A reduced labeled samples (RLS) framework for classification of imbalanced concept-drifting streaming data.

Elaheh Arabmakki

University of Louisville

Follow this and additional works at: https://ir.library.louisville.edu/etd

Part of the Computer Engineering Commons, and the Computer Sciences Commons

Recommended Citation


https://doi.org/10.18297/etd/2602

This Doctoral Dissertation is brought to you for free and open access by ThinkIR: The University of Louisville's Institutional Repository. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of ThinkIR: The University of Louisville's Institutional Repository. This title appears here courtesy of the author, who has retained all other copyrights. For more information, please contact thinkir@louisville.edu.
A REDUCED LABELED SAMPLES (RLS) FRAMEWORK FOR CLASSIFICATION OF IMBALANCED CONCEPT-DRIFTING STREAMING DATA

By

Elaheh Arabmakki
B.S., Azad University of NajafAbad, 2006
M.S., Morehead State University, 2011

A Doctoral Dissertation
Submitted to the Faculty of the
J.B. Speed School of Engineering at the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy
in Computer Science and Engineering

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

December 2016
A REDUCED LABELED SAMPLES (RLS) FRAMEWORK FOR CLASSIFICATION OF IMBALANCED CONCEPT-DRIFTING STREAMING DATA

By

Elaheh Arabmakki
B.S., Azad University of NajafAbad, 2006
M.S., Morehead State University, 2011

A Dissertation Approved On

November 16th 2016

by the following Dissertation Committee:

__________________________________________
Dr. Mehmed Kantardzic, Dissertation director

__________________________________________
Dr. Adel S. Elmaghraby

__________________________________________
Dr. Olfa Nasraoui

__________________________________________
Dr. Adrian Lauf

__________________________________________
Dr. Jozaf Zurada
ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Dr. Mehmed Kantardzic for his guidance, advice, and his patience during my PhD studies.

I would like to thank the Computer Engineering and Computer Science department, and all the professors for their effort and care for the students. I would like to thank Dr. Adel Elmaghraby for providing healthy research environment and for his consistent support.

I also would like to thank my parents for their endless support and encouragement through these years.

Finally, there are no words to describe my appreciation towards my dear husband. Without his unconditional support and encouragement, this work would have never came through.
ABSTRACT

A REDUCED LABELED SAMPLES (RLS) FRAMEWORK FOR CLASSIFICATION OF IMBALANCED CONCEPT-DRIFTING STREAMING DATA

Elaheh Arabmakki

November 16th 2016

Stream processing frameworks are designed to process the streaming data that arrives in time. An example of such data is stream of emails that a user receives every day. Most of the real world data streams are also imbalanced as is in the stream of emails, which contains few spam emails compared to a lot of legitimate emails.

The classification of the imbalanced data stream is challenging due to the several reasons: First of all, data streams are huge and they can not be stored in the memory for one time processing. Second, if the data is imbalanced, the accuracy of the majority class mostly dominates the results. Third, data streams are changing over time, and that causes degradation in the model performance. Hence the model should get updated when such changes are detected. Finally, the true labels of the all samples are not available immediately after classification, and only a fraction of the data is possible to get labeled in real world applications. That is because the labeling is expensive and time consuming.

In this thesis, a framework for modeling the streaming data when the classes of the data samples are imbalanced is proposed. This framework is called Reduced Labeled Samples (RLS). RLS is a chunk based learning framework that builds a model using partially
labeled data stream, when the characteristics of the data change. In RLS, a fraction of the samples are labeled and are used in modeling, and the performance is not significantly different from that of the 100% labeling. RLS maintains an ensemble of classifiers to boost the performance. RLS uses the information from labeled data in a supervised fashion, and also is extended to use the information from unlabeled data in a semi supervised fashion. RLS addresses both binary and multi class partially labeled data stream and the results show the basis of RLS is effective even in the context of multi class classification problems. Overall, the RLS is shown to be an effective framework for processing imbalanced and partially labeled data streams.
# TABLE OF CONTENTS

ACKNOWLEDGMENTS ........................................................................... iii
ABSTRACT .............................................................................................. iv
LIST OF TABLES ...................................................................................... ix
LIST OF FIGURES .................................................................................... x

CHAPTER 1 .............................................................................................. 1
INTRODUCTION ....................................................................................... 1

1.1 Classification of the streaming data .................................................. 1
1.2 Concept drift .................................................................................... 4
   a) Conditional drift ............................................................................ 4
   b) Feature drift ................................................................................... 5
   c) Dual drift ...................................................................................... 6
1.3 Imbalanced classes in streaming data .............................................. 6
1.4 Thesis problem specifications .......................................................... 8
1.5 Thesis contributions ........................................................................ 9

CHAPTER 2 .............................................................................................. 11
LITERATURE REVIEW .............................................................................. 11

2.1 Supervised-fully labeled data ......................................................... 12
2.2 Supervised-partially labeled data ..................................................... 15
2.3 Semi Supervised- labeled and unlabeled data .................................... 21

CHAPTER 3 .............................................................................................. 29
REDUCED LABELED SAMPLES (RLS) FRAMEWORK ................................. 29

3.1 RLS framework architecture ............................................................ 29
3.2 How the critical samples are selected? ............................................ 31
3.3 Introduction to Support Vector Machines (SVM) .............................. 34
3.4 RLS-S Algorithm .............................................................................. 35
3.5 Experiments ..................................................................................... 41
3.5.1 Data sets ......................................................................................................................... 41
3.5.2 Evaluation measures of the classification performance .............................................. 45
3.6 Experiments ........................................................................................................................ 48
3.7 Discussion ......................................................................................................................... 53

CHAPTER 4 .................................................................................................................................. 55
USING AN ENSEMBLE OF CLASSIFIERS FOR CLASSIFICATION OF
IMBALANCED STREAMING DATA ........................................................................................... 55
4.1 Introduction ......................................................................................................................... 55
4.2 Maintaining an ensemble of the models ............................................................................ 57
  4.2.1 Classification results from ensemble ............................................................................. 57
  4.2.2 Classification results from each individual model in the ensemble ......................... 59
4.3 RLS-E Algorithm .............................................................................................................. 60
4.4 Experimental results and analysis .................................................................................... 62
  4.4.1 Data sets ......................................................................................................................... 62
  4.4.2 Results .......................................................................................................................... 62
4.5 Experimental comparison with other methods ................................................................. 74
  4.5.1 Comparison of RLS-E with RLS-S ............................................................................... 74
  4.5.2 Comparison of RLS-E with other ensemble methodologies ........................................ 75
4.6 Discussion .......................................................................................................................... 77

CHAPTER 5 .................................................................................................................................. 79
APPLYING SELF ORGANIZING MAP (SOM) TO TACKLE THE SUDDEN
FEATURE DRIFT ....................................................................................................................... 79
5.1 Introduction ......................................................................................................................... 79
5.2 Proposed RLS-SOM .......................................................................................................... 81
  5.2.1 SOM helps in detecting the minority class samples ..................................................... 81
  5.2.2 RLS-SOM Algorithm .................................................................................................... 83
5.3 Experimental results and analysis .................................................................................... 88
  5.3.1 Data sets ......................................................................................................................... 88
  5.3.2 Results .......................................................................................................................... 88
  5.3.3 Experimental comparison of RLS-SOM with other techniques .............................. 90
5.4 Discussion .......................................................................................................................... 99
CHAPTER 6

APPLYING SEMI-SUPERVISED LEARNING IN CLASSIFICATION OF IMBALANCED STREAMING DATA ................................................................. 101

6.1 Introduction ........................................................................................................ 101
6.2 Comparision results of RLS-SSL with RLS-SOM ............................................. 104
6.3 Comparision results of RLS-SSL with other methodologies ......................... 106
6.4 Discussion ........................................................................................................... 107

CHAPTER 7

RLS FRAMEWORK FOR MULTI CLASS DATA STREAMS ..................................... 109

7.1 Introduction ........................................................................................................ 109
7.2 Literature Review ................................................................................................ 110
7.3 RLS-Multi for multi class data sets ................................................................. 114
7.4 Results ................................................................................................................ 118
7.4.1 Data sets ......................................................................................................... 118
7.4.2 Experimental Setup ..................................................................................... 121
7.4.3 Experimental Results .................................................................................. 121
7.5 Discussion ........................................................................................................... 125

CHAPTER 8

CONCLUSIONS ...................................................................................................... 127

8.1 RLS in review .................................................................................................... 127
8.2 Analysis of complexity of RLS Framework .................................................... 131
8.3 Future work ....................................................................................................... 133

REFERENCES ........................................................................................................ 135
CURRICULUM VITAE ............................................................................................. 142
LIST OF TABLES

TABLE 1. DATA SETS CHARACTERISTICS ................................................................. 45
TABLE 2. LABELING PERCENTAGE FOR EACH DATA SET ........................................... 50
TABLE 3. EXAMPLE OF CALCULATION FOR TWO SAMPLES .................................. 58
TABLE 4. FINAL CLASS CALCULATION .................................................................... 58
TABLE 5. THE F-MEASURE FOR “SELECTION OF HIGHEST”, “ENSEMBLE”, AND “INDIVIDUAL MODEL” FOR ALL THE DATA SETS ............................................................................................ 63
TABLE 6. F-MEASURE RESULTS OF THE ELECTRICITY DATA SET FOR DIFFERENT CHUNK SIZES .............................................................................................................................. 68
TABLE 7. F-MEASURE RESULTS OF THE OZONE-1H DATA SET FOR DIFFERENT CHUNK SIZES .............................................................................................................................. 69
TABLE 8. F-MEASURE RESULTS OF THE OZONE-8H DATA SET FOR DIFFERENT CHUNK SIZES .............................................................................................................................. 70
TABLE 9. F-MEASURE RESULTS OF THE CHECKBOARD DATA SET FOR DIFFERENT CHUNK SIZES .............................................................................................................................. 71
TABLE 10. COMPARISON RESULTS OF RLS-E WITH THREE OTHER TECHNIQUES SA, HD, HDIG .............................................................................................................................. 76
TABLE 11. DATA SET DESCRIPTION AND PARAMETERS SELECTION ....................... 97
TABLE 12. COMPARISON RESULTS OF RLS-SOM WITH OTHER METHODOLOGIES(DITZLER & POLIKAR, 2013) ........................................................................................................ 98
TABLE 13. DATA SETS CHARACTERISTICS .................................................................. 120
TABLE 14. EXPERIMENTAL SET UP .......................................................................... 122
TABLE 15. EXPERIMENTAL RESULTS AND COMPARISON OF RLS-MULTI WITH OTHER TECHNIQUES(MIRZA ET AL., 2015) ................................................................................. 125
LIST OF FIGURES

FIGURE 1. CLASSIFICATION OF THE DATA STREAM USING A CHUNK-BASED LEARNING PROCESS ................................................................. 2
FIGURE 2. CONDITIONAL DRIFT. (t_1 < t_2) ........................................................................................................... 5
FIGURE 3. CONDITIONAL DRIFT IN A STREAM OF DATA FOR MULTIPLE CHUNKS
(t_1 < t_2 < t_3 < t_4) ........................................................................................................... 5
FIGURE 4. FEATURE DRIFT. (t_1 < t_2) ........................................................................................................... 6
FIGURE 5. MAKING THE TRAINING SET BALANCED (GAO, DING, FAN, HAN, & YU, 2008). 12
FIGURE 6. SIMPLE MARGIN WILL CHOOSE b (TONG & KOLLER, 2002) ........................................... 17
FIGURE 7. MAX-MIN MARGIN WILL CHOOSE b(TONG & KOLLER, 2002) ........................................... 18
FIGURE 8. RATIO MARGIN WILL CHOOSE e (TONG & KOLLER, 2002) ........................................... 18
FIGURE 9. AING LEARNS THE TOPOLOGY OF DATA. LEFT: DIFFERENT DATA DISTRIBUTION SHAPES IN A 2 DIMENSIONAL SPACE. RIGHT: GRAPH TOPOLOGY G OF NEURONS
(BOUGUELIA ET AL., 2013) ........................................................................................................... 23
FIGURE 10. THE GENERAL A2ING SCHEME (BOUGUELIA ET AL., 2013) ........................................... 23
FIGURE 11. PROPOSED RLS FRAMEWORK ................................................................................................. 30
FIGURE 12. EXAMPLE OF THE CLASSIFICATION OF THE DATA, A) DATA WITH TWO CLASSES,
  B) CLASSIFICATION MODEL SHOWN IN RED, C) SUPPORT VECTORS SHOWN IN BOLD .... 32
FIGURE 13. AN EXAMPLE OF DETECTING SAMPLES CLOSE TO THE SUPPORT VECTORS FOR IMBALANCED CHUNK OF DATA ........................................................................................................... 33
FIGURE 14. AN EXAMPLE OF DETECTING SAMPLES CLOSE TO THE SUPPORT VECTORS AS WELL AS MINORITY CLASS SAMPLES FOR IMBALANCED CHUNK OF DATA ............... 33
FIGURE 15. GRAY AREA SHOWS CLOSEST NEIGHBORHOOD TO BORROW SAMPLES ......... 40
FIGURE 16. ROTATING CHECKBOARD EXPERIMENTS \( \alpha \in \{0, \pi\} \) (DITZLER & POLIKAR, 2013) ........................................................................................................... 42
FIGURE 17. SHIFTING HYPERPLANE DATA WHEN A) THRESHOLD=4 B) THRESHOLD=7 ...... 43
FIGURE 18. ROC CURVE ............................................................................................................................................... 47
FIGURE 19. THE F-MEASURE RESULTS SHOWN FOR THE FIRST 20 CHUNKS WHEN THE
  THRESHOLD VARIES FROM 3% TO 70% FOR HYPERPLANE DATA SET ........................................... 49
FIGURE 20. F-MEASURE RESULTS WITH (p%) AND 100% LABELING FOR A) OZONE-1H,
  B) OZONE-8H, C) ELECT, AND D) WEATHER DATA SET .............................................................. 51
FIGURE 21. F-MEASURE RESULTS WITH (p%) AND 100% LABELING FOR COVERTYPE,
  ADULT, CHECKBOARD, AND HYPERPLANE DATA SET .............................................................. 52
FIGURE 22. F-measure results for the a) Weather b) Adult data sets for chunk sizes (400,600,800,1000,1200), and labeling percentages (5,10,15,20,25,30,50,70,100) ................................................................. 64
FIGURE 23. F-measure results for the a) HyperPlane b) CoverType data sets for chunk sizes (400,600,800,1000,1200), and labeling percentages (5,10,15,20,25,30,50,70,100) ................................................................. 66
FIGURE 24. Labeling percentage for the a) Electricity b) Ozone-1h c) Ozone-8h), and d) CheckBoard data sets ................................................................. 67
FIGURE 25. Running time vs chunk size ................................................................. 72
FIGURE 26. Effect of ensemble size on F-measure for the a) HyperPlane, and b) Electricity and CheckBoard data sets ................................................................. 73
FIGURE 27. a) F-measure and b) G-mean results of RLS-E and RLS-S for Adult, Electricity, HyperPlane, Weather, Ozone-1h, Ozone-8h, CheckBoard, and CoverType data sets ................................................................. 74
FIGURE 28. Feature drift between Chunk 1 (C1) and Chunk 2 (C2) of the streaming data ........................................................................................................ 80
FIGURE 29. RLS-SOM illustration ........................................................................ 82
FIGURE 30. The effect of different map size on labeling ........................................ 89
FIGURE 31. The F-measure for all the datasets when the labeling percentages vary from 5% to 100% ................................................................. 90
FIGURE 32. Labeling percentage for the Electricity, CheckBoard, Ozone-1h, and Ozone-8h for the RLS-SOM and RLS-E ................................................................. 91
FIGURE 33. F-measure results for the Electricity, CheckBoard, Ozone-1h, and Ozone-8h for the RLS-SOM and RLS-E ................................................................. 91
FIGURE 34. The F-measure results for Electricity, HyperPlane, CheckBoard, and Weather datasets ........................................................................ 94
FIGURE 35. The F-measure results for Adult, CoverType, Ozone-1h, and Ozone-8h datasets ........................................................................ 95
FIGURE 36. F-measure results of the a) Ozone-1h, b) Ozone-8h, c) Electricity, d) CheckBoard data sets for all the chunk sizes ................................................................. 105
FIGURE 37. F-measure results of the a) Adult, b) HyperPlane, c) Weather and d) CoverType data for all the chunk sizes ................................................................. 106
FIGURE 38. Comparison F-measure results of RLS-SSL with other techniques for the CheckBoard, HyperPlane, Electricity, and Weather data sets ........................................................................ 107
FIGURE 39. a) SVM used for modeling a multi class data (OAO) b) Samples are projected to a 2D map using SOM ........................................................................ 115
FIGURE 40. Varying the labeling percentage to obtain high G-mean performance ........................................................................ 121
Figure 41. G-mean performance over time when there is no model update (red dotted line), and when the concept drift is detected and handled in RLS-Multi (blue solid line) on real world data sets.
CHAPTER 1
INTRODUCTION

1.1 Classification of the streaming data

Data stream is generated in many real world applications, such as sensor networks, computer network traffic, web logs, credit card transactions, remote sensors, and web click streams (Aggarwal, Han, Wang, & Yu, 2004; Babcock, Babu, Datar, Motwani, & Widom, 2002). The research in the classification of the streaming data has gained a high attraction due to the importance of its applications and the increasing generation of streaming information (Babcock et al., 2002; Gaber, Zaslavsky, & Krishnaswamy, 2005; Hulten, Spencer, & Domingos, 2001). For example in the computer network traffic, the traffic is classified into a number of traffic classes according to various parameters, or in the credit card transactions, classification is used to detect the fraud transactions from legitimate ones.

Unlike the traditional data sets, data streams arrive continuously at varying speed, those are fast changing, temporally ordered, potentially infinite and massive (J. Han & Kamber, 2006). Due to their huge volume, and its need for fast processing, data stream can not be stored in the memory for further analyses and should be analyzed through methods such as incremental learning or chunk-based learning approaches. In the incremental learning process, the system learns from each single data sample when arrives. Whereas in a chunk-based learning process, system learns from a chunk of n data samples when arrive (Read, Bifet, Pfahringer, & Holmes, 2012).
As it is stated in (Read, Bifet, Pfahringer, & Holmes, 2012b), the performance of a chunk-based learning process is similar to the incremental learning approach. One disadvantage of incremental learning is that it may require massive number of samples to learn, and may not adapt naturally to the changes in the data. Whereas a chunk-based learning process automatically deals with changes in the data stream (Read, Bifet, Pfahringer, & Holmes, 2012a). Therefore in this research, data stream is analyzed using a chunk-based learning process. Figure 1 depicts one step in a chunk-based learning process in the classification of the data stream.

![figure](image)

Figure 1. Classification of the data stream using a chunk-based learning process

Figure 1 shows the learning process for the first few chunks in the data stream (C1, C2, and C3). In order to initiate the data stream mining, the true labels of all the samples in the first chunk of data stream (C1) is obtained from the human expert. Hence, in Figure 1, 1) an initial classification model is built using C1. 2) When next chunk (C2) which contains unlabeled data samples arrive, this model is used to predict the class labels of the samples in C2. 3) The predicted labels of the samples in C2 are obtained
from the model. 4) The true labels of the samples are obtained from the human expert and 5) this chunk is used to update the classification model. 6) The updated model is used to classify the unlabeled samples in the next chunk ($C3$). This process is continued for the all other chunks of data that arrive in time.

As it is shown in Figure 1, in order to adapt with the changes in the data stream, the model is updated per chunk. This needs the availability of the true labels of the samples for each chunk which may cause a lot of problems. First of all, labeling is very costly, since a labor is involved in the process. Second, it is time consuming and labeling all the samples in each chunk may impose a lot of delay to the system. Third, if there is no change in the data from one chunk to another, expert effort has been wasted since there was no need to update the model.

Many studies on the streaming data mining assume relatively balanced classes in the data stream (Abbaszadeh, Amiri, & Khanteymoori, 2015; Bifet, Holmes, Pfahringer, Kirkby, & Gavaldà, 2009; H. Wang, Fan, Yu, & Han, 2003). However, many applications can involve concept-drifting data stream with skewed distribution of the classes (Lyon, Brooke, Knowles, & Stappers, 2013). In data with skewed distribution of the classes, each data chunk has few samples which belong to Class (+), and many samples which belong to Class (-). In this thesis, the former is called minority class samples and the latter majority class samples.

An example of an application involving data stream with imbalanced classes is bank account transactions, in which the majority of the transactions are legitimate, and very small number of them may be fraudulent. Knowing information about the previous cases of legitimate and fraudulent transactions, the model is built which can predict the
labels of the future transactions. However, hackers may use novel techniques that cause a fraudulent transaction to be classified as legitimate. Therefore, the previous model is no longer able to detect the legitimate transactions from fraudulent ones and must be rebuilt to adapt with the changes.

In the next section, we explain the types of changes in the data stream. These changes are called concept drift.

1.2 Concept drift

Data streams undergo concept drift considerably over time. If \( x \) is the feature vector, and \( y \) is the class label, the concept drift is defined as changes in \( p(x, y) \) in the following joint probability distribution for the classification problem (Gao, Fan, Han, & Philip, 2007):

\[
p(x, y) = p(y \mid x).p(x)
\]

(1)

Here, \( p(x) \) is the feature probability and \( p(y \mid x) \) is the class label conditional probability. From a predictive perspective, only the changes that affect the prediction decision require adaptation. Therefore, three types of concept drift: a) Conditional drift, b) Feature drift, and c) Dual drift are described as follows:

a) Conditional drift

Conditional drift occurs when there is a change in \( p(y \mid x) \), or in the classification model. For example, in the information filtering domain a stream of documents is labeled as relevant or irrelevant. Conditional drift in this case occurs if relevance of some
documents changes over time, that is the class label of some documents changes from relevant to irrelevant or vice versa (Borchani, Larrañaga, & Bielza, 2011).

![Model A at time t1 and Model A at time t2](image)

**Figure 2.** Conditional drift. (t₁ < t₂)

Figure 3 also shows how conditional drift occurs in a data stream for multiple chunks.

![Conditional drift in a stream of data for multiple chunks](image)

**Figure 3.** Conditional drift in a stream of data for multiple chunks (t₁ < t₂ < t₃ < t₄)

**b) Feature drift**

Feature drift corresponds to change in the \( p(x) \) or in the data distribution. That is, some infrequent feature vectors may become more frequent or vice versa. For instance, in the information filtering domain, the relative frequency of some documents changes over time, which may affect their relevance (Borchani et al., 2011). Figure 4 shows an example of this type of drift. In Figure 4(a) at time t₁, Model B is built over the data. In
time \( t_2 \), as shown in Figure 4(b), the distribution of the data has changed. That caused a new model to be built over the new data.

![Diagram showing feature drift](image)

**Figure 4. Feature drift. \((t_1 < t_2)\)**

c) **Dual drift**

Both type of feature and conditional drift will occur meaning both \( p(x) \) and \( p(y \mid x) \) changes.

In addition, concept drift can be either gradual or sudden. In the gradual drift, the concept changes gradually whereas in the sudden drift, the concept changes abruptly from one concept to another. Figure 3 shows an example of gradual conditional drift.

1.3 **Imbalanced classes in streaming data**

Learning from imbalanced data stream has been recognized as one of the crucial problems in machine learning and data mining (Longadge & Dongre, 2013). The most significant issue in this kind of data is that, minority class samples are often of much significance and interest to the user compared to the majority class samples. For example, in the spam detection applications, the classes are imbalanced, (spam vs healthy emails), and there is a high interest to detect the spam emails.
Many of the classification algorithms result in good performances for data with balanced classes. However, when applied on the data with imbalanced classes, the classification performance drops. This happens because learning from such imbalanced streams causes the learner to become biased towards the majority class; thus the minority class samples are likely to be misclassified. Therefore, the main issue in the classification of imbalanced data is maintaining the accuracy of the minority class samples. There are many real world applications for imbalanced data streams, some of which are listed and explained as follows:

- **Financial fraud detection:** In financial fraud detection, majority of the financial transactions are legitimate, and very small number of them may be fraudulent (Bay, Kumaraswamy, Anderle, Kumar, & Steier, 2006).

- **Network intrusion detection:** Usually in the network, there are thousands of access requests every day. Among all these requests, the number of malicious connections is very small compared to the number of normal connections (Wu, Xiong, Wu, & Chen, 2007).

