under different level of redundancy, and the proposed aIncNDA learning system is capable of learning with less number of instances, but has more often an improved discrimination performance, than an passive incremental NDA.

Index Terms

Nonparametric Discriminant Analysis, Incremental NDA, Active Learning, Active Mode Incremental NDA Learning.

I. INTRODUCTION

Active learning technique is crucial for classification as it iteratively selects distinctive information for training the classifier. Active rather than passive learning is preferred as it performs selective sampling, which enables the learning, immune to noise and data scarcity problems. Owing to its adaptive,
evolving and dynamic characteristics it is potentially useful for targeted learning tasks and works well particularly for nonlinear dataset/data stream. By now, active learning has been successfully used in the field of internet security, bioinformatics [25] and text classification [14].

Active learning fundamentally consists of two main components namely the selective sampling engine and the base classifier. Selective sampling is carried out based on a certain criterion, which selects informative instances from the given chunk of data to better the learning function. Thus active learning technique is principally more accurate and computationally efficient than passive learning.

In supervised machine learning for class discrimination, the nonparametric discriminant analysis (NDA) is similar to Linear Discriminant Analysis (LDA) [21], which seeks a transformation towards a maximum separation between classes and minimum separation within classes. Classic NDA is a passive batch learning approach, assumes the entire dataset for training is truly informative and is presented in advance. However in real world applications, data is often being presented at different times in a stream of random chunks, and the quality of data is often not guaranteed due to noise affection. Incremental NDA (IncNDA) [19] somehow has solved the difficulty of NDA and empowered the NDA with an flexibility of incremental learning that accommodate a data stream sequentially. But in spite of that, IncNDA still conducts a rigid learning because IncNDA does not make any instance choices before actual learning, just passively learns whatever instances that are confronted/provided.

In order to overcome NDAs passive learning limitation, we have proposed an active mode incremental NDA learning approach, which incorporates incremental NDA (IncNDA) and selective sampling technique together to form an online active learning. The proposed aIncNDA allows constant informative update of NDA eigenspace obtained from the incoming data.

The rest of paper is structured as follows: Section 2 describes related researches and motivations. Section 3 introduce previous passive NDA learning approaches. Section 4 presents the proposed methodology been used in this experimentation. Section 5 contains comparative experimental results of IncNDA and aIncNDA. Finally in Section 6 conclusion is given along with future work directions.

II. RELATED RESEARCHES AND MOTIVATIONS

The concept of Active learning has only been explored recently. The key to active learning lies in its adaptive selective sampling technique, which selects the most informative instances or data, and eventually boosts the performance of the classifier. The selected data will be assimilated into the training set to retrain the classifier in order to achieve improved level of performance. This procedure can be iterative, since the objective is to achieve a targeted level of performance with least amount of data and high number of informative instances. In our method, incremental NDA is addressed for active learning implementation.
A. Approaches of Active Learning

There are varieties of selective sampling approaches used in active learning models. Amongst them, one of the most commonly used technique is Pool-based active learning. However it suffers from multiple drawbacks. Most of the pool-based active learning iteratively selects random samples from the pool which may be informative or irrelevant [2]. Moreover, selecting the samples to be included in the pool itself is a time consuming process. Another selective sampling approach is membership query which selects samples directly from the dataset. Membership query scheme does not have the drawbacks posed by the pool-based scheme. It also reduces the predictive error rapidly and is less computationally intensive.

Clustering [6] and Batch mode active learning [7] are some of the other common flavors of active learning which aims at decreasing the redundancy amongst the selected instances, consequently providing more unique instances for the refinement of classifiers. Lastly, Query by Committee technique [8] is an effective approach, where selective sampling is based on the disagreement amongst ensemble of hypotheses. Some of the frequently used ensemble in this type of active learning includes techniques such as Bagging and Boosting.

For application, incorporation of active learning with support vector machine has been commonly used especially in the field of bioinformatics and text categorization [14]. However majority of them have made use of pool-based technique, which suffers from multiple drawbacks stated above, therefore it is recommended that though incorporation of active learning with SVM is good, other approaches such as membership querying or batch mode active learning should be used as they negate the drawbacks introduced by pool based learning.

B. Incremental Discriminant Analysis Approaches

It is well known that Linear Discriminant Analysis (LDA) [21] seeks a transformation towards a global maximum separation between classes and minimum separation within classes. In contrast, another known discriminant analysis approach, Nonparametric Discriminant Analysis (NDA) relies on local eigenvectors for obtaining discriminant knowledge from the entire dataset. The advantage of NDA over LDA is that, NDA does not rely on assumptions that instances are drawn from a given probability distribution, therefore are more robust than parametric methods such as LDA, and suits particularly on those nonlinear datasets. Similar to LDA, NDA requires the entire dataset for training presented in advance, thus is often called batch NDA in the literature. For incremental learning of NDA, Raducanu et. al [19] proposed an incremental version of NDA, which allows us to maintain a constantly updated NDA eigenspace. However, both batch NDA and incremental NDA are merely a passive learning approach, learning passively whatever data is being given/confronted.
C. Motivation of Active Mode Incremental NDA Learning

To enable active learning of NDA, we incorporated incremental NDA and selective sampling technique together to form a new active learning technique, which delivers constant informative updating of NDA eigenspace, therefore minimizing concept drift and computational cost.

III. PASSIVE NDA LEARNING APPROACHES

Classic NDA [1] assumes that the entire training dataset is provided in advance, the learning is passively done in one batch. Incremental NDA (IncNDA) is capable of learning incoming instance continuously, but IncNDA also learns inactively whatever instances are confronted. The computation of Batch NDA and IncNDA are briefed as follows.

A. Nonparametric Discriminant analysis (Batch NDA)

Assuming that the data samples we have belong to N classes. Let \( C_i \) represents samples belonging to one of the class \( i, i = 1, 2, 3, ..., N \). Then, a NDA discrimination eigenspace according to [19] can be computed to express the class separability of data,

\[
\Omega = tr(S_w^{-1}S_b)
\]  

In above \( \Omega \), \( S_w \) is the within class covariance matrix defined as:

\[
S_w = \sum_{i=1}^{C_N} \sum_{j \in C_i} (x_j - \mu_{C_i})(x_j - \mu_{C_i})^T;
\]  

\( S_b \) is the between class covariance matrix defined as,

\[
S_b = \sum_{i=1}^{C_N} \sum_{j=1,j \neq i}^{C_N} \sum_{q=1}^{w_{C_i}} W_{ijq}(x_q^i - \mu_{NN}(x_q^i, C_j))(x_q^i - \mu_{NN}(x_q^i, C_j))^T.
\]

where \( \mu_{C_i} \) is the mean vector of class \( C_i \), and \( w_{C_i} \) is the number of samples in class \( C_i \).

In \( S_b \), \( \mu_{NN}(x_q^i, C_j) \) is defined as a local K-NN mean,

\[
\mu_{NN}(x_q^i, C_j) = \frac{1}{K} \sum_{t=1}^{K} NN_t(x_q^i, C_j)
\]

where \( NN_t(m_q^i, C_j) \) represents the tth nearest neighbor from vector \( m_q^i \) to class \( C_j \). \( W_{ijq} \) is defined as a weighting function,

\[
w_{ijq} = \frac{d^\alpha(x_q^i, NN_t(x_q^i, C_i))(x_q^i, NN_t(x_q^i, C_j))}{d^\alpha(x_q^i, NN_t(x_q^i, C_i)) + (x_q^i, NN_t(x_q^i, C_j))},
\]

where \( \alpha \) denotes control parameter for sample weights which can be selected between zero and infinity.
B. Incremental Nonparametric Discriminant Analysis (IncNDA)

Consider new instances are presented in the future. Incremental NDA [19] incorporates the discriminant knowledge presented in the new coming sample as: given new instance \( y \) is coming in, then the current NDA model \( \Omega \) is required to be updated as,

\[
\Omega' = f(\Omega, y) = tr(S_w^{-1} \cdot S_b')
\]  

(6)

This means that \( S_w \) and \( S_b \) are required to be updated respectively.

According to , the updated between class \( S_b' \) and within class \( S_w' \) covariance matrix can be calculated as follows:

\[
S_b' = S_b - S_b^{in}(C_L) + S_b^{in}(C_L') + S_b^{out}(y^{C_L})
\]  

(7)

\[
S_w' = \sum_{j=1, j \neq L}^{C_N} S_w(C_j) + S_w(C_L')
\]  

(8)

where \( S_b^{in}(C_L) \) represents the covariance matrix between the existing class and the class newly presented, \( S_b^{out}(y^{C_L}) \) gives the covariance matrix between the existing class and the updated class \( C_L' \), and \( S_w(C_L') \) signifies the updated within class covariance matrix. For further computation approaches on \( S_b^{in}(C_L), S_b^{out}(y^{C_L}), \) and \( S_w(C_L') \), please refer to [19].

The above IncNDA can be used to construct an agent capable of updating the current discriminant knowledge \( \Omega(t) \) by \( \Omega(t + 1) = F(\Omega(t), y) \) whenever a new instance \( y \) is confronted by the agent in the future. However, the IncNLDA is counted as a passive learning approach, because the IncNDA learns passively every instance confronted, even if the instance is confirmed redundant or noise data.

IV. THE PROPOSED ACTIVE INCNDA (AIncNDA)

For active learning, we consider here an active learning way (AIncNDA) to empower the IncNDA with the ability of detecting the discriminative interestingness of data before it is delivered for IncNDA learning. That is, the above IncNDA can be renovated to conduct incremental learning in an active learning way,

\[
\Omega(t + 1) = \begin{cases} 
F_c(\Omega(t), y) & \text{if } L(t) > \xi \\
\Omega(t) & \text{otherwise.}
\end{cases}
\]  

(9)

where only discriminative instances are delivered for IncNDA learning. \( \xi \) is the threshold identifying discriminative criterion of NDA. The smaller \( \xi \) leads to the bigger number of instances learned by IncNDA.

Recall that the nature of NDA learning lies at the discriminability difference between the NDA transformed space and the original space. Straightforwardly, \( L(t) \) can be represented as a type of mathematical residue that reflects the discriminability difference between the NDA transformed space and the original space.
Given one new instance presented at one time, similar to [20], the discriminability difference between the NDA transformed space and the original space of the IncNDA at time \( t \) by a classification performance evaluation as,

\[
L(t) = Ad(t) - Ao(t),
\]

where \( Ad(\cdot) \) is the classification accuracy on discriminant eigenspace, and \( Ao(\cdot) \) is the accuracy on original space. It could be any type of classification performance evaluation by any classifier.

However, such performance-based residue calculation involves a serious problem. That is, the \( L(t) \) is highly classifier dependent. For example, suppose a K-NN method is used for performance evaluation \( Ad(\cdot) \) and \( Ao(\cdot) \), then the selected instances for incremental learning is meaningful only for K-NN classification and the category of prototype-based methods, but not for the classification using any other methods such as hyperplane-based support vector machines (SVM) and decision-tree based C4.5.

A. Discrimination Residue Ratio

The idea of discrimination residue ratio is adapted from the weighting function (i.e. Eq. (5)) used in NDA, where \( NN_k(x^i, C_i) \) and \( NN_k(x^i, C_j) \) emphasize local within class distances and local between class distances. As we know, the principle of NDA, similar to LDA, seeks simultaneously minimizing within class distances and maximizing between class distances. The difference between NDA and LDA is, LDA is global model, whereas NDA focus on local instances distribution.

Given \( M \) new instances \( Y = \{ y_1, y_2, \ldots, y_M \} \) presented as one chunk at time \( t \), for each instance \( y_i \in Y \), we can quickly estimate the within-class residue to the class mean vector \( \mu_{C_i} \):

\[
\| NN_k(y^i, C_i) - \mu_{C_i} \|,
\]

also the between-class residue to any other the class mean vector \( \mu_{C_j}, j = 1, \ldots, C_N, j \neq i \):

\[
\| NN_k(y^i, C_j) - \mu_{C_j} \|.
\]

Thus, the contribution of incoming instance \( y_i \) to the NDA fundamental maximum \( tr(S_w^{-1}S_b) \) criterion can be estimated as the following discrimination residue ratio of within-class to between-class scatter estimates

\[
\nu(y_i) = \frac{\| NN_k(y^i, C_i) - \mu_{C_i} \|}{\| \frac{1}{C_N} \sum_{j=1, j \neq i}^{C_N} NN_k(y^i, C_j) - \mu_{C_j} \|}
\]

if \( \nu(y_i) > 1 \), then the contribution of \( y_i \) to NDA discrimination is positive, otherwise is negative.

However, it is noticeable that the above discrimination residue ratio varies in practice largely depending on individual dataset. Thus, it is hard for us to determine a suitable threshold value for a given dataset. To overcome this difficulty, we compute the discrimination residue ratio for every instance of the \( Y \), then the above \( \nu(y_i) \) can be normalized as,

\[
\nu_{y_i} = \frac{\nu - \bar{\nu}}{\sqrt{\frac{1}{M} \sum_{m=1}^{M} (\nu_m - \bar{\nu})^2}}
\]
where \( \bar{\nu} = \frac{1}{M} \sum_{m=1}^{M} \nu_m \) is the chunk mean discrimination residue ratio. Thus, \( L(t) \) in Eq. (15) can be implemented by \( \nu_y \) as a chunk data filter.

\[
\Omega' = \begin{cases} 
F_c(\Omega, y) & \text{if } \nu(y) > \xi \\
\Omega & \text{otherwise.}
\end{cases}
\]  \quad (15)

V. EXPERIMENTS AND DISCUSSIONS

In this section, we have examined the efficiency and accuracy of the proposed aIncNDA method, and compared to IncNDA. Particularly, we investigate the relationship between 1) the discriminability and number of instances, 2) the redundancy and number of instances. To experiment on data with different discriminative characterization, we used datasets from two database resources. One resource is from UCI Machine Learning Repository [23], where we selected 8 datasets that have different application backgrounds and the features 100% of continuous/integer values and no missing value. The other resource is the MPEG-7 face database [24], which consists of pose and light two subsets, total 1355 face images of 271 persons, 5 different face images per person and each face image has the size of \( 56 \times 46 \).

A. Experimental Setup

To implement the proposed aIncNDA for incremental learning, we select randomly, for each dataset, 10% for initial batch NDA training, and divide the remaining data into 10 random chunks for incremental learning test. We collect every instance learned by aIncNDA, and evaluate the performance of aIncNDA and IncNDA on discrimination contribution at every learning stage. For performance evaluation, we compared the eigenspace from the proposed aIncNDA with the eigenspace from IncNDA by a leave-one-out kNN (k=1) classification over all data presented by current learning stage. Note that we use the term learning stage instead of the usual time scale since the events of data arriving in the above incremental learning may not happen in a regular time interval. Here, the number of learning stages is equivalent to the number of instances that have been learned by incremental models.

