ENSEMBLE LEARNING FOR MULTIPLE DATA MINING PROBLEMS

by

TRI SI DOAN

B.S., Computer Science, HoChiMinh city, Technical University, Viet Nam 1990

M.S., Computer Science, Kansas State University 2012

A dissertation submitted to the Graduate Faculty of the

University of Colorado Colorado Springs

in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

Department of Computer Science

2017
This dissertation for the Doctor of Philosophy degree by

TRI SI DOAN

has been approved for the

Department of Computer Science

by

Jugal Kalita, Chair

Terrance Boult

Rory Lewis

Jonathan Ventura

Abhijit Bendale

Date May 11, 2017
Doan, Tri Si (Ph.D., Engineering)

Ensemble Learning For Multiple Data Mining Problems

Dissertation directed by Professor, Chair Jugal Kalita

ABSTRACT

Data mining practitioners often face problems of the unavailability of all training data at the same time and the inability to process a large amount of data due to constraints such as lack of adequate system memory. Building a data mining system with whatever data available at a certain time is data is a practical solution. Our hypothesis is that a learning model should be able to update on incoming data in an incremental manner. Another challenge arises when new classes are introduced into a trained system during testing because the learned model does not have an ability to handle unknown classes. Traditional mining models fail to detect and learn new classes. While current solutions have been well studied in computer vision, the challenge of how computer systems deal with unknown classes has received less attention particularly in text classification tasks in Natural Language Processing (NLP). It is in this realm that this dissertation will focus its resources while overcoming the aforementioned challenges. In this thesis, we extend the ensemble learning approach to overcome the large scale data challenge. First, we introduce our solution to select an algorithm for each partitioned region using meta-learning approaches. Next, we propose a solution to aggregate the final prediction. The problem with the majority vote methodology is that the majority outvotes the minority of trained classifier results, often rendering an incorrect final prediction.
ACKNOWLEDGMENTS

I would like to express the deepest appreciation of my advisor Professor Jugal Kalita for his guidance and support during my study at UCCS. Without his guidance and persistent help, this dissertation would not have been possible. I am also grateful to the members of my committee for their patience and support in overcoming numerous obstacles I have been faced through my research: Especially, Prof. Boult with several discussions on the open set problem and help with a poster conference; Prof Lewis for his generous support and his inspiration me to follow the area of data mining, and Prof Ventura for his discussion; and Dr. Abhijit Bendale, a scientist at Samsung Research America for providing his dissertation templates and valuable comments.

I would like to thank my fellow students at the Linc Lab for their feedback; particularly Michael Bankston, and Connor Clark for their willingness to help. My special thanks go to Thomas Conley who used his spare time to read my dissertation and give help with Latex. In addition, I would like to express my gratitude to the Writing Center staff, Heather Gutekunst, Makaela Worden, and Malcome Campbell, for help with my dissertation writing.

I am also grateful to Micheal Bihn for checking typos. I would like to thank Ethan Rudd for his kind help and troubleshooting skills, Chad Mello for his charming support. Last but not the least, I would like to thank Ali Langfels for administrative support.
I would like to express my special thanks to my parents and my wife who have sacrificed tremendously and missed me for more than seven years. Finally, I dedicate this Ph.D. dissertation to my two lovely daughters, Vy and Anh who have been with me during my study.
# TABLE OF CONTENTS

## CHAPTER

### I INTRODUCTION

1. Purpose of study .................................................. 2
2. Scope of study .................................................... 4
3. Methodology ...................................................... 5
   3.1 Components of ensemble learning ............................ 6
   3.2 Diversity measurement for ensemble learning ............... 7
   3.3 How diversity can improve accuracy of ensemble learning ...... 8
4. Ensemble modeling ................................................ 10
   4.1 Loss function ................................................ 11
   4.2 Bias and variance trade-off .................................. 13
   4.3 Conventional ensemble model ................................ 16
   4.4 Negative correlation based ensemble ........................ 16
   4.5 Deep learning based ensemble ............................... 17
   4.6 Combination methods for ensemble learning ................ 18
5. Two types of ensemble models ................................. 19
   5.1 Ensemble learning with same base classifiers ............... 19
2 IMDB movie review dataset .................................. 130
3 Other datasets used in this study .......................... 131

B BAYESIAN INFERENCE 132
1 Pearson correlation coefficient .................................. 133
2 Statistical tests .................................................. 134
LIST OF FIGURES

1.1 Improving prediction with an ensemble model ........................................ 2
1.2 Illustration of different ensemble models ................................................. 4
1.3 Example of the wisdom of the crow in an ensemble model ......................... 9
1.4 Adapt: Understanding the Bias-Variance Tradeoff from Scott Fortmann-Roe. 13
1.5 Bias, Variance and Trade-off ................................................................. 14
1.6 A CNN example .................................................................................. 17
1.7 Decision tree and ensemble model .......................................................... 19

2.1 New split outside current boundary. Adapt (Lakshminarayanan et al., 2014) 36
2.2 New split inside boundary Adapt: (Lakshminarayanan et al., 2014) ............ 37
2.3 Random Forests vs Mondrian Forests. Adapt from (Lakshminarayanan et al., 2014) ................................................................. 37
2.4 Decision tree and a forest of trees ............................................................ 39
2.5 ROCs for 3 experiment settings .............................................................. 44
2.6 Accuracy comparison among incremental learners .................................... 47

3.1 Illustration of open set problem in the real world ..................................... 52
3.2 Outlier detection illustration .................................................................... 59
3.3 Class representation with Nearest Class Mean ......................................... 61
3.4 Reachability from a core point .................................................................. 69
3.5 Example of initial values for a new ball in Online NNO ......................... 70
3.6 Example of initial values for a new ball of the same class in NCC ............ 71
3.7 Second case of initial new ball of the same class in NCC ....................... 71
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.8</td>
<td>Expansion of boundary after update</td>
<td>74</td>
</tr>
<tr>
<td>3.9</td>
<td>A local nearest prediction with different classes</td>
<td>74</td>
</tr>
<tr>
<td>3.10</td>
<td>Accuracy on unknown test cases on Amazon dataset</td>
<td>79</td>
</tr>
<tr>
<td>3.11</td>
<td>Accuracy on unknown test cases on the 20 newsgroup dataset</td>
<td>81</td>
</tr>
<tr>
<td>3.12</td>
<td>Accuracy on unknown test cases on the IMDB domain dataset</td>
<td>82</td>
</tr>
<tr>
<td>4.1</td>
<td>Illustration of common ensemble approach</td>
<td>89</td>
</tr>
<tr>
<td>4.2</td>
<td>Problem of ensemble combination given different training.</td>
<td>95</td>
</tr>
<tr>
<td>4.3</td>
<td>An ensemble member not trained on a certain class, during testing</td>
<td>96</td>
</tr>
<tr>
<td>4.4</td>
<td>An ensemble model diagram.</td>
<td>101</td>
</tr>
<tr>
<td>0.1</td>
<td>The trend of reviews posted on Yelp by years</td>
<td>126</td>
</tr>
<tr>
<td>1.2</td>
<td>Star vs useful votes between elite and non-elite users</td>
<td>127</td>
</tr>
<tr>
<td>1.3</td>
<td>Distribution of votes</td>
<td>128</td>
</tr>
<tr>
<td>1.4</td>
<td>Words in Yelp review vs. star rating</td>
<td>129</td>
</tr>
<tr>
<td>1.5</td>
<td>Timeline of Yelp review posted by years</td>
<td>130</td>
</tr>
<tr>
<td>1.6</td>
<td>Review evaluated by others</td>
<td>130</td>
</tr>
<tr>
<td>2.7</td>
<td>IMDB’s review of Kong Island 2017</td>
<td>131</td>
</tr>
<tr>
<td>0.1</td>
<td>Bayesian framework for data mining</td>
<td>132</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

1.1 Improving predicting with an ensemble model .......................... 9

2.1 Comparing accuracy performance on Yelp dataset ...................... 43
2.2 Comparing accuracy performance on IMDB dataset ..................... 43
2.3 Comparing accuracy performance on Amazon dataset .................. 43
2.4 Accuracy performance comparison of tree-based models ............... 46
2.5 Results of the paired t-tests ............................................ 48