- **Spam email detection:** Another example of imbalanced data stream is spam email detection, in which the majority of the emails are legitimate, whereas a few are reported as spam (Godase & Attar, 2012).

- **Health care:** In the health care, the rare diseases affect insignificant amount of people, but the consequences involved are very severe. It is extremely vital to correctly detect and classify the rare diseases and the affected patients (Godase & Attar, 2012).
1.4 Thesis problem specifications

In the classification of the streaming data, when concept drift is detected, the model must be rebuilt to maintain the performance.

For building a new model, samples in the current chunk of the data stream should be labeled. However, labeling is a costly process, because the acquisition of the true labels for a learning problem often requires a skilled human agent. Therefore, it is beneficial to design frameworks which build accurate models requiring a few labeled samples. Labelling only partial of samples in the data stream is even harder in the presence of imbalanced data stream. For example if we have a balanced chunk of data and if we label 10% of the chunk, most likely we get 5% of the majority class samples and 5% of the minority class samples, or close. But in case of imbalanced chunk of data, it could happen all the 10% of the chunk, returns majority class samples and we get 0% of the minority class samples. In this case, additional samples need to be labeled to reach 5% for the minority class samples.

In this thesis, the problem of labeling for the classification of imbalanced data stream with concept drift is addressed. Hence, a Reduced Labeled Samples (RLS) framework for the imbalanced data stream is proposed, which is designed to overcome concept drift problem by building new models utilizing small amount of labeled samples. In RLS framework, only the data stream samples that contain important information regarding variability of the data are found and labeled. Based on this partially labeled data stream, a classification model is built.
1.5 Thesis contributions

The main contributions of this study are as follows:

- Defining the architecture of the RLS framework for classification of imbalanced data stream
- Defining a technique for partial labeling based on Support Vector Machine (SVM) and Self-Organizing Map (SOM)
- Handling concept drift including both gradual and sudden changes in data stream
- Handling binary and multi class classification in data stream
- Building a new model only when the concept drift occurs, by using partial labeling while maintaining the quality of results not significantly different from that of the 100% labeling
- Detection of the minority class samples for labeling in highly imbalanced data stream with labeling small number of samples

The rest of this thesis is organized as follows:

- Chapter 2 provides the literature survey for classification of the data stream with concept drift, and with skewed distribution of the classes. These techniques used either partially, or fully labeled data stream in building the model.
- Chapter 3 describes the architecture of the proposed RLS framework, and explains the RLS-S (RLS-Single) algorithm which handles concept drift with changes close to decision boundary. The experiments on 6 real world and 2 synthetic data sets are shown for the proof of concept.
- Chapter 4 explains RLS-E (RLS-Ensemble) which maintains an ensemble of classifiers instead of using a single model. The experimental results are shown
when RLS-E is applied on the data streams with imbalanced classes for both synthetic and real world data sets. The experimental comparisons of RLS-E with RLS-S, and other techniques are also demonstrated.

- Chapter 5 defines RLS-SOM (RLS-Self Organizing Map), to use SOM for detecting the minority class samples if data stream suffers from sudden changes. The experimental results and comparison with RLS-E and other partially and fully labeled approaches in this domain are also presented.

- Chapter 6 presents RLS-SSL (RLS-Semi Supervised Learning) which uses S4VM (Chen & He, 2011) as its semi-supervised learning method in conjunction with our partially labeled method to improve the quality of performance.

- Chapter 7 demonstrates RLS-Multi (RLS- Multi class classification) which is the extension of RLS framework to handle imbalanced and multi class data stream, and builds a model using partially labeled data stream. The experimental results and comparison with other techniques are also demonstrated.

- Chapter 8 provides the scalability of the framework, the conclusion and the future work.
CHAPTER 2

LITERATURE REVIEW

Various classification methods presented in the literature to deal with the skewed data streams. Many of these approaches work with fully labeled data (Gao et al., 2007; H. Han, Wang, & Mao, 2005). Methods which uses partially labeled data are relatively new approaches for the data streams (Godase & Attar, 2012). When we say partially labeled data, we mean only p% of the data is labeled, and when we say fully labeled data we mean 100% of the data is labeled. The research in the domain of classification of the imbalanced streaming data may be divided into three different categories:

- Supervised approaches for classification of the imbalanced data stream with fully labeled data (Supervised-fully labeled data)
- Supervised approaches for classification of the imbalanced data stream using partially labeled data, that is building a model using the information from few labeled samples (Supervised-partially labeled data)
- Semi-Supervised approaches for classification of the imbalanced data stream with few labeled and large number of unlabeled data, that is building a model using the information from few labeled samples and abundant unlabeled samples (Semi Supervised- labeled and unlabeled data)
2.1 Supervised-fully labeled data

Many research have been done to address the class imbalance problem with concept drift, most of them considering all the samples as labeled (Gao et al., 2008; Gao et al., 2007; Lichtenwalter & Chawla, 2009; Sheng & Haibo, 2009). (Gao et al., 2007) presented an algorithm where streaming data is analyzed through a chunk-based learning process, and it is assumed the chunks are imbalanced. To build a model, researcher tried to make a balanced set for the training through the process shown in Figure 5. All the minority class samples from all the previous chunks $C_1, C_2, \ldots C_m$ are collected and will be placed in $T_s$ which is the training set. The majority class samples of the most current chunk $C_m$ is under sampled and will be placed in $T_s$ as well. This process makes the classes balanced. Then a model is built using $T_s$. In this approach, it may happen that the minority class samples become majority in the future due to the concept drift, and therefore these data samples are becoming irrelevant in the future.

![Figure 5. Making the training set balanced (Gao, Ding, Fan, Han, & Yu, 2008)](image)

Sheng and Haibo (2009) proposed SERA, short for Selectively Recursive Approach, which is similar to the framework presented by (Gao et al., 2008). Instead of
accumulating all the minority class samples from the previous chunks, only those similar to the minority class samples of the current chunk are selected. To find the similar samples, a similarity measure is used which is based on Mahalanobis distance. The Mahalanobis distance is formulated as follows:

$$\Omega = \sqrt{(x_i - \mu)^T \sum^{-1} (x_i - \mu)} \quad (2)$$

Where \( \Omega \) is the Mahalanobis distance from a set of \( n \) samples (accumulated minority class samples) with mean of \( \mu = [\mu_1, ..., \mu_n]^T \) and covariance matrix \( \sum \) to each of the minority class samples of the current chunk \( x = [x_1, ..., x_n]^T \). Based on the Mahalanobis distance, those samples close to the minority class samples of the current chunk are selected and finally a balanced chunk is obtained for the training purpose. SERA can be implemented either by building a single model, or by using a biased bagging that manually increases the minority class samples’ weights. SERA is less prone to the problem of drifting minority class samples compared to the method presented by (Gao et al., 2007), and both of these methods work best if the minority class is not drifting. Since SERA only relies on the information from current chunk, it may discard a significant part of knowledge from previous chunks. Therefore, motivated by approach in SERA, Chen and He (2009) proposed MuSeRA (Multiple Selectively Recursive Approach) which maintains all the hypothesis built on previous training data chunks. These hypotheses are weighted as an ensemble to predict the class label of the current testing data chunk. MuSeRA is using uncorrelated bagging, and same similarity measures as SERA (Chen & He, 2009), which assumes that the minority class carries the information about no more than one concept. Hence, (Chen & He, 2011) proposed Recursive Ensemble Approach.
(REA) which uses K-nearest neighbors as the similarity measure to address this issue. REA uses similar approach to SERA and MuSeRA by selectively accumulating similar minority samples to the current chunk as well as employing an ensemble approach to perform classification.

Lichtenwalter and Chawla (2009), proposed a methodology for resampling a highly imbalanced data to achieve better performance. For the minority class samples, Borderline-SMOTE (H. Han et al., 2005) was used. In this technique, the minority class samples which are at or close to the boarder line are selected to oversample. Borderline samples are more prone to be misclassified compared to the samples which are far from the borderline (these are the samples close to the classification model). In Borderline-SMOTE, the border line minority class samples are over sampled by creating new synthetic samples along the line between these minority class samples and their neighbors. For the majority class samples, all the majority class samples that the current model misclassified were accumulated. Later, these majority class samples are under-sampled to ensure the class proportion remains at the desired level. This method is called Boundary Definition (BD) (Lichtenwalter & Chawla, 2009). Lichtenwalter and Chawla (2010), proposed an extension to the work presented by (Gao et al., 2008), using the Boundary Definition (BD). In this work to address the feature drift, a combination of information gain and Hellinger distance was used to determine the similarity of the current chunk to the other chunks of data. The more similar chunks will introduce the same concept. The Hellinger distance is defined as follows:

\[ HD(X, Y, f) = \sqrt{\sum_{v \in f} \left( \frac{|X_{f=v}|}{|X|} - \frac{|Y_{f=v}|}{|Y|} \right)^2} \]  (3)
where $X$ and $Y$ are two chunks of data for a given feature $f$.

The information gain (IG) for a chunk $X$ is defined as the decrease in entropy $H$ of a class $c$ conditioned upon a particular feature $f$:

$$IG(X, f) = H(X_c) - H(X_c | X_f)$$

(4)

The authors observed the product of Hellinger distance and Information gain acts well for most of the usual situations. Finally, with simple smoothing, the following distance function for a single feature was found:

$$HDIG(X,Y,f) = HD(X,Y,f) *(1 + IG(X,f))$$

(5)

All the mentioned methodologies assume that the true labels of the classes are available after classification and fully labeled data were used in training.

### 2.2 Supervised-partially labeled data

Some approaches in the classification of the imbalanced data stream reduce the need for the labeled samples and build the model using partially labeled data (Lindstrom, Delany, & Mac Namee, 2010; Lyon, Brooke, Knowles, & Stappers, 2014). Lindstrom et al. (2010) proposed a method for labeling small amount of samples based on SVM. Author used the idea that a good selection approach is to choose samples close to the separating hyperplane, since the classifier has the least confidence in predicting their true class labels (Tong & Koller, 2002). (Tong & Koller, 2002) assumed data samples as a pool of unlabeled data and based on Support Vector Machines (SVMs), it is decided what are the best samples from the pool to label.

(Tong & Koller, 2002) used the idea that the hyperplanes in parameter space $W$ correspond to the points in feature space $F$ and vice versa (Cortes & Vapnik, 1995). In
their approach, a version space $V_i$ is defined which is a set of hyperplanes that separates the data in the feature space. Thus, samples that are chosen from the pool are the ones which divide the version space to two equal parts, $V$- and $V+$ corresponding to the two classes -1 and +1.

Authors try to find $w_i$ which is called unit normal vector of SVM. That maximizes the minimum distance to any of the hyperplanes. That is, SVMs find the center of the largest radius hypersphere whose center can be placed in the version space and whose surface does not intersect with the hyperplanes corresponding to the labeled samples.

Three techniques to select the best samples to label were proposed which are called Simple Margin, Max-Min Margin, and Ratio Margin. In Simple Margin, having the data samples $\{x_1 \ldots x_i\}$ with class labels $\{y_1 \ldots y_i\}$, authors try to find the SVM unit vector $w_i$ for this data. Authors check the unlabeled sample from the pool to see how close their corresponding hyperplanes in $W$ come to the centrally placed $w_i$. For each unlabeled sample $x$, the shortest distance between its hyperplane in $W$ and the vector $w_i$ is $|w_i \cdot \Phi(x)|$. This is the distance between the feature vector $\Phi(x)$ and the hyperplane $w_i$ in the feature space $F$. Therefore, SVM is learned on the labeled data, and the next sample for labeling is chosen if it comes closest to the hyperplane in $F$.

An illustration of this method is shown in Figure 6. The white area is the version space $V_i$ which is bounded by the labeled samples shown with solid lines. The unlabeled instances are presented with five dotted lines. The circle represents the largest radius hypersphere that can fit in the version space. Since the sample $b$ is closest to the SVM $w_i$, $b$ is chosen to label.
There are two assumptions for the Simple Margin: the version space is symmetric and that $w_i$ is centrally placed. However, these assumptions can significantly fail (Herbrich, Graepel, & Campbell, 2001). Therefore, Tong and Koller (2002) proposed Max-Min approximation to overcome these problems. Having data $\{x_1, \ldots, x_i\}$ and class labels $\{y_1, \ldots, y_i\}$, the SVM unit vector $w_i$ is the center of the largest hypersphere that can fit inside the current version space $V_i$ and the radius $m_i$ of the hypersphere is proportional to the size of the margin of $w_i$. The radius $m_i$ is now considered as an indication of the size of the version space (Cortes & Vapnik, 1995).

The relative size of the resulting version space $V^−$ can be estimated by labeling $x$ as $-1$, and finding the SVM obtained from adding $x$ to the labeled training data and looking at the size of its margin $m^−$. Similar calculation is done for the $V^+$ by relabeling $x$ as belonging to class $+1$ and finding the resulting SVM to obtain the margin $m^+$. Then authors choose to select the unlabeled samples for which the quantity $\min (m^−, m^+)$ is the greatest. Max-Min Margin is shown in Figure 7 which chooses sample $b$ to label.
If the current version space is extended, for some $x$ in the pool both $m^-$ and $m^+$ may be small simply because of the shape of the version space. Thus authors choose to select the $x$ for which $\min \left( \frac{m^-}{m^+}, \frac{m^+}{m^-} \right)$ is the largest. This is called Ratio Margin and is shown in Figure 8 and it chooses $e$ from the pool to label.

Lindstrom et al. (2010) used the (Tong & Koller, 2002) idea for partial labeling and handling concept drift in a text data stream. In their strategy, documents are unlabeled samples which are analyzed through a chunk-based learning process. The classification is performed on the first chunk as training set and will be applied to the next chunk as test set. After classifying documents in the test chunk, the model is rebuilt.
on a new training set. That is the old training set plus the \( n \) documents which their true class labels obtained from the expert. The \( n \) documents are the closest samples to the hyperplane. To keep the class distribution of the training set constant, from the training set the oldest documents of the same class as \( n \) documents are removed. This method further compared with the random sampling, which the \( n \) documents were chosen randomly from each chunk for labeling. The results show the effectiveness of the Lindstrom method compared to the random sampling.

Lyon et al. (2014) proposed to use Hellinger distance in the Very Fast Decision Tree (VFDT) methodology to build a classification model on the imbalanced data stream. VFDT (Hulten et al., 2001) is an incremental anytime decision tree learner suitable for the data stream. It permits the model updates in time proportional to the tree depth and data dimensionality. VFDT uses first samples in the data stream to choose split features at the root. Subsequent samples are passed through the tree to be used for choosing the split features there as well and so on. VFDT uses Hoeffding bound (Hoeffding, 1963; Maron & Moore, 1993) to decide how many samples are needed for each decision. Suppose we have a real random variable \( r \) with the range \( R \). If we made \( n \) independent samples from this variable \( r \), and we find the mean \( \bar{r} \), the Hoeffding bound states that the true mean of the variable \( r \) is at least \( \bar{r} - \varepsilon \), when the probability of \( 1 - \delta \). \( \varepsilon \) is defined as follows:

\[
\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}} \quad (6)
\]

The Hoeffding bound is independent of the distribution of the samples, and in fact, it takes more samples to reach the same \( \delta \) and \( \varepsilon \). Hence, the goal is to find the features using \( n \) samples which, with high probability, would be the same features when
using infinite samples in the data stream. For finding the features, a measure $G(X_i)$ such as information gain or Gini index is used. Let’s consider $X_a$ is found as the first choice feature and the $X_b$ is the second choice, and $\Delta \overline{G} = \overline{G}(X_a) - \overline{G}(X_b) \geq 0$. Then given $\delta$, the Hoeffding bound guarantees that $X_a$ is the correct choice with probability of $1 - \delta$ and $\Delta \overline{G} > \varepsilon^2$.

Lyon et al. (2014) considered Hellinger distance as a skew insensitive split criterion in VFDT which helped to split features based on their true class labels. If the features are normally distributed, and the distribution for the minority class samples is $P$ and for the majority class samples is $N$, the Hellinger distance between these two normal distributions is defined as follows:

$$d_H(P, N) = \sqrt{1 - \frac{2\sigma_1\sigma_2}{\sigma_1^2 + \sigma_2^2} e^{-\frac{1}{4} \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}}} \quad (7)$$

where $\mu_1$, and $\mu_2$ are the means, and $\sigma_1$ and $\sigma_2$ are the standard deviations of the $P$ and $N$ respectively. This approach is called GH-VFDT (Gaussian Hellinger Very Fast Decision Tree) and was tested on the data with different levels of class imbalance.

For dealing with the unlabeled streaming data, author used incremental learning process. To start the process, a classification model is built on the samples $x_1$ to $x_{i-1}$, as the labeled training set. This model is used to predict the class label of the sample $x_i$ which arrives at time $i$. Since the stream is unlimited, it is not possible to obtain the true class labels for each sample at $i+1$. Hence, the class label is obtained by expert only for those samples which are predicted to belong to the minority class. Therefore, a varying proportion of the samples in the stream (10%-50%-75%-100%) receive their true class
labels based on this process. In this approach, author did not consider the effect of concept drift.

2.3 Semi Supervised- labeled and unlabeled data

There are many semi-supervised approaches for the classification of the data stream (Bertini Jr, de Andrade Lopes, & Zhao, 2012; Masud, Gao, Khan, Han, & Thuraisingham, 2008). However, the research in the domain of semi-supervised learning with imbalanced distribution of classes has not been explored much. Masud et al. (2008) presented a semi-supervised clustering algorithm which uses a cluster-impurity measure based on the limited labeled data contained in each cluster. Semi-supervised clustering uses the information from limited labeled samples and unlabeled samples. In the semi-supervised clustering problem, having \( m \) data points \( X = \{x_1, x_2, \ldots, x_m\} \), and their class labels, \( y_j \in \{\varnothing, 1, \ldots, C\} \), if a data point \( x_j \in X \) has class label \( y_j = \varnothing \), then it is unlabeled. The goal is to create \( K \) clusters, having the assumption that all the labeled points in the same cluster have the same class label. Given a limited amount of labeled data, the goal of impurity-based clustering is to create \( K \) clusters by minimizing the intra-cluster dispersion and at the same time minimizing the impurity of each cluster. Author called this problem as K-means with Minimization of Cluster Impurity (MCI-Kmeans). A cluster is completely pure if it contains the labeled data points from only one class (along with some unlabeled data). Thus, the objective function should penalize each cluster for being impure. The objective function which should be minimized is:

\[
O_{MCIKmeans} = \sum_{i=1}^{K} \left( \sum_{x \in L_i} ||x - u_i||^2 + W_i \cdot Imp_i \right)
\]

(8)

where \( L_i \) is a set of all labeled data points in cluster \( i \), \( W \) is :
\[ W_i = \sum_{x \in L_i} ||x - u_i||^2 \] (9)

which is the sum of dispersions of all labeled points from the cluster centroid, and \( \text{Imp}_i \) is:

\[ \text{Imp}_i = \text{ADC}_i \times \text{Ent}_i \] (10)

where \( \text{ADC}_i \) is the “aggregated dissimilarity count” of the cluster \( i \), and \( \text{Ent}_i \) is the entropy of the cluster \( i \). After creating \( K \) clusters, the summary of the statistics of the data points in each cluster is extracted and saved. These saved summaries are called “micro-cluster” and serve as a classification model to classify the unlabeled data. To classify a sample using this model, K-Nearest Neighbor (K-NN) algorithm is used to find the \( K \) nearest micro-clusters from that sample and to select the class that has the highest frequency of the labeled data in these \( K \) clusters. The researchers assumed 5% of the data is labeled. The experimental results show comparable or better results compared to the methods which used fully labeled training data (Aggarwal, Han, Wang, & Yu, 2006).

(Bertini Jr et al., 2012) proposed a graph-based semi-supervised approach for dynamic streaming data. The \( K \)-associated graph is defined as a graph \( G=(V,E) \) which consists of a set of the labeled nodes \( V \) and a set of edges \( E \) between them. An edge \( e_{ij} = (v_i, v_j) \) connects \( v_i \) and \( v_j \) if they belong to the same class. Then, the empty nodes are merged. This graph is able to spread the labels in the training set, and is updated with time for the streaming process. A2ING short for Adaptive Incremental Neural Gas is another graph-based semi supervised learning (Bouguelia, Belaïd, & Belaïd, 2013) for classification of a document data stream. In this work, the most informative samples are detected based on an uncertainty measure to be used in classification model along with unlabeled data samples. The labels of these samples are actively requested from a human.
annotator. A2ING is the extension of AING which is a clustering algorithm. AING maintains a model as a graph topology $G$ of document-representatives which are called neurons. Each neuron $y \in G$ is a feature-vector, and AING continuously updates the neurons. A new document $x$ is assigned to a neuron $y$ in the graph $G$ if the distance between $x$ and $y$ is higher (respectively smaller) than a threshold $T_y$. $T_y$ is defined as the mean distance from $y$ to its neighboring neurons. The graph topology is shown in Figure 9.

Figure 9. AING learns the topology of data. Left: different data distribution shapes in a 2 dimensional space. Right: graph topology $G$ of neurons (Bouguelia et al., 2013)

Figure 10. The general A2ING scheme (Bouguelia et al., 2013)
To perform the classification task in A2ING, the labeled instances are needed. In A2ING, algorithm decides what samples to label. The process is shown in Figure 10. At the beginning, several documents are labeled to build an initial model $G$ with some neurons. Each new document then is presented as a feature vector $x$, and shown in (1). The model $G$ is used to classify these samples and to produce the class label $c$ (2). The classification methods provide us with uncertainty level of each sample. If the method is uncertain about the class label of the mentioned samples, this sample is considered as informative and its class label is queried from human expert (3). The classified sample $(x,c)$ is used to update the model $G$ (4). The uncertainty is found using KNN, and for each sample, a probability for belonging to each class is estimated.

(Ru, Andromeda, & Marsono, 2014) proposed a selective self-training for classification of the data stream. In this work, the samples are classified when arrived, and the classification model is updated incrementally. To build a model, there are three steps: offline pre-training, online classification & learning, and cluster reduction. In the pre-training, the base classifier model is prepared and the labeled data is partitioned into $k$ clusters using k-means clustering. Later, the Clustering Features ($CF$) is provided which is an information summary of each cluster.

When a new data sample $x_i$ arrives, its label is predicted by assigning that sample to the nearest cluster, by finding the distance of $x_i$ to the centroid of the clusters as shown in equation (11):

$$D(x_i, \mu_j) = \sqrt{\sum_{m=0}^{d} (x_{im}^m - \mu_{jm}^m)^2}$$ (11)
where $\mu_j$ is the centroid of cluster $j$ which is $\mu_j = \frac{\bar{LS}}{N}$, and $\bar{LS} = \sum_{i=1}^{N} \bar{x}_i$. $\bar{x}_i$ are the $N$ $d$-dimensional samples which are in the cluster $j$.

After obtaining the labels, in order to reduce the classification error, only samples with high prediction confidence are selected for learning. The high confidence prediction is obtained under two criteria: 1) the two nearest neighbor clusters should belong to the same class. 2) the distance to the nearest cluster should be within the average radius of that cluster ($R$). $R$ is calculated as follows:

$$R(\mu_j) = \sqrt{\sum_{m=0}^{d} SS_{\mu_j}^m - \left( \frac{\sum_{m=0}^{d} (LS_{\mu_j})^2}{N_{\mu_j}} \right)}$$

(12)

where $SS = \sum_{i=1}^{N} (\bar{x}_i)^2$.

These selected samples then will be added to the nearest cluster. In the cluster reduction phase, if a cluster is not contributed to the decision making, it will be deleted.

Another semi-supervised approach called Incremental semi-supervised learning via Self-Representative Selection (IS$^3$RS) was proposed by (Feng, Wang, Yang, & Jiao, 2016), which has three phases: self-representative selection, co-training, and final decision. In the self-representatives selection phase, relevant unlabeled samples are selected from the sequential data chunk. In this step, informative samples which represent the whole data should be selected. Selecting these samples can be reduced to the following optimization problem as follows:
\[
\begin{aligned}
\min & \|X - Xs\|_F^2 \\
\text{s.t.} & \|S\|_{row,0} \leq k
\end{aligned}
\] (13)

where \( S \in \mathbb{R}^{N \times N} \) is the coefficient matrix and \( \|S\|_{row,0} \) counts the number of non-zero rows of \( S \). In other words, it is expected to select at most \( k(k \ll N) \) samples in \( X \) that can best represent \( X \).