In the experiment, parameter \( \xi \) is relevant to the number of curiosity instances and the discriminability of the resulting NDA. For each experiments, we fixed \( \xi \) by the rule that the instances are significantly selected with, at most, minor sacrifices in discriminability.

B. Synthetic Dataset

We first experimented the proposed aIncNDA with a synthetic data set that has 3 classes 475 instances. The data distribution is a mixture of several 2D \((X1, X2)\) Gaussian distributions as shown in Fig. 1.

Fig. 2 gives the distribution of the 257 informative instances learned by aIncNDA. As compared to the data distribution of the entire 475 instances, the discriminative representativeness of the selected
instances by aIncNDA is clear because those 257 instances includes all critical instances for class distinction, such as instances involving class-mixture, and major representative instances of the independent class.

Fig. 2 illustrates the whole procedure of incremental learning with a comparison to IncNDA, where the horizontal and vertical axis represent the incremental stage and the classification accuracy from k-NN (k=1). As seen from the figure, the proposed aIncNDA and IncNDA is compared on the classification error at every incremental learning step. The classification accuracy difference between two methods is +0.842105, which indicates that the proposed aIncNDA achieves better learning effectiveness of the original IncNDA, although aIncNDA learns only 54.10% of total 475 instances.

C. UCI Datasets

Table I gives an comparison of aIncNDA versus IncNDA on the incremental learning of 8 UCI datasets. In the table, $\xi$ is fixed for each dataset by the rule described above, the number of instances and the percentage to the number of all instances is denoted as ‘No. Instances(rate)’, and the classification accuracies is denoted as ‘Acc.’. The discriminability difference (denoted as ‘Diff.’) is calculated as the
Fig. 2. The comparison of aIncNDA and IncNDA on the performance of incremental learning.

TABLE I

COMPARISON OF AINCNDA VERSUS INCNDA ON_INCREMENTAL LEARNING OF INSTANCES OVER 8 UCI DATASETS.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>aIncNDA</th>
<th>IncNDA</th>
<th>Dif.[%]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ξ</td>
<td>No. Instances(rate[%])</td>
<td>Acc.[%]</td>
</tr>
<tr>
<td>Iris</td>
<td>0.75</td>
<td>56 (37.3)</td>
<td>94.5</td>
</tr>
<tr>
<td>Liver-disorder</td>
<td>0.8</td>
<td>51 (22.2)</td>
<td>63.3</td>
</tr>
<tr>
<td>Vehicle</td>
<td>3.0e-3</td>
<td>251 (29.7)</td>
<td>77.6</td>
</tr>
<tr>
<td>Glass</td>
<td>0.98</td>
<td>50 (23.4)</td>
<td>60.1</td>
</tr>
<tr>
<td>Wine</td>
<td>0.95</td>
<td>162 (92.7)</td>
<td>83.7</td>
</tr>
<tr>
<td>Wisconsin</td>
<td>0.95</td>
<td>443 (95.7)</td>
<td>84.3</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>0.7</td>
<td>291 (83.1)</td>
<td>76.2</td>
</tr>
<tr>
<td>Heart</td>
<td>0.65</td>
<td>33 (11.1)</td>
<td>53.2</td>
</tr>
</tbody>
</table>

proposed aIncNDA minus IncLDA in terms of the K-NN LOO classification performance at the final learning stage.

As seen in the table, the proposed aIncLDA method, ignores 4.7%-88.9% instances of the whole dataset, constructs discriminant eigenspaces on the remaining 11.1%-95.3% selected instances. However, the discriminability of the obtained eigenspace from composed instance subset, compared to the eigenspace from all instances (using IncLDA), has no decrease, reversely, most of case has a slight increase. This suggests that the proposed active IncNDA learning is valid, and the selected instances by aIncNDA have the expected discriminative representativeness.
D. Performance under different discriminative redundancy

To test the performance of the proposed method under different level of discriminative redundancy, we carried out face recognition (FR) and face membership authentication (FMA) experiments [26], [27], [28] using the same face database described above. FMA is to distinguish the membership class (cls. 1) from the non-membership class (cls. 2) in a total group through a binary class classification. FMA involves more discriminative redundancy than face recognition problem, because the size of membership in FMA is often smaller than that of nonmembership, which indicates that not every instance are discriminatively important for FMA.

Over the 271 persons 1355 faces data, we conducted FR and FMA, respectively. For the FMA experiment, we set the membership size as 71 (cls. 1/cls. 2 is 71/200) without loss of generality. Thus, we compared the proposed \( aIncNDA \) with the \( IncNDA \) on incremental learning of 271 classes (i.e. FR) and 2 classes (i.e. FMA) data, respectively.

Fig. 3(a) shows the comparison of NDA discriminability between the proposed \( aIncNDA \) and the \( IncLDA \) for both FR and FMA experiments, and Fig. 3(b) reports corresponding the number of instances learned by \( aIncNDA \).

As seen in Fig. 3(a), the proposed \( aIncNDA \) learns NDA for FR on 1331 of total 1355 instances, only 24 instances are found redundant. Whereas for FMA, \( aIncNDA \) learns 1093 of 1355 which is only about 20.0% of total 1355 instances are reduced. However, the performance of the proposed \( aIncNDA \) for both FR and FMA as given in Fig. 3(a) outperforms in most cases, the performance of the \( IncNDA \) on all 1355 instances. This indicates that the proposed \( aIncNDA \) is able to suit itself automatically to data with discriminative redundancy, and select a suitable number of instance to build an correct NDA model. This also can be reflect from Fig. 3(b), where \( aIncNDA \) is shown actively selecting different number of instance for incremental learning.

VI. CONCLUSION AND FUTURE WORKS

Method based on passive learning prove to be inadequate in real world application. To overcome this limitation, we have developed Active Mode Incremental NDA which performs adaptive discriminant selection of instances for incremental NDA learning. Performance evaluation carried out on benchmark UCI datasets show that Active Mode Incremental NDA performs on par and in many cases better then incremental NDA with less number of instances. Given the nature of network data which is large, streaming, and constantly changing, we believe that our method can find practical application in the field of internet security.

Over the datasets from different resources, the proposed \( aIncNDA \) learning method is evaluated on: (1) \( aIncNDA \) versus \( IncNDA \), and (2) performance under different level redundancy, where face recognition and face membership authentication are studied, respectively. The experimental results
Fig. 3. Comparison of aIncNDA and IncNDA on FR and FMA, (a) the performance of aIncLDA versus IncNDA on incremental learning; (b) the number of learned instances by aIncNDA at every learning stage.

demonstrate that the proposed aIncNDA learning helps more efficient NDA learning with fewer instances, but with no performance deduction. One limitation of the proposed method concerns, as the original IncNDA retains raw data at every step of incremental learning, the data processing in aIncNDA is not one-pass.

As future work, the presented methods application in intrusion detection system will be exploited along with added enhancements to the selective sampling criterion. Also, the use of incremental classifier will be researched to serve as an extension to our present model which will eliminate the need for retraining further enhancing the processing speed while been computationally efficient.
ACKNOWLEDGEMENT

The authors would like to thank Prof. Jie Yang for the useful discussion on this topic of discrimination analysis modelling during Dr. Pang’s research visit at the institute of image processing and pattern recognition, Shanghai Jiao Tong University, in Dec. 2008.

REFERENCES


Hierarchy Minimum Enclosing Balls for Multi-label Classification with Reference to Intrusion Detection

Ye Chen, Shaoning Pang, Nikola Kasabov

Abstract

As network intrusions are usually associated with multiple labels in a hierarchy structure, the classification of network intrusions naturally possess a hierarchical multi-label classification (HMC) problem, in which every instance may belong to more than one class. Addressing HMC problem, we propose in this paper a novel approach of hierarchy minimum enclosing balls (HMEB), where a label hierarchy is modeled as data HyperSphere hierarchies, called minimum enclosing ball (MEB). The MEBs are separating, encompassing and overlapping with each other formed as tree-like structure. Given an unlabeled sample, HMEB seeks a MEB enclosing the sample, and label the sample according to the MEB’s position in the MEB hierarchy. The proposed method has been validated on the hierarchy multi-label problem from KDD’99 and Yeast dataset, respectively. The experimental results show that the proposed HMEB improves the classification accuracy of U2R from 13.2% to 82.7% and R2L from 8.4% to 45.9%, as compared to the winner of KDD’99. Also, the efficiency of HMEB is highlighted, as the computational time stays steady while the size of training data exponentially manifolds in another experiment of Yeast data.

Index Terms

Multi-label, Minimum enclosing ball(MEB), Hierarchical Multi-label Classification(HMC), Network Intrusions

Knowledge Engineering & Discovery Research Institute, Auckland University of Technology, Private Bag 92006, Auckland 1020, New Zealand Email: {zhf0202,spang, nkasabov}@aut.ac.nz.
I. INTRODUCTION

Classification refers to the task of categorizing a set of instances for learning. Such that we can use it to predict the class of previously unseen instances. Hierarchical multi-label classification (HMC) differs from normal classification in two aspects. Firstly, an instance may belong to more than one class simultaneously. Secondly, instances from any classes belong automatically to its ancestor classes, because classes are structured as a hierarchy (i.e. a tree structure). The problem of HMC has been addressed in several domains, including text classification [1], functional genomics [2], and object recognition [3].

Along with the explosive growth of Internet users, the issues relating to Internet security are also increasing. It has become a challenging issue to develop a system for safeguarding the users/system from these threats. As a preventive measure, several intrusion detection systems have been devised. However network intrusion detection still has underlying issues relating to categorization of the attack types.

Previous researches focus on grouping any network attacks into several major clusters thereby reducing the detection rate/efficiency due to that the lower level information of intrusion attacks been disregarded. Therefore, splitting a major cluster into sub-clusters would give more specific information on the attack type, which may eventually result in a more effective intrusion detection (as seen in Fig.1). For example, spam can be simply divided into several major categories, such as junk mail, IM spam, TXT spam etc. However, for an accurate spam detection, it is advantageous to drill into each category, and further divide the division of junk mail into Unsolicited bulk e-mail (UBE) and Unsolicited commercial e-mail (UCE), as this gives obviously more insight into the nature of spam. These attack types have been hierarchically organized.

Motivated by this, we consider intrusion detection as multi-label hierarchical classification problem, where attacks of different types have been coagulated using hierarchal minimum enclosing balls (HMEB). The minimum enclosing ball (MEB) computes a hypersphere of minimum radius which encloses a specific attack/intrusion into separate MEBs. Given an unlabeled instance/attack, HMEB seeks a MEB that encloses similar attack type, and labels the sample according to the MEBs position in the hierarchy. HMEB is akin to tree structure, in which the MEBs are separating, encompassing, and overlap with each other.

We have authenticated the proposed HMEBs classification proficiency and computational
efficiency. On comparison with Bernads method on KDD’99, the proposed HMEB exhibits an outstanding classification accuracy for those classes with smaller number of instances. On the other hand, the capability of HMEB for handling large amount of data is demonstrated in another experiment with multi-labelled Yeast dataset.

The rest of paper is structured as follows: Section 2 contains motivation and review of the previous works done in this area. Section 3 states the problem specification. The Methodology been used is explained in Section 4. In Section 5, we cover experimentation and results. Lastly, in Section 6 we draw our conclusion and state future directions.

II. RELATED RESEARCHES AND MOTIVATION

A. Review of Multi-label Classification Approaches

The existing multi-label classification methods can be grouped into two main categories: a) problem transformation methods and b) algorithm adaptation methods.

The problem transformation methods were defined in [4] as those methods that transform the multi-label classification problem either into one or multiple single-label’s classification or regression [5], [6]. One common ‘problem transformation method’ is to train \(|L|\) binary classifiers \(H_l: X \rightarrow \{l, \neg l\}\), one for each individual label \(l\) in \(L\). Thus, the original data set is transformed into \(|L|\) data sets, in which each dataset \(D_l\) contains the same instance of the original dataset. For each dataset \(D_l\), instances are labeled as \(l\) or \(\neg l\), then a binary classifier is applied for the classification on label \(l\). Given a unlabeled instance \(x\) for classification, a set of labels are produced by \(|L|\) classifiers,

\[
H(x) = \bigcup_{l \in L} \{l\} : H_l(x) = l.
\]  

(1)

Algorithm adaptation methods extend the existing learning algorithms to handle multi label data directly. Adaboost.MH and Adaboost.MR [7] are two extensions of AdaBoost [8] for multi-label classification. Both extensions are used on weak classifiers represented in the form \(H: X \times L \rightarrow R\). In AdaBoost.MH, if the output sign of the weak classifiers is positive, then the new instance \(x\) is labeled as \(l\); otherwise, this instance is labeled as \(\bar{l}\). In AdaBoost.MR, the output of the weak classifiers is considered for ranking each of the labels in \(L\).
B. Review of MEB Modeling

The MEB problem computes the ball(s) of minimum radius enclosing a given set of points. Traditional algorithms for finding exact MEBs [9], [10] do not scale well with the dimensionality d of the labeled data points. Consequently, recent attention has been on the approximation algorithms development by [11], [12]. In particular, a breakthrough was obtained by Badoiu and Clarkson, who showed that an \((1 + e) - approximation\) of the MEB can be efficiently obtained using core sets. Generally speaking, in an optimization problem, a core set is a subset of input points such that we can get a good approximation on the original input by solving the optimization problem directly on the core set. An interesting property of [11] is that the size of its core set was shown to be independent of both d and the size of the point set. As the MEB problem arises in a number of important applications, often requiring to be solved in relatively higher dimensions. Kumar et al. developed methods to compute the MEB in a higher hypersphere dimension.

C. Motivations of MEB for HMC

Addressing HMC for network intrusion detection, we investigate the suitability of MEBs. We extend the problem transformation methods [5], [6] and the high dimensional MEB [11], [12], and propose a constructive hierarchy minimum enclosing balls (HMEB) method. Our motivations of using MEB for HMC are summarized as follows:

1) More specific information can be learned from the dataset by separating it into more subsets and marking as hierarchical multi-label.

2) MEB is simple to implement and results in faster computation in higher dimensions therefore matching the processing requirements of large network data.

3) The MEBs are organized into a tree-like structure, allowing the multi-label problem to be solved easily by analyzing the relationship between the MEBs.

4) The classification performance is independent of the dataset size.