3.1 F-score results on Amazon product review dataset with 10, 20 domains . 76
3.2 F-score results on Amazon product review dataset with 30 and 40 domains . 76
3.3 F-score results on Amazon product review on dataset with 50 domains . . 77
3.4 F-score results on 20 newsgroup dataset ................................ 77
3.5 F-score on IMDB dataset with 6 and 12 domains .......................... 78
3.6 F-score on IMDB dataset with 18 domains ................................ 78
3.7 Pairwise t-test comparison ............................................... 84

4.1 F-scores in Experiment 1a on Amazon product review dataset ......... 102
4.2 F-scores in Experiment 1b on Amazon product review dataset .......... 102
4.3 F-scores in Experiment 1c on Amazon product review dataset .......... 102
4.4  F-scores in Experiment 1a on IMDB movie review  . . . . . . . . . . . . . . . . 104
4.5  F-scores in Experiment 1b on IMDB movie review  . . . . . . . . . . . . . . . . 104
4.6  Accuracy of classifiers in Experiment 2  . . . . . . . . . . . . . . . . . . . . . . . . . 104
4.7  Accuracy of combining methods in Experiment 2  . . . . . . . . . . . . . . . . . . 105
4.8  t-tests in Experiment 2  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 105
CHAPTER I

INTRODUCTION

Two heads are better than one.

John Heywood

Human decision-making usually prefers a conclusion that is convincing from different angles or perspectives. The main reason is to guarantee that the final conclusion has sufficient support. It is commonly referred to as wisdom of the crowd because reasoning shared among many is usually considered better than that of an individual. Trial by jury is one example that shows how the aggregated answer among a group can be obtained. In machine learning, ensemble learning is similar in concept (Rokach, 2010) where multiple classifiers or experts solve and combine the results. Each model plays the role of an expert in the crowd.

In several data mining fields, increasing volumes of data are a big challenge because computational powers do not suffice. With existing computing systems, ensemble learning can be an alternative feasible solution. In fact, ensemble learning has been extensively studied in data mining literature (Cesa-Bianchi and Lugosi, 2006), (Chernov et al., 2010) and has excelled in machine learning competition (Vovk and Zhdanov, 2009).
The popular uses of the ensemble approach consist of a long list of domain applications such as computer vision (Bosch et al., 2007), (Samat et al., 2014), (Du et al., 2015), object tracking (Kalal et al., 2012), (Zhang et al., 2014), bioinformatics and biomedicine (Yang et al., 2010), (Chen and Ishwaran, 2012), spam detection (Saeedian and Beigy, 2012), (Neumayer, 2006), Natural Language Processing (NLP) (Xu and Jelinek, 2004), (Whitehead and Yaeger, 2010), (Luca, 2011) and network intrusion detection (Tavallaee et al., 2009), (Gogoi et al., 2012), (Maglaras et al., 2016).

![Figure 1.1: Improving prediction with an ensemble model](image)

### 1 Purpose of study

Ensemble learning is a framework that combines a set of classifiers to improve accuracy performance in many classification problems. Ensemble models often place high in data mining competitions and thus demonstrate the advantage of ensemble learning over any single data mining algorithm. Our goal is to develop a data mining model that scales well to data. To achieve this goal, we investigate a feasible solution using ensemble learning that allows us to deal with three different challenges as follows:

- training data is introduced in an incremental order,
- unknown classes are able to detect and learn, and
- the new combined approach is able to balance between a minority and majority of classifiers.
The first challenge is tackled by a proposed incremental ensemble model. Later, we explore its use to address the problems posed by large datasets that can be only processed fractionally due to system memory constraints. In the second challenge, we extend our study on ensemble learning to deal with text classification under open set settings to address limitations of current models. Finally, we investigate enhancing the ability of the ensemble model using a new combining method.

Most ensemble models work well under an implicit assumption of having a single set of data points. However, a divide-and-conquer approach can generate several datasets which may have different distribution such that ensemble members may work well only in particular data regions. A prediction generated by aggregating predictions from all ensemble members may be in a question if a majority of ensemble member performs poorly in data regions. We overcome this challenge with a new method for combining the predictions of different classifiers. This solution provides crucial support for the scalable model proposed in this thesis. We summarize the goals of our approach as the following: **dynamic, being able to handle real world assumptions** and **scalability**, making it more amenable to dealing with large scale data mining and machine learning problems.

- ‘Dynamic’ describes an incremental learning approach where a proposed model can incrementally update itself with incoming data.
- ‘Real world assumption’ addresses the problems that arise when training data is available in incremental order.
- ‘Scalability’ requires an alternative and optimal solution to combine solutions to the first and second problems.
2 Scope of study

Our objective is to propose a scalable algorithm with ensemble learning. Our study focuses on the classification problem using ensemble learning even though similar solutions can be achieved for the regression task also. Particularly, we follow an incremental approach where training data are learned in chunks. We use incremental learning to handle large scale data by building a model with available data and allowing the model to update itself efficiently. Figure 1.2 illustrates two ensemble models. The left figure describes a situation where ensemble members are created from different subsets of data with sampling. The right figure describes an ensemble model where different ensemble members are created from different subsets of data by partition.

State-of-the-art ensemble models such as Random Forests and eXtreme Gradient Boosting are developed with the left figure approach. Two types of ensemble classifiers built from the same type of base classifiers are tree-based and Nearest centroid-based classifiers.
An ensemble of tree-based models is known for its high performances whereas one built from Nearest Centroid classifiers is a natural choice to deal with the open set problem. We investigate solutions with both approaches. We further study the second approach (refer to the left Figure 1.2) where ensemble members are different types of classifiers.

3 Methodology

Ensemble learning refers to a learning technique that achieves high predictive power by aggregating or combining the predictions of a set of different models (supports the concept of “wisdom of the crowd”). In general, an ensemble model consists of several base algorithms and a combiner. As mentioned, the primary objective of using ensemble learning is to improve the performance of a system as a whole. Our methodology is described as follows.

- Top-down approach with ensemble learning: A problem is be decomposed to several sub-problems where ensemble learning is applied. Breiman et al. demonstrate that optimum accuracy can be achieved by aggregation of several classifiers (Breiman, 1999). In this work, we focus on the ability of incremental learning model which does not support a distributed approach.

- Algorithm selection for a given data space: Searching a classifier for a dataset is often difficult since a poor choice may affect the final prediction of the ensemble model. An automatic advice system may provide a better way to deal with an unfamiliar data domain, high consulting costs of experts, or time-consuming experiments.

- Combined prediction methods of ensemble members: Current combination methods work well under certain specified assumptions such as prior knowledge of testing classes. Given the open set problem, a combining method needs to address the problem of classifiers untrained on certain classes outvoting the minority of classifiers that have actually been trained on these classes.
3.1 Components of ensemble learning

A combination of arbitrary classifiers may not improve the accuracy of performance (Džeroski and Ženko, 2004). In order to achieve improved performance, two main components define an ensemble learning system: 1) the diversity in classification algorithms, and 2) combining strategy (Dietterich, 2002). Diversity aims at learning from different errors by a variety of classifiers (Brown et al., 2005a). Without diversity, there is no effective learning gain because the same errors are always treated the same way by the system (Brown et al., 2005a), (Peterson and Martinez, 2005). It is noted that diversity requires that the classifiers be independent of each other. (Chandra and Yao, 2006) further consider the interactions among an ensemble’s classifiers by introducing a correlation penalty coefficient into the cost function of each ensemble member. As a result, each classifier minimizes its mean square error (MSE) together with the remaining ensemble members. However, independence among classifiers does not guarantee the best outcome in the combined classifier (Kuncheva et al., 2000). Combination of many independent and poor performing classifiers may not improve the ensemble model’s performance. Diversity can be achieved in several ways: manipulation of data, and hyper-parameter, and by using a suitable architecture.

- Data diversifies generate multiple datasets from the original data source. Different training results are produced in different models.
- Different hyper-parameter settings generate different predictors.
- The architecture is the structure of an ensemble model that may distinguish one ensemble from another.