In the co-training phase, the features of each sample are split randomly into two dependent portions, and the labels of these samples are predicted using KNN classifier. Later, the most confident samples with their class labels are grouped to form a new labeled set. Finally, a classification model is built using this set.

Semi-Supervised Support Vector Machines (SS-SVM) learning techniques have been used for many of the real world applications such as sentiment orientation, intrusion detection, and cognitive psychology (Zhu, 2011). S3VM (Bennett & Demiriz, 1999; Joachims, 1999b; Vapnik & Vapnik, 1998) is one of the most popular one which has been used in many real world applications such as text classification (Joachims, 1999b), bioinformatics (Kasabov & Pang, 2003), and image processing (L. Wang, Chan, & Zhang, 2003). S3VM is an extension of the supervised SVM for the semi-supervised learning. It is based on low density assumption, in which by training a low-density separator, going across the low density region in the space. It simultaneously learns the optimal hyperplane and the labels for the unlabeled data. However, it has been proven in many cases that S3VM may degrade the classification performance (Chapelle, Schölkopf, & Zien, 2006; Chapelle & Zien, 2004; L. Wang et al., 2003; Zhang & Oles, 2000).

Unlike S3VM which uses one optimal low-density separator, (S4VM) was presented by (Li & Zhou, 2011) to address this problem which focuses on the multiple candidate low-density separators. Usually having large number of unlabeled samples and
small amount of labeled samples, there is more than one large-margin low-density separator, and it is hard to decide which one to pick. Although all coincide with the limited labeled samples well, their diversity leads to using a wrong selection that causes huge loss and ultimately results in a poor performance. Moreover, because of the limited training data, the optimal objective value may deviate from the ground-truth. Therefore, S4VM tries to consider all the candidate low-density separators and optimizes the label assignments for the unlabeled samples in the worst case. If the ground-truth label assignment can be recognized by a low-density separator, S4VMs will never degrade the classification performance.

There are several weaknesses with the aforementioned methodologies. For example, the methodologies mentioned in the first category (Gao et al., 2008; Lichtenwalter & Chawla, 2009; Sheng & Haibo, 2009) assume the labels for the samples are available after classification, which is not a true assumption in the real world applications. Since the labeling process is expensive and time consuming. The weakness with the second category (Lindstrom et al., 2010; Lyon et al., 2014) is the authors do not explicitly detect changes, and a new model is generated whether or not drift occurs. In the semi-supervised approaches introduced in the third category, mainly the balanced distribution of classes has been addressed. Also, as it was mentioned before, in some Semi Supervised approaches such as S3VM, using of unlabeled data may decrease the performance (Chapelle et al., 2006; L. Wang et al., 2003)

We propose RLS framework for classification of imbalanced data streams that builds a model dynamically using partially labeled samples when concept drift occurs.
RLS solves three problems which are important for the real world problems. It solves the problem of the classification with:

- the imbalanced data stream,
- using data stream which is partially labeled, and
- assuming concept drift occurs in the data stream.
CHAPTER 3

REDUCED LABELED SAMPLES (RLS) FRAMEWORK

3.1 RLS framework architecture

The general architecture of the RLS framework is shown in Figure 11. RLS uses a chunk-based learning process, and it is assumed that the class distribution of the samples in each chunk is highly imbalanced and all the class labels are available only for the samples in the first chunk of the data stream. RLS has two phases: 1) Initial learning phase and 2) Streaming phase.

1) Initial learning phase:

In this phase, an initial model is built using the data in the first chunk \( C_1 \) of a data stream. This initial model is applied on the upcoming chunks until it gets updated.

2) Streaming Phase:

In this phase, the process is repeated for every new chunk \( C_2, C_3, \ldots, C_n, C_{n+1}, \ldots \) in the data stream. When the next chunk arrives, the following steps are performed:

- **Selection of critical samples:** from the current chunk, the critical samples should be selected. These samples have the important information regarding the variability of the data, and are used to detect the changes in the data.

- **Labeling:** The Critical samples are provided to the expert to be labeled.

- **Applying the Model:** The most updated model is applied on the current chunk
Decision Making (Concept Drift detection): To detect whether or not concept drift has occurred. If it has occurred, the model should be updated. Hence, RLS builds a new model and replaces the old one (Single Model approach), or adds the new model to the ensemble of classifiers (Ensemble of Models approach). To build a new model, RLS either uses supervised approach (default); that is to use the information only from labeled data, or builds a model using Semi-Supervised Learning approaches which use the information from both labeled and unlabeled data samples. In this thesis, RLS with single model is called RLS-S, RLS with ensemble of the models is
recognized as RLS-E, and RLS with semi supervised learning is called RLS-SSL. If
data stream faces sudden feature drift, RLS uses Self Organizing Map. This version is
called RLS-SOM. RLS-SOM also can handle multi class classification, that we call
RLS-Multi (RLS-Multi class classification).
If no concept drift is detected, the model is not updated. Therefore, RLS continues to
read the next chunk of data stream.

- **Model Selection**: A performance comparison is performed between single model,
ensemble of the models, and semi-supervised learning and the model with the highest
performance is selected and is added to the *Updated Model* box to be applied for the
next chunk in the data stream.

3.2 **How the critical samples are selected?**

RLS framework aims to reduce the amount of labeled samples when there is a
need to build a new model. Hence, it finds and labels only those samples which have the
important information regarding variability of the data. If the changes occur close to the
classification model, Support Vectors (SVs) are a good indicator of these changes.
Support vectors are the samples which are closer to the classification model than the
other samples. These samples are the ones that a classifier has the least confidence in
predicting their class labels (Tong & Koller, 2002). Therefore, it is wise to perform
labeling only for the samples which are close to the classification model. Support vectors
can be found using Support Vector Machines (SVM) classification algorithm.

Figure 12(a) shows an example of a data with two classes (triangle and circle). In
Figure 12(b), a classification model (red line) is built for the given data. In Figure 12(c),
support vectors samples which are closer to the classification model than the other samples are shown in bold for each class.

In the streaming data mining based on a chunk-learning process, the effort is to find the informative (Critical) samples of the next chunk, with the help of previous chunk. Therefore, these informative samples are found with the aid of support vectors of the previous chunk. That is, from the current chunk, the closest samples to the support vectors of the previous chunk are selected. This approach would work if the data has the balanced distribution of classes, since selecting the samples close to the support vectors would give the samples from both classes. However, in the case of highly imbalanced data, it may not work.

Assume we have the following imbalanced data samples in the first chunk of data stream (chunk1) as shown in Figure 13(a). Majority class samples are shown in triangle, and minority class samples are shown in circle. In the first chunk, we have the labels for all the samples, and thus a model is built using SVM for this chunk. The support vectors are shown in bold in Figure 13(a). In Figure 13(b), when chunk2 with the unlabeled samples arrives (unlabeled samples are shown in X), we want to find the samples in chunk2 which are close to the support vectors of chunk1. In Figure 13(c), the samples in
the blue area show the support vectors of the chunk1. The red area shows the closest neighborhood to the blue area. As it is shown, since the chunk is imbalanced, we may detect only the samples from the majority class as shown in gray in Figure 13(c). This way the classification task can not be performed as we expect to have the labels for the both classes.

![Figure 13](image.png)

**Figure 13.** An example of detecting samples close to the support vectors for imbalanced chunk of data

Our approach to detect the samples from the both classes is shown in Figure 14.

![Figure 14](image.png)

**Figure 14.** An example of detecting samples close to the support vectors as well as minority class samples for imbalanced chunk of data

Instead of considering only support vectors from chunk1, we also consider all of the minority class samples in the chunk1. In Figure 14(a), a classification model is built
on the chunk1 and the support vectors and the minority class samples in this chunk are shown in bold. In Figure 14(b), chunk2 arrives which contains the unlabeled samples. In Figure 14(c), the blue area shows the support vectors and the minority class samples from chunk1, and red area shows the closest neighborhood to these samples. In this neighborhood, there are some samples from both classes in chunk2 which are shown in gray. These samples are called *Critical samples* and will be labeled and used in building the model.

### 3.3 Introduction to Support Vector Machines (SVM)

RLS uses SVM classification algorithm to detect the support vectors. SVM is a binary classification algorithm which identifies a separating hyperplane that maximizes the margin between the data samples in the different classes (Boser, Guyon, & Vapnik, 1992; Cortes & Vapnik, 1995). SVM will be described as follows:

Assume we are given the training samples \{x_1,...,x_n\} that are vectors in some space \(\chi \subseteq \mathbb{R}^d\). We also have the class labels \(y_i \in \{-1,+1\}\).

In the linear SVM, all the data samples on one side of the hyperplane are labeled as -1, and all the samples on the other side are labeled as +1. The data samples that are closest to the hyperplane are called support vectors. The linear SVM classifier recovers an optimal separating hyperplane which maximizes the margin of the classifier. The classifier for the linear SVM can be represented by \(f(x) = wx + b\), which can be rewritten as:

\[
    f(x) = (\sum_{i=1}^{n} \alpha_i y_i x_i \cdot x_j + b)
\]  

(14)

where \(w\) is the classifier weight vector and can be computed as:
\[ w = \sum_{i=1}^{n} \alpha_i y_i x_i \]  

(15)

\( w \) is used for the prediction, \( b \) is a biased term, and \( \alpha \) is the Lagrange multiplier.

For non-linear classification, SVMs allow projecting the data from the space \( \chi \) to another high dimensional feature space \( F \). Therefore, a set of classifiers can be written in the form of:

\[ f(x) = (\sum_{i=1}^{n} \alpha_i y_i K(x_i, x_j)) \]  

(16)

where \( K(x_i, x_j) \) is the kernel function used for the non-linear classification. Kernel \( K(x_i, x_j) \) takes two inputs and gives their similarity in the feature space \( F \). Two commonly used kernel functions are as follows (Boser et al., 1992; Cortes & Vapnik, 1995):

1- Polynomial kernel function: \( K(x_i, x_j) = (x_i, x_j + 1)^d \)

2- Radial basis kernel function \( K(x_i, x_j) = (\exp(-\gamma \| x_i - x_j \|^2)) \)

where \( \gamma \) is the scale parameter that controls the decay rate of the distance. Depending on the data, either linear or non-linear kernel for the SVM is used.

3.4 RLS-S Algorithm

As shown in Figure 11, RLS-S is a unit in the RLS framework. In RLS-S, if concept drift is detected, a single model is built and the old model is replaced with the new one. The RLS-S procedure is shown in Algorithm 1. The data stream is analyzed as follows:
Initial Learning Phase: First chunk ($C_1$) of data stream arrives which is the $TrainSet$ (line 1). An initial classification model $m$ is built on $TrainSet$ using Decision Tree. SVM is also applied on the $TrainSet$ to find Support Vectors (line 2). Please note that in the RLS framework, SVM is used to provide us with the support vectors. For the classification purposes, either SVM, or any other classification algorithm can be used. Next, all the support vectors ($SV$) and the minority class samples ($MinS$) of the $TrainSet$ are kept and we refer to these as the Borrow ($B$) samples (line 3).

<table>
<thead>
<tr>
<th>Notations:</th>
<th>CS_A: Critical Samples A</th>
<th>SV: Support Vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MinS: Minority class Samples</td>
<td>m: model</td>
</tr>
<tr>
<td></td>
<td>MajS: Majority class Samples</td>
<td>B: Borrow</td>
</tr>
<tr>
<td></td>
<td>DT: Decision Tree</td>
<td>C: Chunk</td>
</tr>
</tbody>
</table>

**Input:**
- Streaming data in chunks: $C_1C_2...C_nC_{n+1}......$

**Goal:**
- Classifying imbalanced streaming data with partial labeling

// **Initial Learning phase**
1. $TrainSet \leftarrow C_1$
2. Build $m$ on $TrainSet$ using DT, SVM on $TrainSet$ to get $SV$s
3. $B \leftarrow \{SV_m \cup MinS_{C_1}\}$: $SV = \{SV_1, SV_2, ..., SV_n\}$, $MinS = \{MinS_1, MinS_2, ..., MinS_m\}$

// **Streaming phase**
for $C_i$, $i=2: \infty$ do
4. $D \leftarrow$ Euclidean Distance ($B$, $C_i$)
5. $CS_A \leftarrow$ top closest samples from $C_i$ to $B$ using $D$ in Neighborhood
6. Obtain true labels for $CS_A$
7. Obtain predicted labels for $CS_1$ using $m$
8. Find classification performance of $CS_A$
9. if ($MinS \in CS_A \& MajS \in CS_A$) then
   10. if (Performance $C_i$ - Performance $C_{i+1}$ > thr) then
       //conditional drift
       11. Build $m$ using $CS_A$
       12. $B \leftarrow \{SV_m \cup MinS_{C_i}\}$ //Update Borrow
   end
12. else if ($MinS \notin CS_A$)
13. Increase Neighborhood, go to 5
end

Algorithm 1. RLS-S to address the conditional concept drift and maintaining a single model
Streaming phase: In this phase, all other chunks in data stream are analyzed. When the current chunk ($C_i$) arrives, we want to find samples in the $C$, which are closest samples to the $B$. These samples are the Critical Samples (CS) which will be used for labeling. The detailed procedure of finding these samples (CS) is as follows:

Let $C = [s_1 s_2 \ldots s_n]$ be the current Chunk with $n$ samples and $B = [b_1 b_2 \ldots b_m]$ be the Borrow with $m$ samples. We find the Euclidean distance of each sample in the Chunk to each sample in the Borrow (line 4). So let $s_i b_j$ denotes the Euclidean distance between $s_i$, and $b_j$, where $i = 1, \ldots, n$, and $j = 1, \ldots, m$. Therefore, we have the following matrix $D$ as follows:

$$D = \begin{bmatrix} s_1 b_1 & \ldots & s_1 b_m \\ \vdots & \ddots & \vdots \\ s_n b_1 & \ldots & s_m b_m \end{bmatrix}_{n \times m} \tag{17}$$

In the matrix $D$, $s_1 b_1$ means the distance of the first sample in $C$ to the first sample in $B$. Suppose the columns of the matrix $D$ is sorted independently in ascending order. Let $D_s$ be the resulting matrix to maintain the original positioning of the elements of $D$. $D_s$ contains the neighborhood samples of the $B$ as follows:

$$D_s = \begin{bmatrix} (s_i b_1)_r & \ldots & (s_i b_m)_r \\ \vdots & \ddots & \vdots \\ \ldots & \ldots & \ldots \end{bmatrix}_{n \times m} \tag{18}$$

(i and $i'$ may not be equal)

Where $r$ denotes the current position of the elements in each column. In $D_s$, For finding the closest samples of the Chunk to the Borrow samples, we define a threshold and we call it Neighborhood size ($N_s$). We split $D_s$ to different Neighborhood sizes.

Let $N_s=20, 40, \ldots, 120$, where $N_s$ is the number of rows in $D_s$, and Let $D_s^{N_s}$ be the matrix obtained from the first (Top) $N_s$ rows of the matrix $D_s$.
\[ D_s^{N_s} = \begin{bmatrix} (s_ib_1)_r & \cdots & (s_ib_m)_r \\ \vdots & \ddots & \vdots \\ \cdots & \cdots & \cdots \end{bmatrix} \quad \text{Nsxm} \]  
\( (i \text{ and } i' \text{ may not be equal}) \)

Now in \( D_s^{N_s} \), we want to compute the occurrence of \( s_i \) for \( i = 1, \ldots, n \) per row and per column. It is computed as follows:

\[ Occurrence_{s_i} = \sum_{r=1}^{k} \sum_{j=1}^{m} O_{ij}^r \]  
\( \text{(20)} \)

where

\( O_{ij} = 1 \) if sample \( s_i \) observed in \( b_j \), and \n\( O_{ij} = 0 \) if sample \( s_i \) is not observed in \( b_j \)

for

\( N_s = 20, 40, \ldots, 120 \)
\( i = 1, \ldots, n \)
\( j = 1, \ldots, m \)

For example, if we want to find the occurrence of \( s_1 \), for \( N_s = 20 \), it will be calculated as follows:

\[ Occurrence_{s_1} = \sum_{r=1}^{20} \sum_{j=1}^{m} O_{1j}^r \]

\[ Occurrence_{s_1} = (O_{11}^1 + O_{12}^1 + \ldots + O_{1m}^1) + \]
\[ (O_{11}^2 + O_{12}^2 + \ldots + O_{1m}^2) + \]
\[ \cdots \]
\[ (O_{11}^{20} + O_{12}^{20} + \ldots + O_{1m}^{20}) \]

We then sort the \( Occurrence_{s_i} \) in descending order. We call this vector Sorted Occurrence (SO) as follows:

\[ SO = \text{sort}\left(Occurrence_{s_i}\right) \]  
\( \text{(21)} \)
The samples on the top of the SO are Critical Samples (CS) and are chosen to get labeled. Therefore:

\[
\text{Critical Samples} = L\% \times \text{size(SO)}
\]  
(22)

Let’s say we want to label 10% of the samples in SO. If the size of SO is 100, 10% will give 10 of them. These 10 samples will be provided to expert to get labeled.

To clarify better, we show an example of the whole process. Let’s say we have a Chunk of 100 samples \( C = [s_1 s_2 \ldots s_{100}] \) and a Borrow vector of 5 samples \( B = [b_1 b_2 \ldots b_5] \). The matrix \( D \) is calculated as follows:

\[
D = \begin{bmatrix}
    s_1 b_1 & \cdots & s_1 b_5 \\
    s_2 b_1 & \cdots & s_5 b_5 \\
    \vdots  & \ddots & \vdots \\
    s_{100} b_1 & \cdots & s_{100} b_5 
\end{bmatrix}
\]

By sorting the matrix \( D \) per column, we will get the matrix \( D_s \) as follows:

\[
D_s = \begin{bmatrix}
    (s_{21} b_1) & (s_8 b_2) & (s_{12} b_3) & (s_{33} b_4) & (s_{31} b_5) \\
    (s_{8} b_1) & (s_{21} b_2) & (s_9 b_3) & (s_{40} b_4) & (s_{50} b_5) \\
    (s_{12} b_1) & (s_9 b_2) & (s_{3} b_3) & (s_{50} b_4) & (s_{9} b_5) \\
    (s_{9} b_1) & (s_{12} b_2) & (s_{31} b_3) & (s_{63} b_4) & (s_{12} b_5) \\
\vdots  & \vdots  & \vdots & \vdots  & \vdots \\
\vdots  & \vdots  & \vdots & \vdots  & \vdots 
\end{bmatrix}
\]

\( N_s=20 \)

We keep the first 20 rows of the \( D_s \) (Neighborhood size \( N_s=20 \)) and we count the occurrence of \( s_i \) for \( (i = 1, \ldots, 100) \) per row and per column. The results are as follows:

<table>
<thead>
<tr>
<th>Samples</th>
<th>( s_8 )</th>
<th>( s_9 )</th>
<th>( s_{12} )</th>
<th>( s_{21} )</th>
<th>( s_{31} )</th>
<th>( s_{33} )</th>
<th>( s_{40} )</th>
<th>( s_{50} )</th>
<th>( s_{63} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Occurrence</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
The occurrence shows sample $s_8$ is close to the 3 of the Borrow samples, sample $s_9$ is close to the 4 of the Borrow samples, and etc. For those samples which have not been observed in the first 20 rows of the $D_s$, the occurrence is 0. Our goal is to find samples in $C$ which are close to the majority of the $B$ samples. Therefore, we sort the occurrence in descending order, and remove those 0 occurrences. The results are in $SO$:

<table>
<thead>
<tr>
<th>Samples</th>
<th>$s_{12}$</th>
<th>$s_9$</th>
<th>$s_8$</th>
<th>$s_{21}$</th>
<th>$s_{31}$</th>
<th>$s_{33}$</th>
<th>$s_{50}$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SO$</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>...</td>
</tr>
</tbody>
</table>

If we assume the size of $SO$ is 50, labeling 10% would give the top 5 samples in $SO$. Therefore, the samples which have the highest priority to be given to the expert for labeling are in the following order: $s_{12}$ $s_9$ $s_8$ $s_{21}$ $s_{31}$. In Figure 15, Borrow ($B$) Samples are shown in the blue area named as $b$. The neighborhood of size 20 around the $B$ is shown in the grey area, and the chunk samples in the neighborhood are shown in $S$. Based on the above methodology, the critical samples are shown in red. These samples are close to the majority of the $B$ samples ($b_1, b_2, b_3, b_5$). The other samples which are outside the gray area, are the chunk samples which are not located in the closest neighborhood of the Borrow samples.

![Figure 15. Gray area shows closest neighborhood to Borrow samples](image)

40
When CS are found based on the above methodology (we call them Critical SamplesA (CSA)) for the first neighborhood (line 5), their true labels are obtained from the expert (line 6). The predicted labels of CSA are also obtained using m, and the classification performance is calculated (line 7-8). If CSA contains samples from both majority and minority class (line 9), we can then check for the conditional concept drift. If the performance dropped more than a threshold (line 9), a new model is build based on the CSA (line 11). From the current chunk Ci, the Support Vectors (SV) and the minority class samples (MinS) are kept as the new Borrow samples (B) (line 12), and the process will be continued for the next chunks. However, if we could not find minority samples in the CSA (line 12), we increase the neighborhood size (Ns) until we find enough samples of minority class for labeling in that neighborhood (line 13).

The CSA provides the information regarding the concept drift. Labeling the CSA ensures that the labeling effort is spent wisely on the samples. The CSA includes both majority and minority class samples and the combination of this set of samples is almost balanced. If this set is not balanced, to maximize the performance, ADASYN(He, Bai, Garcia, & Li, 2008) over sampling technique is used before building the new model. Therefore RLS-S helps in the classification of concept drifting and imbalanced data stream, and with using small amount of samples in building the classification model.

3.5 Experiments

3.5.1 Data sets

The experimentations were performed using MATLAB. For the purpose of experimentations and also for the comparison with other approaches, two synthetic data
sets from (Ditzler & Polikar, 2013), and six real world data sets from UCI repository (Asuncion & Newman, 2015), and (NOAA, 2012) were used. To simulate the data stream, the data was partitioned into several imbalanced chunks. These data sets are as follows:

3.5.1.1 Synthetic data streams

- Rotating CheckBoard dataset

  This data set is a generalization of the classical nonlinear XOR problem. Two parameters control the experiment. One is the angle of the rotation: \( \alpha \in \{0, \pi/2, \pi, 3\pi/2, 2\pi\} \) and the other one is the relative side length of each square with respect to the total length of the board. The rotation makes the data to have gradual and reoccurring concept (Ditzler & Polikar, 2013). The data set has 2 features, 630044 samples, 2 classes (0, 1), and the minority class is 5% of the data set. A snapshot of this data set is shown in Figure 16.

![Rotating checkboard experiments \( \alpha \in \{0, \pi\} \) (Ditzler & Polikar, 2013)](image)

- Shifting HyperPlane dataset

  This data set was first proposed by (Street & Kim, 2001). It later was modified in (Ditzler & Polikar, 2013) to induce the imbalance class distributions. This data has
feature drift since the distribution of the data is changing. The data has 3 features, where the first two are relevant and the third one is noise. An instance is assigned to class “1” if the sum of its relevant features \((x_1 + x_2)\) falls below a threshold \(\theta\), and assigned to class “0”, otherwise. The threshold is changing in the order of \(\{4 \rightarrow 7 \rightarrow 4 \rightarrow 7\}\). The data has 343658 samples, 2 classes \((0, 1)\), and the minority class is 7% of the total size of data. Data also contain 5 percent class label noise. A snapshot of this data set when the threshold is changing from 4 to 7 is shown in Figure 17.

![Figure 17. Shifting HyperPlane data when a) threshold=4 b) threshold=7](image)

### 3.5.1.2 Real world datasets

- **Adult**

  This data set contains 30162 samples, 104 features, and 2 classes. The imbalance ratio in this data is 0.25. The task is to predict whether or not a person makes over 50K a year.

- **CoverType**

  The task in this data set is to predict forest cover types or types of vegetation from cartographic variables. This data set has 581012 samples, 54 features, and has 7 classes.
To make the data appropriate for the imbalanced learning, we selected subset of the data, including class 2 samples and class 4 samples. Class 2 has the maximum number of samples and class 4 has the minimum number of samples. This new version includes 82200 samples, 54 features, and two classes (2, 4). The imbalance ratio is 0.03.

- **Ozone**

  Another data set from UCI repository is Ozone, which is a streaming problem concerning local ozone peak prediction, that is either based on one hour peak set (Ozone-1h) or based on eight hour peak set (Ozone-8h). Those data were collected from 1998 to 2004 at the Houston, Galveston and Brazoria area. Ozone-1h has 2536 samples, and Ozone-8h has 2534 samples. Both data has 73 features, and 2 classes. The minority class for the Ozone-1h, and Ozone-8h is 3% and 7% respectively.