III. HMC PROBLEM INVESTIGATION

In the example of network intrusion detection, Fig. 1 describes a typical HMC adapted from KDD’99 dataset, where different network connection types are divided into 4 major categories, and 21 subcategories, and the relationship between these subcategory connection types and major
connection category types are structured as a Markov Tree. The major connection types are set as the first level, and those subcategory types as the second level of the tree. In this way, the problem of Internet intrusion detection is interpolated into the learning of hierarchy multi-label classification. Here, the definition of HMC task is briefed as follows [13]:

**Given:**

1) a set of $m$ instance $S = \{x_1, \ldots, x_m\}$, where each $x_i \in \mathbb{R}^d$, 
2) a class hierarchy $(C, \leq_h)$, where $C$ is a set of classes and $\leq_h$ is a rooted tree representing the superclass relationship ($\forall c_1, c_2 \in C : c_1 \leq_h c_2$, if and only if $c_1$ is a superclass of $c_2$), and
3) a set $T$ of examples $(x_i, C_i)$ with $x_i \in S$ and $C_i \subseteq C$ such that $c \in C_i \Rightarrow \forall c' \leq_h c : c' \in C_i$.

**Find:** a function $f : x \rightarrow 2^c$ where $2^c$ is the power set of $C$, such that $c \in f(x) \Rightarrow \forall c' \leq_h c : c' \in f(x)$. In this article, the function $f$ is represented with a set of hierarchical MEBs. We compute the MEB $B_{c_1}$ where $(x, c_1) \subset B_{c_1}$, and $B_{c_2}$ where $(x_i, c_2) \subset B_{c_2}$. If $c_1 \leq_h c_2$ and $x_i \subset x$, then the instance located only in the overlap space $O = B_{c_1} \cap B_{c_2}$ represents the distribution of the data of classes $c_1$ and $c_2$. In our case, we need to find the overlap space $O$ between the main category data distribution with its subcategories distribution.

### IV. Methodology

#### A. Minimum Enclosing Ball

Given a set of points $S = \{x_1, \ldots, x_m\}$, where each $x_i \in \mathbb{R}^d$, the minimum enclosing ball of $S$ (denoted by $\text{MEB}(S)$) is the smallest ball that contains all the points in $S$. Since the traditional algorithm given by [10] and [9] does not scale well with the dimensionality $d$ of the points. A reformative solution was proposed by [11], who show that an $(1 + \epsilon)$-approximation of the MEB can be efficiently obtained based on Core Set. The core set is a subset of given dataset contains the instances located at the outer area.

Let $B_S(c, r)$ be an exact MEB of the data set $S$ with center $c$ and radius $r$, and $B_Q(\tilde{c}, \tilde{r})$ be another exact MEB with center $\tilde{c}$ and radius $\tilde{r}$. Note that, different from $B_S$, MEB $B_Q$ is constructed on the Core Set of $S$: $(Q), Q \subset S$. Given an $\epsilon > 0$, a ball $B_Q(\tilde{c}, (1 + \epsilon)\tilde{r})$ is a $(1 + \epsilon)$-approximation of $B_S(c, r)$, if $S \supseteq B_Q(\tilde{c}, (1 + \epsilon)\tilde{r})$ and $\tilde{r} \leq r$.

Formally, subset $Q$ is judged as the core set of $S$, if an expansion by a factor $(1 + \epsilon)$ of its MEB contains $S$ (i.e. $S \subset B_Q(\tilde{c}, (1 + \epsilon)\tilde{r})$). Figure 2 gives an example of exact MEB, Core set
Fig. 1: The label hierarchy of network connection types adapted from KDD’99 dataset

MEB, and Core set MEB expansion, where the dotted circle identifies the exact MEB of the entire dataset $B_S$, and the inside solid line circle gives the exact MEB of Core set $B_Q$ (denoted as points in square). $B_Q$ does not cover the whole data points, but its $(1 + \epsilon)$ expansion (the outside circle) does.

B. Kernel MEB

One major issue with the original MEB method is that it cannot be used for kernel learning (i.e. support vector machine). To solve this problem, [14] proposed the kernel MEB method.

Let $\varphi$ be the feature map corresponding to kernel $k$. Given a set of $\varphi$-mapped points $S_\varphi = \{\varphi(x_1), \ldots, \varphi(x_n)\}$, its MEB is denoted by $B(c^*, r^*)$ with center $c^*$ and radius $r^*$. It is the smallest ball that encloses all these data points in $S_\varphi$ [14]:

$$(c^*, r^*) = \arg \min_{c,r} r^2 : \|c - \varphi(x_i)\|^2 \leq r^2 \ \forall i.$$
The corresponding Wolfe dual is:

$$
\max_{\alpha} \sum_{i=1}^{m} \alpha_i k(x_i, x_i) - \sum_{i,j=1}^{m} \alpha_i \alpha_j k(x_i, x_j) \alpha \geq 0, \quad i = 1, \ldots, m, \quad \sum_{i=1}^{m} \alpha_i = 1, \quad (3)
$$

where $\alpha = [\alpha_1, \ldots, \alpha_m]$ are the Lagrange multipliers and $K_{m \times m} = [k(x_i, x_j)]$ is the kernel matrix. The MEB center $c$ and radius $r$ can be recovered from the optimal $\alpha$ as [14]:

$$
c = \sum_{i=1}^{m} \alpha_i \varphi(x_i), \quad r = -\sqrt{\alpha' \text{diag}(K) - \alpha' K \alpha}. \quad (4)
$$

In this way, the kernel learning problem is resolved by setting the instances in core set as the support vectors, to create a boundary for the corresponding class.

C. Hierarchy Minimum Enclosing Ball

To solve the multi-label problem, we have extended the usage of minimum enclosing ball.

Given $S = (X \times Y), X = \mathbb{R}^d$ and $Y = c_1, c_2, \ldots, c_n$. Where $c_i$ represents the set of labels for each corresponding class levels. Firstly, we have aligned the multi-label to a hierarchical rooted tree $T$ where each level indicates a set of labels $c_i$ and the nodes of each level represent the labels contained in $c_i$. For each class $y_l^i \in c_i$ we compute a MEB $B_l^i$, and replace the nodes of the original rooted tree $T$ by the corresponding MEB. As seen in Fig. 3, the tree on left hand
side represents the hierarchy multi-label structure and the tree on right hand side represents the transformed hierarchy MEB structure. We mark each MEB $B^k_i$ with not only the corresponding class $l_i$ it encloses, but also the parent class $p_i$ it belongs to, thus $B^k_i$ is transferred to $B^{l_i,p_i}_i$. Note that the parent class for the MEBs located at the top level will be labeled as class 0. In this way, each MEB is able to find its parent or child MEB, allowing the structure of the rooted tree to be encoded into the labels of the MEBs as shown in Fig. 3.

![Hierarchy Labels to Hierarchy MEBs](image)

**Fig. 3:** Transfer hierarchy multi-label to hierarchy minimum enclosing ball

For training HMEB, we consider each class as a binary problem where alternatively every class is considered positive and the remaining classes as negative. Then we compute a set of MEBs, which encloses only the positive instances by the above discussed method. In addition, the model of this class can be formed by assigning the corresponding core sets as support vectors.

In order to classify an unlabeled instance $x$, we seek $n$ MEBs whose center is closer to $x$ then the other MEBs. To compute the distance between $x$ and center $c_t$ belonging to MEB $B_t$ we extended the given formula 4. Putting $x$ into a feature map $\varphi$, we replace it with a kernel $k(x, x) = \kappa$. And the distance between $k(x, x)$ and $c_t$ is computed as:

$$
\|c_t - \varphi(x)\|^2 = \sum_{z_i, z_j \in S_t} \alpha_i \alpha_j k(z_i, z_j) - 2 \sum_{z_i \in S_t} \alpha_i k(z_i, x) + k(x, x). \quad (5)
$$

Then proceed by checking whether $x$ is inside the boundary formed by the corresponding core sets of support vectors derived from the nearest MEBs. We continue this iterative process till we find a set of $n$ nearest MEBs having $x$, resulting in the formation of a complete tree path
from the root to the terminal level of the MEB tree. The instance $x$ seeks MEBs starting from the root hierarchy, and progressively seeks child MEBs, assigning itself the labels of ancestor MEBs under which the $x$ instance falls.

Since, the MEB core set generates an approximate boundary of the corresponding class, it may cover non-essential area. In order to resolve this problem, one method is to create multiple sub-core sets, such that it includes more data points giving optimally minimum enclosing space while discarding the non-essential areas. We assume that the data points for each of the child classes are closely grouped in different area of the parent class distribution space. Since, the instance is placed in parent class distribution area, it must be present in at least one of the child class distribution area. Thus, we can optimize the original area $P$ for the super-class $C$ as

$$P_{optimized} = P \bigcap (P_1 \bigcup P_2 \cdots \bigcup P_n),$$

where \{ $P_1, P_2, \ldots, P_n$ \} denotes the distribution area of the sub-class of $C$. Figure 4 depicts the situation with one parent class and 3 child classes.

![Diagram](image)

Fig. 4: Super-class core set boundary VS Super-class core set boundary $\bigcup$ 3 sub-classes core set boundaries
D. The Proposed HMEB Algorithm

According to the theories discussed above, we summarized and developed the following 3 algorithms for HMC.

Algorithm 1 to computer a MEB and the core set by given a set of instances.

To train a given dataset $S = (X \times Y)$, $X = \mathbb{R}^d$ and $y_i^j \in Y$, $i = \{1, 2, \ldots, n\}, j = \{1, 2, \ldots, m_i\}$, where $n$ indicates the number of hierarchy levels, and $m_i$ denotes the number of class in $i^{th}$ level. We proposed Algorithm 2 using kernel method to compute the MEB and core for each class $S_i = (X_i, y_i) \subset S$, and hierarchically label the MEBs and core sets according to the corresponding level and its parent class.

We also developed a HMEB testing algorithm (Algorithm 3) to classify a new instance $x$.

---

**Algorithm 1** MEB algorithm

**Input:** Set of points $S \in \mathbb{R}^d$, parameter $\epsilon = 2^{-m}$, subset $Q_0 \subset S$

**Output:** A $(1 + \epsilon)$-approximation $\text{MEB}(S)$ and $O(1/\epsilon)$-size core set

1: for $i \leftarrow 1$ to $m$ do
2:   loop
3:     $Q \leftarrow Q_{i-1}$
4:     Computer $B_{c,r} = \text{MEB}(Q)$
5:     if $S \subset B_{c,(1+\epsilon)r}$ then
6:        Return $Bc, r, Q$
7:     else
8:        $p \leftarrow$ point of $S$ maximizing distance, $\|cp\|$, from $c$
9:     end if
10:    $Q \leftarrow Q_{i-1} \cup \{p\}$
11:   end loop
12: $Q_i \leftarrow Q$
13: end for
**Algorithm 2** Hierarchy Minimum Enclosing Ball Training Algorithm

**Input:** Set of points \( S = (X \times Y), y_i^j \in Y, i = \{1, 2, \ldots, n\}, j = \{1, 2, \ldots, m\}, \epsilon \)

**Output:** A \((1 + \epsilon)\)-approximation MEB(S) and \(O(1/\epsilon)\)-size core set

1. for \( i \leftarrow 1 \) to \( n \) do
2. for \( j \leftarrow 1 \) to \( m_i \) do
3. \( S = \{\varphi(x)\}, x = (X, y_i^j) \) and \( Q = 0 \).
4. \([\text{MEB}(B), \text{CoreSet}(C)] \leftarrow \text{Run Algorithm 1 with input } S, Q, \text{ and } \epsilon.\)
5. \( p_{i,j} \leftarrow \text{parent class label for } y_i^j \)
6. \( B_{p_{i,j},y_i^j} \leftarrow B, C_{p_{i,j},y_i^j} \leftarrow C \)
7. end for
8. end for

V. Experiments and Discussions

While testing the capability of the proposed HMEB for multi-label classification, particularly on hierarchy multi-label problem, we experimented the proposed method specially on KDD’99 and Yeast datasets, respectively. Here, KDD’99 gives a hierarchy multi-label problem and Yeast poses a traditional single level multi-label structure.

A. Experimental setup

The proposed HMEB is implemented in MATLAB version(7.6.0), on a 1.86Hz Intel Core 2 machine with 2GB RAM. In our experiments, we use non-linear Gaussian kernel \( k(x, y) = \exp(-\|x - y\|^2 / \beta) \) with \( \beta = \frac{1}{m^2} \sum_{i,j=1}^{m} \|x_i - y_j^2\|. \) We set the \( \epsilon \) as \( 1e^{-6} \) for \((1+\epsilon)\)-approximation of MEB, and the Gaussian kernel parameter \( \beta \) as 1000. The parameter \( n \) in algorithm 3 identifies the number of the nearest MEBs to the input instance. Normally, it should be a number greater than or equal to the layer number of the label hierarchy. Thus, for Yeast, we set \( n \) as 7, since the maximum number of labels for an instance is 7.

In practice, \( n \) can be determined by a cross validation test procedure. For example, we set \( n \) as 3 for KDD’99 dataset as observed in Fig. 5, where different \( n \) values are tested, and \( n \) is fixed where the highest 93.6% accuracy is achieved by HMEB.
Algorithm 3: Hierarchy Minimum Enclosing Ball Testing Algorithm

**Input:** An instance $x$, of $t$ MEBs where $B = \{B_1^{l_1,p_1}, B_2^{l_2,p_2}, \ldots, B_t^{l_t,p_t}\}$ with $m$ hierarchy levels, where $l, p$ denote the corresponding class label and the super-class label respectively, parameter $n$

**Output:** $Label$ which is a set of $m$ labels

1. Transfer $x$ to kennel $k(x, x)$ by putting it in feature map $\varphi$
2. $Label \leftarrow 0$
3. $index = \{1, 1, \ldots, 1\}$
4. $i \leftarrow 1$
5. while $i \leq m$ do
   6. for $j \leftarrow \text{index}(i)$ to $n$ do
      7. Find MEB $B_k^{l_k,p_k} \in B$ where $p_k = Label(i)$, $B_k$’s center is $j^{th}$ nearest to $k(x, x)$
      8. if $k(x, x)$ inside of boundary formed by core set of $B_k$ then
         9. $Label \leftarrow Label \cup l_k$
         10. Break
      11. else if $j == n$ then
         12. $Label \leftarrow Label - Label(i)$
         13. $i \leftarrow i - 1$
      14. end if
   end for
   16. if $Label$ is empty then
      17. Return $Label \leftarrow 0$
   end if
   19. $index(i) = index(i) + 1$
   20. $i \leftarrow i + 1$
7. end while

B. Experiment on multi-label dataset yeast

The Yeast dataset is formed by micro-array expression data and phylogenetic profiles with 1500 genes in the learning set and 917 in the testing set. The dataset dimensionality is 103.
Each instance represents a gene sequence, and each gene is associated with a set of functional classes consisting of 14 different classes [15].