Diversified data can be obtained by Bagging (Breiman, 1996), Boosting (Freund et al., 1996), or the use of Random subspaces (Ho, 1998) while hyper-parameter searches can
produce a large number of predictors from a base classifier. Ensemble learning can use more than one layer where the outcome of one classifier is an input to another classifier.

The method of combining classifiers determines how a final decision is made from the ensemble classifier. Common combining methods include majority voting such as Unweighted or Weighted Majority voting (Kim et al., 2011), and borda count which takes the vote for each class into account. Voting methods such as Unweighted/Weighted majority voting implement a winner-take-all strategy in which the vote is assigned only to the class chosen by the classifier but not to other classes. Borda count considers the class with the most support as well as the class with the second most support, etc. The votes are summed and the class with the most votes is represented as an ensemble decision.

### 3.2 Diversity measurement for ensemble learning

Currently, ensemble learning is assessed using two approaches, pairwise and non-pairwise measures. In the pairwise approach, the diversity of an ensemble learner with $n$ classifiers is defined as an average of all $\frac{n(n-1)}{2}$ pair values. To compute a single value, all pairs are be averaged. Several metrics include the disagreement measurement to determine the probability that two classifiers disagree on their results (Ho, 1998), and the double-fault measurement to determine the probability of both classifiers being incorrect (Ruta and Gabrys, 2001).

On the other hand, the non-pairwise approach takes all classifiers into account at once. Included in this category are entropy measurement (Cunningham and Carney, 2000), Kohavi-Wolpert variance (KW variance) (Hansen and Salamon, 1990), Interrater agreement (Fleiss
et al., 2013), and measurement of difficulty (Hansen and Salamon, 1990). With the entropy measurement, 1 denotes the most diversity while 0 indicates no diversity. KW variance measures the variance of predicted labels over different training sets by classifiers, and computes the probability of correct or incorrect predictions. Interrater agreement computes the number of times the individual learners agree on correct labels while measurement of difficulty estimates the shared difficulty among the classifiers that data points present in classification (e.g., data points that are often given incorrect labels) (Giacinto and Roli, 2001). Each metric is required to have a minimum threshold to measure the diversity and the value depends on a specific problem.

3.3 How diversity can improve accuracy of ensemble learning

In a top-down approach, a problem can be decomposed into independent sub-problems in order to solve it efficiently. In the same vein, ensemble learning breaks down the data space into several sub-regions and performs the learning task in each sub-region. Known as the wisdom of the crowd approach, diversity of experts is vital to produce the best decision. This concept that inspired ensemble learning assumes that a diversity among ensemble classifiers (known as a base classifier) is important for the accuracy of the model. Practically, the performance of ensemble learning is desired to be at least comparable to the performance of its best classifier member (Džeroski and Ženko, 2004). Understanding diversity is vital to justify the choice of algorithms used in ensemble learning. Considering an ensemble model with majority vote scheme. Let A, B and C denote 3 binary classifiers. Suppose we have a test set of 10 examples and the ground truth is all “1”. We can see that an ensemble
Table 1.1: Improving predicting with an ensemble model

<table>
<thead>
<tr>
<th>A, B and C are high performers</th>
<th>A and B are high performers but not C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111111100 = 80% accuracy</td>
<td>1111111100 = 80% accuracy</td>
</tr>
<tr>
<td>1111111100 = 80% accuracy</td>
<td>1101111110 = 80% accuracy</td>
</tr>
<tr>
<td>1011111100 = 70% accuracy</td>
<td>1011111100 = 70% accuracy</td>
</tr>
<tr>
<td>Combined votes:</td>
<td>Combined votes:</td>
</tr>
<tr>
<td>1111111100 = 80% accuracy</td>
<td>1111111110 = 90% accuracy</td>
</tr>
</tbody>
</table>

may sometimes actually boost the final accuracy performance using an error correcting method (case 2) while there is no gain in the case 1. Taking a closer look, all three classifiers A, B and C have agreed on most of the testing examples. That is when A classifies 1 so does B, and even C except in the second testing example. In case 2, both A and B achieve 80% accuracy but they disagree on the third and ninth examples. C misclassifies the second, ninth and tenth examples resulting in 70% accuracy. But a Majority vote works well to correct errors and improve the final prediction.

Additionally, an ensemble model outperforms any single ensemble member if the members are independent or uncorrelated. To see this, there is another example with even lower performing individuals as seen in the following. Given 10 test examples with ground truth all “1”, Figure 1.3 illustrates how the ensemble can produce correct prediction from the predictions of the individual learners.

Figure 1.3: Example of the wisdom of the crow in an ensemble model

We now observe that a more diverse set of algorithms is better suited for an ensemble model. In other words, an ensemble model is only efficient when the ensemble members
are less correlated. Importantly, these uncorrelated ensemble members may include low performing classifiers, but not classifiers that do not under-perform. In fact, an ensemble of under-performing (e.g., random guessers) will not improve the final performance.

It is sufficient to say that when all training examples are correctly labeled by the same number of base classifiers, ensemble margins (referred to also as the uniformity condition) have been achieved with diversity. However, (Tang et al., 2006) indicate that maximizing diversity does not come with the desired improvement of accuracy unlike maximizing classification margins. Their argument is that the not-easy-to-achieve uniformity condition leads to a weak relationship between diversity and performance. To complicate the issue, high diversity may lead to high performance of ensemble learning in one study but poor performance in others. What this means is that if we were to choose an ensemble by maximizing diversity, the results may be unpredictable, posing a question of how diversity is considered either good or bad (Brown and Kuncheva, 2010).

4 Ensemble modeling

In an earlier work, (Dietterich, 2000) indicated the fundamental reasons for the use of ensemble learning: computational, statistical and representational benefits that accrue. This work supports the bias-variance decomposition finding in (Kohavi et al., 1996) and later well-known Random Forests model proposed by (Breiman, 2001). Considering a classification task where we have a set of training examples of the form \((x, y)\), a classification model will predict which class a particular example belongs to. Let a general classification function be defined as
\[ y = f(x, \theta) \quad y \in \mathcal{Z} \] (I.1)

where \( x \) denotes a new observation vector of \( n \) features or \( \bar{x} = x_1, x_2, \ldots, x_n \), \( y \) denotes a predicted class and \( Z \) denotes a set of nominal values. In the same way, a regression task uses a similar function, but an output \( y \) is defined as continuous outcome

\[ y = f(x, \theta) \quad y \in \mathcal{R} \] (I.2)

where \( \mathcal{R} \) denotes a set of real values. In both cases, \( \theta \) denotes parameter values.

It is known that traditional classification and regression models require all training loading into the system memory at the beginning of a specific learning stage. After a training process is finished, any new training example can be learned by starting a new training process again. Instead of resetting a system, incremental learning is designed to learn a new concept on the fly. This characteristic gives it an ability to learn with incomplete training examples. Learning by chunking a subset of examples is an approach we suggest to tackle the large scale problem.

### 4.1 Loss function

Ensemble methodology is defined as a divide and conquer when a problem is decomposed into subproblems, and solutions are aggregated into a final solution. Predicted outcomes from all ensemble members are aggregated to a second layer of classifiers where another classifier produces a final prediction. To be able to outperform each ensemble member, a diversity among ensemble members is very important. With a diversity, Windeatt et al.
(Windeatt, 2005) demonstrate that the ensemble model can achieve smaller square error than the average squared error of the based predictors for a given proper ensemble predictor in a regression problem. For multiple datasets, Domingos et al. (Domingos, 2000) formulated the expected error of an ensemble model as following:

$$E[(\bar{f} - t)^2] = \text{bias}^2 + \frac{1}{S} \text{var} + (1 - \frac{1}{S}) \text{covar}$$

$$\text{bias} = \frac{1}{M} \sum_i (E[f_i] - t),$$
$$\text{var} = \frac{1}{S} \sum_i E[f_i - E[f_i]]^2,$$
$$\text{covar} = \frac{1}{S(S-1)} \sum_i \sum_{j \neq i} E[f_i - E[f_i]](f_j - E[f_j])$$

where \(E(.)\) denotes Expected error, \(S\) denotes size of the ensemble, \(f_i\) denotes outcome of \(i_{th}\) ensemble member, and \(t\) denotes target. Bias are the simplifying assumptions made by a model to make the target function easier to learn. Variance (refers to as var) indicates the choice of training datasets affecting built models.