- **Electricity pricing dataset**

  This is a benchmark data set for concept drift analysis. The data set provides time and demand fluctuations in the price of electricity in the New South Wales(NSW), Australia (Harries, 2012). The task is to predict if the NSW electricity demand will be higher or lower than Victoria (VIC) electricity demand. The data originally has 8 features, and before running the experiments the two features related to the price should be removed. The data also was under sampled to have the minority class as approximately as 5.5 % of total data size. The new data has 17038 sample and, 6 features, and 2 classes (1, 0).
• The weather dataset

This data set consists of weather data from hundreds of locations around the world (NOAA, 2012). The classification task is whether or not the rain precipitation was observed on each day. In this experiment, the “Offutt Air Force Base in Bellevue, Nebraska” was chosen which has over 50 years worth of data, providing not only cyclical seasonal changes, but also possibly long-term climate change. The data has 18159 samples, 8 features and 2 classes (1,0). Minority class is 30% of the total size of the data. All the data set characteristics are summarized in the Table 1.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Number of samples</th>
<th>Number of features</th>
<th>Number of classes</th>
<th>Imbalance (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckBoard</td>
<td>630044</td>
<td>2</td>
<td>2</td>
<td>5%</td>
</tr>
<tr>
<td>HyperPlane</td>
<td>343658</td>
<td>2</td>
<td>2</td>
<td>7%</td>
</tr>
<tr>
<td>Adult</td>
<td>30162</td>
<td>104</td>
<td>2</td>
<td>25%</td>
</tr>
<tr>
<td>CoverType</td>
<td>82200</td>
<td>54</td>
<td>2</td>
<td>3%</td>
</tr>
<tr>
<td>Ozone-1h</td>
<td>2536</td>
<td>73</td>
<td>2</td>
<td>3%</td>
</tr>
<tr>
<td>Ozone-8h</td>
<td>2534</td>
<td>73</td>
<td>2</td>
<td>7%</td>
</tr>
<tr>
<td>Electricity</td>
<td>17038</td>
<td>8</td>
<td>2</td>
<td>5.5%</td>
</tr>
<tr>
<td>Weather</td>
<td>18159</td>
<td>8</td>
<td>2</td>
<td>30%</td>
</tr>
</tbody>
</table>

3.5.2 Evaluation measures of the classification performance

For machine learning problems with imbalanced streaming data, classification accuracy is not a good measure since majority class samples, will dominate the results (Godase & Attar, 2012). For example, if we have a dataset which has 1% minority class samples, and 99% majority class samples, a simple default strategy of guessing the majority class samples would result in 99% correctly classifying majority class samples, however 0% for the minority class samples. Therefore, the evaluation metrics other than accuracy should be used which will provide a better evaluation of the results for imbalanced streaming data.
There are measures, which are used specifically for the imbalanced learning problems (Godase & Attar, 2012; Lyon et al., 2014). In the thesis, F-measure, G-mean, and AUC ROC are used because these measures have the ability to interpret the performance of the minority class as well, and they are described as follows:

### 3.5.2.1 F-measure

F-measure is a measure of the test's accuracy in the binary classification. It combines the trade-offs of the precision and recall, and outputs a single number reflecting the goodness of a classifier in the presence of the minority class (Buckland & Gey, 1994). Precision is the fraction of the retrieved samples that are relevant, while recall is the fraction of the relevant samples that are retrieved (true positive rate). F-measure reaches its best value at 1 when the precision and recall both are 1. F-measure is computed as follows:

\[
F\text{-}\text{measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]  

(23)

Where precision and recall are defined as follows:

\[
\text{precision} = \frac{TP}{TP + FP}
\]  

(24)

\[
\text{recall} = \frac{TP}{TP + FN}
\]  

(25)

and:

TP is number of items correctly labeled as belonging to the minority class,

TN is number of items correctly labeled as belonging to the majority class,

FP is number of items incorrectly labeled as belonging to the minority class, and

FN is number of items incorrectly labeled as belonging to the majority class.
3.5.2.2 G-mean

G-mean is a metric that measures the performance of a learning algorithm for both of the classes. This measure tries to maximize accuracy for both of the classes at the same time (Kubat, Holte, & Matwin, 1998). That is to increase the number of minority class samples correctly recognized as well as the number of majority class samples correctly recognized. G-mean is defined as follows:

\[
Gmean = \sqrt{\text{recall} \times \text{specificity}} = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}} \tag{26}
\]

where the \text{recall} is the true positive rate and \text{specificity} is the true negative rate.

3.5.2.3 AUROC

It is the Area Under the ROC Curve. ROC (Receiver Operating Characteristic) is a curve that is created by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. In Figure 18, ROC Curve is shown in blue, and the Area Under the Curve (AUC) is shown in gray. The closer the ROC curve is to the upper left corner, the AUC is higher which leads to higher accuracy of the test. AUC reaches its best value at 1.

![Figure 18. ROC Curve](image.png)
3.5.2.4 Measure of change in performance: \( P_{\text{drop}} \)

For detecting the concept drift, we define Performance drop \( (P_{\text{drop}}) \) as follows:

\[
P_{\text{drop}} = \left( \text{Performance}_{(CC)} \right) - \left( \text{Performance}_{(PC)} \right)
\]

(27)

\[ P_{\text{drop}} > \text{thr} \]

Where

\( CC = \text{Current Chunk} \)

\( PC = \text{Previous Chunk} \)

\( \text{thr} = \text{a certain threshold set for detecting the concept drift} \)

The latest model is applied on the current chunk and its F-measure and G-mean is compared which the previous chunk. If for any of them or both, the difference is more than a threshold, we consider concept drift has occurred. The value for the threshold is obtained via experiments in the section 3.5.

3.6 Experiments

- **Threshold value for detecting the concept drift**

  As stated before, a new model is built if \( P_{\text{drop}} \) drops above a certain threshold. We are to find the best threshold for the concept drift detection. In RLS, we update the model when either the F-measure or the G-mean performances drop more than a certain threshold (3%, 7%, 10%, 15%, 20%, 30%, 70%). We show the F-measure results of our experimentation on HyperPlane data set for the first few chunk of data stream. The results on other data sets follow the same trend. The results are shown in Figure 19.
Figure 19. The F-measure results shown for the first 20 chunks when the threshold varies from 3% to 70% for HyperPlane data set.

Figure 19 demonstrates a significant drop for the threshold of 30% and 70%. At threshold 3%, the F-measure results are higher, since the model is updated frequently, regardless of concept drift. This means that if the number of misclassified samples is very few, which could be the error of any classifier, a new model is built. On the other hand, threshold of 7%, 10%, and 15% obtained the comparable results. Results show that the lower threshold, updates the model frequently, which may increase the overhead of the system. This is not significantly detecting any changes, only blindly rebuilds a new model. The higher threshold, however, does not update the model when it is needed, causes a gradual drop in performance. Therefore based on our experimentations, the threshold for detecting the changes is good to be chosen in the range of [0.07-0.15]. In the rest of our experimentation we use threshold of 10% (0.1) for detecting the concept drift.
• **Experimental Results**

An extensive experimentation has been conducted on each data set for different chunk sizes (400-600-800-1000-1200). Figure 20 and Figure 21 shows F-measure when 100% labeling, as well as F-measure when partial (p%) labeling was used in building the model. The optimal labeling percentages which results in equivalent performance as 100% labeling are reported in Table 2. The labeling percentages are averaged over different chunk sizes for each data set.

Table 2. Labeling percentage for each data set

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>Labeling %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ozone-1h</td>
<td>25%</td>
</tr>
<tr>
<td>Ozone-8h</td>
<td>15%</td>
</tr>
<tr>
<td>Electricity</td>
<td>31%</td>
</tr>
<tr>
<td>Weather</td>
<td>20%</td>
</tr>
<tr>
<td>CoverType</td>
<td>5%</td>
</tr>
<tr>
<td>Adult</td>
<td>25%</td>
</tr>
<tr>
<td>CheckBoard</td>
<td>28%</td>
</tr>
<tr>
<td>HyperPlane</td>
<td>25%</td>
</tr>
</tbody>
</table>

To handle the concept drift, if the classification performance drops more than 10%, we build a new model and add it to the ensemble. The initial value for the neighborhood size is set to 20. If the framework could not obtain enough minority samples for labeling in this neighborhood, we increment the neighborhood size by 20 continuously until we found them.

Figure 20 Shows the F-measure results for the Ozone-1h, Ozone-8h, Elect, and Weather data set when partial (p%) and 100% labeled samples are used in building the model. As demonstrated in Figure 20, the chunk size does not have a very significant effect for Electricity and Weather data sets, but for the Ozone-1h, and Ozone-8h the performance is improved at chunk size 800 or higher.
The results also indicate for the Ozone-1h, and Ozone-8h, the equivalent quality of performance as 100% labeling have been obtained, when partial labeling of 25% and 15% used for the Ozone-1h, and Ozone-8h respectively.

In Figure 21, the F-measure results are shown for CoverType, Adult, CheckBoard, and HyperPlane data sets. In CoverType data set, the equivalent performance as 100% labeling has been achieved for all the chunk size. Adult data set achieved equivalent performances as 100% with only 25% labeling at chunk size 1000.
HyperPlane data set also obtained good performance with 25\% labeling mostly at chunk sizes 400, 600, and 1200.

The CheckBoard data set has a very complex decision boundary to learn and as the results show, this data set needs more labeled samples (28\%) in order to build a model which has equivalent performance as 100\% labeling. We compared the average results over all the chunk sizes for the partially labeled versus the fully labeled data using Wilcoxon test(Wilcoxon, 1950). The results indicate that there is no significant difference.
between the results from partially labeled and fully labeled at $\alpha=0.01$. Overall, the results of our experiments show the same quality of performance has been obtained using less labeling.

3.7 Discussion

In this chapter, the RLS framework for classification of the imbalanced data stream has been introduced. RLS framework addresses concept drift in data streams and builds a new model using partial labeled samples. RLS aims to maintain the quality of performance with less labeling, since labeling is very time consuming and expensive. In RLS, Support Vector Machine (SVM) is used to address the conditional drift. The idea of using SVM is motivated by that only those samples which are close to the classification model would carry useful information needed to build a classification model. As shown in the RLS framework in Figure 11, RLS-S module builds a new model when concept drift occurs, and the old model is replaced with the new one. Experimental results for RLS-S on 2 synthetic data sets (HyperPlane and CheckBoard) and 6 real world data sets (Electricity, Ozone-1h, Ozone-8h, Weather, Adult, and Cover Type) leads us to the four important conclusions applicable for the future analysis of the real world streaming data: 1) the best value of the threshold for detecting concept drift is in the range of [0.07-0.15], 2) the initial value for the neighborhood size is set at 20. If the algorithm could not obtain enough samples to label, it will be incremented by 20. 3) It shows with labeling only up to 30% of the samples in the stream, RLS-S would achieve the results comparable with 100% labeling. The experimental results prove that our method decreases the labeling,
and saves expert time and effort. Building an accurate model with less labeling in data stream environment is very desirable in the real world applications.
CHAPTER 4

USING AN ENSEMBLE OF CLASSIFIERS FOR CLASSIFICATION OF IMBALANCED STREAMING DATA

4.1 Introduction

The RLS-S is based on one single classifier/model. As mentioned earlier, in the RLS-S, an initial model is built based on the first chunk, and this model is periodically rebuilt based on the new training samples in the stream if the concept drift occurs. In the RLS-S, we do not keep the previous models in the memory and each time a new model is built, the old one is forgotten. However, this solution may not be efficient if an old trend in the system appears again. In this case, a new model should be built which causes extra cost, whereas if the models from the past have been maintained, a lot of cost to the system could have been reduced. One way to keep and combine the models in the memory is using an ensemble of classifiers.

There are many research activities proposed by (Chen & He, 2009; Gao et al., 2007; Kolter & Maloof, 2007; Lichtenwalter & Chawla, 2010; Muhlbaier & Polikar, 2007; Street & Kim, 2001) in the domain of imbalanced data stream, which use an ensemble of classifiers. In these approaches, instead of building a single model at the time of concept drift, several models are built and combined into an ensemble of classifier to improve the performance (Kantardzic, 2011).
The main idea of the ensemble learning is to train a number of individual learners in order to learn a target and combine their predictions (Polikar, 2006). It is theoretically proved that using an ensemble of classifiers outperforms a single one if the classifiers are independent of each other, and the accuracy of each is greater than 0.5 (Dietterich, 2000; Kantardzic, 2011).

We added an ensemble of classifiers to our RLS framework to improve the performance of the classification model. That is called RLS–Ensemble (RLS-E). RLS-E uses Dynamic Weighted Majority (DWM) (Kolter & Maloof, 2007) to obtain the final class labels from ensemble. In DWM (Kolter & Maloof, 2007) an online learner such as naïve Bayes is used to train an ensemble in which the final voting is obtained by dynamic weighted majority voting. The algorithm works in a way that the voting weight of each classifier is set to 1 when created, and is reduced when that classifier misclassifies a sample. The model is removed from ensemble when it’s weight falls below a certain threshold.

We had one additional hypothesis: There could be one individual model in the ensemble which may produce much higher prediction than the ensemble. Therefore, in RLS-E, in addition to obtaining classification results from combined output of ensemble models, we also used the results from each individual model as well. In our experiments, RLS-E selects an individual model, if its prediction outperforms ensemble’s output.

One parameter, when maintaining an ensemble of models, is the number of the models which should be maintained in an ensemble. Due to memory restriction, we cannot keep all the models and after some times, we should remove some of them. In (Hansen & Salamon, 1990), it is argued that ensembles containing ten classifiers is
sufficient for reducing the error rate. We also perform experiments for varied ensemble size to determine the effect of ensemble size on performance.

4.2 Maintaining an ensemble of the models

RLS-E maintains an ensemble of the models instead of maintaining a single model. However, RLS-E does not solely rely on the prediction from ensemble, since there could be one individual model in the ensemble with the performance higher than the one from ensemble. Hence, in RLS-E the classification results are obtained in two different ways. a) RLS-E obtains a classification from combined output of all the models in the ensemble, and b) It also evaluates the results from each individual model in the ensemble. RLS-E uses one individual model’s classification if its performance outperforms the ensemble output.

4.2.1 Classification results from ensemble

The ensemble used in the RLS-E is using a similar procedure as used in Dynamic Weighted Majority (DWM) (Kolter & Maloof, 2007), but for a chunk based learning framework. For each sample in the current chunk, a classification is obtained from each individual model in the ensemble. The weights of those models which produce the incorrect classification are reduced by multiplying by 0.5. Regardless of the correctness of the classification, each individual model’s classification and the weights are used to compute a weighted sum for each class. The class with the most weight is set as the ensemble classification output.
Table 3 shows an example of calculation of the class labels in ensemble for the samples $s_1$ and $s_2$. Here, $w_1$ is referred to the weight for the $s_1$, and $w_2$ is referred to the weight for $s_2$. For $s_1$ and $s_2$, their class labels are obtained from all the models $m_1$ to $m_5$.

Let’s say the true label of $s_1$ is 1 and true label of $s_2$ is 0. In case of $s_1$, the $w_1$ for all the $m_1$ to $m_4$ is 1 since they predicted the class accurately. For $m_5$, since the predicted class is wrong, its weight is reduced to 0.5. The same process is done for $s_2$ as well.

Table 3. Example of calculation for two samples

<table>
<thead>
<tr>
<th>model</th>
<th>Classification of $s_1$</th>
<th>$w_1$</th>
<th>Classification of $s_2$</th>
<th>$w_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>m2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>m3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>m4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>m5</td>
<td>0</td>
<td>0.5</td>
<td>1</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 4. Final class calculation

<table>
<thead>
<tr>
<th>Classes</th>
<th>Sum of Weights- $s_1$</th>
<th>Sum of Weights- $s_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 0</td>
<td>0.5</td>
<td>1+1+1 =3</td>
</tr>
<tr>
<td>Class 1</td>
<td>1+1+1+1 =4</td>
<td>0.5+0.5 =1</td>
</tr>
</tbody>
</table>

For each class, a label is calculated based on weighted sum and is shown in Table 4. The final class for each sample is the one which has the highest score. Therefore, for $s_1$ is 1 and for $s_2$ is 0. The performance obtained from classification results of ensemble is called $P_E$.

After obtaining the class labels, the weights of all samples in the entire chunk are added to produce a final weight for each model in the ensemble. The weights are then scaled uniformly so that after transformation, the maximum weight is 1. These weights are carried over through analyzing the data stream. When the next chunk arrives, after
class prediction and calculating the weights of classifiers for this chunk, the new weights and the old weights are averaged to produce an updated weight for each model in the ensemble.

If a concept drift has occurred, a new model with weight of 1 is added to the ensemble. To consider the recency factor as well, the newly added model slightly should maintain higher weight than the other ensemble members. Therefore, all the previous models weights are multiplied by a decaying factor $\alpha=0.95$. This factor is used to decrease all other models’ weights smoothly over the time, and to maintain only the newly added model with the weight of 1.

The RLS-E ensemble technique considers both models’ history, by transferring the weights throughout analyzing the data stream, as well as considering the recent model with slightly higher weight compared to the old models in the ensemble.

4.2.2 Classification results from each individual model in the ensemble

Each individual model in the ensemble produces classification results on the current chunk. The performance of all these classifications is compared together and the highest one is selected:

$$m = \arg \max_{i=1,2,...,k} \{\text{performance}_i\}$$  \hspace{1cm} (28)

Here $k$ is the number of the models maintained in the ensemble, and $m$ is the model index in which the performance is maximized. The classification obtained from model $m$ is chosen over ensemble classification if its performance outperforms the $P_E$. 
4.3 RLS-E Algorithm

RLS-E is shown in Algorithm 1a which is a modification of Algorithm 1 in chapter 3. In this version, the classification performance is found using our Dynamic Model Selection as shown in Algorithm 2.

<table>
<thead>
<tr>
<th>Notations:</th>
<th>CSₐ: Critical Samples A</th>
<th>SV: Support Vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MinS: Minority class Samples</td>
<td>m: model</td>
</tr>
<tr>
<td></td>
<td>MajS: Majority class Samples</td>
<td>B: Borrow</td>
</tr>
<tr>
<td>E: Ensemble</td>
<td></td>
<td>C: Chunk</td>
</tr>
<tr>
<td>DT: Decision Tree</td>
<td></td>
<td>k: number of models</td>
</tr>
</tbody>
</table>

**Input:**
- Streaming data in chunks: C₁C₂…CₙCₙ₊₁......

**Goal:**
- Classifying Imbalanced Streaming data with partial labeling

// Initial Learning phase
1. E←0, k←0
2. TrainSet←C₁
3. Build m₁ on TrainSet using DT, SVM on TrainSet to get SVs, m← m₁
4. E← m₁∪E
5. B←{SV∪MinS}: SV={SV₁,SV₂,…,SVₙ}, MinS={MinS₁, MinS₂,…, MinSₘ}

// Streaming phase
6. for C₁,…, i=2:∞ do
7. D← Euclidean Distance(B, Cᵢ)
8. CSₐ ← top closest samples from Cᵢ to B using D in Neighborhood
9. Obtain true labels for CSₐ
10. Obtain predicted labels for CSₐ using E
11. Find classification performance of CSₐ in Algorithm 2: Dynamic Model Selection
12. if (MinS∈ CSₐ & MajS∈CSₐ)
13.     if (Performance Cᵢ - Performance Cᵢ₋₁ > thr) then
14.         //conditional drift
15.             Build mₖ using CSₐ, k++
16.             E← mₖ∪E , B←{SVₘₖ∪MinSₗₐ}
17. else if (MinS∉CSₐ )
18.         // Increase Neighborhood, go to 5
19. end
20. end

Algorithm 1a. RLS-E to address the conditional concept drift and maintaining an ensemble of models

The RLS-E works in a same way as RLS-S, with some additional differences:
In RLS-E, an ensemble of classifiers is maintained, so the ensemble \((E)\) and the number of models \((k)\) is initialized to 0 (line 1).

When initial model is built, it is added to Ensemble (line 4).

When concept drift occurs (line 13), a new model is built using Critical Samples \(_A\) \((CS_A)\) (line 14), and is added to an ensemble of classifiers (line 15).

Classification performance of the \(CS_A\) is obtained from our Dynamic Model Selection as shown Algorithm 2 (line 11).

\[ \text{Algorithm 2: Dynamic model selection} \]

\[
\begin{align*}
\text{Input:} & \quad -\text{Set of samples} \\
\text{Output:} & \quad -\text{Classification performance} \\
& \quad \text{// Use Dynamic Weighted Majority to get classification performance from ensemble} \\
1. & \quad P_E \leftarrow \text{F-measure}(E) \\
& \quad \text{// find performance of each individual model in the ensemble} \\
2. & \quad \text{for } m=1 \text{ to } k \\
3. & \quad \quad [P]_k \leftarrow \text{F-measure}(m) \\
& \quad \text{end} \\
4. & \quad [P]_k \leftarrow P_E \cup [P]_k \\
& \quad \text{// choose the model with maximum performance} \\
5. & \quad m = \arg \max_{i=1,2,...,k} \{[P]_k\} \\
6. & \quad \text{Result } \leftarrow \text{classification performance from } m
\end{align*}
\]

In Algorithm 2, classification of \(CS_A\) is obtained from:

- Ensemble of classifiers using dynamic weighted majority, and its performance is preserved in \(P_E\) (Alg2, line 1),

- Each individual model in the ensemble, and the results are maintained in array \([P]_k\) in which \(k\) is the number of the models in the ensemble (line 2-3).

After evaluation, the model \(m\) with the highest performance is selected (Alg2, line 4-5), and the classification performance from this model is selected (Alg2, line 6).
4.4 Experimental results and analysis

4.4.1 Data sets

For the analysis, the same data sets from Chapter 3 were used. These data sets are 2 synthetic data sets: HyperPlane, and CheckBoard (Ditzler & Polikar, 2013), and 6 real world data sets: Ozone-1h, Ozone-8h, Weather, Electricity, Adult, and CoverType retrieved from (Asuncion & Newman, 2015), and (Ditzler & Polikar, 2013).

4.4.2 Results

4.4.2.1 Dynamic model melection results

Our hypothesis has been tested on all the data sets. For these data sets, we tested our framework with:

1) maintaining only the results from ensemble,
2) maintaining only the results from one individual model which produces higher performance than other models,
3) The highest results between 1) and 2).

If for a chunk size, an individual model performance dominates the results from the ensemble, its performance is selected over ensemble performance. Table 5 reports the average F-measure results over all the chunks. In this table, “ensemble” results shows the average F-measure when only the results from ensemble is chosen; “individual model” performance shows when the model with highest performance among all the models in the ensemble is chosen; and the “selection of highest” reports the average F-measure results when the performance of individual model is chosen over ensemble.
Table 5. The F-measure for “selection of highest”, “ensemble”, and “individual model” for all the data sets

<table>
<thead>
<tr>
<th>Data sets</th>
<th>selection of highest</th>
<th>ensemble</th>
<th>individual model</th>
</tr>
</thead>
<tbody>
<tr>
<td>HyperPlane</td>
<td>0.63</td>
<td>0.63</td>
<td>0.53</td>
</tr>
<tr>
<td>CheckBoard</td>
<td>0.55</td>
<td>0.54</td>
<td>0.41</td>
</tr>
<tr>
<td>Elect</td>
<td>0.31</td>
<td>0.33</td>
<td>0.13</td>
</tr>
<tr>
<td>Adult</td>
<td>0.80</td>
<td>0.80</td>
<td>0.58</td>
</tr>
<tr>
<td>Weather</td>
<td>0.81</td>
<td>0.81</td>
<td>0.54</td>
</tr>
<tr>
<td>Ozone-1h</td>
<td>0.73</td>
<td>0.72</td>
<td>0.53</td>
</tr>
<tr>
<td>Ozone-8h</td>
<td>0.83</td>
<td>0.84</td>
<td>0.57</td>
</tr>
<tr>
<td>CoverType</td>
<td>0.97</td>
<td>0.98</td>
<td>0.94</td>
</tr>
</tbody>
</table>

The results in Table 5 indicates that the “individual model” selection produces performance less than the ensemble, and our hypothesis for the “selection of highest”, shows the obtained results are not significantly different from “ensemble” results. Therefore, for the rest of our experiments, the classification results based on the ensemble is used.