For constructing HMEB, we implement the Algorithm 2 on the Yeast dataset. As testing, Algorithm 3 checks the $n$ nearest MEBs, and labels each new instance as the classes of the MEBs enclosing the instance.

For performance evaluation, we compare HMEB with SVM [16] on the computational time cost for training and testing. Here, the SVM uses a radial basis kernel, whose parameters are determined by cross validation tests.

Fig. 6a gives the comparison results, where ROC-like curves present the variation of algorithms computational time cost (in CPU time) against the increase of training data size. As seen, the proposed HMEB is more computational expensive than SVM as the size of training set is small. But, the time cost of HMEB increases more slowly than the SVM. On comparison of testing time, Fig. 7 shows that HMEB’s time cost is only half of the SVM’s time cost. This could be explained by SVM’s use of one-against-one method [17] for the multi-label testing, whereas HMEB simply seeks the $n$ nearest MEBs, which indicates that HMEB is capable of dealing with huge volume dataset.
C. Experiment on hierarchy multi-label dataset KDD’99

1) KDD’99: KDD Cup 1999 dataset contains 5 million Internet connection records. Each record encodes 41 connection features including 1 class label, 34 continuous features, and 7 symbolic features. The class label identifies one of 22 connection types (i.e. normal, buffer_overflow, guess_password etc.). All connection features are assumed to be conditionally independent. Apart from the normal connection type, the rest of 21 attack types are associated with 4 major categories of attack, they are:

1) DOS, denial of service, e.g. back;
2) R2L, unauthorized access from a remote machine, e.g. guessing password;
3) U2R, unauthorized access to local superuser (root) privileges, e.g. buffer overflow;
4) PROBE, information gathering, e.g. port sweep.

2) **Data Preprocessing:** For data preprocessing, we replace the original label for each instance with a set of numerical labels by the following rules: the major attack categories stay at the head followed by its subcategories. For example, 'back', as the first subcategory of DOS attack, is represented as (1,1), and the second subcategory 'land', as (1,2). In this way, all labels can be transformed into a numerical hierarchical structure as shown in Fig. 8. Additionally, we normalize every continuous feature into [0, 1], and encode each symbolic feature to binary digit.

![Fig. 8: KDD99 data classification label replacement](image)

3) **Comparison Results:** For constructing HMEB, we compute MEBs and core sets for 21 subcategories and the 4 major categories of attacks, respectively. For each categories, including subcategories, we use the instances of the core set as the support vectors. As a result, we obtain total 25 support vector sets corresponding to 4 major categories plus 21 subcategories of attacks. Table I presents our classification results with a comparison to the results achieved by Bernhard who is the winner of KDD’99 cup.

As seen from the table, Bernhard (1999) achieved an extremely high classification accuracy of 99.5% on normal connect type, however U2R and R2L showed poor classification performance, with none of them exceeding 15%, because of their class size being smaller than the other classes. Although the overall classification accuracy of the proposed HMEB is slightly lower than that of the Bernhard’s method for normal connection type and Dos type attacks, the proposed HMEB wins Bernhard’s method on the classification of 3 most important classes.
TABLE I: Compare the classification accuracy with (Bernhard 1999) who is the winner of KDD’99 cup

HMEB particularly increases the classification accuracy of U2R and R2L by 70% and 35%, respectively. This demonstrates the advantage of the proposed HMEB, which encloses all the data points of corresponding class, enabling a more accurate approximation of class information, even for very small dataset.

VI. CONCLUSION

In this paper, we studied a hierarchy multi-label problem with reference to network intrusion detection. On investigating the minimum enclosing ball, which encloses the data points into a smallest HyperSphere, we proposed a novel ‘hierarchy minimum enclosing ball’ method, where MEBs are separating, encompassing and overlapping with each other representing an organized multi-label tree structure. Moreover, we have modified the original MEB algorithm using the Gaussian kernel method, enabling MEBs to be applicable for kernel computing in a higher dimensional space. In our experiments, we have implemented the HMEB on benchmark datasets from UCI archives such as KDD’99 and yeast. The experimental results show that the proposed HMEB is clearly more favorable than the traditional methods, especially for the U2R and R2L problem of the KDD’99 dataset.
In general, the proposed method has the following desirable properties. First, the proposed method models a difficult hierarchical multi-label problem as a simple MEB association analysis. Second, MEB excludes the existing sparseness of data, which enables HMEB to approximate data more accurately. However, the proposed method inhibits a limitation that the classification performance of the terminal classes might be inferior to that of non-terminal classes since the boundaries for the terminal classes in a multi-label hierarchy are not optimized.

In our further work, we will address the limitations involved with the terminal classes by investigating on measuring the density of the MEB and exploit new models based on 'hierarchy minimum enclosing ball'.

REFERENCES


Spanning SVM Classification Tree along the Grid of Data Spareness

Shaoning Pang, Tao Ban, Youki Kadobayashi, and Nikola Kasabov

Abstract

Addressing the capability of a given dataset that could be better modelled by a modular method for classification, this paper proposed a new type of SVM aggregating method, called gridding SVM Classification Tree (gSVMT). The proposed gSVMT achieves to decompose a problem without loss of the original discriminant information, discover subregions contained SVM knowledge in the data through a recursive SVM-supervised data part, and for each obtained subregion, an individual SVM is allocated for approximating the subregion knowledge, and a group of such SVMs are aggregated along the grid in an order that they are explored. Experiments with a Gaussian synthetic data, 13 benchmark machine learning datasets, have highlighted the utility of gSVMs on its very competitive classification capability. In particular, gSVMs is demonstrated absolutely better than single SVM for those datasets with high sparseness and/or class-imbalance. This has been further demonstrated with its very successful real application on face membership authentication.

Index terms— Sparseness, Class Imbalance, SVM Aggregating, Modular Classifier, Data Partitioning.

I. INTRODUCTION

Given a unknown pattern for classification, one often considers, for the first instance, to use one of available single discriminative classifiers such as a well known k-nearest neighbor classifier, feed-forward neural network, or support vector machines (SVM) to solve the problem quickly in one-stage process. But more often, when complex patterns are confronted in real-world applications, single-stage approaches are unable to solve the problem promptly as we expected. In this case, one may fall back on data preprocessing techniques, e.g. feature extraction, to reduce the classification difficulty, or alternatively just turn to using multi-stage approaches. The idea
is to observe, measure, and break the problem into a union of several simpler decisions, and finally summarize to yield a global decision. This generally is called ‘divide-and-conquer’. The representative multi-stage classification methods are decision tree classifiers [19] and modular classifiers [20], e.g. SVM ensemble [10], and SVM classification tree (SVMT) [15].

Profiting from the divide-and-conquer strategy, the advantages of a multi-stage classifier over a single-stage classifier include quick convergence, parallel training, and better generalization. In the literature, multi-stage classifiers have been successfully applied in a diversity of areas including character recognition, remote sensing, medical diagnosis, expert systems, and speech recognition [4], [6]. However, given a new data set, there is no existing guidelines on how to choose such an appropriate multi-stage classifier that suits properly the specific problem with the preferable performance guaranteed. Thus, one often confronts an embarrassment to implement and evaluate a number of different types of methods on the data set. Obviously, this is a frustrating procedure because it is not just a lot time and labor demanding, but sometimes could be eventually resultless, considering the case of a real world data set with a large scale rendering the methods under testing mostly noneffective.

In this paper, we focus our discussion on the usability and construction of a novel multi-stage classifier. The contribution of the paper is twofold. First, we introduce two measurements characterizing the distribution of the data set, namely the sparseness and the class-imbalance. This helps to identify weather a data set can be well fitted by a multi-stage approach so that a generalization performance better than a state-of-the-art single-stage approach can be expected. Hence, via a preceding exploratory analysis step, we learn if experiments with multi-stage approaches are necessary.

Second, we propose a novel SVM-based modular classifier termed gridding SVM Classification Tree (gSVMT). The name of gSVMT comes from the fact that the component SVMs are aggregated over a grid learned from data. The proposed gSVMT implements a ‘divide-and-conquer’ strategy as: first perform data partitioning at the sparse region of the data distribution, then conduct decision-making through the aggregation of a group of regional SVMs. Owing to that every component SVM not only explores the discriminant knowledge but also simultaneously imposes a balancing requirement in the subregion, the gSVMT demonstrates a strong capability for modelling sparse and/or class-imbalanced data sets.

The organization of the paper is as follows. Section 2 gives a brief review on related researches
on multi-stage classifiers. Section 3 presents our pattern observation measurements: sparseness and class-imbalance. Section 4 describes the detail of the proposed gSVMT methodology and its training and testing algorithms. In Section 5, experimental results on 13 UCI data sets and an application of the proposed method to a face membership authentication problem are reported. Section 6 concludes the paper and presents future work.

II. RELEVANT RESEARCHES AND MOTIVATIONS

In this section, we first review some properties of the multi-stage approaches for classification and then discuss a certain group of modular classifiers which has enlightened our idea.

A. Modular Classifiers

Modular classifier, in general, is defined as the type of learning model whose decision making is conducted cooperatively by more than one module that operates on inputs with/without communication to other modules [20], [6], [9]. Unlike a single-stage classifier where input data are fed directly to the learning model, a modular classifier performs normally a first step of data observation and analysis. Further steps of data partitioning and decision making are conducted only when a certain criterion is satisfied. This criterion reflects the design philosophy of the employed modular approach. For example, for SVM ensemble, the diversity of component SVMs in the ensemble is a criterion. A new SVM is significant for ensemble only if it is diverse to other existing SVMs in the ensemble. Such diversity criterion that derives SVM ensemble with the better generalization performance than single SVM.

Despite the mentioned successful applications in various areas, multi-stage classifiers also have confronted some failures: Huang and Ling [7] reported that decision trees performs obviously no better than single-stage method Naive Bayes and SVM on nonuniform data classification in terms of the Receiver Operating Characteristics (ROC). Drucker et. al [3] had reported that an empowered decision tree, called boosting decision tree, is not as successful as a single SVM for email spam categorization. Also, Andrew Senior [21] had found that a standard decision tree can not achieve a satisfied performance on fingerprint classification, unless the decision tree is combined properly with a hidden Markov model (HMM). This raises a problem that how a multi-stage system can be better constructed over different case studies from real world applications.
Given the learning task presented in a set of objects associated with their labels, there are three possible sources of generalization errors for a multi-stage method: 1° selection of learning models based on domain knowledge or a preceding analysis of the input data; 2° the decomposition of the problem and the aggregation of the solutions of the subproblems; 3° decision rules obtained by individual component classifiers. For errors from 2°, most multi-stage classifiers impose cost functions to guide the decomposition procedure and use statistical analysis to reduce the aggregation error as much as possible. For errors from 3°, one may employ a powerful classifier, such as an SVM or a Neural Network, as the component classifier to build the local learners. However, errors from 1° have been overlooked by researchers in most previous works. In fact, analysis of input data and selection of learning models are the basis of all subsequent data processing, especially for multi-stage classifiers which are sensitive to data distribution.

This paper addresses the errors from 1°, focusing on two data characteristics that are common to see in real applications, namely the sparseness and class-imbalance. The sparseness of a data set indicates the capability of the data to be decomposed, without losing too much discriminant information presented in the data. The class-imbalance measures the property of a data set that the samples are not evenly distributed into the two classes (in the case of binary class dataset, the class with the smaller number of samples is called ‘skewed’ class, and the other class is called ‘overweighted’ class). In practice, the class-imbalance problem has led to degenerated classification performance for many single-stage approaches including SVM. This is because that most classifiers have take the assumption of a balanced prior probability, which is not true for real applications. This also gives us a very challenging difficulty to model a robust multi-stage learner in the case of imbalanced class distribution. To solve this problem, during module aggregating, a straightforward method is taken to allocate more individual modules on the skewed-class.

B. Related works of SVM-based Modular Classifiers

Our work can be viewed as a new attempt under the framework of SVM aggregating [14]. As an SVM based modular approach, SVM aggregating aims to create SVM-based multi-core cooperative computing techniques towards optimized problem-solving for computational intelligence. SVM ensemble learning [10] is a type of SVM aggregation in the principle of deriving a mixture of SVM experts to improve the generalization ability of single SVM decision
making. SVM ensemble learning assumes that the number of SVMs in aggregation should be known in advance as prior knowledge. In fact, this number is often difficult to determine in real applications. The SVM classification tree (SVMT) was first proposed in [15], in the context of face membership authentication application [16]. SVMT overcomes the difficulty of SVM ensemble learning by determining the number of SVMs automatically during the learning.

III. CHARACTERIZING THE DATA

In this section, we introduce two measurements to characterize the distribution of a data set. The data are assumed to be presented in a vectorial form.

A. Data Sparseness

we often conduct a divide-and-conquer on a dataset, but do not care about if the dataset is dividable (i.e. be partitioned without losing information for classification), and what if an improper data partition is performed? Due to this concern, it is interesting to see if we can define a measurement on the feasibility of a data set to be partitioned.

The sparseness describes the uniformity of a data distribution. From the view point divide-and-conquer, it also identifies the space on which the data distribution could be partitioned without much loss of discriminant information. Given a data set $X = \{x_i\}, i = 1, \cdots, N$, with the attribute set $\{a_1, \cdots, a_M\}$, we analyze the data distribution of $X$ along each attribute $a_i$. By partitioning the range $[\max(a_i), \min(a_i)]$ equally into $K$ unit intervals, a histogram $s_l(l = 1, \cdots, K)$ is calculated by counting the number of points dropping into each interval. Further, the sparseness can be calculated as the ratio of the geometric mean to the arithmetic mean of the list associated with the histogram:

$$S(X) = 1 - \frac{1}{\log 2} \left( \prod_{i=1}^M \frac{1}{N} \sum_{l=1}^N \frac{1}{\log(|s_l| + 1)} \right)^{\frac{1}{M}} .$$

Note that the above sparseness measures an averaged uniformity of the global data distribution, but it does not recognize any spareness caused by data clusters. Fig. 2 gives a comparison on the sparseness of two synthetic data sets with multiple gaussian distribution. Although Fig. 2 (a) and (b) have a comparable sparseness value, obviously the capability of the two data sets for data partitioning is different. With respect to labelled data sets for classification, it is suggested that the proposed sparseness be applied to each of the classes. The obtained class-sparseness reflects the suitability of this data set for the divide-and-conquer based modelling.
B. Class Imbalance

Class-imbalance is another problem which leads to degeneration of the generalization performance for single-stage classifiers including SVM. When class-imbalance occurs in a data set, the number of samples from one class is much larger than those of the other classes. In the case of binary classification, the class with the smaller number of samples is called ‘skewed’ class and
the other class is called ‘over-weighted’ class). In this paper, we measure the class-imbalance of a data set by computing the following class-imbalance ratio (CIR),

\[ C(\omega_1, \omega_2) = 1 - \frac{\min(|\omega_1|, |\omega_2|)}{\max(|\omega_1|, |\omega_2|)}. \]

(2)

where \( |\omega_i| \) is the cardinal number of class \( \omega_i \). Obviously, a larger \( C \) value shows a more imbalanced class distribution.