Equation I.3 shows influences on the ensemble errors by mean bias (between prediction and ground truth), mean variance (among ensemble members) and mean covariance (average pairwise difference among ensemble members). Firstly, when the number of ensemble learning increases, partial variance in this loss function decreases. Secondly, a negative covariance may reduce the expected loss of the ensemble model. It turns out that low correlated (uncorrelated) individual models allow for an increase in ensembles’ performance. For a classification task, the same goal can have been proved by using bias and variance decomposition in (Kong and Dietterich, 1995), (James, 2003).
4.2 Bias and variance trade-off

Given a particular dataset, the results of classifiers can vary often from one classifier to the others. This expected error due to model mismatch is defined as bias, whereas the variation due to the train sample and randomization is referred to as variance. Determining a good trade-off (James et al., 2013) between bias and variance is important to achieve a generalization of a model. For a data point \( x^*, y^* \), an expected prediction error of a classifier is defined as

\[
E[(g(x^*) - y^*)^2] = E[g(x^*)^2] - 2g(x^*)y^* + y^*^2
\]

\[
= E[g(x^*]^2 - 2E[g(x^*)y^*] + E[y^*^2] \quad since \quad E[g(x^*)y^*] = g(x^*)y^*
\]

\[
= E[(g(x^*) - E[g(x^*)]]^2 + E[g(x^*)]^2 - 2E[g(x^*)]f(x^*) + E[(y^* - f(x^*))^2] + f(x^*)^2
\]

\[\text{since } E[X^2] = E[(X - E[X])^2] + E[X]^2 \quad \text{by I.5}\]
Proof:

\[ E[X^2] = E[X^2] - E[X]^2 + E[X]^2 \]
\[ = E[X^2] - 2E[X]^2 + E[X]^2 + E[X]^2 \]
\[ = E[X^2] - 2EX[X] + E[X]^2 + E[X]^2 \]
\[ = E[(X - E[X])^2] + E[X]^2 \]  \hspace{1cm} (I.5)

We can arrange I.4 to

\[ E[(g(x*) - E[g(x*)])^2] + E[g(x*)]^2 - 2E[g(x*)]f(x*) + f(x*)^2 + E[(y* - f(x*))^2] = \]
\[ E[(g(x*) - E[g(x*)])^2] + (E[g(x*)] - f(x*))^2 + E[(y* - f(x*))^2] \]

\[ \text{variance of the estimator} \hspace{1cm} \text{bias of the estimator} \hspace{1cm} \text{variance of noise} \]

Note that the third term does not depend on the estimator \( g(x) \) so it often refers to as irreducible error.

![Image of bias, variance, and trade-off graphs](image1.png)

**Figure 1.5: Bias, Variance and Trade-off**

There are two common ways to reduce this bias-variance problems: Ensemble method (Bauer and Kohavi, 1999) and regularization (Friedman et al., 2001).

- Given \( n \) independent samples, a probability distribution determines by variance \( \sigma \) and mean \( \mu \). By averaging (for example with ensemble), we can achieve a new variance \( \frac{\sigma^2}{n} \) while a mean \( \mu \) is unchanged.

- Regularization uses L1 norm or L2 norm to deal with bias and variance problem.

Ensemble learning averages all classifiers’ outputs to reduce variance (refer Figure 1.5). Random Forests (an ensemble of Decision trees) takes this into account in the design
Breiman (2001) by first building a set of Decision trees. The difference among trees is obtained by sampling examples and sampling features using Bagging (Bootstrap Aggregation). The optimal split decision at an intermediate node relies on a random selection or a linear combination of features. Examples excluded from sampling are defined as “out-of-bag” (OOB), which are used to test each binary tree. The final prediction is obtained with a combiner.

While sampling examples allows for faster training time due to a small subset of examples, randomized feature selection reduces a possible correlation among features that can improve the performance. These features give Random Forests a predictive power that is comparable to the best state-of-the-art algorithms in classification and regression problems. Random Forests possess several properties such as a natural design that allows them to work in distributed framework, naturally multi-classes and robust to noisy system. As a result, Random Forests have become an attractive method for a variety of applications in computer vision such as image classification, pedestrian detection (Marin et al., 2013), and facial key point tracking (Lepetit and Fua, 2013). Many variants of Random Forests have been successfully tailored to specific data mining needs. For example, Hough Forests, a variant of Random Forests, implement Hough transformation during the process of building trees to identify features of a particular shape within an image for object detection (Gall and Lempitsky, 2013). Nearest Class Mean Forests (Ristin et al., 2014) use nearest class mean (mean descriptor for each class) at decision nodes to identify objects. Given a large dataset, Random Forests emerge as a prominent candidate of choice. However, most Random Forests models have been designed and implemented for batch learning (refers to as off-line) where all training data is available at the training time. The general issue of these off-line
techniques is that all training data and grown trees remain in the main memory.

4.3 Conventional ensemble model

Ensemble models with Bagging, Boosting and Stacking, are considered as conventional ensemble models that are well-studied in machine learning literature. Bagging is a Bootstrap Aggregating example with a replacement which allows for creating different classifiers. A special ensemble model, for e.g., a Random Subspace applies Bagging to feature space instead of sample spaces. Stacking or Stack Generalization (Wolpert, 1992) as well as Blending (Jahrer et al., 2010) is a generalized method to Bagging. It uses multi-layers where results from the current layer aggregates to the next layers. The classifier in the second layer estimates weights for the outputs from based learners (Muja and Lowe, 2009).

4.4 Negative correlation based ensemble

An uncorrelated model is considered as a main factor to determine diversity among ensemble models. The ensemble models can perform independently before aggregating their predictions to the next layers. On the contrary, negative correlation ensemble (Liu and Yao, 1999) trains its neural network members simultaneously using a correlation penalty term in its loss function. In both regression and classification tasks, this model uses the simple averaging method. Negative correlation model with regression (Brown et al., 2005b) is designed with two neural networks, Multi-Layer Perceptron (MLP) and radial basis function neural network. A Negative Correlation model for classification proposed by (Dam et al., 2008) uses the Artificial Neural Network as a base learner.
4.5 Deep learning based ensemble

The emergence of neural networks in data mining is fueled by their high performance in face recognition tasks. For example, the Convolutional Neural Network (CNN) architecture (LeCun et al., 1998) is a feedforward neural network that has multiple layers including convolution, activation and pooling layers, followed by a fully connected and a final output layer. The CNN is an example of a deep neural network or deep learning.

![A CNN example](image)

In a deep structure, one CNN can stack on top of another CNN to form multiple layers of non-linear operation. An ensemble of deep SVMs (Abdullah et al., 2009) uses one SVM in the first hidden layer for feature extraction. This layer is used as an input for the next layer to approximate target function. An ensemble of deep learning for regression (Qiu et al., 2014) uses Support Vector Regression (SVR) as the aggregate algorithm. Neural Networks (NN), particularly Deep learning (including CNN) have demonstrated their predictive power in the computer vision domain. We believe that it is feasible to apply Deep Learning in the field of Natural Language Processing (NLP). However, words and sentences in NLP will replace the image pixels of computer vision. Word, for example Word2vec (Le and Mikolov, 2014) or GloVe (Pennington et al., 2014) use vectors that index words in one-hot encodings. Such word vectors can be put together in matrix form for computation.
4.6 Combination methods for ensemble learning

Word vectors in natural language processing are sometimes referred to as word embedding, a low dimensional representation. An image of a 64 x 64 matrix in computer vision can be reimagined as 64 x 64 matrix of 64 words in 64-dimensional embeddings for NLP representation. Character-embedding using CNN has also been used for part-of-speech tagging (dos Santos and Zadrozny, 2014). A Deep Learning architecture with 9 layers has been used by (dos Santos and Zadrozny, 2014) for sentiment analysis and (Zhang et al., 2015b) for text categorization tasks. (Kim et al., 2015) use the output of the character level as the input to an LSTM (Sak et al., 2014) and apply to different languages.