4.4.2.2 Effect of chunk size and labeling percentage on performance

An extensive experimentation has been conducted on each data set for different chunk sizes (400-600-800-1000-1200) while a number of labeling percentages (5-10-15-20-25-30-50-70-100) have been tried to find the effect of these parameters on the quality of the performance. To handle the concept drift, the following threshold from previous experiments has been established: if the classification performance dropped more than 10%, a new model is built and is added to the ensemble.

To perform the experiments for all the labeling percentages (5-10-15-20-25-30-50-70-100), each of these is fed to the system as the required labeling budget. Then, it is examined if within that labeling percentage, samples from both classes can be obtained.
Since the data sets are imbalanced, this could happen that the minority class is missing in lower labeling percentages (such as 5%). If the minority class is missing, the labeling percentage is increased (we call real labeling) until we find minority class samples to be able to perform classification task. The results for the Weather, Adult, HyperPlane, and CoverType data sets are shown in Figure 22 and Figure 23.

Figure 22. F-measure results for the a) Weather b) Adult data sets for chunk sizes (400,600,800,1000,1200), and labeling percentages (5,10,15,20,25,30,50,70,100)
In Figure 22(a), the F-measure is reported for the weather data set. For this data set, at lower labeling such as 5%, the F-measure is also lower compared to 100% labeling for all the chunk sizes. However, increasing the labeling causes an increase in performance. The model could achieve the same quality of performance while using few labeling percentages (20%, 25%) compared to 100% labeling for all the chunk sizes. Figure 22(b) also reports the F-measure for the Adult data set.

Figure 22(b) shows achieving the same quality of performance as 100% labeling is possible using at least 20% labeling. The experimental results also demonstrate that the chunk size does not have a very significant effect on the performance when labeling starts at 20%, for both data sets. In Figure 23, the F-measure results are shown for the HyperPlane and CoverType data sets. The results indicate that the framework needs 20% and 5% partial labeling for the HyperPlane and CoverType respectively, to build an accurate model and to maintain the quality of performance in all the chunk sizes.

The experimental results for the four data sets Weather, Hyperplane, CoverType and Adult data sets show with labeling up to 20%, the same quality of the results as 100% labeling have been obtained. These four data sets also remained in the budget, meaning that the required labeling budget is equal to the real labeling. In other world, RLS framework could find samples from both classes, even at the lower labeling percentage 5%, while the imbalance characteristic of the data sets makes it a more challenging problem. The results, however, for the Electricity, Ozone-1h, Ozone-8h and CheckBoard data sets show these data sets exceeded the required labeling budget.
Figure 23. F-measure results for the a) HyperPlane b) CoverType data sets for chunk sizes (400,600,800,1000,1200), and labeling percentages (5,10,15,20,25,30,50,70,100).

Figure 24 shows the required labeling and the real labeling for these data sets. If required labeling is equal to the real labeling, the points are located in the black line, which shows the data set stays within the labeling budget. If real labeling is greater than
the required labeling, the raised percentage is shown in the left top of each figure, above the diagonal line.

Figure 24. Labeling percentage for the a) Electricity b) Ozone-1h c) Ozone-8h), and d) CheckBoard data sets

Figure 24(a) shows for the Electricity data sets, the real labeling in higher than the required labeling, and this data set needs more labeling to find samples from minority class in order to build an accurate model. For instance, if the required labeling is 5%, and the chunk size is 400, the real labeling has been increased to 20%. However, at higher labeling percentages (10% or more) and when the chunk size is larger (1000 or 1200) the real labeling is getting closer to the required labeling, as it gets closer to the black line.
The labeling budget for the CheckBoard, Ozone-1h, and Ozone-8h demonstrates similar trend meaning these data set exceeded the required labeling budget. Figure 24(b) and Figure 24(c) shows the real labeling for the Ozone-1h, and CheckBoard data sets has been increased, and this is more visible at the chunk size 400; while increasing the chunk size, causes to decrease the real labeling for building an accurate model. This issue is less visible in the Ozone-8h as shown in Figure 24(c). Although at Ozone-8h, the real labeling also has exceeded the budget, however, the increase is not much compared to Electricity, CheckBoard, and Ozone-1h data sets. Overall, the results show for our RLS-E setting, higher chunk sizes such as 1000, 1200 needs less labeling to find samples from both classes, in order to build an accurate model. The F-measure results of these data sets are summarized in the Table 6, Table 7, Table 8, and Table 9.

Table 6. F-measure results of the Electricity data set for different chunk sizes

<table>
<thead>
<tr>
<th>Required Labeling Budget</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>50</th>
<th>70</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Labeling Budget</td>
<td>18</td>
<td>19</td>
<td>21</td>
<td>24</td>
<td>27</td>
<td>30</td>
<td>50</td>
<td>70</td>
<td>100</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.406</td>
<td>0.351</td>
<td>0.304</td>
<td>0.422</td>
<td>0.415</td>
<td>0.350</td>
<td>0.409</td>
<td>0.405</td>
<td>0.450</td>
</tr>
<tr>
<td>Real Labeling Budget</td>
<td>13</td>
<td>16</td>
<td>18</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>50</td>
<td>70</td>
<td>100</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.357</td>
<td>0.432</td>
<td>0.453</td>
<td>0.384</td>
<td>0.339</td>
<td>0.492</td>
<td>0.432</td>
<td>0.387</td>
<td>0.463</td>
</tr>
<tr>
<td>Real Labeling Budget</td>
<td>9</td>
<td>11</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>50</td>
<td>70</td>
<td>100</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.304</td>
<td>0.309</td>
<td>0.509</td>
<td>0.419</td>
<td>0.446</td>
<td>0.485</td>
<td>0.442</td>
<td>0.416</td>
<td>0.518</td>
</tr>
<tr>
<td>Real Labeling Budget</td>
<td>6</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>50</td>
<td>70</td>
<td>100</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.309</td>
<td>0.398</td>
<td>0.396</td>
<td>0.455</td>
<td>0.475</td>
<td>0.430</td>
<td>0.464</td>
<td>0.509</td>
<td>0.519</td>
</tr>
<tr>
<td>Real Labeling Budget</td>
<td>7</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>50</td>
<td>70</td>
<td>100</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.28</td>
<td>0.34</td>
<td>0.34</td>
<td>0.31</td>
<td>0.40</td>
<td>0.41</td>
<td>0.40</td>
<td>0.44</td>
<td>0.48</td>
</tr>
</tbody>
</table>
In these tables, the “Required Labeling Budget” is the labeling budget at the beginning, and the “Real Labeling Budget”, is the actual labeling that should be done in order to build an accurate model. Table 6 shows the F-measure results of the Electricity data sets. The Required Labeling (5, 10, 15, 20, 25, 30, 50, 70, 100) are shown on the top of the table, and for each chunk size, the Real Labeling is shown as well.

Table 6 demonstrates the Real labeling percentage is higher than Required labeling budget at lower chunk sizes, and it will be remained within the budget at higher chunk sizes. The F-measure results for Electricity data sets shows this data needs 25-30% labeling to build an accurate model as 100% labeling for the chunk sizes (800-1200) as shown in bold. Table 7 shows for the Ozone-1h data set, at chunk size (800-1200), the Required labeling is equal to the Real labeling (15%), and the F-measure results is equivalent to 100% labeling as shown in bold.

Table 7. F-measure results of the Ozone-1h data set for different chunk sizes

<table>
<thead>
<tr>
<th>ChunkSize</th>
<th>Required Labeling Budget</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>Real Labeling Budget</td>
<td>0.696</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.731</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.768</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.611</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.688</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.739</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.762</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.742</td>
</tr>
<tr>
<td>600</td>
<td>Real Labeling Budget</td>
<td>0.738</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.777</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.729</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.770</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.803</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.691</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.688</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.880</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.823</td>
</tr>
<tr>
<td>800</td>
<td>Real Labeling Budget</td>
<td>0.635</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.750</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.806</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.829</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.822</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.835</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.870</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.876</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.873</td>
</tr>
<tr>
<td>1000</td>
<td>Real Labeling Budget</td>
<td>0.610</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.680</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.787</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.780</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.749</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.795</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.801</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.821</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.815</td>
</tr>
<tr>
<td>1200</td>
<td>Real Labeling Budget</td>
<td>0.63</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.78</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.85</td>
</tr>
</tbody>
</table>
Likewise as is shown in Table 8, the Real labeling percentage is 20% for the Ozone-8h data sets, where the same quality of F-measure results (shown in bold) has been obtained when compared with 100% labeling.

Table 8. F-measure results of the Ozone-8h data set for different chunk sizes

<table>
<thead>
<tr>
<th>Required Labeling Budget</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>50</th>
<th>70</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChunkSize=400</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real Labeling Budget</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-measure</td>
<td>0.533</td>
<td>0.606</td>
<td>0.596</td>
<td>0.615</td>
<td>0.613</td>
<td>0.646</td>
<td>0.729</td>
<td>0.766</td>
<td>0.794</td>
</tr>
<tr>
<td>ChunkSize=600</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real Labeling Budget</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-measure</td>
<td>0.712</td>
<td>0.683</td>
<td>0.782</td>
<td>0.774</td>
<td>0.758</td>
<td>0.817</td>
<td>0.864</td>
<td>0.816</td>
<td>0.844</td>
</tr>
<tr>
<td>ChunkSize=800</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real Labeling Budget</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-measure</td>
<td>0.731</td>
<td>0.852</td>
<td>0.784</td>
<td>0.817</td>
<td>0.857</td>
<td>0.857</td>
<td>0.901</td>
<td>0.906</td>
<td>0.885</td>
</tr>
<tr>
<td>ChunkSize=1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real Labeling Budget</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-measure</td>
<td>0.743</td>
<td>0.780</td>
<td>0.818</td>
<td>0.814</td>
<td>0.879</td>
<td>0.841</td>
<td>0.893</td>
<td>0.927</td>
<td>0.885</td>
</tr>
<tr>
<td>ChunkSize=1200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real Labeling Budget</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-measure</td>
<td>0.79</td>
<td>0.86</td>
<td>0.87</td>
<td>0.89</td>
<td>0.87</td>
<td>0.92</td>
<td>0.91</td>
<td>0.95</td>
<td>0.94</td>
</tr>
</tbody>
</table>

The results for the CheckBoard data set is shown in table 9. For chunk size 400, increasing the labeling causes improvement in performance, and finally at 50%, the performance is not significantly different from the one at 100% labeling. The reason is the decision boundary in CheckBoard is very complex, and increasing the labeling definitely helps in building an accurate model. However, the results show increasing the chunk sizes causes a decrease in labeling to build an accurate model as 100% labeling. For example, for the chunk size 1000, and 1200, 20% labeling could lead to results not
significantly different from 100% (shown in bold). Moreover, at higher chunk sizes, the real labeling is equivalent to the required labeling budget.

4.4.2.3 Effect of chunk size on Running Time

Having larger chunks of data may allow the system to process data at higher rates. However, it also increases the end-to-end latency between receiving data chunk and getting the generated results from it. Hence, it is necessary for the system to operate at a chunk size that minimizes latency while ensuring that the data is processed as fast as it is received (Das, Zhong, Stoica, & Shenker, 2014). The effect of chunk size on running

<table>
<thead>
<tr>
<th>ChunkSize=400</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>50</th>
<th>70</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-measure</td>
<td>0.548</td>
<td>0.488</td>
<td>0.464</td>
<td>0.542</td>
<td>0.552</td>
<td>0.589</td>
<td>0.659</td>
<td>0.680</td>
<td>0.696</td>
</tr>
<tr>
<td>ChunkSize=600</td>
<td>7</td>
<td>11</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>50</td>
<td>70</td>
<td>100</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.454</td>
<td>0.518</td>
<td>0.570</td>
<td>0.570</td>
<td>0.644</td>
<td>0.621</td>
<td>0.695</td>
<td>0.720</td>
<td>0.705</td>
</tr>
<tr>
<td>ChunkSize=800</td>
<td>6</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>50</td>
<td>70</td>
<td>100</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.468</td>
<td>0.452</td>
<td>0.505</td>
<td>0.591</td>
<td>0.604</td>
<td>0.622</td>
<td>0.623</td>
<td>0.641</td>
<td>0.654</td>
</tr>
<tr>
<td>ChunkSize=1000</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>50</td>
<td>70</td>
<td>100</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.381</td>
<td>0.370</td>
<td>0.521</td>
<td><strong>0.554</strong></td>
<td>0.526</td>
<td>0.535</td>
<td>0.546</td>
<td>0.558</td>
<td>0.562</td>
</tr>
<tr>
<td>ChunkSize=1200</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>50</td>
<td>70</td>
<td>100</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.47</td>
<td>0.36</td>
<td>0.41</td>
<td><strong>0.57</strong></td>
<td>0.52</td>
<td>0.52</td>
<td>0.49</td>
<td>0.47</td>
<td>0.641</td>
</tr>
</tbody>
</table>

Overall, the results indicate higher chunk sizes causes to stay in the labeling budget, while building an accurate model with partial labeling.
time for all the data sets are shown in Figure 25. The times are normalized in the range of [0,100].

![Figure 25. Running time vs chunk size](image)

Figure 25 demonstrates the running time is higher at larger chunk sizes such as 1000 or 1200 for all the data sets. For the Adult, Hyperplane, and CheckBoard, and CoverType data sets the running time is a little higher in chunk size 400. This could be because when the chunk size is small, the algorithm may not learn the current concept properly and should build a new model, which results in increasing the running time. For these data sets, the running time is decreased at 600 and starts to increase after that. The running times demonstrated at the Figure 25 show an ascending order from lower chunk sizes to larger chunk sizes. Although less running time is associated with the lower chunk sizes, lower chunk size may miss the current concept. Higher chunk sizes may also operate at higher running time. Therefore, empirical results demonstrate that the desired chunk size varies with data rates and other operating conditions.
4.4.2.4  Effect of ensemble size on performance

A series of experiments have been performed with varying the ensemble size, to see how it affects the performance. In Figure 26 the experimental results for the Electricity, HyperPlane, and CheckBoard data sets are shown. For all other selected data sets, the number of models is a few, since either the data is not drifting much, or the data set is small. Here, the results are shown for the chunk size 400, while maintaining a varying number of models in the range of [1:25] in ensemble. To preserve the ensemble size, the models with lower weights are removed from ensemble.

![Figure 26. Effect of ensemble size on F-measure for the a)HyperPlane, and b)Electricity and CheckBoard data sets](image)

Figure 26 shows increasing the ensemble size generally increases the performance. For HyperPlane, CheckBoard, and Electricity data sets, increasing the number of models from 1 to 10 has caused an increase in F-measure performance. However, having the ensemble size in the range of [11:25] does not have a very significant effect on performance. This shows maintaining the higher number of models does not necessarily helps, and the results of having ensemble size of 10 is not significantly different when more than 10 models are maintained in the ensemble.
Memory restrictions would require the framework to maintain limited number of models. Figure 26 suggests that maintaining 10 to 15 models in the ensemble is close to optimal, and that higher number of models show limited improvement.

4.5 Experimental comparison with other methods

The experimental results of RLS-E are compared with several methods in the domain of imbalanced data stream. Firstly, it is compared with our previous work (RLS-S) (Arabmakki, Kantardzic, & Sethi, 2014) in which only a single model is maintained. RLS replaces the old model with new one when concept drift occurs. Second, RLS-E is compared with some other ensemble methods in the domain.

4.5.1 Comparison of RLS-E with RLS-S

The comparison results of G-mean and F-measure for RLS-S (which maintains a single model) and RLS-E (which maintains an ensemble of classifiers) are shown in Figure 27.

![Figure 27](image_url)

a) F-measure  
b) G-mean

Figure 27. a) F-measure and b) G-mean results of RLS-E and RLS-S for Adult, Electricity, HyperPlane, Weather, Ozone-1h, Ozone-8h, CheckBoard, and CoverType data sets
In Figure 27, the blue line, and red line show the results for the RLS-S and RLS-E respectively. The results are shown when the chunk size is 800, the labeling percentage is up to 20%, and the number of models in the ensemble is 10.

For all the data sets, the F-measure and G-mean is higher in RLS-E compared to the RLS-S except for CoverType which maintained the same quality of performance. Based on the results shown at Figure 27 for CoverType data set, it seems that the data is not drifting much and maintaining one single model contributes to the results no significantly different from the results of ensemble. Overall, the comparison results prove that RLS-E increases the classification performance when compared to RLS-S which relies only on prediction from a single model.

4.5.2 Comparison of RLS-E with other ensemble methodologies

We compared the RLS-E with three ensemble based methodologies in the domain of streaming data. These methods are introduced in Section 2 and a brief explanation is given as follows:

1- In (Gao et al., 2007), the minority class sample are accumulated from previous chunks and are added to the current chunk. The majority class samples are also undersampled to build a balanced training set. This method used a simple average voting (SA) to combine probability outputs from $k$ models as follows:

$$f^E(x) = \frac{1}{K} \sum_{i=1}^{k} f^i(x)$$

(29)

where $f_i(x)$ is the posterior probability output of each classifier $C_i$ for each sample $x$. 

75
2- In (Lichtenwalter & Chawla, 2010), along with propagated minority class samples from previous chunks, miss-classified majority class samples are also accumulated. Authors used a combination of Hellinger Distance (HD) and Information Gain (IG) to address concept drift. This method was explained in details in section 2. Ensemble weights are applied in two different ways:

- Weights are applied to each model in the ensemble using Hellinger Distance (HD),
- Weights are applied to each model using (HDIG) distance which is a combination of Hellinger Distance (HD) and Information Gain (IG).

In Table 10, the experimental results of F-measure and AUC for RLS-E with 20% labeling, and other techniques such as SA, HD, and HDIG with 100% labeling are reported. The results are reported for four of the data set Adult, CoverType, Ozone-1h, and Ozone-8h which were used in other methodologies as well. The chunk size is 1000 and 10 models were maintained in the ensemble.

Table 10. Comparison results of RLS-E with other techniques (Lichtenwalter & Chawla, 2010)

a) F-measure

<table>
<thead>
<tr>
<th></th>
<th>RLS-E (20%)</th>
<th>SA(100%)</th>
<th>HD(100%)</th>
<th>HDIG(100%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult</td>
<td>0.79</td>
<td>0.631</td>
<td>0.631</td>
<td>0.631</td>
</tr>
<tr>
<td>CoverType</td>
<td>0.96</td>
<td>0.751</td>
<td>0.752</td>
<td>0.751</td>
</tr>
<tr>
<td>Ozone-1h</td>
<td>0.78</td>
<td>0.104</td>
<td>0.100</td>
<td>0.100</td>
</tr>
<tr>
<td>Ozone-8h</td>
<td>0.81</td>
<td>0.072</td>
<td>0.142</td>
<td>0.142</td>
</tr>
</tbody>
</table>

b) AUC ROC

<table>
<thead>
<tr>
<th></th>
<th>RLS-E (20%)</th>
<th>SA(100%)</th>
<th>HD(100%)</th>
<th>HDIG(100%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult</td>
<td>0.81</td>
<td>0.892</td>
<td>0.892</td>
<td>0.892</td>
</tr>
<tr>
<td>CoverType</td>
<td>0.98</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
</tr>
<tr>
<td>Ozone-1h</td>
<td>0.89</td>
<td>0.784</td>
<td>0.795</td>
<td>0.794</td>
</tr>
<tr>
<td>Ozone-8h</td>
<td>0.9</td>
<td>0.748</td>
<td>0.747</td>
<td>0.746</td>
</tr>
</tbody>
</table>
Table 10 shows for Ozone-1h, and Ozone-8h data sets, there is a significant improvement in both AUC and F-measure. For CoverType, and Adult, significant improvement achieved in F-measure, while no significant difference is reported in AUC. In RLS-E, only 20% of the samples are used in building the model, whereas other techniques used 100% labeling. The experimental results prove RLS-E is an efficient ensemble framework which can produce high classification performance using partial labeled samples in building the model.

4.6 Discussion

In this chapter, an ensemble method for classification of the imbalanced streaming data called RLS-E was proposed. The class labels are predicted in two different ways:

1) we applied the ensemble of the models on the current chunk. The outputs of all the models are combined using Dynamic Weighted Majority (DWM) to produce ensemble prediction.

2) we evaluated each individual model in the ensemble on the current chunk. After evaluation, the classification result from the model with higher performance is selected.

Our hypothesis was based on the fact that there may be one model in the ensemble that produces a higher prediction than the ensemble prediction. We compared our classification performance from: 1) ensemble, 2) One individual model which produces higher results among others, and 3) Choosing highest between 1) and 2). Our results indicate that one individual model always produce lower results compared to the ensemble results. Moreover, the results show there is no significant difference when we choose one individual model classification results over ensemble results.
Our results indicate that RLS-E achieved high classification performance with partial labeled samples. Our experimental results and comparison of RLS-E with RLS-S on 8 data sets show that RLS-E improved the performance at least 10% for the majority of data sets, especially for the F-measure. Comparison results with other state of the art fully labeled techniques indicate that RLS-E improved F-measure and AUC performance using 20% of the data samples in building the model. Based on the experiments, we propose several recommendations for using the RLS-E as follows:

- For all the data sets, the smaller chunk size needed more labeling. Hence, the recommended chunk size for the RLS-E setting is suggested to be within the range [800-1200]. Usually there should be a trade-off between chunk size and labeling percentage. We found this range good enough since it is not that big to pose additional running time to the system, while requiring less labeling when analyzing the data stream.

- The recommended ensemble size is in the range of [10-15]. The experiments prove that maintaining additional models in the ensemble may not contribute to significant improvement, and the highest error reduction will take place when 10 to 15 models are maintained in the ensemble.

- The recommended initial labeling percentage is suggested to be within the range of [15%-20%]. Based on our experiments, the lower labeling percentages such as 5% or 10% usually exceeded the labeling budget. The labeling percentage within the range [15%-20%] has been found as a good initial labeling percentage to the system.
CHAPTER 5

APPLYING SELF ORGANIZING MAP (SOM) TO TACKLE THE SUDDEN FEATURE DRIFT

5.1 Introduction

In the RLS-S and RLS-E framework, we addressed the problem of the conditional drift. As mentioned earlier, if the changes occur in the classification model, there is a conditional drift in the data stream. In case of conditional drift, the samples close to the hyperplane are the ones which classifier has least confident in predicting their class labels (Tong & Koller, 2002). These samples are from both minority and majority classes.

However, data may suffer from sudden feature drift. That is the distribution suddenly shifts far away from classification model. This issue is shown in the Figure 28. Assume we have the first chunk of the data stream in $C_1$ as shown in Figure 28(a). After building the model, expert tries to label the samples close to the classification model in $C_1$. Now assume we get the second chunk of data stream ($C_2$) as shown in Figure 28(b) in a part of space which is far from the classification boundary. In this case, those already labeled samples are not helpful in building the model. In this case the entire feature space should be searched to find enough samples to build an accurate model. Since RLS is based on partial labeled samples, the goal is by reducing the labeling as much as possible, finding samples from both classes to build a new model while achieving the same classification performance when compared to fully labeled setting.
If data has sudden feature drift, and is imbalanced, specially finding the minority class samples for labeling is a challenging task. RLS-S and RLS-E fails when there is sudden feature drift, since the neighborhoods around the classification model should be searched to find such samples, and that will increase the labeling dramatically.

In order to address the sudden feature drift and locate the minority class samples in the feature space with less labeling, we employed the Self-Organizing Map (SOM)(Kohonen, 1982). SOM is an unsupervised learning algorithm which may project the high dimensional data into a two-dimensional regular spacing in a hexagonal or rectangular grid. The map is useful in analyzing the patterns in the input space (Kohonen, 1990). SOM is used in many applications such as DNA classification (Naenna, Bress, & Embrechts, 2003), speech recognition (Venkateswarlu & Kumari, 2011), etc. Compared to other clustering algorithms, SOM is shown to be scalable for very large and complex data sets in terms of both the number of records and the number of clusters (Singh, Singh, & Dubey, 2010).

Figure 28. Feature drift between Chunk 1 (C_1) and Chunk 2 (C_2) of the streaming data
This extension to the RLS-E is called RLS-SOM (RLS-Self Organizing Map). It maintains an ensemble of classifiers and addresses both changes close and far to the classification boundary, while building a new model using partial samples.