Single-stage classifiers including SVM often fail to classify the skewed class correctly [24], [23], [12], [22], since most classification algorithms assume a balanced prior probability [18]. To solve the problem caused by class-imbalance, data sampling method is a straightforward solution. Popular approaches include undersampling, which reduces the number of samples in the over-weighted class closer to the smaller class; oversampling, which repeatedly takes equal numbers of random subsamples from the two classes [1]; and resampling, which is a mixture of undersampling and oversampling [5], [25]. Algorithmic methods, e.g. adjusting decision class probability, imposing different penalty for different classes, [12], [22] modifying learning function to adjust the class boundary [24], [23], are also commonly used alternatives. Since class imbalance often occurs along with class-overlapping in many applications, the above approaches still have the following problem: improvement on the recognition rate of the skewed-class will generally leads to too much degeneration of the recognition rate on over-weighted class, especially when serious class-overlapping is involved.

Instead of a single-stage SVM with a complicated classification hyperplane, in this paper, the problem is modelled as an aggregation of a set of component SVM classifiers, e.g. linear SVMs or RBF kernel SVMs. Hence the class-imbalance problem can be treated easily by allocating more computing power, i.e. individual SVMs, for the classification of the skewed-class.

IV. THE PROPOSED gSVMT AGGREGATION METHOD

Towards a modular SVM aggregation classifier with high generalization ability, we address three aspects of problem-solving: 1° the convergence of an SVM classification tree (SVMT), 2° problem decomposition with the minimum discriminant knowledge loss, and 3° the creation of inter-module communication amongst individual SVMs for aggregation.
A. The Convergence of SVMT

Mathematically, a modular SVM system can be formulated as a composite structural model as follows: Given a 2-class data set \( X \) for classification, and a predefined data partitioning model \( P \) on \( X \), the data set \( X \) can be partitioned via an optimized \( P^* \) into \( L \) partitions \( \{g_1, g_2, \cdots, g_L\} \).

Then, an SVMT for binary classification is modelled as,

\[
T_{SVMT}(P, f_{SVM}^{<2>}, i = 1, \cdots, L),
\]

(3)

where \( f_{SVM}^{<2>} \) is a regional 2-class decision maker modelled as a standard alone SVM classifier. Parameter \( L \) is determined during the SVMT modelling process. Then we can have the decision function of SVMT \( \hat{f} \) as,

\[
\hat{f}(x) = f_{SVM}^{<2>} \text{ if } P(\{x\}) \in g_i.
\]

(4)

Obviously, errors may occur when \( \hat{f} \) is used to predict the class label of new coming samples, because of the difference between \( \hat{f} \) and the true classification function, \( f \). We use a real-valued loss function \( L = L(\hat{f}, f) \) to capture the extent of this error. The expectation of \( L \) is therefore data dependent:

\[
\hat{L} = \int |\hat{f}(x) - f(x)|p(x)dx = \int |\hat{f}(x) - y|p(x)dx,
\]

(5)

where \( y \) is the true class labels offered by the training set, and \( p(x) \) is the density function.

Given a data partitioning \( P(\{X\}) = \{g_1, \cdots, g_L\} \), the minimal \( \hat{L} \) can be obtained by training individual SVMs over all data partitions \( X_i \). The detailed proof can be found in Appendix.

B. Data Partitioning

Partitioning methods, such as K-means clustering, partition data normally on the basis of similarity. Yet in the spirit of searching for better individual partitions, we are aiming at partition data, which is to decompose data into partitions of different sizes, taking both input similarities and discriminant ability into consideration.

Assume the data are presented as vectors of attributes, \( X = \{x_1, \cdots, x_N\} \). Let \( A = \{a_1, \cdots, a_M\} \) be the attributes set, and \( Y = \{y_1, \cdots, y_N\} \) the class labels. The recursive data partitioning function \( P \) over \( G_0 = \{X\} \) is noted as

\[
P(G_0) = P^j(G_{J-1}, A_J, E_J) = \{g_{1}^{j}, \cdots, g_{L_j}^{j}\},
\]

(6)
where $G_j$ is the partition learned at iteration $J$ and $A_J$ the selected set of attributes on which the partitioning is performed. The number of partitions, $L_J$, and the number of iterations, $J$, are determined during the learning. $E_J$ is the cost function ensuring an optimal data partitioning, which will be made clear later.

The iteration process is formulated as follows.

**Step 1.** For each partition $g^J_j$, an optimal attribute $a^J_i$ can be picked from attribute set $A$, embodying a data partitioning with the minimal discriminant information loss for current subset $g^J_j$. In our implementation, we select $a^J_i$ as

$$a^J_i = \arg \min_{a_i \in A} \frac{|u_{i1} - u_{i2}|}{\sqrt{\frac{s_{i1}}{n_{i1}} + \frac{s_{i2}}{n_{i2}}}},$$

where $u_{ik}$ and $s_{ik}$ are the mean vector and variance of the samples in class $k$, respectively.

Along the selected attribute $a^J_i$, the data partitioning described in Eq.(6), noted as a data separation $T(a^J_i)$, is implemented by a crisp median-point split, or more properly by fuzzy histogram thresholding [2].

**Step 2.** For each subset $g^J_i$ obtained from the above grid data partitioning, a cost function $E^J_i$ is defined as

$$E(g^J_i) = \sum_{x_r \in g^J_i} (y_r - \sum_s (y_s^* \alpha_s^* K(x_r, x_s) + b^*)), \quad (8)$$

where $\sum_s y_s^* \alpha_s^* K(x, x_s) + b^*$ is the decision function of SVM and $K(\cdot, \cdot)$ is a user defined kernel function. Equation (8) is applied to estimate the applicability of a given data partition $g^J_i$ to act as a component classifier in SVMT. As discussed in Section II.A, this guarantees that the loss function of SVMT converges in the end.

**Step 3.** A threshold parameter $\xi$ is put on $E^J_i$ to see whether $g^J_i$ bears a class discriminability large enough to be treated as a component classifier. Hence, every data partition obtained with its $E^J_i$ less than $\xi$ are extracted out, and the remaining data are forwarded to the next iteration of data partitioning.

C. SVMs Aggregating over Grid

From Eq. (6), we can obtain a collection of subsets via $J$ iterations of recursive partitioning. For regional decision making, a stand alone SVM is trained over each subset to build a 2-class classifier $f_{SVM}^{<2>}$ . Towards final global decision making, these local SVM classifiers are
aggregated onto a grid, whose columns and rows correspond to the iteration of data partitioning and selected attributes by Eq. (7), respectively.

Specifically, given that \( q_j \) attributes are selected by Eq. (7) at iteration \( j \), and totally \( Q (Q \leq M) \) attributes are selected for \( J \) iterations. Then all data partitions extracted by Eq.(6), together with the corresponding partitioning functions or regional SVM classifiers, composes a composite model \( \Omega \) as

\[
\Omega = \left\{ \begin{array}{c}
\{V_{11}\} \\
\{V_{21}, V_{22}\} \\
\{V_{31} \cdots V_{3q_1}\} \\
\vdots \\
\{V_{ji} \cdots V_{jq_j}\} \\
\vdots \\
\{V_{J1} \cdots V_{Jq_J}\}
\end{array} \right.
\]

(9)

where \( \{V_{ji}\} \) denotes a group of functions that use attribute \( a_i \) for data partitioning at the \( j \)th iteration. The functions could be either regional SVM classifiers \( f_{SVM_i}^{<2>} \) or regional data partitioning functions \( T(a_j^i) \). \( \Omega \) models a \( Q \)-dimensional grid which partitions the data into \( L \) grids, with the discriminant information in the dataset well preserved in the subsets.

Thus, over the grid learned from data, an SVM tree is constructed by building connections between every two lines of composite model \( \Omega \).

D. Coping with Class-Imbalance

During the aggregation, the class-imbalance is addressed in the following two parallel ways.

The first approach implements a recursive data zooming-in schema. Data size with comparatively large sample sizes, especially for the subsets from the over-weighted class, are extracted out in the first few rounds. As a result, the class-imbalance is alleviated for the remaining data. Data with class-overlapping/class-mixture are left for the next step partitioning. Hence, as we drilled down in SVMT, decisions are made only at an appropriate scale where the decision is reachable.

In the second approach, the skewed-class is protected by SVMT via turning SVM loss function \( E \) in Eq. (8) to the skewed class as,

\[
L_{SVM} = \frac{1}{n} \sum_{r=1}^{n} (y_r - \sum_i (y_s \alpha_s^* K(\mathbf{x}, \mathbf{x}_s) + b^*)) 
\]

(10)
where \( n \) is the sample size of the regional skewed class, and \( x \) a sample from the class.

Note that the regional skewed class represents the skewed-class within a data partition. In other words, a regional skewed class also could be a globally over-weighted class, if its class distributes are skewed regionally. Hence, Eq. (10) does no overbalanced adjustment as most data resampling methods do to favor the skewed class by using the skewed-class data repeatedly.

In fact, the regional skewed class loss function Eq. (10) protects equally both the skewed class and the over-weighted class from being underestimated in the regional decision functions and thus helps to overcome the class-imbalance. In practice, Eq. (10) eventually favors the skewed class, because the probability for the globally skewed class to be regionally skewed is often much higher than that of the globally over-weighted class.

**E. The Proposed gSVMT Algorithm**

To build a gSVMT, Eq. (6) is implemented recursively in two steps. First, the grid partitioning is employed to split the data recursively. During this procedure, data partitions with a cost function \( E_i^j \) smaller than threshold \( \xi \) are extracted out. For each of these partitions, an SVM classifier is built and associated with a leaf node of the gSVMT structure. To mitigate the overfitting of decision tree learning, gSVMT merges neighboring SVM nodes at the level of terminal node, uses only merged SVM node for final decision making.

To test a constructed gSVMT, an input sample \( x \) is first judged by the test function \( T(x) \) at the root node of the gSVMT. Depending on the decision made by the root node, \( x \) will be redirected to one of the children of the root node. This procedure is repeated until a leaf node is reached. The final prediction \( \hat{f}(x) \) is made by the SVM classifier associated with the leaf node. Algorithm 2 presents the testing procedure of a gSVMT.

In the above algorithms, \( K \) specifies the type of SVM used in gSVMT construction. \( \xi \) is the permitted minimal loss function value, which gives a criterion to select partitions which can be well modelled by a stand alone SVM classifier. The default value for \( \xi \) is zero. However, in practice, a small positive value near zero should be given. Obviously, the smaller \( \xi \), the more time cost to train the gSVMT.
Algorithm 1. gSVMT Training

V. EXPERIMENTS AND DISCUSSION

In this section, the experimental results on a synthetic data, 13 UCI Machine Learning benchmark data sets and a facial membership authentication application are reported.
Function: gSVMT testing

Input:
T; /* a constructed gSVMT */
x; /* a testing instance */

Output:
C; /* output classifier */

Variables:
Current; /* current node number of T */
Next; /* the node next to Current node */
T_i; /* The test function on node i */

Function Calls:
SVMTest(x, K); /* trained SVM classifier */
SgnTest(x); /* trained one-class classifier */
SearchNode(T, i); /* search the node next to i in T */

begin
  1 Current ← 0;
  2 while (SearchNode(T, Current) == Null)
  3     Next ← SearchNode(T, Current);
  4     Current ← Next;
  5 if T_Current is 2-class
  6     C ← SVMTest(x, K, Current);
  7 else
  8     C ← SgnTest(x, Current);
  9 return C;
end

Algorithm 2. gSVMT Testing

A. Synthetic Dataset

We first experimented the proposed gSVMT with a synthetic Chess data set. The data distribution is a 3x3 chessboard formed by a mixture of numbers of 2D ([X1 X2]) Gaussian distributions. The number of Class 2 samples is approximately 1/5 of Class 1, thus the class-imbalance of the dataset is 0.75. The sparseness by Eq. (1) is 0.9925 for the whole data set. The t-test value by Eq. (7) is 0.8973 for X1, and 0.7834 for X2. The t-test value of X1 being lower than that of X2.
Fig. 3. (a) The data distribution of the synthetic Chess data. (b) the produced grid by the proposed gSVMT algorithm, (c) the classification boundary of the gSVMT with a comparison to single SVM

indicates that the class separability from X2 is more significant than that from X1. In another words, X1 is the sparseness axis of the 2D space.

When gSVMT is constructed over the dataset of Fig. 3 (a), the data is zoomed in recursively by a recursive data partitioning described in Section IV-B, where every step of data partitioning is being supervised by an SVM. As a result, all steps of data partitioning form a grid presented in Fig. 3 (b), where each bar of the grid is the boundary of one step data partitioning, the the entire grid represents the footprint of the gSVMT data zooming-in. As seen from the grid, the grid bar appears irregularly along X1 or X2, means that the spareness of a regional data partition varies dynamically as the data is being zoomed-in by gSVMT. In general, the number of bar on X1 is seen about two times of those on X2. This indicates that gSVMT is more likely to partition data on X1, which is the sparseness axis.

About classification generalization ability, Fig. 3 (c) compares the proposed gSVMT and
single SVM on the classification hyper-plane learned from the Fig. 3 (a) data. As seen from the figure, while the SVM boundary loses the track of chessboard grid for some subregions, the gSVMT boundary surprisingly gives conversely nearly perfect matches with the chessboard grid. It is noticeable that such superiority occurs only at some subregions (i.e. not globally at every subregion), but the gSVMT aggregates those pieces of regionally optimal boundaries and achieves an overall better chessboard approximation.

Fig. 4 gives an example of the gSVMT generated by Algorithm 1. As can be seen, the gSVMT is a binary tree with ‘P0’ as the root node. Every ellipse here represents an SVM node, and every internal node is a testing function for data partitioning. The size of ellipse identifies the size of training dataset over the SVM node.