Let $D = \{D_1, \ldots, D_n\}$ denote an ensemble model of n classifiers and $\omega = \{\omega_1, \ldots, \omega_c\}$ denote a set of classes. There are two common outcomes of a classifier: a class label, and a probability of belonging to one class (or probability of membership) for a classification task, or a real value for the regression task.

(Kuncheva, 2004) proposes a probabilistic framework for combining label outputs. Let $P(\omega_k \text{ is the true class } | s)$ or for short $P(\omega_k | s)$ be the probability of a predicted class for class label output $s = \{s_1, \ldots, s_L\}$. We can write

$$P(\omega_k | s) = \frac{P(\omega_k)}{P(s)} \times \prod P(s_i | \omega_k). \quad (I.6)$$

For an ensemble of classifiers, there are two ways to generate the final decision: 1) combining and 2) generating. A combining method commonly uses Majority Vote (hard vote), (Randomized) Weighted Majority vote (soft vote), and a Naive Bayes combiner, while a generating method often uses average and product. Since classifiers may generate different predictions, Weighted Majority vote is often used to improve the prediction.
5 Two types of ensemble models

5.1 Ensemble learning with same base classifiers

Decision trees are well-known for high accuracy in the field of data mining (Quinlan, 2014). Decision trees are also sensitive and unstable to training data. This means that a different set of data or additional data during training will result in different trees. The process of classification with Decision trees is as follows. In Figure 1.7, an incoming data example is classified by following down from the top of the tree (root) to the leaf. At each intermediate node, a splitting decision is made to determine the left or the right direction this example should follow. To measure the possible classification error, we define a loss function as follows. A training example $\mathcal{D}$ is defined as $\mathcal{D} = X, y$ where $X = x_1^m, ..., x_p^m$ denotes data and $y = y_1^m, ..., y_n^m$ denotes label at the current node $m$. The reduction in error by splitting at node $m$ is determined by $\delta(y^m) = L(y^m) - \left[ \frac{n_{m_l}^m}{n_m^m} L(y_{m_l}^m) + \frac{n_{m_r}^m}{n_m^m} L(y_{m_r}^m) \right]$ with $n_{m_l}^m$ and $n_{m_r}^m$ being the number of examples that go to the left or the right of the split, and $L(.)$ being the loss.
Two loss functions are often presented in classification problems including Gini loss function (refer to I.7) and Entropy loss function (refer to I.8).

\[
L_{\text{ent}}(y^{(m)}) = - \sum_{d \in D^{(m)}} p^m(d) \log[p^m(d)]
\]  

(I.8)

As a Decision tree grows, the predicted model is prone to overfitting. To ameliorate this problem, stopping criteria are often introduced. These include the depth of the tree reaching a certain threshold, the number of data points or the homogeneity of distribution in the terminal nodes going below certain parameterized threshold values. The main objective of using the stopping criterion is to prevent a Decision tree from growing too complex so that errors in testing cases do not deteriorate compared to errors in training. Pruning the tree is another method to deal with overfitting when a tree has grown to its fullest. Such approaches may lead to different prediction results because the trees become different.

5.2 Ensemble learning with different base classifiers

This approach has emerged as a popular practice with the diversity of the ensemble members coming from different types of classifiers instead of relying on different subsets of examples generated by sampling techniques. When such diversity is guaranteed, each classifier is distinguished from each other on predictive performance and usually the final final performance is better.

A common method to combine predictions of all committee members is by a Majority
voting. The simplest form is a democratic system where consent of more than half of the committee members determines a final decision. For an ensemble of classifiers where each classifier is selected as a different expert, different weights may need to used. (Littlestone and Warmuth, 1989) in their study, discover the benefit of a Weighted Majority of votes and propose a new process to address the drawback that rises due to high bound for weighted majority of votes. (Zamani et al., 2014) introduce weighted majority of votes with cascade. However, these current solutions only work with the closed set assumption where each ensemble member has prior knowledge of all classes.

The cold start problem arises when a new member has a relevant knowledge of a new class w.r.t the remaining members. A majority of votes from the remaining members can outvote the knowledgeable voter and result in incorrect predictions until classifiers with relevant prior knowledge of the new class become sufficient. We investigate more on the combination approach in Chapter IV while tackling a large scale problem. Particularly, our goal is to propose a new weighted majority vote, able to work on open set scenarios. To achieve this goal, a proposed combination method should be able to deal with the cold start problem when a number of classifiers with untrained classes in an ensemble model is a majority.

6 Summary of work

Our work can be summarized as follows. **Problem 1: Incremental ensemble learning.** We investigate issues that arise when data are not available at once or arrive continuously. This often results in large data sets. We explore the incremental learning approach and
our candidate is an ensemble of tree-based models, which is an example of ensemble of homogeneous models. Inspired by the recent introduction of the Mondrian process (Roy and Teh, 2009) and Mondrian Forests (Lakshminarayanan et al., 2014), we explore the advantages and disadvantages of the current version of Mondrian Forests in an incremental approach. In the current implementation of Mondrian Forests (Lakshminarayanan et al., 2014), we observe that individual sampling of features degrades the performance, particularly when the selection of informative features is not guaranteed. Mondrian Forests also suffer from a critical drawback as extensive memory is required because it maintains all training samples in memory to compute a split in each node. This is an acute problem particularly when it reaches about 500,000 samples. Our goal is building a new model that applies to sentiment analysis on review datasets, which evolve over time. This work has not been attempted before despite the fact that the reviews are often generated in a time series. Using a stationary approach for analysis of streaming data will lose a lot of information, resulting in a misleading conclusion. In business, a bad review without reference to a considered time line can make or break the business. One challenge of our study is that the current version of Mondrian Forests is unable to work with sparsity or high dimensionality. These characteristics are commonly seen in many NLP problems. As a result, we perform a redesign, although we keep the main idea of a sampling approach using the Mondrian process.

**Problem 2: Ensemble learning with unknown classes.** Traditional supervised learning works well with prior knowledge of all classes at the training time. In the real world, examples of unknown classes are commonly seen during testing. This scenario likely occurs in continuous data due to the unavailability of training data at once. Much of the significant
work on open set problems has been done by a group of researchers at the VAST Lab, UCCS (Scheirer et al., 2013a), (Jain et al., 2014b), (Bendale and Boult, 2015a). Their work has inspired our curiosity in this new idea in data mining. In our research, we found that the concept of an open set of classes had been used in earlier literature, such as (Li et al., 2006) in stylometry, (Sommer and Paxson, 2010) in intruder network detection, and (Koppel et al., 2011) in authorship attribution. For example, (Stolerman et al., 2011) demonstrate their solution to tackle the problem of determining an unknown author. The fundamental problem in forensic stylometry relies on linguistic analysis to determine the genuine author of an unknown text. (Sommer and Paxson, 2010) and (Stolerman et al., 2011) depend on an approximate solution that computes probability of new classes to solve this challenge.

Open set recognition and open world problems (Scheirer et al., 2013a), (Jain et al., 2014b), (Bendale and Boult, 2015a) have gained new attention among data mining practitioners after these papers’ publication. In this work, we improve the online version which has been introduced from (Bendale and Boult, 2015a) and successfully apply this to text classification problems.

**Problem 3: Ensemble learning for a large scale problem.** Large scale problems are the most challenging for data mining tasks, especially with recent advanced technologies to collect data in many fields such as communication, health care, or banking. A lot of attention among data mining communities focus on a distributed approach such as (Lin and et al., 2014). While a top-down approach in problem solving is very popular, ensemble learning can use this approach to tackle the large scale problem in a two-step processes.

First, the decomposition of a dataset into a subset of data can be performed with random sampling of features and/or random sampling of examples. Then a solution for each subset
of data with the best selected algorithm (an expert for the specific subset) can be obtained. Second, solutions for a set of sub-problems can be aggregated and integrated into the final solution. Ensemble learning uses this approach (Ho, 1998) is limited to a single base classifier (tree-based classifier), rather than different base classifier as recent top performers in a data mining competition.