The idea of using SOM is based on the approach proposed by (Patra & Bruzzone, 2014), in which authors used SOM to classify an image data set using only informative samples. In this approach, SVM is used to train an initial labeled set of samples. By using the trained SVM, the unlabeled samples which classifier has low confidence in predicting their class labels are identified. These samples then are mapped to a two dimensional space using SOM. Based on the cluster assumptions, authors claimed samples in the low dense region of the map are the samples close to the decision boundary and are the most informative ones. However, this assumption may not work in many real world applications involving the imbalanced class distributions. In RLS-SOM, we use SOM in a data stream environment where the class distribution is imbalanced. For simplicity, hereafter, we call the sudden feature drift as feature drift in this thesis.

5.2 Proposed RLS-SOM

5.2.1 SOM helps in detecting the minority class samples

In RLS-SOM approach, SOM provides a way to find the minority class samples. An illustration of the process is shown in the Figure 29. In Figure 29(a), our approach for when the changes occur close to classification model is shown. This is the same approach as the one in RLS-S and RLS-E. In Figure 29(a1), first chunk of data stream (chunk1) arrives. The majority class samples are shown in circles, and minority class samples are shown in stars. A classification model (black line) is built on chunk1 called model1. The margin area for model1 is shown in the gray area enclosed with dotted lines.
In Figure 29(a2) next chunk (chunk2) arrives with a change in the data distribution. In chunk2 the samples closest to the margin area of the model1 should be selected. In Figure 29(a2) the margin area of model1 is shown in the gray area. There are
some chunk2 samples in the margin area of model1 which are shown in bold. These samples are selected for labeling and appear to have the most information regarding variability of the data. Since the change is gradual, in the margin area we have samples from both minority and majority classes. In Figure 29(a), these samples are used to build a new classification model 2 (black line). The new margin area for model2 is shown in gray area.

In Figure 29(b), our approach for sudden changes far from classification model is shown. In Figure 29(b1), first chunk of data stream (chunk1) arrives. In Figure 29(b2) when chunk2 arrives, it contains changes compared to chunk1 (the position of the minority class samples has totally changed). In this case, data samples in the margin area only contain majority class samples shown in bold. Since no minority class sample is detected, SOM is applied to look for the minority class samples in the feature space. In Figure 29(b3), SOM project chunk2 into a 2-D map. SOM preserves the topology of the input vector and therefore, each node in the map contains several samples belonging to either majority or minority classes. Labeling starts from center of each node until minority class samples are found. The selected samples in the map are shown in red bold and are used in building a new model. The algorithm is described in details in section 3.2.

5.2.2 RLS-SOM Algorithm

RLS-SOM handles both conditional and feature drift. It maintains an ensemble of classifiers and produces the predictions based on Dynamic Weighted Majority (Kolter & Maloof, 2007). The Algorithm1 from section 4.3 has been slightly changed to Algorithm1b.
Analyzing the data stream begins through a chunk-based learning algorithm. In initial learning phase, the ensemble \(E\) and the number of models \(k\) is initialized to 0, and an initial Decision Tree \(DT\) model \(m_1\) is built on the TrainSet which is first chunk of data stream \(C_1\), hence the \(k\) is incremented. SVM is also applied on TrainSet for detecting support vectors (line 1-3).

| Notations: | \(d_1, d_2\): map dimensions | SV: Support Vectors |
| CS\(_A\): Critical Samples A | m: model |
| CS\(_B\): Critical Samples B | B: Borrow |
| MinS: Minority class Samples | \(K\) : number of models |
| MajS: Majority class Samples | C: Chunk |
| DT: Decision Tree | E: Ensemble |

**Notations:**
- \(d_1, d_2\): map dimensions
- \(CS_A\): Critical Samples A
- \(CS_B\): Critical Samples B
- \(MinS\): Minority class Samples
- \(MajS\): Majority class Samples
- \(DT\): Decision Tree
- \(SV\): Support Vectors
- \(m\): model
- \(B\): Borrow
- \(K\) : number of models
- \(C\): Chunk
- \(E\): Ensemble

**Input:**
- Streaming data in chunks: \(C_1, C_2, \ldots, C_n, C_{n+1}, \ldots\)

**Goal:**
- Classifying Imbalanced Streaming data with partial labeling

**// Initial phase**
1. \(E \leftarrow 0, k \leftarrow 0\)
2. TrainSet \(\leftarrow C_1\)
3. Build \(m_1\) on TrainSet using DT, SVM on TrainSet to get SVs , \(m \leftarrow m_1\)
4. \(E \leftarrow m_1 \cup E\)
5. \(B \leftarrow \{SV \cup MinS\}: \ SV = \{SV_1, SV_2, \ldots, SV_n\}, \ MinS = \{MinS_1, MinS_2, \ldots, MinS_m\}\)

**// Iterative phase**
6. for \(i = 2, \infty\) do
7. \(D \leftarrow \text{Euclidean Distance}(B, C_i)\)
8. \(CS_A \leftarrow \text{top closest samples from } C_i \text{ to } B \text{ using } D \text{ in Neighborhood}\)
9. Obtain true labels for \(CS_A\)
10. Obtain predicted labels for \(CS_A\) using \(E\)
11. Find classification performance of \(CS_A\)
12. if (\(MinS \in CS_A \land MajS \in CS_A\))
13. \(\text{if (Performance } C_i, \text{- Performance } C_{i+1} > \text{thr) then}\)
14. \(\text{//conditional drift}\)
15. \(E \leftarrow m_k \cup E, B \leftarrow \{SV_{mk} \cup \text{MinS}_{Ci}\}\)
16. else if (\(MinS \notin CS_A\) ) // feature drift
17. **CS\(_B\)** selection in \textit{Algorithm3: SOM Sample Selection}
18. Build new model \(m_k\) using \(\{CS_A \cup CS_B\}\), \(k++\)
19. \(E \leftarrow m_k \cup E, B \leftarrow \{SV_{mk} \cup \text{MinS}_{Ci}\}\)

**Algorithm 1b. RLS-SOM algorithm**
The Decision Tree model is added to the ensemble ($E$) (line 4). From the first chunk, the support vectors ($SVs$) and the Minority class Samples ($MinS$) are preserved to an array called Borrow ($B$) (line 5).

In the streaming phase, all other chunks are processed (line 6). In order to detect the conditional drift, array $B$ is used. From current chunk $C_i$, the closest samples to the $B$ based on distance matrix $D$ are selected, and is called Critical samples$_A$ ($CS_A$) (line 8). Then, the true labels of $CS_A$ is obtained from the expert (line 9), and its prediction is obtained from Ensemble $E$ (line 10) and its classification performance is calculated (line 11).

When the classification performance of $CS_A$ is obtained, the framework checks whether or not a conditional drift has occurred:

- If the $CS_A$ contains samples from both minority ($MinS$) and majority ($MajS$) classes, that means changes in the data, if any, exist close to the decision boundary. In this case, if there a significant drop in performance compared to the last chunk, it is claimed that conditional drift has occurred (line 13). Therefore, samples in the $CS_A$ are the most informative ones since these are the closest samples to the decision boundary. These samples are used to build a new model $m_k$. The model is then added to ensemble $E$ (line 14). The model is added to the ensemble and the $B$ is updated (line 15).

- If the $CS_A$ did not contain any sample from minority class (line 16), this means the changes occurred far from the decision boundary. In such case, the feature drift is detected, and we need to look for the minority class samples in some other parts of the feature space.
Finding minority class samples in case of feature drift is very hard, since these samples could be located in any part of the feature space. Partial labeling in such case is also very challenging, as we do not have access to the class labels of all the samples. Finding minority class samples through methods such as random selection of the samples completely fails since for the high imbalanced ratio, with high probability it selects majority class samples. Therefore, in order to detect the minority class samples, Self-Organizing Map (SOM) is used (line 17). The sample selection procedure is shown in Algorithm 3.

In such case, the current chunk is projected to a 2-D map using SOM. The map provides the information regarding the center point of each node. Starting from node1, the closest sample to the center of the node is selected and is labeled (Alg3, line 1-2):

<table>
<thead>
<tr>
<th>Algorithm 3. SOM sample selection</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> - Set of samples</td>
</tr>
<tr>
<td><strong>Output:</strong> - Selected samples for labeling</td>
</tr>
<tr>
<td>1. for j=1 to d1×d2</td>
</tr>
<tr>
<td>2. S ← find and label closest sample to centroid</td>
</tr>
<tr>
<td>3. if (S ∈ MinS)</td>
</tr>
<tr>
<td>4. [Temp]_n← [S]_n, n&lt;\text{\textTheta} // label other closest sample from current node</td>
</tr>
<tr>
<td>5. CS ← S ∪ [Temp]_n</td>
</tr>
</tbody>
</table>

- If the selected sample belongs to minority class samples (Alg3, line 3), with high probability this node contains samples from the minority class, under the assumption that minority class appears in compact clusters. Hence, other closest samples to the node center, up to a threshold \text{\textTheta}, are selected and labeled (Alg3, line 4). These samples are saved in an array called Temp. If we reach the
threshold \( \Theta \) and no minority class sample was obtained, we just keep the previously selected samples (Alg3, line 5) and we move to the next nodes.

- If the center sample belongs to the majority class, this means most probably this node is related to the majority class samples. Therefore, we keep this sample to and we move to the next node (Alg3, line 5).

By the time we exit the map, we have \( CS_B \) which contains both samples from minority and majority classes. Afterward, a new model is retrained using an array contains \( CS_A \) (obtained previously) and \( CS_B \) (line 18). The new model is added to the ensemble \( E \), and the \( B \) is reconstructed as well(line 19). The \( CS_A \) in this case contains representative samples, which demonstrate a good sense of distribution. The \( CS_B \) contains both the informative samples, which are samples close to decision boundary as well as some representative samples. The process is continued for all other chunks of the data stream.

With searching the entire map, we may obtain diverse samples from all over the feature space, which may contain samples from decision boundary for both of the classes. Since SOM preserves the topology of the high dimensional data, samples in each node usually belong to a common class. Since the center is the representative of the samples in each node, labeling samples closest to the center point should reveal what class that node belongs to. Therefore, SOM helps us to find the diverse samples from both majority and minority classes.
5.3 **Experimental results and analysis**

5.3.1 **Data sets**

The experiments ran over several synthetic data sets and several publicly available data sets from UCI repository (Asuncion & Newman, 2015) as used in Chapter 3. The data sets are: Adult, Ozone-1h, Ozone-8h, CoverType, Electricity, Weather, Checkboard, and Hyperplane. To simulate the data stream, the data was partitioned into several imbalanced chunks. For detailed explanations of these data sets please refer to Chapter 3.

5.3.2 **Results**

5.3.2.1 **The effect of SOM map size on the performance**

Self-Organizing Map can have different dimensions. We used different number of map size in our experiments to see the effect of having different map sizes on labeling percentage. Large maps (such as 15*15) produce a large number of small but "compact" nodes and therefore the samples assigned to each node are quite similar. Small maps (such as 5*5) produce less but more generalized nodes. Since the data sets are imbalanced, and we are looking for the compact clusters of minority classes, the larger map could help rather than small ones. Figure 30 shows the results of our experiments for Electricity, HyperPlane, and CheckBoard data sets when chunk size is 400.

The results show the labeling percentage is usually more when the map size is larger, specially this is more visible in the Electricity and CheckBoard data sets. That is because the minority class are more scattered when the map is large (for example 15*15) and the algorithm should look for more nodes to find them. That increases labeling.
The results show for the map size of 7*7 and 10*10 the labeling is less compared to the other map sizes. Although the map size is dependent of the data and the applications, having the map size in the range of [7*7-10*10] for the high imbalance data set is recommended.

5.3.2.2 Experimental results

The F-measure results for the 8 data sets are shown in Figure 31. The results are reported for the chunk size of 1000 and for the varied labeling percentage of (5,10,15,20,25,30,50,70,100). The map size used in our experiment is 10*10.

Figure 31 shows Weather, Ozone-1h, Ozone-8h data sets needs 10% labeling while Adult, CoverType, and HyperPlane data sets needs 5% to build a model with equivalent F-measure as 100% labeling. On the other hand, Electricity and CheckBoard data sets at least needs 20%, and 15% labeling respectively to obtain comparable results with 100% labeling. The RLS-SOM results indicate with labeling up to a quarter of the
data stream, an accurate model could be built which its performance is not significantly different from the model which is built using 100% labeling.

![Graph showing F-measure for all datasets when the labeling percentages vary from 5% to 100%]

Figure 31. The F-measure for all the datasets when the labeling percentages varies from 5% to 100%

5.3.3 Experimental comparison of RLS-SOM with other techniques

5.3.3.1 Comparison of RLS-E with RLS-SOM

The comparison results of RLS-E and RLS-SOM are done for the four data sets which had increased the Real Labeling Budget in Chapter 4. These data sets are Electricity, CheckBoard, Ozone-1h, and Ozone-8h. The labeling comparison between RLS-E and RLS-SOM for all the chunk sizes, when the original labeling percentage is 10%, is shown in Figure 32.

Figure 32 demonstrate there is a decrease in labeling percentage in RLS-SOM compared to RLS-E. For the Ozone-8h data sets, the labeling has decreased in chunk size 400, and for the other data sets the labeling has decreased at chunk sizes 400, and 600. As it was also previously observed, the labeling percentage was much higher in lower chunk size than higher ones. Figure 32 indicates RLS-SOM was able to decrease the labeling
and finding the minority class samples using fewer samples to label. Hence, RLS-SOM optimizes the search over the map and finds the minority class by less labeling.

Figure 32. Labeling percentage for the Electricity, CheckBoard, Ozone-1h, and Ozone-8h for the RLS-SOM and RLS-E

In Figure 33, the comparison results of F-measure for the four aforementioned data sets, as well as HyperPlane data set which suffers from feature drift is shown.

Figure 33. F-measure results for the Electricity, CheckBoard, Ozone-1h, and Ozone-8h for the RLS-SOM and RLS-E
The results indicate for Ozone-1h and Ozone-8h equal performances have been achieved. Although at chunk size 400 the F-measure in RLS-SOM has been decreased, but the difference is not significant. As it was discussed in Figure 32, the labeling percentage has been decreased in RLS-SOM compared to the RLS-E. In other word, with less labeling, the same quality of performance has been achieved. In case of Electricity data sets, higher F-measure has been achieved in RLS-SOM compared to RLS-E, while reducing the labeling at chunk sizes [400,800]. For CheckBoard data set, labeling has been decreased in the chunk size [400,600] as well, however with not significantly different F-measure when compared with RLS-E. The results indicate that the RLS-SOM can maintain the labeling in the budget more than the RLS-E for the lower chunk sizes, while maintaining the same or better quality of performance. In HyperPlane data set, the results have been increased at least 10% in RLS-SOM compared to RLS-E, which shows the ability of RLS-SOM in detecting informative samples when data faces feature drift. HyperPlane data also achieved higher quality of results with labeling at least 5%, while in the RLS-E, the labeling percentages was much higher (as shown in Figure 23(a)).

5.3.3.2 Comparison of RLS-SOM with partially labeled techniques

RLS-SOM results are compared with three different partially labeled algorithms mostly applicable to balanced data stream. We aim to compare our RLS-SOM with these techniques to show how these techniques fails when they are applied on the imbalanced data stream. We implemented these algorithms and used them in the labeling section of RLS framework. These techniques are as follows:
• **Random Sampling (base line):** This strategy picks the samples randomly from each chunk for labeling, instead of looking for the informative samples to build a new model.

• **Uncertainty Sampling:** The idea is to select and label samples which our model is least confident to produce the class label (Settles, 2010). To accommodate this strategy, only the samples closest to the decision boundary of the last chunk are selected and labeled, as opposed to RLS-SOM, in which the closest samples to the decision boundary and the minority class of the last chunk is selected. The closest samples are found using K-Nearest Neighbor (KNN). Uncertainty sampling, however, only addresses the conditional drift, and it totally fails if data stream experience a sudden feature drift.

• **Uncertainty Sampling + Randomization:** As it is mentioned, the uncertainty sampling always label the samples close to the decision boundary, and it misses the changes occur in other part of the space. Thus, randomization is used in addition to the uncertainty sampling to capture the changes if sudden drift occurs. This method is used in many studies such as (Ienco, Zliobaite, & Pfahringer, 2014) and (Žliobaité, Bifet, Pfahringer, & Holmes, 2014).

Figure 34 and Figure 35 shows the classification performance of RLS-SOM, Random sampling, uncertainty sampling, and Uncertainty sampling with randomization for all the data sets. The F-measure results are shown for the chunk size 400, and for the
labeling percentages (5-10-15-20-25-30-50-70-100). In order to evaluate the results, we show how varying the labeling percentage affects the performance results.

Figure 34. The F-measure results for Electricity, HyperPlane, CheckBoard, and Weather datasets

Figure 34 shows the results for the Electricity, HyperPlane, CheckBoard, and Weather data sets. From Figure 34 we can observe that RLS-SOM outperforms other techniques in Electricity, CheckBoard, and Weather data sets. RLS-SOM obtains higher performance at lower labeling percentages in CheckBoard data set which has a very complex decision boundary to learn. The three other techniques, however, could improve
the performance at higher labeling percentage. For the HyperPlane data sets, RLS-SOM obtained higher F-measure at lower labeling percentages, while it obtained comparable F-measure at high labeling percentage. HyperPlane data set has feature drift, and the results show how the RLS-SOM could capture the changes and act accordingly. However, the Uncertainty with random sampling method in this data set obtains lower F-measure than Uncertainty, showing randomization did not help and caused to lower performance at lower labeling percentages. This method, however, could increase the results at 25% labeling percentage, while is still lower when compared to RLS-SOM.

![Graphs](image)

a) Adult  

b) CoverType  

c) Ozone-1h  

d) Ozone-8h

**Figure 35.** The F-measure results for Adult, CoverType, Ozone-1h, and Ozone-8h datasets
Figure 35 shows the performance results of Adult, CoverType, Ozone-1h, and Ozone-8h data sets. The results indicate RLS-SOM obtained higher performance when compared to all other methods in Ozone-1h, and Ozone-8h. However, for the CoverType data set, the difference is not significant. The performance in Adult data set, however, is slightly increased compared to the other techniques since the imbalance ratio in this data is 25%, and that is why the other techniques perform relatively well.

To wrap up this set of experiments, we can state that RLS-SOM obtained higher performance when compared with all other techniques which basically are used for the balance data stream. These methods fail for the imbalanced data set, as it may not find any samples from the minority class samples in order to build the model. These results underline that using SOM positively affects the performance of the data stream, by helping to find informative and diverse minority class samples from all over the data space.

5.3.3.3 Comparison of RLS-SOM with Fully labeled techniques

The RLS-SOM is also compared with four algorithms in the domain of data stream mining which used 100% labeling in building their model. These techniques are as follows:

- **UCB**: Uncorrelated bagging (UCB) is based on a bagging framework. In this algorithm, all the minority class samples from previous chunks are accumulated and will be added to the current chunk. From the current chunk, the majority class samples are under sampled to make the class distribution balanced (Gao et al., 2007). To avoid minority class samples becoming majority, a sliding window was used in
(Ditzler & Polikar, 2013) that ensures it is not trained on more minority samples that majority class samples.

- **SERA:** Similar to UCB, it also accumulates the minority class samples, those that are similar to the minority class of the current chunk (Chen & He, 2009).

- **Learn++.CDS:** This algorithm does not accumulate the minority class and it does not need the information from previous chunks. Learn ++CDS is a integration of Learn++NSE and SMOTE to deal with imbalance classes (Ditzler & Polikar, 2013).

- **SEA:** The original SEA (Streaming Ensemble Algorithm) was designed for data stream with balance distribution (Street & Kim, 2001). The implementation later modified to include the class imbalance problem by (Ditzler & Polikar, 2013). We compare RLS-SOM with the SEA designed for the imbalanced class distribution.

The parameters used in our experiments such as data sets chunk sizes, imbalance ratios, and labeling percentage are shown in Table 11. For each data set, 10 models are maintained in ensemble, and the map size is 10*10.

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>Chunk Size</th>
<th>Imbalance Ratio</th>
<th>Label %</th>
<th>% RLS-SOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckBoard</td>
<td>1000</td>
<td>5%</td>
<td>15%</td>
<td></td>
</tr>
<tr>
<td>HyperPlane</td>
<td>1000</td>
<td>7%</td>
<td>10%</td>
<td></td>
</tr>
<tr>
<td>Electricity</td>
<td>1000</td>
<td>5.5%</td>
<td>15%</td>
<td></td>
</tr>
<tr>
<td>Weather</td>
<td>1000</td>
<td>10%</td>
<td>15%</td>
<td></td>
</tr>
</tbody>
</table>

Experimental results of AUC and F-measure, and the comparison of RLS-SOM with other algorithms (Ditzler & Polikar, 2013) are shown in Table 12.
Table 12. Comparison results of RLS-SOM with other methodologies (Ditzler & Polikar, 2013)

a) AUC

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>RLS-SOM</th>
<th>UCB</th>
<th>SERA</th>
<th>SEA</th>
<th>Learn++.CDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckBoard</td>
<td>0.80</td>
<td>0.919</td>
<td>0.808</td>
<td>0.658</td>
<td>0.909</td>
</tr>
<tr>
<td>HyperPlane</td>
<td>0.84</td>
<td>0.858</td>
<td>0.801</td>
<td>0.820</td>
<td>0.852</td>
</tr>
<tr>
<td>Electricity</td>
<td>0.66</td>
<td>0.697</td>
<td>0.624</td>
<td>0.585</td>
<td>0.606</td>
</tr>
<tr>
<td>Weather</td>
<td>0.69</td>
<td>0.711</td>
<td>0.635</td>
<td>0.734</td>
<td>0.729</td>
</tr>
</tbody>
</table>

Table 12 shows the comparison results of AUC and F-measure, between RLS-SOM, and other algorithms. Labeling percentage for each data set in RLS-SOM is reported in Table 11. Other techniques used 100% labeling. In CheckBoard data set, RLS-SOM obtained higher AUC and F-measure compared to the SEA, as well as obtaining higher F-measure compared to UCB. It obtained higher F-measure than SERA while maintaining equal AUC using 15% of labeling. For HyperPlane, RLS-SOM shows higher F-measure than SERA, and SEA, while maintaining equivalent performance as UCB, and Learn++.CDS. This indicates RLS-SOM could capture the minority class samples in existence of the feature drift in this data set, using only 10% of the samples to label. In Electricity data set, the performance is better or equal compared to all the methods in F-measure, showing the improvement in classification of the minority class samples as well at 15% labeling. In Weather data set, RLS-SOM achieved higher F-
measure compared to all other algorithms, meaning the detection of the minority class samples with only 15% of the labeling, while maintaining comparable performance for AUC. We used Nemenyi test (Demšar, 2006a) for the pairwise comparison between RLS-SOM and the other algorithms. The test shows there is no significant difference between the AUC and F-measure obtained from RLS-SOM and the other algorithms. The experimental results and the significant test prove that RLS-SOM maintains the classification performance while using partial labeling.

5.4 Discussion

In this chapter, RLS-SOM methodology proposed which is a framework for classification of the imbalanced data stream with concept drift. RLS-SOM uses partial labeled samples in building the model to maintain the classification performance when concept drift occurs. It also addresses both types of conditional and feature drift. When concept drift occurs, RLS-SOM searches for the samples from both classes to build an accurate model. Since data is imbalanced, finding minority class samples is often challenging. In case of conditional drift, SVM is used to find the samples closest to the decision boundary (margin area of the model). These samples contain both majority and minority classes to build a new model. However, if no minority class sample is detected in the margin, the current chunk is mapped to a 2-D grid using SOM to search for the minority class samples in the whole feature space. The map represents the data in several nodes. The labeling starts from center of each node to detect the minority class samples. Since SOM preserves the topology of the input vector, we should be able to locate the cluster of minority class samples. Our experiment shows that the labeling is increased
while increasing the map size. Hence, the recommended size for SOM grid is in the range of \([7*7-10*10]\). The experimental comparison of RLS-SOM with other partially labeled techniques shows that our technique is applicable for the highly imbalanced data stream and produces higher performance. The experimental comparisons of our method with other fully labeled techniques in the domain of imbalanced learning such as UCB, SEA, SERA, and Learn ++.CDS which used fully labeled data stream show RLS-SOM maintains the same quality of classification performance using 15\% of the samples to label, on average. RLS-SOM is able to detect the minority class samples with labeling less, and to use partial labeled samples to build an accurate model. RLS-SOM uses only the information from labeled data samples to build a model. In the next chapter, we use semi supervised techniques to use the information from unlabeled data samples as well.
CHAPTER 6
APPLYING SEMI-SUPERVISED LEARNING IN CLASSIFICATION OF
IMBALANCED STREAMING DATA

6.1 Introduction

Semi supervised learning combined with imbalanced, and partially labeled data stream is a very challenging task. As data stream are huge in volume, and fast changing, updating the classification model needs a new set of labeled samples. However, the assumption that entirely labeled data stream is available often violated in many real worlds problem (Woolam, Masud, & Khan, 2009). RLS framework aims to reduce the labeling and to build an accurate model when compared to fully labeled setting. RLS-S, RLS-E, and RLS-SOM are based on supervised learning which use p% (partial) of the data samples in building the model, while disregarding huge amount of unlabeled data samples that may contain the information regarding the variability of the data. Unlabeled data are not used in the supervised learning set up, but are used in the semi supervised approaches. These samples are used to improve the performance of the classification model. In a semi-supervised learning, the classification model is built over small amount of labeled data and a large amount of unlabeled data. In semi-supervised learning approaches, there are several assumptions (Zhu & Goldberg, 2009):

1) Data samples which are close to each other are more likely to share same label.