![Example of gSVMT generated by Algorithm 1 over Fig. 3 (a) dataset.](image)

**B. UCI Datasets**

We test the proposed gSVMT algorithm on 13 UCI data sets. Table 1 reports the error rates obtained by the proposed gSVMT method with a comparison against previously reported results on SVM classifiers [13]. We divide the 13 data sets into two categories according to the sparseness value. Six data sets with the sparseness greater than 0.3 are taken as sparse data sets, and the remaining 7 data sets are counted as non-sparse data sets.
As shown in the table, the proposed gSVMT matches SVMs on 5 out of 6 sparse data sets when proper RBF kernels are employed. It is surprising that gSVMTs wins SVM over 15% on the classification accuracy of 2 non-sparse data sets, Flare-solar and Titanic. As we exploring the distribution characteristics of these two data sets in their feature space, it is found that both of them have some discrete attributes. In other words, both two data sets are a type of discrete data set, which is a special case of sparseness covered by our sparseness definition. The extraordinary performance on these two data sets suggests that the proposed gSVMT is very efficient on the classification of discrete data sets. The proposed gSVMT performs not as well as single SVMs on those non-sparse data sets, this possibly because that the grid partitioning of gSVMT has destroyed too much information useful for classification.

C. Face Membership Authentication Application

To test gSVMTs on problems where dynamic class-imbalance and class-overlapping are prevalent, we have also applied the algorithm to the face membership authentication (FMA) problem [17], [16], [15]. The objective of an FMA system is to distinguish the membership
class from the non-membership class in a whole group, where the size of membership is usually smaller than that of non-membership group. The original definition of MA problem can be found in [15].

FMA is counted as a problem involving sparse data, because the similarity within and amongst the class is rather low, and facial appearance of one person may look very different from another one, despite that they both belong to the same membership group. In another words, face samples from one person may distribute far away from samples from another person in the feature space. On the other hand, FMA is also a challenging problem with severe class-imbalance: the size of membership group is typically much smaller than that of non-membership, and such class-imbalance varies dynamically over the time in real applications.

![Fig. 5. The illustration of FMA of 5 members against 100 non-members](image)

In a 271 persons environment, where 5 different face images are taken for each person, we extracted face feature using a principle component analysis (PCA), and selected 100 top energy eigenfeatures for our FMA experiments. The calculated sparseness by our definition is 0.6605. Although this value does not really reflect the sparseness of the dataset, the capability of the dataset for data partitioning is obvious. Fig. 5 presents a feature space for the FMA of 5 members against 100 non-members. As we can imagine that a complicated classification boundary is required for distinguishing these 5 members all together from the rest of 100 non-
members. However, such classification difficulty has been reduced largely in Fig. 5, where 5 members could be individually authenticated by a separated, but much simpler classification boundary. This indicates that the capability of the FMA problem to be decomposed then solved is large.

To test the robustness of gSVMT to class imbalance, we formulate a sequence of FMA problems with the membership group size increasing from 10 ($C_1 = 0.962$) to 120 ($C_\infty = 0.205$), with a step size of 10 persons. Here, as an extreme case of class-imbalance, results for the member group size of 135 ($C_\infty = 0.007$) are also reported. Note that the sparseness for the whole data set does not change over time, however, that of the membership class data is decreasing as shown in Fig. 6.

![Graph showing the sparseness variation for membership class data under the condition of membership group size increasing from 10 to 135.](image)

Fig. 6. The sparseness variation for membership class data under the condition of membership group size increasing from 10 to 135.

In particular, when the size of the membership group is 10, it indicates that 10 members scatter randomly in the feature space among the remaining 261 non-members. In this case, although the sparseness for the whole data set is the same as before, the sparseness within membership group reaches the maximum because the within-class similarity of the 10 members is less than cases with more than 10 members. Hence, membership group data distribution shows a large sparseness value.

Fig. 7 gives a comparison of output member authentication rate (i.e. the classification rate of
the skewed class) versus input imbalance (i.e. the class imbalance from training data distribution). Four curves has recorded how four classifiers under comparison is performed under the variation of the class-imbalance from training data distribution, respectively. As we can see that, single SVMs get a large variation on the classification of the skewed class (i.e. member), while the membership group size is growing from a small group size 10 to the maximum 135. However, the proposed gSVMT shows a strong robustness to such class-imbalance variation in that gSVMT are shown keeping consistently at a level of 80% member autointoxication rate under all class-imbalance situations.

Especially for cases where the sparseness is large, it is extremely difficult for ordinary SVMs to find a complicated but continuous decision rule to distinguish a small number of members accurately from a large number of non-members. On the contrary, the proposed gSVMT has shown an very good performance on the authentication of such small membership groups. It is distinct that the sparseness of the data set here is an advantage for the proposed gSVMT, but it is certainly a disadvantage for ordinary SVMs.

![Graph showing classification rates](image)

Fig. 7. The proposed gSVMT is compared with 3 single SVMs on FMA

VI. CONCLUSION AND FUTURES WORK

Addressing the problem caused by the sparsity and class-imbalance of data distribution, this paper proposes a new type of SVM-based modular method. The properties of the proposed
gSVMT are summarized as follows. First, gSVMT keeps every iteration of data partitioning to be performed only at the sparse region of the data, minimizing the loss of discriminant information during the SVMT construction. Second, gSVMT addresses the class-imbalance problem by adjusting the partitioning cost function on the skewed-class, so that the skewed-class is emphasized by every SVM node in gSVMT. The proposed gSVMT is experimented on 13 UCI data sets and a real Facial Membership Authentication application which is characterize by the dynamic class-imbalance and class-sparseness property. The experimental results demonstrated that, compared with stand alone SVMs, the proposed gSVMT is able to treat with the sparseness and class-imbalance of the input data and delivers competitive classification performance.

From the viewpoint of SVM aggregating intelligence [14], the proposed gSVMT follows the same route of recursive data partitioning plus aggregating as the previous SVM tree methods. The gSVMT is different from all previous SVMT aggregating methods [15] because 1º grid partitioning is used for problem decomposition; 2º every internal node of the gSVMT is a data partitioning function, and every leaf node is an SVM node. Note that gSVMT has no one-class node, and decisions are made by the leaf SVM node.

The proposed gSVMT is merely sustainable for 2-class problems. It can be simply extended to m-class problems by learning m binary classifiers for each class versus the others, or by learning m(m – 1) binary classifiers for each class versus another class. This topic is generally discussed under the framework of SVM decision forests [8]. Alternatively, in the principle of SVM aggregating, a multi-class gSVMT (m-gSVMT) can be constructed by decomposing a m-class task into a certain number of 2 to m classes subtasks as

\[ T_{m-gSVMT}(P^n, f_{SVM}^{<2>}, \ldots, f_{SVM}^{<m>}, x), \]

where \( P^n \) represents a partitioning function which is able to split a set into multiple subsets. Similar to 2-gSVMT, \( f_{SVM}^{<i>} \) is an i-class (2 \( \leq \) i \( \leq \) m) local SVM classifier. Accordingly, the classification function of a m-gSVMT can be written as,

\[ \hat{f}(x) = f_{SVM}^{<i>} \text{ if } P^n(x) \in g_j. \]

As a measurement of the sparsity of a data set, spareness is an important concept for divide-and-conquer data modelling. However, it is also a concept difficult to give an accurate and consistent evaluation, just as has been reflected in our work. Still, sparsity does exist for most
life data, thus for better generalization ability of gSVMT, it will be an interesting future work to derive a new sparseness measurement consistent for most data sets.

APPENDIX

**Theorem 1** Given a data partitioning $\mathcal{P}(\{X\})$ for constructing SVMT $\hat{f}$, the loss function $\mathcal{L}$ of SVMT is converged when every SVM in the SVMT is trained regionally.

**Proof.** The constructed SVMT $\hat{f}$ may differs from the truth classification function $f$, thus the SVM loss function $\mathcal{L} = \mathcal{L}(\hat{f}, f)$ is therefore data dependent:

$$\mathcal{L} = |\hat{f} - y|$$  \hspace{1cm} (13)

where $y = f(x)$ represents the true classification value. As $\hat{f}$ is applied to datasets drawn from the whole data $D$ under a distribution of $g$. The expected loss can be quantified as,

$$E[\mathcal{L}] = \int |\hat{f} - y|g(D)dD,$$  \hspace{1cm} (14)

where $y$ can be fixed as the class label of the training dataset $D$.

Substituting Eq.(3) to $\mathcal{L}$ calculation, the above SVMT loss function becomes:

$$\mathcal{L} = \sum_{i=1}^{I} |f_{Svmi} - f_i|\theta_{2i} + \varepsilon_P$$  \hspace{1cm} (15)

where $I$ is the number of regional SVMs in the constructed SVMT. $f_i$ is the local true values of $f$. $\theta_{2i}$ represents the distribution probability of the $i$th partition that contains both class 1 and class 2 data. $\varepsilon_P$ is error from partitioning function $\mathcal{P}$.

The $\varepsilon_P$ is a constant, as the partitioning function $\mathcal{P}(\{X\})$ is given beforehand. Thus, $\mathcal{L}$ is determined completely by reginal SVMs $f_{Svmi}$, $i = 1, \ldots, I$. Provided every SVM in the SVMT with the same kernel and penalty parameters, the loss function $\mathcal{L}$ of SVMT is converged only when $f_{Svmi} - f_i$ is minimized (i.e. SVM is trained regionally).

REFERENCES


Appendix: A

International Workshop on Data Mining and Artificial Intelligence
In conjunction with
11th IEEE International Conference on Computer and Information Technology
December 24-27, 2008, Khulna, Bangladesh
gSVMT: Aggregating SVMs over a Dynamic Grid
Learned from Data

Shaoning Pang, Tao Ban, Youki Kadobayashi, and Nik Kasabov

Abstract—Addressing the problem of adaptively modelling a classifier as a modular system, a new type of SVM aggregating method termed gridding SVM Tree (gSVMT) is proposed in this paper. The proposed gSVMT achieves to discover data subregions with principal discriminant knowledge through a recursive SVM-supervised data partitioning procedure. For each subregion, an individual SVM is allocated to extract the subregion knowledge. A set of such SVMs are aggregated in a specific order, resulting in a globally reliable decision rule to predict new coming samples. Experiments on a synthetic Gaussian data set and 13 benchmark machine learning data sets, have highlighted the usability of the gSVMT on its competitive classification capability. In particular, the proposed gSVMT is found to have better generalization performance than SVM classifiers for data sets with high sparseness and/or class-imbalance. Its performance has been further demonstrated with the successful real application on a face membership authentication system.

I. INTRODUCTION

W hen faced with a new classification problem, one often considers, for the first instance, to employ one of the available discriminative classifiers such as the well known \(k\)-nearest neighbor classifier, feed-forward neural networks, or support vector machines (SVMs) to solve the problem swiftly in a single-stage mode. But more often, when complex applications are confronted, just as in most real-world applications, the problem could come into a scale too large and a nature too complicated for a single-stage model to be solved efficiently in terms of the convergence time in the training phase and the response time in the testing phase. In such a case, one may fall back on data preprocessing techniques, e.g. feature extraction and feature selection, to reducing the difficulty of the problem, or alternatively turn to multi-stage approaches. The idea to observe, measure, and break the learning problem into a collection of simpler subproblems and then summarize solutions of the subproblems to yield a global decision, is generally known as the ‘divide-and-conquer’ methodology. In this paper, to make difference between a single-step classifier, we call a classification method adopting decision tree classifiers [19] and modular classifiers [20], e.g. SVM ensemble [10], and SVM classification tree (SVMT) [15].

Profiting from the divide-and-conquer strategy, the advantages of a multi-stage classifier over a single-stage classifier include quick convergence, parallel training, and better generalization. In the literature, multi-stage classifiers have been successfully applied in a diversity of areas including character recognition, remote sensing, medical diagnosis, expert systems, and speech recognition [4], [6]. However, given a new data set, there is no existing guideline on how to choose an appropriate multi-stage classifier that can well suit the specific problem with a preferable performance guaranteed. Thus, one often confronts the embarrassment to implement and evaluate multiple methods on the data set. This procedure is usually frustrating because it is not only time and labor demanding, but sometimes could be resultless as well, considering the case of a dataset with a large scale that rendering some of the methods noneffective.

In this paper, we focus our discussion on the usability and construction of a novel multi-stage classifier. The contribution of the paper is twofold. First, we introduce two measurements characterizing the distribution of the data set, namely the sparseness and the class-imbalance. They help to identify whether the dataset can be fitted well by a multi-stage approach so that a generalization performance better than a state-of-the-art single-stage classifier can be expected. Hence, via a preceding exploratory analysis step, we can learn whether experiments on multi-stage classifiers are necessary. Second, we present an SVM based modular classifier termed gridding SVM Tree (gSVMT). The name of gSVMT comes from the fact that the component SVM classifiers are aggregated over a grid learned from data. gSVMT implements a ‘divide-and-conquer’ strategy by first performing data partitioning at the sparse region of the data distribution, and then conducting decision-making through aggregation of the collection of component SVM classifiers. Since each component SVM classifier not only explores the discriminant knowledge but imposes a balancing requirement in the subregion as well, gSVMT demonstrates a strong capability on sparse and/or class-imbalanced data sets.

The organization of the paper is as follows. Section II gives a brief review on related researches on multi-stage classifiers. Section III presents our pattern observation measurements: sparseness and class-imbalance. Section IV describes the detail of the proposed gSVMT methodology and its training and testing algorithms. In Section V, experimental results on 13 UCI data sets and an application of the proposed method to a face membership authentication problem are reported. Section VI concludes the paper.
II. RELATED RESEARCHES

In this section, we first review some properties of the multi-stage approaches for classification and then discuss a certain group of modular classifiers which has enlightened our idea.

A. Problem of Classifiers

A multi-stage classifier is generally defined as a learning model where the computation performed by the classifier is decomposed into more than one module that operates on inputs with/without communicating with each other [20], [6], [9]. Different from a single-stage classifier where input data are directly fed to the learning model, a multi-stage classifier normally performs a first step of data observation and analysis. Further steps of data partitioning and decision making are conducted only when a certain criterion is satisfied. This criterion reflects the design philosophy of the employed multi-stage approach.

Despite the mentioned successful applications in various areas, multi-stage classifiers also have confronted some failures: Huang and Ling [7] reported that decision trees performs obviously no better than single-stage method Naive Bayes and SVM on nonuniform data classification in terms of the Receiver Operating Characteristics (ROC). Drucker et. al [3] had reported that an empowered decision tree, called boosting decision tree, is not as successful as a single SVM for email spam categorization. Also, Andrew Senior [21] had found that a standard decision tree can not achieve a satisfied performance on fingerprint classification, unless the decision tree is combined properly with a hidden Markov model (HMM). This raises a problem that how a multi-stage system can be better constructed over different case studies from real world applications.