We tackle the large dataset problem in two related sub-problems: finding a ranked list of algorithms for a given dataset (a subset of data space) and finding the best combination of ensemble classifiers. The problem consists of choosing a high performing algorithm and weight assignment for each algorithm of an ensemble member. Published papers from winning data mining competitions focus on the model architecture rather than the logic behind each ensemble model. Trial and error is a main practice of many data mining practitioners. We tackle this challenge in two studies: algorithm selection, and by developing an ensemble model for data mining tasks.

A common challenge of ensemble learning is to choose its members. Choosing the base classifiers usually relies on personal experience or opinions from experts. We investigate how algorithms can be chosen for ensemble learning based on a meta-learning framework. Given an unknown dataset, our approach generates a ranked list of algorithms in which the top $k$ ($k$ is a parameter indicating the number of algorithms in the final ensemble model) algorithms will be of interest to an ensemble learner.

A combination of multiple classifiers is crucial for the success of ensemble learning. A selection of best performers (ensemble member) may not guarantee the best prediction of an ensemble model. A naive method is to use a brute-force approach to search through an entire combination of ensemble members. However, it is problematic. A common practice for the
combination to obtain is a majority of the votes. It was actually used in early development of an ensemble model. When classifiers are distinct, each classifier may provide different predicted results for a given set of data examples. The Weighted Majority vote that assigns different weights for ensemble members is a better method. In our work, we suggest a new solution for weight assignment using a multi-arm bandit approach.

The era of big data has emerged in many fields with unprecedented growth of data at a magnitude of million or even more records per day. Big data are testing the limitations of data mining algorithms as well as computer resources. When amount of big data are generated as a result of streaming, new patterns may evolve as time elapses. The hidden patterns may represent a new concept, referred to as a concept drift by (Widmer and Kubat, 1996), requiring the system to adapt without interfering with the working model.

The approaches in data mining can be classified into two types: static and dynamic. A static method obtains a solution with a single data mining algorithm, commonly known as a distributed method. This is also known as scale-out where multiple-computer systems perform parallel processing of data mining tasks, e.g., a Hadoop distributed system. Recently, the Spark distributed system has been used effectively to implement parallel functions of a distributed system in a single computer. Alternatively, a dynamic method studies a solution for continuous data but its approach works for non-streaming data too. This method addresses the limitation of computer resources, especially with memory. Commonly, data mining algorithms are required to load and store an entire training dataset, which may not be feasible due to memory constraint.

A solution is breaking data into chunks and reading them in a sequence. Streaming data also shares a similar characteristic in that data examples come over time and need
to be processed in sequence. To deal with this problem, incremental learning starts at the
time of incoming data, and applies its adjustment for the next prediction task. As a result,
incremental learning can be studied as a means to deal with the large data problem where
data size is bigger than the system memory.

Ensemble learning, on the other hand, integrates the main concept of incremental learning
in its own design. Incremental learning can be implemented with a single classifier of an
ensemble model (e.g., incremental Decision tree), an ensemble model with a single type of
base classifiers (e.g., incremental Random Forests) or ensemble model with different base
classifiers. For the sake of simplicity, we use the term incremental ensemble learning to refer
to the ensemble model using an incremental approach while the term ensemble learning is
used to emphasize the combination of multiple models.

6.1 Contributions of the thesis

1. Incremental ensemble model: In Chapter II, we address the challenges that tradi-
tional machine learning algorithms face, including unavailability of entire large dataset
at one time due to high cost of collection, nature of data (data examples introduced
in incremental order), and the problem of loading the entire dataset for training. A
feasible solution is building a model on the currently available data and updating with
new training data examples as they become available. While Random Forests are is
considered a best feasible solution for incremental learning, its drawback is that it
is unable to reintroduce the split at current node making it hard to obtain efficient
performance. Our proposed model tackles this problem and outperforms other tree
based ensemble models.

2. **Tackle unknown classes** : In Chapter III, we show that streaming data can present an open set problem by their basic nature. We review several previous works in computer vision as well as natural language processing on this topic. We study state-of-the-art solutions and their concerns. We present a new solution to the open set problem, along with the challenge of sparse data. We indicate the problem of using traditional tree based ensembles that fail to extend to deal with the open set problem. We propose our ensemble model and demonstrate it in text classification.

3. **Large scale data with ensemble learning**: In Chapter IV we address the other challenges posed by a big volume of data. A distributed approach may not be a good choice for streaming data. Instead, we use the divide and conquer approach by decomposing data into sub-datasets, and building models by training on each region, and aggregating all predictions into a final result. This is a challenge because there is no guaranteed that a testing example may belong to the same region of training. We demonstrate that ensemble learning has the ability to correct this error. The contribution is that we can use this outcome in the previous study and perform the prediction with our proposed combination model.

In the following chapters, the above mentioned problems and proposed solutions are discussed in detail\(^1\).

\(^1\)Each chapter mentioned in this thesis is presented as a “stand alone” techniques. The symbols and notations are specific to each chapter, unless specified otherwise.
6.2 Publications

This thesis has resulted in following publications (current and in progress)

1. T. Doan, J. Kalita “Sentiment Analysis of Restaurant on Yelp review” IEEE International Conference Machine Learning and Application (ICMLA) Dec 2016, Anaheim, CA (Chapter II)


5. T. Doan, J. Kalita “Predicting run time of classification algorithms under meta-learning approach” Journal Machine Learning and Cybernetic, June 2016
CHAPTER II

INCREMENTAL ENSEMBLE MODEL

Data generated at high speed in real time from network traffic monitors, log records from click streams, and news feeds result in an increase in demand for scalable algorithms that can handle a massive number of data points. Despite the fact that some current state-of-the-art powerful computational systems can meet this high demand, they are not practical posed by large amounts of data for most in data mining communities. This thesis focuses on providing a suitable tool to discover in-depth knowledge for application domains under the constraint of finite memory resources by processing data in chunks.

Mining with evolving data is often referred to as an incremental or online approach where the main difference is the ability to phase out old knowledge to reserve space for the new. While both approaches give a system the ability to update for incoming training data, online learning requires extra resources to deal with dynamic change in contents known as concept drift. One state-of-the-art method that is the first candidate in this thesis is incremental learning using the Support Vector Machines (SVM). The goal is to adapt learning for new concepts without sacrificing the performance, because retraining the
entire model is not desired.

Ensemble learning supports the principal characteristic of incremental learning: navigating the space for learning new concepts in an additive manner. In ensemble learning, each batch of data (referred to as a sample of examples) can be used to construct a new model member or classifier. Ensemble models train several members where each member is constructed with a subset of data space. Since each classifier in ensemble learning is constructed using only data examples from an individual batch, the strategy for sampling data is important to construct predictive models of ensemble learning. Importantly, an ensemble model also supports traditional data mining algorithms by loading the entire training data. This feature makes an incremental ensemble model advantageous in real world applications.

In practice, finite memory resources hinder the use of many data mining algorithms due to the fact that many of them require that all the training data is residing in memory and this is not conducive to incremental computation. There are two types of incremental approaches: batch and instance. The batch incremental ensemble learning (or simply ensemble learning) requires specification of a batch size, determined based on the total available memory in a system. In an instant incremental ensemble learning, each example arrives and is processed one at a time. Online learning uses this approach.

With incremental learning, the new example may cause the underlying distribution of the data to change. As a result, the prediction may suddenly become less accurate. Typically, two approaches are used to improve the accuracy of incremental learning, 1) sampling data such that we guarantee that the distribution of data is preserved, and 2) using a sliding window method to deal with concept drift. Inspired by a new method of sampling data (Roy and Teh, 2009), this dissertation investigates a novel design of incremental batch
learning based on KD structures. Herein, the number of ensemble members are established a-priori and training data arrives sequentially, requiring adjusting of ensemble members. To deal with larger datasets, we also propose a strategy that maintains and updates the set of ensemble classifier members so that we can update the old model by dropping an ensemble member and/or including a new ensemble member classifier.