2) If we have several clusters in data, samples in the same clusters more likely have the same label.
In this section, we used a semi supervised learning approach in the RLS. Semi supervised approaches have been used in the research involving the balanced distribution of classes in data stream. The imbalanced characteristics of many real world applications make the problem harder as the correct classification of minority class samples is of much interest. As it was discussed in chapter 2.3, the semi supervised learning method has received tremendous attention in terms of balanced distribution between the classes (Bertini Jr et al., 2012; Masud et al., 2012). However, the imbalanced characteristics of many of the real world data stream, in conjunction with concept drift and partial labeling, make the problem very challenging.

In this section, we propose RLS-SSL (RLS-Semi Supervised Learning) which is an extension to RLS-SOM. RLS-SSL contains several characteristics:

- It maintains an ensemble of classifiers
- It addresses both conditional and feature drift
- It uses the labeled data and unlabeled data in building a classification model

As previously mentioned in section 2.3, there are several semi supervised algorithms such as S3VM (Bennett & Demiriz, 1999), and S4VM (Li & Zhou, 2011).

S3VM is an extension of SVM to semi supervised learning algorithm which is simultaneously learning the optimal hyperplane and the labels for the unlabeled data (Bennett & Demiriz, 1999). S3VM realizes the low density assumption and enforces the hyperplane to lie in the low density region. If we are given \( x_i \) which is a set of labeled data, with class label \( y \{ -1, +1 \} \), and \( \tilde{x}_j \) which is set of unlabeled data, the goal of S3VM is to find a function \( f: \chi \rightarrow \{ +1, -1 \} \) and the label \( \tilde{y} \in \{ +1, -1 \}^n \) for unlabeled data in order to minimize the following objective function:
\[
\begin{equation}
\begin{aligned}
    h(f, \hat{y}) &= \frac{1}{2} \| f \|_2^2 + \sum_{i=1}^{l} l(y_i, f(x_i)) + C_2 \sum_{j=1}^{u} l(\hat{y}_j, f(\hat{x}_j)) \\
\end{aligned}
\end{equation}
\]

(30)

where \( l \) is the number of labeled, \( u \) is the number of unlabeled. \( h \) is the Reducing Hilbert Kernel Space (RKHS) induced by kernel function \( k \), \( l(y, f(x)) = \max(0, 1 - yf(x)) \) is the hinge loss, \( C_1 \) and \( C_2 \) are the regularization parameters trading off the complexity and the experimental error for the labeled and unlabeled data,

The formulation of S3VM is non-convex, which may results in S3VM to get stuck in the local minima (Li & Zhou, 2011). To tackle this problem, four approaches have been studied: One is based on global combinatorial optimization, which results in good performance on small data sets (Bennett & Demiriz, 1999). The second approach is based on heuristic search, as it was done in TSVM (Joachims, 1999a). That increases the difficulty of the non-convex problem slowly. The third approach uses a convex relaxation which transfer the equation (30) to a relaxed convex problem, one example of this work is minimax relaxation as has been done in (Li, Kwok, & Zhou, 2009; Li, Tsang, Kwok, & Zhou, 2009).

The forth approach is S4VM which tries to prevent the local minima while trying to find the optimal solution for equation (30) (Li & Zhou, 2011). S4VM considers all the large margin low-density and diverse separators and then keep the candidates with large diversity. S4VM tries to minimize the following function:

\[
\begin{equation}
\begin{aligned}
    \min \sum_{i=1}^{T} h(f_i, \hat{y}_i) + M\Omega(\{\hat{y}_i\}_{i=1}^{T}) \\
\end{aligned}
\end{equation}
\]

(31)

Where \( h(f_i, \hat{y}_i) \) is the equation (30), \( T \) is the number of separator, \( \Omega \) is a quantity of penalty for the diversity of separators, and \( M \) is a large constant which ensures the large diversity.
Therefore, S4VM considers all the candidate low-density separators and optimizes the label assignments for the unlabeled samples. If the ground-truth label assignment can be recognized by a low-density separator, S4VMs will never degrade the classification performance.

The Semi-Supervised learner which was used in RLS-SSL is S4VM. S4VM is designed for the data with balanced distribution of classes. Li and Zhou (2011) provided the implementation of S4VM, and we utilize the S4VM in the data stream environment with imbalanced distribution of classes.

In RLS-SSL, the information from unlabeled, along with labeled data, is used. To address the class imbalance problem in S4VM, we used ADASYN (He et al., 2008) to over sample the minority class samples before importing to S4VM as its set of labeled samples. ADASYN is an over sampling technique which is an extension of the SMOTE (Chawla, Bowyer, Hall, & Kegelmeyer, 2002). It is expected that using unlabeled data samples improves the performance.

6.2 Comparision results of RLS-SSL with RLS-SOM

The F-measure results of RLS-SSL when is compared with RLS-SOM is shown in Figure 36 for all the chunk sizes and when 15% of the samples are used in building the model.

Figure 36 shows for the Ozone-1h and CheckBoard data sets, the RLS-SSL obtained the higher performance than the RLS-SOM for all the chunk sizes. In Ozone-8h, RLS-SSL results are higher at chunk size 400, but the difference between F-measure of RLS-SSL, and the one in RLS-SOM is not significant for all other chunk size.
In Electricity data set, RLS-SSL could increase the F-measure performance in chunk size 1200, but not a significantly increase for other chunk sizes. In chapter 5 we observed that for checkboard data sets, we needed to label more in order to build an accurate model. The semi-supervised learner for this data set helps a lot since in CheckBoard data set has a very complex decision boundary to learn. Hence, by including the information from unlabeled data, semi supervised learner increases the performance for this data much more than a supervised learner. Moreover, in Figure 37 for Adult, and CoverType, the results of RLS-SSL are not significantly different from the F-measure results in RLS-SOM. The RLS-SSL increases the performance in Weather and Hyperplane data set. Since there is a feature drift in HyperPlane data set, it is obvious
using the information from unlabeled data samples helps much in increasing the performance.

Figure 37. F-measure results of the a) Adult, b) HyperPlane, c) Weather and d) CoverType data for all the chunk sizes

The comparison results indicate that using semi supervised learning algorithm helps in improving the performance, specifically for the data sets which have a complex boundary to learn, as well as the data which contains feature drift.

6.3 Comparison results of RLS-SSL with other methodologies

RLS-SSL is compared with the 4 methodologies as described in the section 5.3.3.3. These methodologies are UCB, SERA, Learn++.CDS, and SEA. The comparison results of F-measure for these methodologies when compared with the RLS-SSL are shown in the Figure 38. For the RLS-SSL, and RLS-SOM, the chunk size is 800, and
15% labeling percentage is used in building the model. For all other methodologies, 100% labeling was used.

![Figure 38. Comparison F-measure results of RLS-SSL with other techniques (Ditzler & Polikar, 2013) for the CheckBoard, HyperPlane, Electricity, and Weather data sets](image)

The comparison results demonstrate that RLS-SSL improved the F-measure for the HyperPlane, Electricity, and Weather data sets when compared with all other methods which used 100% labeling. In CheckBoard data set, RLS-SSL outperforms other techniques except Learn++.CDS. Overall, the comparison results indicate RLS-SSL improves the performance by taking the information from unlabeled data to account.

### 6.4 Discussion

In our RLS-SOM, we used the information from only the labeled samples to build the model. These samples include critical samples which have the information regarding the variability of the model. However if the decision boundary is very hard to learn, such as the one in CheckBoard data set, semi-supervised learning methods help a lot by taking the information from unlabeled data samples to account. We extended the RLS-SOM to
use a semi supervised learner rather than a supervised learner. We call this version RLS-SSL (Short for the RLS-Semi Supervised Learning). The semi supervised learning method in RLS-SSL is S4VM. To deal with the class imbalance problem, we used ADASYN over sampling technique. In RLS-SSL by feeding our critical samples to S4VM along with other unlabeled data samples in each chunk, we could improve the performance of the classification model for the Ozone-1h, HyperPlane, CheckBoard, and Weather data sets compared to RLS-SOM. For all other data sets, the performance is not significantly different in RLS-SSL, compared to RLS-SOM. The comparison results between RLS-SSL and other fully labeled techniques demonstrate that RLS-SSL achieved higher performance for Hyperplane, and Weather data set, while maintaining same quality of results for Electricity data set.

Our experimental results indicate that semi supervised learning improved the performance by taking the information from unlabeled data samples to the account.
CHAPTER 7

RLS FRAMEWORK FOR MULTI CLASS DATA STREAMS

7.1 Introduction

Initially, RLS framework was proposed and the experiments have been done on the data with two classes. However, the classification is a very challenging task if the data has more than two classes. The research studies in the classification of data stream with multiple classes, especially when one or more classes are imbalanced, and coupled with partially labeled data streams, have not been explored much. Techniques such as Support Vector Machines (SVM) were originally designed for binary classification (Hsu & Lin, 2002). Later, in several research studies it was extended effectively for multi-class classification (Schölkopf, Platt, Shawe-Taylor, Smola, & Williamson, 2001). On the other hand, most existing imbalanced learning methods are mainly designed for two class problems. These methods are less effective or even cause a negative effect in dealing with the multiclass tasks (Zhou & Liu, 2006).

As stated in (Hsu & Lin, 2002), solving a multi class problem in general is computationally more expensive than a binary classification problem with the same number of data. In recent studies, there are approaches such as One-Against-All (OAA) (Vapnik & Vapnik, 1998), and One-Against-One (OAO) (Bishop, 2006) to solve a multi class problem. The multiclass problem is solved using class decomposition in these approaches.
One of the simplest multiclass classification schemes is OAA. In this method, if K is the number of classes, K different binary classifiers are trained; each one trained to distinguish the samples in a single class from the samples in the all other classes. Therefore, if we are given a K-class problem (K> 2), a decomposing scheme works as follows: it labels one class as positive and merges other classes and labels them as negative for creating sub-problems. Continuously, each class becomes the positive class once, and K-binary classifiers are produced to give the final output. On the other hand, the number of classifiers in OAO is K(K-1)/2 classifiers. Each classifier has been trained on the data from two classes.

While the aforementioned methods solve a multi class problems using class decomposition, there are several other methods which consider all the data at once, called “all-together”. These techniques are trying to solve a multi class problem by solving one single optimization problem (Vapnik & Vapnik, 1998; Weston & Watkins, 1998). Similar to OAA approach, in these methods K two-class rules are constructed, and therefore, there are K decision boundaries which are obtained by solving only one problem.

A comparison study between the aforementioned methods shows that the difference in their performance is not statistically significant. However, regarding the training time, OAO is faster and authors suggested to use OAO for practical applications (Hsu & Lin, 2002).

### 7.2 Literature Review

For imbalanced and multi class data stream, several methods such as MDO(Abdi & Hashemi, 2016), MOOB, and MOUB(S. Wang, Minku, & Yao, 2016), etc. have been
proposed. MDO (Mahalanobis Distance-based Over-Sampling technique) (Abdi & Hashemi, 2016) is an over-sampling technique for multi-class imbalanced data stream. In this method, the synthetic samples are created in the condition to have the same Mahalanobis distance, as other minority class samples, from their class mean. Their results represent the over-sampling technique can improve the performance of learning from multi-class imbalanced data sets, considerably. For the multi class classification, authors used Decision tree, KNN, and Ripper. MOOB (Multi-class Oversampling-based Online Bagging), and MUOB (Multi-class Undersampling-based Online Bagging) was proposed by (S. Wang et al., 2016) for imbalanced data stream in both stationary and dynamic environment. To overcome the imbalanced problem, the data is either over or under sampled; and the multi class data is processed without using class decomposition. To handle the multi class directly, either neural networks or Hoeffding trees can be used(S. Wang et al., 2016). There results show that MOOB performs better that MOUB as under sampling techniques may cause to lose some of the information contained in the data. These techniques, as well as MDO, do not consider the concept drift in the multi-class data sets. Effect of drifting and varying class imbalance in the data stream has been considered in (Pang et al., 2013), by adding dynamic class imbalance learning (DCIL) to incremental LPSVM (Linear Proximal Support Vector Machine). To deal with the multi class in data stream, OVA approach has been used. DCIL-IncLPSVM adapts its weights dynamically to cope with any class imbalance ratio, whenever a chunk of data is arrived, or removed.

McELM (Savitha, Suresh, & Kim, 2014) is another method for classification of multi class data stream with imbalanced distribution of classes. In order to predict the
class label of unseen samples, McELM consist of two components: a meta cognitive, as well as a cognitive component. The cognitive component is a three-layered extreme learning machine (ELM) classifier. ELM is a fast learning single hidden layer network which can be used for classification. The meta-cognitive component of McELM has a dynamic model of cognitive component, and has a learning mechanism that decides about an incoming new sample. When a new sample arrives, the meta-cognitive component compares the knowledge supplied in the sample, with that of the cognitive part, using Hinge loss error, and the predicted class label. Based on this information, it decides what samples to learn, as well as when-to-learn, and how-to-learn. If the new sample does not contain novel knowledge, it can be deleted from the training sequence. The new sample is deleted if the following criterion holds:

\[ c' = \hat{c}' \text{ AND } E' \leq E_d \]  

(32)

Here, \( c' \) is the true label of the new sample, and \( \hat{c}' \) is its predicted label. \( E_d \) is the delete threshold, and \( E' \) is the maximum absolute hinge error and is calculated as follows:

\[ E' = \max_{j=1,2,...,C} \left| e'_j \right| \]  

(33)

Where \( e'_j \) is the hinge loss error defined as:

\[
e'_j = \begin{cases} 
0 & \text{if } y'_j \hat{y}'_j > 1 \\
\quad y'_j - \hat{y}'_j & \text{otherwise}
\end{cases} \quad j=1,2,...,C
\]  

(34)

On the other and, if the new incoming samples does not meet the sample deletion rule, McELM will use it in the learning. The learning in ELM begins with zero hidden neurons, and based on the incoming samples, either a neuron is added or parameters of the existing network are updated. A neuron is added if:
\[ c' \neq \hat{c'} \text{ AND } E' > E_a \]  
(35)

where \( E_a \) is the self-regulating neuron addition threshold. However, if:

\[ c' = \hat{c'} \text{ AND } E' \geq E_a \]  
(36)

the current sample is used to update the output weights of the cognitive component.

Here, \( E_u \) is the self-adaptive meta-cognitive parameter update threshold. If the sample does not satisfy the deletion or the learning strategy, it is reserved in the training set for future use. The drawback of McELM is that it can not handle highly imbalanced data stream.

VWOS-ELM (Voting based Weighted Online Sequential Extreme Learning Machine) (Mirza, Lin, Cao, & Lai, 2015) is the first sequential learning classifier that can handle multi class problem in the domain of class imbalance. VWOS-ELM is the extension of WOS-ELM algorithm, in which the weights are computed in more efficient manner. The weights basically are assigned to reduce the imbalance class problem, by assigning higher weights to the minority class. In WOS-ELM, weights tuning are assigned to optimize the G-mean:

\[
G\text{-}mean = \left( \prod_{j=1}^{q} \left( \frac{s_\bar{j}}{s_j} \right)^{1/q} \right)
\]  
(37)

Here, \( s_i \) is the number of samples in class \( j \), and \( s_\bar{j} \) is the number of correctly classified samples in class \( j \), and \( q \) is the number of classes.

WOS-ELM can tackle the imbalanced learning both in chunk learning, and incremental learning. However, the WOS-ELM assumes that the data is not changing over time. Since WOS-ELM can handle the binary classification only, it is extended to
VWOS-ELM for the multi class classification. VWOS-ELM contains M independent WOS-ELM networks initialized with different random hidden node parameters and the same number of hidden neurons. When a new sample is coming, the weighing matrix is tuned to optimize G-mean in each WOS-ELM. If for one of the classes, the G-mean performance is low, it tries to increase it by checking the class specific weight setting, while the weights of other classes remained fixed to the inverse of their respective class sizes as follows:

$$w_{k,i} = 1 / s^j_k$$  \hspace{1cm} (38)

where, $s^j_k$ is the total number of samples in class $j$ up till time step $k$. For each sample, the predicted label is obtained from each WOS-ELM, and the final label is obtained based on majority voting.

The aforementioned methodologies use 100% labeled samples in building the model. MOOB, MOUB and MDO do not handle the concept drift in the data. McELM is also not applicable to highly imbalanced data. In the next section, we propose RLS-Multi (RLS-Multi class classification) which is an extension of RLS-SOM to handle multi class and imbalanced data stream.

7.3 RLS-Multi for multi class data sets

In RLS-Multi, SVM and SOM are used to track the gradual and sudden changes in the data stream respectively. The ability of these methods makes them suitable for multi class learning as well. Both SVM and SOM ability help towards detecting single or multi minority class samples in an imbalanced data set when drift occurs, and will provide partial samples from all the minority classes to build a model.
Support vectors help us to detect the changes close the classification model. When a new chunk arrives, the framework checks the closest samples to our previously built model. If one or more of the minority class is missing, the data suffers from feature drift and it is projected to a 2D map using SOM (Self-Organizing Map) to look for the remaining classes. The multi class learning in RLS-Multi is OAO which is faster than OAA, and it is suitable for practical application (Hsu & Lin, 2002). In Figure 39, SVM and SOM in dealing for multi class imbalanced are shown.

Figure 39. a) SVM used for modeling a multi class data (OAO) b) Samples are projected to a 2D map using SOM

In Figure 39(a), an imbalanced and multi class data with three classes is demonstrated. Figure 39(a) shows how the SVM is applied for modeling using OAO. For example the bold line separates class 1 from class 3. In SVM, the support vectors can help us to detect if any conditional drift occurs. Figure 39(b) demonstrates when an
imbalanced and multi class data is projected to a 2D map using SOM. The neighboring nodes forms cluster of the data and hence, the location of the minority class samples can be found using SOM. RLS-Multi procedure is shown in Algorithm 1c.

**Notations:**
- **CS$_A$:** Critical Samples A
- **CS$_B$:** Critical Samples B
- **MinS$_{(\text{min classes})}$:** Minority classes Samples
- **MajS$_{(\text{maj classes})}$:** Majority classes Samples
- **MinC:** Minority classes
- **MajC:** Majority classes
- **SV:** Support Vectors
- **m:** model
- **B:** Borrow
- **K:** number of models
- **C:** Chunk
- **E:** Ensemble
- **d$_1$, d$_2$:** map dimensions

**Input:**
- Streaming data in chunks: $C_1, C_2, \ldots, C_n$

**Goal:**
- Classifying Imbalanced Streaming data with partial labeling

**// Initial phase**
1. $E \leftarrow 0$, $k \leftarrow 0$
2. TrainSet$\leftarrow C_1$
3. Build $m_1$ on TrainSet using SVM, $k++$
4. $E \leftarrow m_1 \cup E$
5. $B \leftarrow \{SV \cup \text{MinS$_{(\text{min classes})}$}\}: SV = \{SV_1, SV_2, \ldots, SV_n\}, \quad \text{MinS$_{(\text{min classes})}$} = \{\text{MinS}_1, \text{MinS}_2, \ldots, \text{MinS}_m\}$

**// Iterative phase**
6. for $C_i$, $i=2, \infty$ do
7. D$\leftarrow$ Euclidean Distance($B$, $C_i$)
8. CS$_A$ $\leftarrow$ top closest samples from $C_i$ to $B$
9. Obtain true labels for CS$_A$
10. Obtain predicted labels for CS$_A$ using E
11. Find classification performance of CS$_A$
12. if (MinS$_{(\text{min classes})} \in$ CS$_A$ & MajS$_{(\text{maj classes})} \in$ CS$_A$) //if all the classes have been found
13. \hspace{1cm} if (Performance $C_i$ - Performance $C_{i-1}$ $> \text{thr}$) then
14. \hspace{2cm} //conditional drift – Use ADASYN if highly imbalanced
15. \hspace{2cm} Build $m_k$ using CS$_A$. $k++$
16. \hspace{2cm} $E \leftarrow m_k \cup E$, $B \leftarrow \{SV_{mk} \cup \text{MinS$_{C_1}$}\}$
17. \hspace{1cm} end
18. \hspace{1cm} else if (MinS$_{(\text{min classes})} \not\in$ CS$_A$) // feature drift
19. \hspace{2cm} //only look for the classes which have not been found in CS$_A$
20. CS$_B$ selection in Algorithm 3: SOM Sample Selection
21. Build new model $m_k$ using $\{CS_A \cup CS_B\}, k++$
22. $E \leftarrow m_k \cup E$, $B \leftarrow \{SV_{mk} \cup \text{MinS$_{C_{i-1}}$ (min classes)}\}$
23. end

**Algorithm 1c.** The RLS-Multi with multi class
In the initial phase of Algorithm 1c, a model $m_i$ is built on the first chunk of the data ($TrainSet$) using SVM (line 4). This model is added to the Ensemble ($E$) (line 5), and from the $TrainSet$, all the minority class samples and the support vectors are maintained in an array called $Borrow(B)$ (line 6).

```
<table>
<thead>
<tr>
<th>Input:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Set of samples</td>
</tr>
<tr>
<td>Output:</td>
</tr>
<tr>
<td>- Selected samples for labeling</td>
</tr>
</tbody>
</table>

```

```
//map the data to a 2D map using SOM
1. for j = 1 to d1×d2
2. S ← find and label closest sample to center of the node
   // check if selected sample belongs to minority class
3. if (S ∈ MinC (missing classes))
4.   [Temp]n ← [S]n, n < θ
   // label other closest sample from current node
end
5. CS ← S U [Temp]n
end

Algorithm 3a. SOM sample selection
```

In the iterative phase, when the next chunk of data arrives, the Euclidean distance between this chunk and $B$ is calculated (line 7). The top closest samples of this chunk to $B$ are used for labeling, and we call them Critical Samples$_A$ ($CS_A$) (line 8-9). If $CS_A$ contains the samples from all the classes, then the performance obtained on $CS_A$ is compared with the last chunk. If it drops more than a $thr$, it is said that conditional concept drift is detected (line 13). Therefore a new model is built and will be added to the ensemble $E$, and the $B$ is updated (line 14-15). Before building a new model, ADASYN (He et al., 2008) is used to over sample the minority classes in the $CS_A$. If either one or more of the minority class is missing from $CS_A$ (line 16), feature drift is detected, and SOM is used to find the remaining classes in the data for labeling (line 17).

Algorithm 3a shows when data is projected to a 2D map using SOM. In each node, we find the closest sample to the center of the node and label it (line 2). If the
labeled samples belong to a one of the missing minority class, under the assumption that minority class appears in compact clusters, we label more sample from this node by selecting the second closet sample to the center. If after labeling up to a threshold \( \Theta \) we did not find samples of minority class, we move to the next node. After returning from SOM, we have \( CS_B \) (Critical Samples\( _B \)) which contains the samples from our missing minority classes. Then in Algorithm 1c (line 18), \( CS_B \) along with \( CS_A \) are used for building a new model. Like before, the new model is added to the ensemble \( E \) and the \( B \) is updated (line 19).

The critical samples contain diverse samples from all over the feature space. Since SOM preserves the topology of the high dimensional data, samples in each node usually belong to a common class. In addition, labeling samples closest to the center of each node should reveal what class that node belongs to. Therefore, SOM helps us to find the diverse samples from both majority and minority classes.