Given the learning task presented in a set of objects associated with their labels, there are three possible sources of generalization errors for a multi-stage method: 1° selection of learning models based on domain knowledge or a preceding analysis of the input data; 2° the decomposition of the problem and the aggregation of the solutions of the subproblems; 3° decision rules obtained by individual component classifiers. For errors from 2°, most multi-stage classifiers impose cost functions to guide the decomposition procedure and use statistical analysis to reduce the aggregation error as much as possible. For errors from 3°, one may employ a powerful classifier, such as an SVM or a Neural Network, as the component classifier to build the local learners. However, errors from 1° have been overlooked by researchers in most previous works. In fact, analysis of input data and selection of learning models are the basis of all subsequent data processing, especially for multi-stage classifiers which are sensitive to data distribution.

To address the errors from 1°, this paper focuses on two data characteristics that are common to see in real applications, namely the sparseness and class-imbalance. The sparseness of a data set indicates the capability of the data to be decomposed, without losing too much discriminant information presented in the data. The class-imbalance measures the property of a data set that the samples are not evenly distributed into the two classes.

In practice, the class-imbalance problem has led to degenerated classification performance for many single-stage approaches including SVM. This is because that most classifiers have take the assumption of a balanced prior probability, which is not true for real applications. This also gives us a very challenging difficulty to model a robust multi-stage learner in the case of imbalanced class distribution. To solve this problem, during module aggregating, a we take a straightforward method to allocate more individual modules on the skewed-class.

B. Related Works

Our work can be viewed as a new attempt under the framework of SVM aggregating [14]. As an SVM based modular approach, SVM aggregating aims to create SVM-based multi-core cooperative computing techniques towards optimized problem-solving for computational intelligence. SVM ensemble learning [10] is a type of SVM aggregation in the principle of deriving a mixture of SVM experts to improve the generalization ability of single SVM decision making. SVM ensemble learning assumes that the number of SVMs in aggregation should be known in advance as prior knowledge. In fact, this number is often difficult to determine in real applications. The SVM classification tree (SVMT) was first proposed in [15], in the context of face membership authentication application [16]. SVMT overcomes the difficulty of SVM ensemble learning by determining the number of SVMs automatically during the learning.

III. CHARACTERIZING THE DATA

In this section, we introduce two measurements to characterize the distribution of a data set. The data are assumed to be presented in a vectorial form.

A. Sparseness

We often conduct a divide-and-conquer on a data set without caring about whether the data is dividable, i.e. be divided without losing information for classification. What if an improper data partition is performed? Due to this concern, it is interesting to see if we can define a measurement on the feasibility of a data set to be partitioned.

The sparseness is defined to describe the uniformity of a data distribution. From the view point of divide-and-conquer, it also identifies the position where the data could be partitioned, without much loss of discriminant information. Given a data set \( X = \{x_i\}, i = 1, \ldots, N \), with the attribute set \( \{a_1, \ldots, a_M\} \), we analyze the data distribution of \( X \) along each attribute \( a_i \). By partitioning the range \([\max(a_i), \min(a_i)]\) equally into \( K \) unit intervals, a histogram \( s_i(l = 1, \ldots, K) \) is calculated by counting the number of points dropping into each interval. Further, the sparseness can be calculated as the ratio of the geometric mean to the arithmetic mean of the list.
Appendix: B

is called ‘skewed’ class and the other class is called ‘over-weighted’ class). In this paper, we measure the class-imbalance of a data set by computing the following class-imbalance ratio (CIR),

$$C(\omega_1, \omega_2) = 1 - \frac{\min(|\omega_1|, |\omega_2|)}{\max(|\omega_1|, |\omega_2|)}.$$  (2)

where $|\omega_i|$ is the cardinal number of class $\omega_i$. Obviously, a larger $C$ value shows a more imbalanced class distribution.

Single-stage classifiers including SVM often fail to classify the skewed class correctly [24], [23], [12], [22], since most classification algorithms assume a balanced prior probability [18]. To solve the problem caused by class-imbalance, data sampling method is a straightforward solution. Popular approaches include undersampling, which reduces the number of samples in the over-weighted class closer to the smaller class; oversampling, which repeatedly takes equal numbers of random subsamples from the two classes [1]; and resampling, which is a mixture of undersampling and oversampling [5], [25]. Algorithmic methods, e.g. adjusting decision class probability, imposing different penalty for different classes, [12], [22] modifying learning function to adjust the class boundary [24], [23], are also commonly used alternatives. Since class imbalance often occurs along with class-overlapping in many applications, the above approaches still have the following problem: improvement on the recognition rate of the skewed-class will generally leads to too much degeneration of the recognition rate on over-weighted class, especially when serious class-overlapping is involved.

Instead of a single-stage SVM with a complicated classification hyperplane, in this paper, the problem is modelled as an aggregation of a set of component SVM classifiers, e.g. linear SVMs or RBF kernel SVMs. Hence the class-imbalance problem can be treated easily by allocating more computing power, i.e. individual SVMs, for the classification of the skewed-class.

IV. THE PROPOSED GSVMT AGGREGATION METHOD

Towards a modular SVM aggregation classifier with high generalization ability, we address three aspects of problem-solving: 1° the convergence of an SVM classification tree (SVMT), 2° problem decomposition with the minimum discriminant knowledge loss, and 3° the creation of inter-module communication amongst individual SVMs for aggregation.

A. The Convergence of SVMT

Mathematically, a modular SVM can be formulated as a composite structural model as follows: Given a 2-class data set $X$ for classification, and a predefined data partitioning model $P$ on $X$, the data set $X$ can be partitioned via an optimized $P^*$ into $L$ partitions $\{g_1, g_2, \ldots, g_L\}$.

Then, an SVMT for binary classification is modelled as

$$T_{SVMT}(P, f_{SVM<2>}), i = 1, \ldots, L,$$  (3)

where $f_{SVM<2>}$ is a regional 2-class decision maker modelled as a standard alone SVM classifier. Parameter $L$ is determined.
during the SVMT modelling process. Then we can have the decision function of SVM \( f \) as,
\[
\hat{f}(x) = f_{\text{SVM}, c>2} \text{ if } P(\{x\}) \in g_i.
\] (4)

Obviously, errors may occur when \( \hat{f} \) is used to predict the class label of new coming samples, because of the difference between \( \hat{f} \) and the true classification function, \( f \). We use a real-valued loss function \( L = L(\hat{f}, f) \) to capture the extent of this error. The expectation of \( L \) is therefore data dependent:
\[
\hat{L} = \int \hat{f}(x) - f(x)\, p(x) \, dx = \int \hat{f}(x) - y\, p(x) \, dx,
\] (5)
where \( y \) is the true class labels offered by the training set, and \( p(x) \) is the density function.

Given a data partitioning \( P(\{X\}) = \{g_1, \ldots, g_L\} \), the minimal \( \hat{L} \) can be obtained by training individual SVMs over all data partitions \( X_i \) (see proof in Appendix).

B. Data Partitioning

Partitioning methods, such as K-means clustering, partition data normally on the basis of similarity. Yet in the spirit of searching for better individual partitions, we are aiming at partition data, which is to decompose data into partitions of different sizes, taking both input similarities and discriminant ability into consideration.

Assume the data are presented as vectors of attributes, \( X = \{x_1, \ldots, x_N\} \). Let \( A = \{a_1, \ldots, a_M\} \) be the attributes set, and \( Y = \{y_1, \ldots, y_N\} \) the class labels. The recursive data partitioning function \( P \) over \( G_0 = \{X\} \) is noted as
\[
P(G_0) = P^J(G_{j-1}, A_j, E_j) = \{g^J_1, \ldots, g^J_{L_j}\},
\] (6)
where \( G_j \) is the partition learned at iteration \( j \) and \( A_j \) the selected set of attributes on which the partitioning is performed. The number of partitions, \( L_j \), and the number of iterations, \( J \), are determined during the learning. \( E_j \) is the cost function ensuring an optimal data partitioning, which will be made clear later.

The iteration process is formulated as follows.

**Step 1.** For each partition \( g^j_i \), an optimal attribute \( a^j_i \) can be picked from attribute set \( A \), embodying a data partitioning with the minimal discriminant information loss for current subset \( g^j_i \). In our implementation, we select \( a^j_i \) as
\[
a^j_i = \arg \min_{a_i \in A} |u_{i1} - u_{i2}| \sqrt{\frac{s_{i1}^2}{n_{i1}^2} + \frac{s_{i2}^2}{n_{i2}^2}},
\] (7)
where \( u_{ik} \) and \( s_{ik} \) are the mean vector and variance of the samples in class \( k \), respectively.

Along the selected attribute \( a^j_i \), the data partitioning described in Eq.(6), noted as a data separation \( T(a^j_i) \), is implemented by a crisp median-point split, or more properly by fuzzy histogram thresholding [2].

**Step 2.** For each subset \( g^j_i \) obtained from the above grid data partitioning, a cost function \( E^j_i \) is defined as
\[
E(g^j_i) = \sum_{x_r \in g^j_i} (y_r - \sum_s (y_s \alpha^*_s K(x_r, x_s) + b^*))\tag{8}
\]
where \( \sum_s y_s \alpha^*_s K(x_r, x_s) + b^* \) is the decision function of SVM and \( K(\cdot, \cdot) \) is a user defined kernel function. Equation (8) is applied to estimate the applicability of a given data partition \( g^j_i \) to act as a component classifier in SVMT. As discussed in Section II.A, this guarantees that the loss function of SVMT converges in the end.

**Step 3.** A threshold parameter \( \xi \) is put on \( E^j_i \) to see whether \( g^j_i \) bears a class discriminability large enough to be treated as a component classifier. Hence, every data partition obtained with its \( E^j_i \) less than \( \xi \) are extracted out, and the remaining data are forwarded to the next iteration of data partitioning.

C. SVMs Aggregating over Grid

From Eq. (6), we can obtain a collection of subsets via \( J \) iterations of recursive partitioning. For regional decision making, a stand alone SVM is trained over each subset to build a 2-class classifier \( f_{\text{SVM}, c>2} \). Towards final global decision making, these local SVM classifiers are aggregated onto a grid, whose columns and rows correspond to the iteration of data partitioning and selected attributes by Eq. (7), respectively.

Specifically, given that \( a_j \) attributes are selected by Eq. (7) at iteration \( j \), and totally \( Q(Q \leq M) \) attributes are selected for \( J \) iterations. Then all data partitions extracted by Eq.(6), together with the corresponding partitioning functions or regional SVM classifiers, composes a composite model \( \Omega \) as
\[
c = \begin{pmatrix}
\{V_{11}\} & \{V_{12}\} & \{V_{13}\} \\
\{V_{21}\} & \{V_{22}\} & \{V_{23}\} \\
\vdots & \vdots & \vdots \\
\{V_{J1}\} & \cdots & \{V_{JQ}\} \\
\end{pmatrix}
\] (9)
where \( \{V_{ij}\} \) denotes a group of functions that use attribute \( a_i \) for data partitioning at the \( j \)th iteration. The functions could be either regional SVM classifiers \( f_{\text{SVM}, c>2} \) or regional data partitioning functions \( T(a^j_i) \). \( \Omega \) models a \( Q \)-dimensional grid which partitions the data into \( L_j \) grids, with the discriminant information in the dataset well preserved in the subsets.

Thus, over the grid learned from data, an SVM tree is constructed by building connections between every two lines of composite model \( \Omega \).

D. Coping with Class-Imbalance

During the aggregation, the class-imbalance is addressed in the following two parallel ways.

The first approach implements a recursive data zooming-in schema. Data size with comparatively large sample sizes, especially for the subsets from the over-weighted class, are extracted out in the first few rounds. As a result, the class-imbalance is alleviated for the remaining data. Data with class-overlapping/class-mixture are left for the next step partitioning. Hence, as we drilled down in SVMT, decisions are made only at an appropriate scale where the decision is reachable.
Appendix: B

In the second approach, the skewed-class is protected by SVM via turning SVM loss function $E$ in Eq. (8) to the skewed class as,

$$L_{\text{SVM}} = \frac{1}{n} \sum_{r=1}^{n} (y_r - \sum_{i} (y_s \alpha_s^* K(x, x_s) + b^*)),$$

(10)

where $n$ is the sample size of the regional skewed class, and $x$ a sample from the class.

Note that the regional skewed class represents the skewed-class within a data partition. In other words, a regional skewed class also could be a globally over-weighted class, if its class distributes are skewed regionally. Hence, Eq. (10) does no overbalanced adjustment as most data resampling methods do to favor the skewed class by using the skewed-class data repeatedly. In fact, the regional skewed class loss function Eq. (10) protects equally both the skewed class and the over-weighted class from being underestimated in the regional decision functions and thus helps to overcome the class-imbalance. In practice, Eq. (10) eventually favors the skewed class, because the probability for the globally skewed class to be regionally skewed is often much higher than that of the globally over-weighted class.

E. The Proposed gSVMT Algorithm

To build a gSVMT, Eq. (6) is implemented recursively in two steps. First, the grid partitioning is employed to split the data recursively. During this procedure, data partitions with a cost function $E_{\text{grid}}$ smaller than threshold $\xi$ are extracted out. For each of these partitions, an SVM classifier is built and associated with a leaf node of the gSVMT structure.

To test a constructed gSVMT, an input sample $x$ is first judged by the test function $T(x)$ at the root node of the gSVMT. Depending on the decision made by the root node, $x$ will be redirected to one of the children of the root node. This procedure is repeated until a leaf node is reached. The final prediction $\hat{f}(x)$ is made by the SVM classifier associated with the leaf node. Algorithm 2 presents the testing procedure of a gSVMT.

In the above algorithms, $K$ specifies the type of SVM used in gSVMT construction. $\xi$ is the permitted minimal loss function value, which gives a criterion to select partitions which can be well modelled by a stand alone SVM classifier. The default value for $\xi$ is zero. However, in practice, a small positive value near zero should be given. Obviously, the smaller $\xi$, the more time cost to train the gSVMT. Fig. 3 gives an example of gSVMT. As can be seen, gSVMT is a binary tree with ‘P0’ as the root node. Every ellipse here represents an SVM node, and every internal node is a testing function for data partitioning. The size of an ellipse identifies the size of training data set associated with the node.

V. Experiments and Discussion

In this section, the experimental results on UCI Machine Learning benchmark data sets and a facial membership authentication application are reported.