Limitations of current ensemble learning with Random Forests include the need for a large number of Decision trees to achieve good accuracy (Denil et al., 2013), and that retraining is slow with batch learning. Finding an efficient way to deal with these issues is an interesting challenge.

1 Related work

Incremental learning is inspired by the need to use data mining algorithms on stream data, where training data instances come along a timeline. An early model using incremental of induction Decision trees (Utgoff, 1989), reconstructs a Decision tree by determining a feasible split after each incoming data instance arrives. The downside of this approach is that it is possible to produce an unstable tree in some rare cases when the splitting feature may be changed repeatedly as a result of incoming data. Furthermore, a single Decision tree has been known to be outperformed by a forest of Decision trees (an ensemble model) that uses consensus opinion. A particular example is (offline learning) Random Forests (Breiman, 2001) where a random selection of features is used for splits. In the opposite direction, Decision Forests proposed by (Criminisi et al., 2012) take an entire set of features as well as combinations into consideration for each split. With training data arriving on a timeline,
incoming data cannot be used to correct a previous split when a decision at a particular node
had already been made. To overcome this problem, (Domingos and Hulten, 2000) keep
several splitting candidates in every leaf and propose a method, known as Hoeffding bound
to estimate the probability of a good split. The use of this split method produces Hoeffding
Trees, which are reported to have better performance (Bifet et al., 2009).

Because incremental learning focuses on continuous data (or data stream), it has to
be able to deal with the problem of concept drift. For example, a proposed incremental
ensemble learning by (Günter and Bunke, 2002) constructs one classifier at a time from
distinct feature subsets or overlapped features between these subsets. New classifier is able
to learn a new concept from most recent data. Incremental learning addresses a principal
weakness of many traditional machine learning algorithms that are unable to learn from an
incoming data example without restarting the entire learning process. One way is to focus
on the ability of learning additional knowledge in an incremental manner rather than dealing
with limited memory resources. For example, the proposed method in (Yang et al., 2009)
maintains old data so that they are accessible while providing an ability to learn on the fly.
An efficient way in incremental learning tries to save space for some new incoming data
examples with a policy to remove the old data examples.

Decision trees were the first types of algorithms used to adapt to incremental approaches.
One example of a successful model is Very Fast Decision Tree (VFDT) (Domingos and
Hulten, 2000) which implements Hoeffding inequality bound to ensure sufficient data in
splitting nodes. Additionally, Incremental Bayesian models support dynamic updates for the
prior probability, when handling incoming data.

A non-tree single learner such as Online Naive Bayes classifier proposed by (John and
Langley (1995) uses updated counters to incrementally compute the probability of the class the new example belongs to. A recently proposed incremental Naive Bayes (in short iNB) algorithm (Ren et al., 2014) computes the posterior probability of 1) test examples in each class and 2) class conditional probability after an incoming data example appears on the bus. Next, it uses this data to adjust the degree of error between classification prediction and observation.

Stochastic Gradient Descent (SGD) (Bottou, 2010) is another example at a classifier that works in an incremental/online setting without making a distribution assumption. SGD updates the model’s weight one sample at a time by optimizing an objective function. Under a batch incremental setting, SGD has advantages of low computation expense and high performance.

SVMs have been also used for incremental learning. SVM classifiers rely on two concepts: 1) projecting inputs to high dimensional features and 2) assuming minimal risk. By maximizing margins for a separating hyperplane, SVMs use support vectors for classification. SVMs need to look at the whole training set to determine these support vectors. As a result, an incremental SVM (Laskov et al., 2006) preserves the current support vectors and adds them to the training set in the next step. However, these support vectors vary depending on the selected kernel function and they become less helpful when a concept drift occurs. Incremental SVMs also have difficulty discarding the old concepts since support vectors depend on the current data points.

In a different direction of study, (Ristin et al., 2014) use a Nearest Class Mean (NCM) classifier to construct their proposed incremental model, named Nearest Class Mean Forest, based on a Random Forest prototype. Their approach is to minimize the cost of updating
summarizes at each leaf node. Here the tree is allowed to incrementally grow using a splitting function. At one particular node, entire subtree can be rebuilt.

Online learning, which shares some characteristics with incremental learning, has also evolved in a similar fashion. Under the framework for online learning called M.O.A. (Massive Online Analysis), online Random Forests proposed by (Bifet et al., 2011) sample the incoming data model based on the Poisson distribution. Here, the growing tree and threshold for propagating samples in each direction of splits are random. The statistics of splits for all random tests are maintained at a node and propagated down to children nodes similarly to the Hoeffding Trees approach. Similarly, a model proposed by (Denil et al., 2013) grows a tree breadth first and keeps all potential children. In this thesis, our goal is to design a new incremental learning algorithm that can address the problem of high computational cost of SVM, but retain the comparable high performance. We focus on a novel sampling technique that identifies the weak performers in ensemble members. We also use a new classifier.

Incremental learning supports two approaches: an instance approach and a batch approach. The batch approach requires a full batch of examples to train so that it learns from the most recent data. Our work focuses on ensemble learning where the ability to integrate a new model or to eliminate an old model is an advantage of the design. Our approach shares the common approach with the Random Forests model, a popular ensemble method that uses a collection of trees instead of a single tree.

Given a Decision tree, the ability to induct a new data instance into it relies on how a new split can be processed. Maintaining all relevant examples at each node is the simplest solution, but costly in terms of memory. Without keeping this information, there is no
guaranteed way to make a new split optimal in response to an incoming data instance due to the recursive design. With an instance incremental approach, different strategies of splitting at a particular node may give significantly different performance results.

Many existing variant models of Random Forests distinguish themselves by the way the trees are grown. For example, Extremely Randomized Trees (Geurts et al., 2006) use entire data examples and decision boundaries are picked at random instead a best feature in Random Forests. Rotation Forests (Rodriguez et al., 2006) split randomly subsets of features and applied Principal Components Analysis (PCA). In another approach, Rotation Forests are built as an ensemble of these new features. Conversely, Gradient-Boosted Trees (GBTs) (Friedman, 2002) train one tree at a time, where each new tree helps correct errors made by previously trained trees. With each tree added, the model becomes more expressive.

Different variants of Random Forests have been implemented on different platforms, including Dato (formerly known as GraphLab) (Low et al., 2014), and Apache Spark (Zaharia et al., 2012). These platforms integrate the advantages of a distributed system that reduces the number of high input/output operations (Dean and Ghemawat, 2008).

In our work, we are attracted to the concept of Mondrian process based on which, a classification algorithm called Online Mondrian forest (Lakshminarayanan et al., 2014) has been developed. Mondrian forest uses a new way for sampling data that guarantees invariant distribution. Those with further interest are referred to (Roy and Teh, 2009). Inspired by this work, this thesis proposes a novel algorithm that overcomes the limitations of online Mondrian forest in more efficient ways. Our proposed work differs from Mondrian forest by the following: We focus on incremental learning. A chunk of data is made available at a time and can be used in batch. We use labels to guide the split of a node whereas Mondrian
forest picks feature to split a node independently with data’ labels. We implement stratified sampling of features to include informative features in each sub region instead of random sampling. Our hypothesis is that this will reduce the risk of constructing a weak model that affects the performance. Our proposed algorithm has two main contributions as follows.

- We propose an incremental learning approach that trains with higher accuracy than other state-of-the-art incremental methods.
- We demonstrate that our solution generates comparable results with offline models at each incremental size.

2 Overview Mondrian process and Mondrian tree

Mondrian process (Roy and Teh, 2009) was introduced as a new class of distributions that supports random partitions in a data space that are not limited to regular grids. Mondrian process uses recursive axis-aligned cuts to partition data space (see Figure 2.3) hierarchically
akin to a Decision tree. This feature gives Mondrian processes the ability to guarantee an invariant distribution to extend to infinite space.

In Figure 2.1, a Mondrian process starts when a new data point is introduced. Instead of introducing a new cutting plane over the data space, the Mondrian process provides the cut on a region determined by existing boundaries. As the new data point enters, a new boundary is redrawn (see Figure 2.2), a similar process is repeated. Noticeably, a cut point is randomly selected uniformly for all intervals in the recursion where the split occurs.