7.4 Results

7.4.1 Data sets

For the purpose of experimentations and also for the comparison with other approaches (Mirza et al., 2015; Savitha et al., 2014), we used five publicly available data sets from UCI repository (Asuncion & Newman, 2015). These data sets are initially very small in size. To make them large enough for processing as a data stream, a method explained by (Hashemi, Yang, Mirzamomen, & Kangavari, 2009) was applied. The data set is repeatedly queued to produce a data stream. To simulate concept changes, every \( S \) samples form a data segment. For example if \( S=10000 \), the data are divided into segments of 10000 samples, and for each segment a feature is randomly selected and changed. If
the feature is nominal, its values are changed in a circular way, that is, \( a_1 \rightarrow a_2 \rightarrow \ldots \rightarrow a_n \rightarrow a_1 \), for all the samples in this segment, while the other features are remained unchanged. If the feature is numeric, all the samples in this segment are ascendingly sorted according to its value. These data sets are described as follows:

- **Page-block**

  This data originally has 5473 samples, 10 features, and 5 classes. The task is classifying all the blocks of the page layout of a document that has been detected by a segmentation process. The classes are \([text (1), horizontal line (2), picture (3), vertical line (4) and graphic (5)]\). The new data set after conversion has 109460 samples.

- **Statelog**

  This is an image data set and the task is to classify the central pixel as one of these classes: \([red soil, cotton crop, grey soil, damp grey soil, soil with vegetation stubble, very damp grey soil]\). The original data has 6435 samples, 36 features, and 6 classes. Having data converted, the new data has 96525 samples, with same number of classes and features. Class ‘damp grey soil’ is also under sampled to make the data highly imbalanced.

- **Dermatology**

  The data originally has 366 samples, 34 features, and 6 classes. The task is to classify the six different erythema-squamous diseases \([psoriasis, seboroeic dermatitis, lichen planus, pityriasis rosea, cronic dermatitis, pityriasis rubra pilaris]\). After conversion, the data stream has 89500 samples.
• **Balance**

This data set was generated to model psychological experimental results. It has 625 samples, 4 features, and 3 classes. The task is to classify a balance scale tip to the right, tip to the left, or be balanced. Hence, the classes are \([\text{Left}, \text{Right}, \text{Balance}]\). After conversion, there are 93750 samples in the data stream.

• **Contraceptive**

This dataset originally has 1473 samples, 9 features, and 3 classes. The problem is to predict the current contraceptive method choice \([\text{no use, long-term methods, short-term}]\) of a woman based on her demographic and socio-economic characteristics. Class ‘long term’ was under sampled to make the data set imbalanced, and after conversion, the data has 103100 samples.

The data sets characteristics and the distributions of classes are shown in Table 13. The Imbalance ratio is calculated as the proportion of the number of samples in majority class over the number of samples in minority class (Orriols-Puig & Bernadó-Mansilla, 2009). In Table 13, we report the inverse of this proportion.

<table>
<thead>
<tr>
<th>Data</th>
<th>Class</th>
<th>Feature</th>
<th>Original Size</th>
<th>Classes Distribution</th>
<th>Data stream size</th>
<th>Imb-R%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page-block</td>
<td>5</td>
<td>10</td>
<td>5473</td>
<td>4913:329:28:88:115</td>
<td>109460</td>
<td>1%</td>
</tr>
<tr>
<td>Statelog</td>
<td>6</td>
<td>36</td>
<td>6435</td>
<td>1533:703:1358:100:707:1508</td>
<td>96525</td>
<td>6%</td>
</tr>
<tr>
<td>Dermatology</td>
<td>6</td>
<td>34</td>
<td>366</td>
<td>112:61:72:49:52:20</td>
<td>89500</td>
<td>18%</td>
</tr>
<tr>
<td>Balance</td>
<td>3</td>
<td>4</td>
<td>625</td>
<td>49:288:288</td>
<td>93730</td>
<td>17%</td>
</tr>
<tr>
<td>Contraceptive</td>
<td>3</td>
<td>9</td>
<td>1473</td>
<td>629:83:511</td>
<td>103110</td>
<td>13%</td>
</tr>
</tbody>
</table>
7.4.2 Experimental Setup

Experiments have been performed in MATLAB and LIBSVM (Chang & Lin, 2011) was used for classification of a multi class data. The SOM grid size was set to 10×10, and the default threshold $\theta$ for selecting the samples from each node is set to 5. In each experiment, 10 models are maintained in the ensemble, and a new model is built and will be added to the ensemble if performance drops more than 10% ($th\alpha=0.1$). For the evaluation measure, G-mean as shown in equation (37) is used.

7.4.3 Experimental Results

7.4.3.1 Labeling percentage

In RLS-Multi, various labeling percentages (5, 10, 15, 20, 25, 30, 50, 100) have been used to find the best value for each data set. The experiments are shown in Figure 40. The algorithm starts at 5% initial labeling, and will increase the labeling until the same quality of results as 100% labeling is obtained.

![Figure 40. Varying the labeling percentage to obtain high G-mean performance](image)

As is shown in the Figure 40, at 5% labeling all the data sets produce lower performance when compared with 100% labeling. However, when the labeling
percentage increases, it causes an increase in the classification performance. For the Page-block, which is a high imbalance data, the labeling percentage at 25% achieved comparable performance with 100% labeling. The labeling is higher in this data due to existence of high imbalanced classes. Statelog, and Dermatology needs 15% while Contraceptive and Balance data set needs more labeling (20% and 50% respectively) in order to achieve high quality of performance. In the Contraceptive data, the minority class may scattered, and thus the algorithm needs to label more in order to locate them. The chunk size and the labeling percentages are shown in Table 14.

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>Chunk Size</th>
<th>Labeling %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page-block</td>
<td>3700</td>
<td>25%</td>
</tr>
<tr>
<td>Statelog</td>
<td>2500</td>
<td>15%</td>
</tr>
<tr>
<td>Dermatology</td>
<td>200</td>
<td>15%</td>
</tr>
<tr>
<td>Balance</td>
<td>200</td>
<td>20%</td>
</tr>
<tr>
<td>Contraceptive</td>
<td>700</td>
<td>50%</td>
</tr>
</tbody>
</table>

### 7.4.3.2 Classification performance over time

The experimental results of G-mean are shown in the Figure 41 for the first few chunk of data stream for all the data sets. The red dotted line shows the G-mean results over time when there is no model update, versus the blue solid line shows the G-mean progression over time when the model is updated upon concept drift detection.

As it is shown in Figure 41, if there is no model update in the data, the performance of all the data gradually decreases. The drop in G-mean is more sever in Balance, Statelog, and Contraceptive, compared to the other data sets. The results indicate that data stream has the concept drift, and the model should get updated when it is detected.
Figure 41. G-mean performance over time when there is no model update (red dotted line), and when the concept drift is detected and handled in RLS-Multi (blue solid line) on real world data sets.

The solid blue line shows when the concept drift is detected in RLS-Multi, and consequently the model gets updated to maintain the performance of the classification. More importantly, partially labeled data stream is used in updating the model. This indicates that RLS-Multi could effectively handle the concept drift in the data, and builds a new model with partial labeled samples.
7.4.3.3 Experimental comparision with other methods

The RLS-Multi was compared with the two methodologies in the domain of multi-class imbalance data stream which uses full labeling in building the model. The goal is to show that RLS-Multi can use partially labeled data stream to build an accurate model, which the performance is not significantly different from the methods which used 100% labeling in building their model. These methods are McELM, and VWOS-ELM which have been described in details in 7.2, here we give a brief explanations:

1) McELM (Savitha et al., 2014) is an approach to handle multi class and imbalanced data stream. It has a meta-cognitive component which decides what sample of the training set is useful in building the model. However, it is not applicable to highly imbalanced data stream.

2) VWOS-ELM (Mirza et al., 2015) is another approach to handle multi class and imbalanced data stream. It checks the G-mean of each class in the process and tries to increase it. The imbalance classes are handled in a way that it assigns more weights to the minority classes.

The results are compared over the original data sets, and the average G-mean results over all the chunks are reported in Table 15. The chunk size and labeling percentage is the same as the one reported in Table 14. Other techniques use 100% labeling.
Table 15. Experimental results and comparison of RLS-Multi with other techniques (Mirza et al., 2015)

<table>
<thead>
<tr>
<th>Data sets</th>
<th>RLS-Multi (partial labeling)</th>
<th>VWOS-ELM (100% labeling)</th>
<th>McELM (100% labeling)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page-block</td>
<td>0.91</td>
<td>92.4</td>
<td>72.6</td>
</tr>
<tr>
<td>Statelog</td>
<td>0.82</td>
<td>87.4</td>
<td>84.3</td>
</tr>
<tr>
<td>Dermatology</td>
<td>0.95</td>
<td>82.8</td>
<td>76.6</td>
</tr>
<tr>
<td>Balance</td>
<td>0.83</td>
<td>91.6</td>
<td>68.0</td>
</tr>
<tr>
<td>Contraceptive</td>
<td>0.65</td>
<td>56.5</td>
<td>54.1</td>
</tr>
</tbody>
</table>

Table 15 shows RLS-Multi obtained comparable performance for the page-block, and Statelog data sets compared to other techniques, using 25% and 15% labeling respectively. In Dermatology and Contraceptive data sets, RLS-Multi also obtained better results compared the two other techniques using 15% and 50% labeling respectively. The performance of the Balance data set in RLS-Multi outperformed the one from McELM, and is still comparable with VWOS-ELM using only 20% labeling. We used Nemenyi test (Demšar, 2006b) for the pairwise comparison between RLS-Multi and the other approaches. The test shows there is no significant difference between the G-mean results obtained from RLS-Multi and the other algorithms. The experimental results indicate that RLS-Multi achieved comparable or higher performance compared with the other fully labeled techniques using partial labeled samples in building the model.

7.5 Discussion

In this chapter, the RLS-Multi was proposed which is an extension of RLS-SOM for the multi class and imbalanced data stream. In RLS-Multi, an initial SVM model is used to classify the samples in each chunk. If changes occur close to the classification model, the samples close to the classification model are selected for labeling. This
includes the samples from all the classes. However, if the data is imbalanced, finding samples from minority classes is a challenging task. Hence, if at least samples from one of the minority classes are missing, the data is projected to a 2-dimensional map using SOM to find and label the missing classes samples. The experimental results on 5 real world data set demonstrates that RLS-Multi reduces the labeling in presence of the multiple class in the data, and it builds a new model using less labeling, as accurate as a model which used 100% labeling. The experimental comparison with two state of the art fully labeled approaches (McELM, VWOS-ELM) for the multi class and imbalanced data stream shows that RLS-Multi is capable of classification of imbalanced and multi class data stream, with partial labeled samples. Overall, RLS-Multi is shown to be effective for classification of the data stream in presence of multiple and imbalanced classes.
CHAPTER 8

CONCLUSIONS

8.1 RLS in review

In this work, we present RLS framework for classification of the imbalanced data streams using partial labeled samples. There are several challenges associated with classification of the imbalanced data stream: First of all, data stream are huge in volume and they can not be stored in the memory for one time processing. Second, if the classes in the data stream are imbalanced, we may lose the performance of the minority class samples. Third, data stream may encounter concept drift over time, which causes degradation in the model performance. Forth, labeling is often needed in the case of data stream since to adapt with the concept drift, a new model should be built. However, labeling is a very costly and time consuming process since human experts are involved in the process.

In this thesis we propose RLS framework to overcome the aforementioned problems. RLS is a chunk based learning framework which maintains an ensemble of the classifiers to increase the performance. RLS detects the concept drift, and rebuilds a model only if it is necessary using partially labeled data stream. It finds the samples which have the most important information regarding the variability of data. It only gets the true label of these samples to build a classification model. Since the classes are imbalanced, finding the samples especially from minority class is a challenging task. RLS
detects such samples by using Support Vector Machine (SVM) and Self Organizing Map (SOM).

RLS in general starts analyzing the data stream by building an initial model using SVM for the first chunk of data stream. The SVM provides us the support vectors which are the samples close to the classification model. The support vectors are the most informative samples for the current chunk, and include the samples from minority and majority classes. RLS is flexible using either the SVM model for the labels’ prediction, or to use other classification algorithms. Upon arrival of the second chunk of the data, several samples in this chunk are selected for labeling. These samples are selected based on their closeness to the support vector of the initial model and the minority class samples of the previous chunk.

It should be noted that since the classes are imbalanced, selecting samples close to the support vector may provide us with the samples only from majority class. Hence, we find the samples which are close to the combination of the support vectors and the minority class of the previous chunk. This will result in finding the minority class samples as well. The selected samples usually have balanced distribution of classes. If it is not balanced, to increase the performance, the ADASYN (He et al., 2008) over sampling technique is used.

The performance of the selected samples is compared with that of the first chunk. If a significant drop occurs (based on a certain threshold), concept drift is detected and therefore a new model should be built to maintain the performance.

Initial version of the RLS maintained only a single model (RLS-S short for the RLS-Single), and therefore the new model replaced the old model. Later, we extended the
RLS to maintain an ensemble of classifier using Dynamic Weighted Majority (DWM) (Kolter & Maloof, 2007). This version of RLS is called RLS-E (short for the RLS-Ensemble). The results show the ensemble of the models improves the performance at least 10%, when compared with that of a single model for majority of data sets. The comparison with techniques such as (SA, HD, and HDIG) indicates RLS-E obtained results not significantly different from that of the 100% labeling using only 20% labeling.

If the selected samples do not contain the minority class samples, the changes have occurred far from the decision boundary. In order to look for the minority class samples, the data is projected to a 2D map using SOM. This version is called RLS-SOM short for RLS-Self Organizing Map. In the SOM grid, for the non-empty nodes, the labeling starts from the center of the node. If the center belongs to the minority class, more samples from this node are selected to get labeled. By the time we exit the map, we have the samples from the minority class as well. Hence, a new model is built on these data samples, and will be added to the ensemble. The process then is continued for all other chunks of data stream. The comparison result of RLS-E and RLS-SOM on the real world data sets shows that RLS-SOM is able to detect the minority class with less labeling compared to the RLS-E, if the changes occur far from the boundary. The comparison of RLS-SOM with fully labeled techniques (USCB, SERA, SEA, learn++, CDS) shows that this technique is able to analyze the data stream with up to 15% labeling, while maintaining the high classification results. Moreover, comparison of RLS-SOM with other partially labeled techniques such as (Random Sampling, Uncertainty Sampling, Uncertainty Sampling with Randomization) shows that these techniques needs
more labeling when they are dealing with the imbalanced classes to build an accurate model. While at lower labeling percentage, RLS-SOM results are at least 15% higher for most of the data sets.

The learning in the RLS is supervised. This means we used only the information from the labeled samples to build the model. The additional extension of RLS is to use the information from unlabeled data samples as well. This version is called RLS-SSL (RLS-Semi Supervised Learning). The comparison of RLS-SSL with RLS-SOM shows that for most of the data sets, using the information from unlabeled data helps. The semi supervised learning specially helps for the data which has a complex boundary to learn such as CheckBoard. The RLS-SSL improves the results at least 10% for the CheckBoard, and HyperPlane data set; while for all other data sets, either the results are increased by 5% or remained the same. The comparison of RLS-SSL with (USCB, SERA, SEA, learn++.CDS) also indicates that RLS-SSL increased the performance over Weather and HyperPlane data sets (that suffers from sudden feature drift) by 10% and 15% respectively, while obtained same quality of results for the Electricity data set.

In addition to use the RLS for binary classification, we extended the RLS for multi class classification, that is called RLS-Multi (RLS-Multi class classification). We used LIBSVM (Chang & Lin, 2011) in our RLS framework to handle multi class in the data stream. For dealing with the imbalanced classes, ADASYN (He et al., 2008) is used. The experimental comparison of RLS-Multi with techniques which used fully labeled data streams (McELM, and VWOS-ELM) indicates that RLS-Multi obtained the
classification results not significantly different when compared with fully labeled techniques, using only 25% of the samples for majority of the data sets.

### 8.2 Analysis of complexity of RLS Framework

The time complexity of the RLS will depend on the time needed for the sample selection, prediction, and retraining when needed. Therefore, the complexity for the iterative phase is calculated as follows:

\[ O(\text{Sample Selection A}) + O(\text{Model updating}) + O(\text{Model Prediction}) \]

The sample selection requires scan of the neighborhood around the classification boundary to find closest samples to the Borrow, this takes \( O(n \cdot \log n) \), where \( n \) is the number of samples in the chunk.

If concept drift is detected and there is a need to update the model, the framework uses the labeled samples (\( n' = CS_A << n \)) to retrain a new model. As mentioned earlier, Critical Samples\(_A\) are the samples which have the most important information regarding variability of the data. Thus the complexity is given by: \( O(f_{\text{train}}(n')) \)

where \( f_{\text{train}} \) is the training time for a base model as a function of number of samples. Let’s say the number of base classifiers in our ensemble \(|E|\) is \( k \), then the time complexity will be:

\[ k \cdot O(f_{\text{train}}(n')) \text{, where } k > 1 \]

In the case of single model, \( k = 1 \)

Meanwhile, if there are \( N \) data chunks to be predicted, the time cost of prediction is linear correlation to the size of instances, denoted as \( O(k \cdot N \cdot n) \). Thus, the combined complexity can be expressed as:

\[ O(N \cdot (n \cdot \log n) + p \cdot k \cdot f_{\text{train}}(n') + (k \cdot n)) \]
where \( p \) is the updating frequency of classifiers. If classifiers are updated in real time and per chunk, the value of \( p \) equals to 1. However, in RLS framework, the threshold \( thr \) is used to decide the updating of classifiers, so \( p<1 \). Correspondingly, the time performance of our algorithm could be improved. If the \( p=0 \), therefore there is no need to update the model, and the time complexity will be limited to time needed for the sample selection and time needed for prediction.

The space requirement for the framework lies in storing of the Borrow Samples, and \( F\)-measure and \( G\)-mean of the last chunk, set of labeled samples \( CS_A \), and the models of the ensemble \( |E| \). The number of models \( |E| \) can be bound by the pruning of old models (with low \( F\)-measure score) as mentioned before. Size \( E=1 \) if one single model is used.

If the changes occur far from the decision boundary, and RLS uses the SOM for sample selection, the time for the SOM clustering component also will be added. Therefore, the time complexity has four parts:

\[
O(\text{Sample Selection A }) + O(\text{Sample Selection B}) + O(\text{Model updating}) + O(\text{Model Prediction})
\]

As mentioned earlier, the time taking for the Critical Sample Selection\(_A\) is:
\( O(n \cdot \log n) \), where \( n \) is the number of samples in the chunk.

\( O(\text{Sample Selection B}) \) would be:

\text{Time to perform SOM} + \text{Time to Sample Selection B}

where

- \text{Time to perform SOM} = O(f_{\text{SOM}}(n))
• **Time to Sample Selection** \( B = d_1 \cdot d_2 \cdot z \cdot \log z \), where on average \( z = \frac{n}{d_1 \cdot d_2} \), \( z < \ll n \),

\( z \) on average is the number of samples per node of SOM grid, \( d_1 \) and \( d_2 \) are the size of SOM grid, \( z \log z \) is the time to find the closest sample to the centroid of each node.

Therefore the complexity can be expressed as follows:

\[
O(N \cdot [(n \cdot \log n) + (f_{SOM}(n)) + (d_1 \cdot d_2 \cdot z \cdot \log z) + p \cdot k \cdot (f_{train}(n')) + (k \cdot n'])]
\]

Where \( N \) is the number of chunks, \( n' = (CS_A + CS_B) \ll n \), \( p < \ll 1 \).

The space requirement for the framework lies in storing of the Borrow Samples, and *F*-measure and *G*-mean of the last chunk, set of labeled samples \( CS_A \) and \( CS_B \), and the models of the ensemble \( |E| \). The number of models \( |E| \) can be bound by the pruning of old models (with low *F*-measure score) as mentioned before.

If the Semi Supervised learner is added to the framework, the complexity \( f_{train} \) will be change to the \( f_{train(S4VM)} \) as follows:

\[
O(N \cdot [(n \cdot \log n) + (f_{SOM}(n)) + (d_1 \cdot d_2 \cdot z \cdot \log z) + p \cdot k \cdot (f_{train(S4VM)}(n'')) + (k \cdot n'])]
\]

Where \( n'' = n' \text{ (labeled samples)} + (n - n') \text{ (unlabeled samples)} \)

To alleviate the complexity of S4VM, this algorithm implemented based on heuristic sampling search (Li & Zhou, 2011).

### 8.3 Future work

There are several directions for the future study of this thesis. In the RLS framework, the SOM is used to find samples from minority classes when a concept drift is detected. RLS assumes that the number of classes will remain the same, and in each chunk, samples from all the classes appear. For the future studies, the RLS framework
can be used in the context of the novel class detections. That means, SOM clustering technique can be used if the novel classes appear in the data stream over time.

In the RLS framework for classification of the multi and imbalanced data streams, the modeling for the multi class data stream is in a supervised fashion. In the future work, it can extend to use the information from unlabeled data samples as well, by incorporating the semi supervised learning. Moreover, in the context of the data stream, changes can occur in a way that the minority classes become majority and vice versa. In addition, the class imbalance ratio for the minority classes can change over time. Hence, in the future studies, one can extend the framework to handle varying class imbalanced in the data stream.

In RLS framework, the concept drift is detected based on the performance degradation over time, and using the information from labeled samples. For the future studies, additional techniques can be used to detect such changes based on the change in the distribution of the data.
REFERENCES


tools on real world data sets. Paper presented at the Methods and Models in Computer Science (ICM2CS), 2010 International Conference on, New Delhi, India.


CURRICULUM VITAE

NAME: Elaheh Arabmakki

ADDRESS: Department of Computer Engineering and Computer Science
University of Louisville
Louisville, KY 40292

EDUCATION

Ph.D. Computer Science and Engineering
University of Louisville, Louisville, KY
2011-2016

M. Sc. Industrial Engineering Technology
Morehead State University, Morehead, KY
2009-2011

B. Sc. Computer Engineering
Azad University of Najaf Abad, IRAN
2002-2006

GRADUATE CERTIFICATE

Data Mining
University of Louisville, Louisville, KY
2013

PROJECT AND RESEARCH EXPERIENCE

- Internship Experience, Mayo Clinic, Rochester, MN
  Research Scientist Intern, Center for the Science of Health Care Delivery,
  Analysis of the EHR (Electronic Health Record) data for the patients with
  diabetes to predict those who are at risk for adverse outcomes when on metformin
  medication.
  May 2015-Aug 2015
- PhD Research Experience, University of Louisville, Louisville, KY
  RLS (Reduced Labeled Samples) framework as PhD dissertation
  Designed a framework to reduce labeling needed for classification of streaming imbalanced data, using Support Vector Machine (SVM); Reduced the labeling percentage from 100% to less than 20% of the samples in the stream. Saved expert time and effort.
  2013- 2016

  Kaggle competition in text analytics “Classification of the insulting tweets vs. natural tweets”: Improved accuracy from 71% to 84% using Logistic Regression. 2013

  Forecasting solar irradiance:
  Using data mining techniques and algorithms such as SVM, and ANN to predict solar irradiance in time-series solar data.
  2013

  Semi-automatic flu outbreak detection system:
  Proposed a system to generate a summary of the flu-related tweets, the summary is presented to medical experts to check if outbreak is happening or not. Achieved 86% accuracy.
  2012

  SAS Analytics and Data Mining Shootout:
  Designed and developed a model for influenza impact on vaccination plans.
  2012

PUBLICATIONS


TEACHING EXPERIENCE

Teaching Assistant: C/C++ Programming, Computer Science and Computer Engineering (CECS) Dept, University of Louisville
2016

Teaching Assistant: Data Structure, Computer Science and Computer Engineering (CECS) Dept, University of Louisville
2015-2016

Teaching Assistant: C/C++ Programming, Computer Science and Computer Engineering (CECS) Dept, University of Louisville
2013-2015
HONORS AND AWARDS

2016, Graduate Dean’s Citation Award
2016, Doctoral Dissertation Completion Award, University of Louisville, KY
2016, CSE Doctoral Award, University of Louisville, KY
2016, Best Paper Award, IEEE IRI
2015, 3rd place at Kentucky Academy of Science (KAS), Graduate Research Competition Awards for Engineering, Northern Kentucky University, KY
2015, Grace Hopper Celebration of Women in Computing Scholarship
2014, Recipient of Student travel grant for IEEE-IRI
2011-2013, Grosscurth Scholarship, Speed School of Engineering, University of Louisville
2012, E-EXPO Student Research Competition, Graduate overall 3rd place and Best in CECS Department, University of Louisville