Function: gSVMTtraining

input:
- $x_{\text{train}}$: /* training data set */
- $\xi$: /* the threshold of SVM loss function */
- $K$: /* the used type of SVM Kernel */

output:
- $T$: /* generated gSVMT */

variables:
- $x$: /* a training instance */
- $T_p$: /* an SVM tree node */
- $X_p$: /* data partition on an SVM node */
- $\lambda_p$: /* merged data partition for SVM node training */

function Calls:
- SVMnode_train($T_p$, $K$): /* train an SVM classifier */
- Partition($X$): /* data partitioning function */
- UpdateTree($T$, $T_p$): /* adding a new node */

begin
1. $T \leftarrow 0$; /* initialize gSVMT as a root node */
2. $d \leftarrow$ variable select($x_{\text{train}}$); /* initialize variable */
3. if $x_{\text{train}}$ is empty
   4. return $T$; /* Iteration stops, and gSVMT generated */
5. $P \leftarrow$ Partition($X$, $d$);
6. for each $X_p \in P$
   7. if $\sum_{\big|X^\text{sm}\big|} |\text{SVM}^{\text{sm}}(x) - \hat{f}(x)| > \xi$
   8. UpdateTree($T$, $T_p$);
   9. $\lambda_{\text{train}} \leftarrow \lambda_{\text{train}} - X_p$;
10. $\lambda_p$;
11. gSVMTTraining($x_{\text{train}}$, $\xi$, $K$)/ /* zooming in */
12. for $T_p \in T$
13. $X_p$;
14. SVMnode_train($X_p$, $K$);
15. SVMnode_train($X_p$, $K$);
end

Algorithm 1. gSVMT Training

A. UCI Data Sets

We test the proposed gSVMT algorithm on 13 UCI data sets. Table 1 reports the error rates obtained by the proposed gSVMT method with a comparison against previously reported results on SVM classifiers [13]. We divide the 13 data sets into two categories according to the sparseness value. Six data sets with the sparseness greater than 0.3 are taken as sparse data sets, and the remaining 7 data sets are counted as non-sparse data sets.

As shown in the table, the proposed gSVMT matches SVMs on 5 out of 6 sparse data sets when proper RBF kernels are employed. It is surprising that gSVMTs wins SVM over 15% on the classification accuracy of 2 non-sparse data sets, Flare-solar and Titanic. As we exploring the distribution characteristics of these two data sets in their feature space, it is found that both of them have some discrete attributes. In other words, both two data sets are a type of discrete data set, which is a special case of sparseness covered by our sparseness definition. The extraordinary performance on these two data sets suggests that the proposed gSVMT is very efficient on the classification of discrete data sets. The proposed gSVMT performs not as well as single SVMs on those non-sparse data set, this possibly because the grid partitioning of gSVMT has destroyed too much information useful for classification.
Appendix: B

Function: gSVM Ttesting
input:
$T$; /* a constructed gSVM */
x; /* a testing instance */

output:
$C$ /* output classifier */

variables:
Current; /* current node number of $T$ */
Next; /* the node next to Current node */
$T$; /* the test function on node i */

Function Calls:
SVMTest($x$, K); /* trained SVM classifier */
SgnTest($x$); /* trained one-class classifier */
SearchNode($T$, i); /* search the node next to i in $T$ */

begin
1 Current ← 0;
2 while (SearchNode($T$, Current) == Null)
3 Next ← SearchNode($T$, Current);
4 Current ← Next;
5 if $T_{Current}$ is 2-class
6 $C$ ← SVMTest($x$, K, Current);
7 else
8 $C$ ← SgnTest($x$, Current);
9 return $C$;
end

Algorithm 2. gSVM Ttesting

### TABLE I

<table>
<thead>
<tr>
<th>Data sets</th>
<th>Sparseness</th>
<th>SVM</th>
<th>gSVM</th>
<th>$\xi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast Cancer</td>
<td>0.1039</td>
<td>0.26 ± 0.05</td>
<td>0.28 ± 0.05</td>
<td>0.1</td>
</tr>
<tr>
<td>German</td>
<td>0.0777</td>
<td>0.24 ± 0.02</td>
<td>0.37 ± 0.03</td>
<td>0.05</td>
</tr>
<tr>
<td>Heart</td>
<td>0.2637</td>
<td>0.16 ± 0.03</td>
<td>0.16 ± 0.03</td>
<td>0.02</td>
</tr>
<tr>
<td>Image</td>
<td>0.2806</td>
<td>0.03 ± 0.006</td>
<td>0.06 ± 0.01</td>
<td>0.5</td>
</tr>
<tr>
<td>Splice</td>
<td>0.0318</td>
<td>0.11 ± 0.01</td>
<td>0.16 ± 0.02</td>
<td>0.01</td>
</tr>
<tr>
<td>Flare-solar</td>
<td>0.0265</td>
<td>0.32 ± 0.02</td>
<td>0.05 ± 0.02</td>
<td>0.01</td>
</tr>
<tr>
<td>Titanic</td>
<td>0.0942</td>
<td>0.24 ± 0.01</td>
<td>0.06 ± 0.04</td>
<td>0.07</td>
</tr>
<tr>
<td>Waveform</td>
<td>0.1136</td>
<td>0.10 ± 0.01</td>
<td>0.13 ± 0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>ringnorm</td>
<td>0.6575</td>
<td>0.02 ± 0.001</td>
<td>0.02 ± 0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>Diabetes</td>
<td>0.3822</td>
<td>0.24 ± 0.02</td>
<td>0.24 ± 0.02</td>
<td>0.0001</td>
</tr>
<tr>
<td>Banana</td>
<td>0.7727</td>
<td>0.12 ± 0.01</td>
<td>0.14 ± 0.01</td>
<td>4</td>
</tr>
<tr>
<td>Thyroid</td>
<td>0.7452</td>
<td>0.05 ± 0.02</td>
<td>0.04 ± 0.02</td>
<td>1</td>
</tr>
<tr>
<td>TwoNorm</td>
<td>0.7043</td>
<td>0.03 ± 0.002</td>
<td>0.02 ± 0.00</td>
<td>0.002</td>
</tr>
</tbody>
</table>

B. Face Membership Authentication Application

To test gSVMTs on problems where dynamic class-imbalance and class-overlapping are prevalent, we have also applied the algorithm to the face membership authentication (FMA) problem [17], [16], [15]. The objective of an FMA system is to distinguish the membership class from the non-membership class based on their facial features presented as images.

FMA is counted as a problem involving sparse data, because the within-class similarity is rather low: Facial appearance of one person may look very different from another, despite the fact that they both belong to the same membership group. In other words, face samples for one person can be far away from samples for another person in the feature space. Another challenging property of a FMA problem is severe class-imbalance: The size of membership group is generally much smaller than that of non-membership, and the class-imbalance may vary dynamically over time in real applications.

In the experiment, we work on a database composed of facial images from 271 individuals, with 5 different images for each. Facial features are extracted using principle component analysis (PCA), and the eigenvector associated with top 100 eigenvalues are used in our experiment. The sparseness of the data set is 0.6605. Although this value does not really reflect the sparseness of the data set, the capability of the data set for data partitioning is obvious. Fig. 4 presents an FMA problem of 5 members against 100 non-members in the feature space defined by the first two eigenvectors. As we can imagine, to distinguish these 5 members from the 100 non-members, a complicated decision function is required. However, such classification difficulty can be reduced greatly, as shown in Fig. 4, if 5 members can be individually authenticated by separated simpler decision rules. This indicates that the FMA problem
Appendix: B

Fig. 5. The sparseness for the membership class data, with membership group size increasing from 10 to 135.

can be well fitted by a divide-and-conquer model.

To test the robustness of gSVMT to class imbalance, we formulate a sequence of FMA problems with the membership group size increasing from 10 ($C_1 = 0.962$) to 120 ($C_2=0.205$), with a step size of 10 persons. Here, as an extreme case of class-imbalance, results for the member group size of 135 ($C_m=0.007$) are also reported. Note that the sparseness for the whole data set does not change over time, however, that of the membership class data is decreasing as shown in Fig. 5.

In particular, when the size of the membership group is 10, it indicates that 10 members scatter randomly in the feature space among the remaining 261 non-members. In this case, although the sparseness for the whole data set is the same as before, the sparseness within membership group reaches the maximum because the within-class similarity of the 10 members is less than cases with more than 10 members. Hence, membership group data distribution shows a large sparseness value.

Figure 6 gives a comparison of the member authentication rate (the classification rate of the skewed class) versus class-imbalance. Four curves have recorded how the four classifiers perform under the variation of class-imbalance. As shown in the figure, SVMs get large variations on the classification of the skewed class (the membership group), while the membership group size is growing from a small size, 10, to the maximum, 135. On the other hand, the proposed gSVMT shows obvious robustness to the class-imbalance variation, keeping the member authentication rate around 80% under all class-imbalance situations.

Especially for cases where the sparseness is large, it is difficult for ordinary SVMs to find a complicated but continuous decision rule to distinguish a small number of members accurately from a large number of non-members. On the contrary, the proposed gSVMT has shown an very good performance on these cases. The sparsity of the data set can be seen as rather an advantage for gSVMT, than a disadvantage for ordinary SVMs.

VI. CONCLUSION AND FUTURES WORK

Addressing the problem caused by the sparsity and class-imbalance of data distribution, this paper proposes a new type of SVM based modular methods. The advantages of the proposed gSVMT are as follows. First, gSVMT keeps every iteration of data partitioning to be performed only at the sparse region of the data, minimizing the loss of discriminant information during the SVMT construction. Second, gSVMT addresses the class-imbalance problem by adjusting the partitioning cost function on the skewed-class, so that the skewed-class is emphasized by every SVM node in gSVMT. The proposed gSVMT is experimented on 13 UCI data sets and a real Facial Membership Authentication application which is characterize by the dynamic class-imbalance and class-sparseness property. The experimental results demonstrated that, compared with stand alone SVMs, the proposed gSVMT is able to treat with the sparseness and class-imbalance of the input data and delivers competitive classification performance.

From the viewpoint of SVM aggregating intelligence [14], the proposed gSVMT follows the same route of recursive data partitioning plus aggregating as the previous SVM tree methods. The gSVMT is different from all previous SVMT aggregating methods [15] because 1st grid partitioning is used for problem decomposition; 2nd every internal node of the gSVMT is a data partitioning function, and every leaf node is an SVM node. Note that gSVMT has no one-class node, and decisions are made by the leaf SVM node.

The proposed gSVMT is merely sustainable for 2-class problems. It can be simply extended to m-class problems by learning m binary classifiers for each class versus the others, or by learning m(m − 1) binary classifiers for each class versus another class. This topic is generally discussed under the framework of SVM decision forests [8]. Alternatively, in the principle of SVM aggregating, a multi-class gSVMT (mgSVMT) can be constructed by decomposing a m-class task into a certain number of 2 to m classes subtasks as

\[
\mathcal{T}_m\rightarrow\text{gSVMT}(\mathcal{P}^n, f_{\text{SVMT}<2>, \ldots, f_{\text{SVMT}<m>}, x),
\]

(11)
Appendix: B

where $\mathcal{P}^n$ represents a partitioning function which is able to split a set into multiple subsets. Similar to 2-gSVMT, $f_{\text{SVMT}}^{<i\downarrow}$ is an $i$-class ($2 \leq i \leq m$) local SVM classifier. Accordingly, the classification function of a $m$-gSVMT can be written as,

$$f(x) = f_{\text{SVMT}}^{<i\downarrow}, \text{ if } \mathcal{P}^n(x) \in g_j. \quad (12)$$

As a measurement of the sparsity of a data set, sparseness is an important concept for divide-and-conquer data modelling. However, it is also a concept difficult to give an accurate and consistent evaluation, just as has been reflected in our work. Still, sparsity does exist for most life data, thus for better generalization ability of gSVMT, it will be an interesting future work to derive a new sparseness measurement consistent for most data sets.

REFERENCES


The last twenty years have witnessed the remarkable progress in computational intelligence modelling for various applications. However, a majority of these researches make one fundamental assumption that sufficient and representative data is required to be provided in advance for training. Because this assumption often does not hold in many real applications, recent efforts on adaptive soft computing techniques, such as evolving connectionist systems (ECOS), incremental learning and other adaptive learning aim to relax the “sufficiency” requirement by continuously updating a model to learn from data streams.

ECOS addresses the learning from a data stream or chunks of data whose underlying distribution changes over time, by training a neural network continuously and adapting its structure and functionality through repeated interactions with the environment or other learning systems. Similarly, incremental learning develops an ability of a computational model to accumulate continuously any knowledge learned at different time from noisy and/or incomplete data. In practice, incremental learning is also featured by one-pass property, which enables the algorithm to work with real time data streams presented only once to the learning machine. Other adaptive soft computing techniques – also called concept drift algorithms, make other assumptions such as restricting the type of change in the distribution, are primarily of heuristic in nature with many free parameters requiring fine-tuning, and have not been evaluated on large scale real-world applications.

It is noticeable that new challenge occurs recently upon the explosive growth of mobile and network communications. The adaptive soft computing is confronting a number of new difficult tasks, such as spam filtering, internet intrusion detection, and malicious software attack prevention, etc. To deal with these problems, a successful method should pay special attention to the following aspects: huge amount of information presented as data streams, real time system response demanded by applications, dynamic property of the data sources, severe sampling bias in the training data, and inequality of misclassification costs.

This thematic issue of “Adaptive Soft Computing Techniques and Applications” intends to explore the most recent advances on the theoretic and enrich the understanding of the practical respects of the topic. In particular, we would like to call forth the attention of the AI society for developing new theories and models for existing challenging problem solving, and on the other hand to encourage the domain experts to introduce new industry applications for driving the advances of the technology.

The topic of interest includes, but not limited to
- Incremental learning; Multi-task learning; Lifelong learning
- Evolving connectionist system Learning in dynamic environments
- Applications to real time intelligent system
- Advanced applications to detect cyber terrorist attacks like intrusion, phishing sites, spam emails, etc.
- Other applications that call for incremental learning or learning from very large database.
- Issues relevant to above mentioned or related fields

Manuscripts are required to be prepared in the standard format of the Memetic Computing journal as indicated in the Information for Authors. All submissions will be peer-viewed by experts in the field. Priority will be given to work with novelty and potential impacts. Manuscripts based on previously published conference papers should be extended substantially. The submissions should be performed through the journal via the Memetic Computing Journal website at: http://www.editorialmanager.com/meme/

Enquiries on this special issue should be sent to guest editor Dr. Paul S. Pang at spang@aut.ac.nz, or Dr. Ban Tao at bantao@nict.go.jp.

Guest Editors:
- Dr. Paul S. Pang, Knowledge Engineering & Discovery Research Institute, Auckland University of Technology, Private Bag 92006, Auckland 1020, New Zealand, Email: spang@aut.ac.nz
- Dr. Tao Ban, Information Security Research Center, National Institute of Information and Communications Technology, Tokyo, 184-8795, Japan, Email: bantao@nict.go.jp
- Xavier Llorà, National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, xllora@illinois.edu

**Submission deadline: April 15th, 2009**