Figure 2.2: New split inside boundary Adapt: (Lakshminarayanan et al., 2014)

Figure 2.3: Random Forests vs Mondrian Forests. Adapt from (Lakshminarayanan et al., 2014)

The main difference between a traditional Decision tree and a Mondrian tree is centered around the splitting method, centered around the refer to Figure 2.3. Looking at Figure 2.3, we see that a cut in the Decision tree is illustrated by the line extending through the data space, while the cut in the Mondrian tree is computed from the range of training data at the time of split. In other words, the cut is restricted to the known region defined by the current data points so long as the new split is consistent with the current tree.
Similar to Random Forests, Mondrian Forests are able to deal with overfitting, even when applied to non-stationary distributions found in online learning. In the Mondrian process (Roy and Teh, 2009), a distribution is chosen by re-sampling and it guarantees a similar distribution as before. Here, we let $MT(\lambda, \mathcal{D})$ denote the distribution of a Mondrian process. Additionally, we can choose a probability distribution $MT'$ including the new example which will result in the new distribution being equal to the distribution $MT$ before including the new example. Originally, $\lambda$ denotes a budget (Roy and Teh, 2009) which is referred to as time in (Lakshminarayanan et al., 2014). In this thesis, we continue using the aforementioned notations so that we can compare our potential contributions. Further details on the splitting mechanism with a Mondrian process are discussed in the (Roy and Teh, 2009), because it guarantees similar distributions in incremental learning. Accordingly, this enables a Mondrian tree to achieve results similar to the trees created using batch learning.

3 Proposed approach

Our proposed model (named iRF) is an ensemble of tree-based classifiers, inspired by the work of (Lakshminarayanan et al., 2014). Each Decision tree is independently built from randomized sampling of examples similar to (Breiman, 2001), and updated or used for prediction. The final prediction for an unseen example is generated by averaging the combination of predictions from all trees. Intuitively, each tree is represented by a KD-tree structure (refer to Figure 2.4) where a node’s ranges are bounded by an axis aligned box. Similar to prepruning of random forests (Breiman, 2001), we use two methods to control
Figure 2.4: Decision tree and a forest of trees

the split: either i) a predefined minimum number of data points or, ii) a required minimum information gain threshold. Splitting dimension is determined with a Mondrian process (Roy and Teh, 2009).

To force phasing out of an old concept, we select a candidate tree with low prediction performance over a time period (equivalent to a half of the predefined number of trees). We also introduce a weight factor for each data point to determine which particular data point will be potentially chosen for removal. Each data point is represented by a tuple \(<\text{index, count}>\) where an \textit{index} denotes an instance’s order of arrival and a \textit{count} denotes the number of data examples arriving after it. Initially, we set the count to zero as the first data point arrives. Herein, it increases at each incoming instance or decreases after removing an instance. For the \(i^{th}\) data instance, its weight is defined as \(W_i = \text{index} / (1 + \text{count})\).

\[W_i = 1 / (1 + 0) = 1\] if it is the only example (i=1).

To incorporate a new arriving example \(d\), we obtain its \textit{count} and update the summary at each node (refer to line 2 in Algorithm 1) when \(d\) traverses to the node of the tree. Two conditions required for the split are the sufficient data points by Hoeffding bound and the depth threshold. In this case (line 6), this tree simply grows (line 7). Otherwise, the process of reconstructing a tree(line 13) using Algorithm 2. In Algorithm 2, the training data is obtained using samples from all current data instances (line 6). Building entire a tree is a
result of low performance during testing. If there is no testing, the oldest data instances are the candidates for removal. A new tree is built with half the examples from recently seen examples and the remaining data points are obtained by sampling the entire set of data points.

In the newly designed incremental learning algorithm, we apply the Mondrian process for sampling data. The idea is illustrated as the following. Utilizing the statistical summaries of internal nodes, we suggest a simpler scoring system. In that, a new data point goes down the tree until it can fall into the boundary of the specific label with its computed class probabilities. When this region has no label, the probability of the nearest ancestor can be used. In fact, we also have the option to use the depth tree’s threshold to control how deep the tree can grow by limiting the total number of splits it is allowed to make.

**Algorithm 1: Incremental Random Forests (iRF)**

- **Input**: A new incoming data instance $d$, a Decision tree
- **Output**: An updated Decision tree including $d_i$

1. Navigate instance $d$ to leaf node starting from the root of the tree
2. Update statistics on attribute values at current node including instance $d$
3. Evaluate gain function $G(A_i, S)$ on current node for all examples seen so far $S$
4. Compute Hoeffding bound $\epsilon = \sqrt{\frac{S^2\ln(1/\delta)}{2n}}$
5. **if** there exists $i$ such that for all $j$, $G(A_i, S) > G(A_j, S) + \epsilon$ **then**
   - **if** training size is less than its threshold and tree depth is less than its threshold **then**
     - split the node with attribute $A_i$
     - initial statistics for each child node
   - **else**
     - select feasible candidate of data example for removal by checking current error of performance of the tree with ADWIN windows
     - process with Algorithm 2 with feasible candidate by ADWIN report
6. **end**
7. **end**
The conditioning test at line 5 checks the minimum number of required data points compared to the approximate number of data points determined with a Hoeffding approach for confidence of splitting (De Rosa, 2016). This approximate strategy has worked with (Domingos and Hulten, 2000) to construct Hoeffding Trees (HT).

As we mention above, the ability to introduce a new parent node (Lakshminarayanan et al., 2014) gives the Mondrian process (Roy and Teh, 2009) an advantage in processing new data examples. This unique feature of the Mondrian process distinguishes it from other random forest methodologies. By maintaining an index table with an updated count of occurrence for each example, we have the ability to select a potential candidate for removal by its lowest weight to save space for incoming data. Furthermore, we adapt a variant of the sliding window approach ADWIN (ADaptive WINdowing) proposed by (Bifet et al., 2009) to detect changes in error performance of each Decision tree. As shown in Table 2.1, this method works more effectively with our design than the estimated average error of all observations as in On-line Random Forests (Saffari et al., 2009).

In addition, if the current boundary at a splitting node does not include the new data instance, it leads to extending the boundary to cover the arriving data instance or replacing it with a new parent and including a sibling to keep any data it does not cover. For detail of sampling with Mondrian process, refer to the paper (Lakshminarayanan et al., 2014).

Our implementation differs from Mondrian forests (Lakshminarayanan et al., 2014) in the following details.

- First, we do not keep any training data at each node, instead we maintain class frequencies and other summaries.
- Second, when a new data instance traverses down the tree, it initializes the update process for the count (the larger the value the older the concept) of each instance (data
Algorithm 2: Create a tree

**Input**: Current training data and error report
**Output**: An updated Decision tree

1. if error report is less than a random guess (as 50%) then
2.    Half from recent data from current node of the tree, and
3.    Randomly sample remaining half from all data instances
4. end
5. else
6.    Sample from all data instances
7. end
8. Build a new tree

point). The count is used to compute each instance’s weight factor, which is used to select the most likely candidate for dropping data instances. The other dropping case involves low performance of a particular Decision tree,

- Third, we also implement Hoeffding bound to determine when there are sufficient data example for splitting a node during increment learning and to use a sliding window to monitor each tree’s performance.

We evaluate our proposed method to perform sentiment analysis with three datasets: restaurant reviews from the Yelp reviews dataset, product reviews from Amazon dataset, and movie reviews from IMDB reviews dataset. For these text review datasets, we first explore the representation of word feature. These pre-experiments use four models: the Online Random Forests (ORF), our proposed model (iRF), Hoeffding Trees, and the incremental Naive Bayes (iNB). Each tree based model has 100 trees. These algorithms are selected because they are incremental learners, which usually are not good at text mining problem.

### 3.1 Feature generation in text data mining

For a review dataset, we generate features using statistical summaries and a bag of words method. A bag of words approach treats each word as a feature. However, many words are rarely used compared to others which results in many zeros in training data. This problem
causes a dimensionality issue in computational linguistics. Furthermore, a simple bag of words may not capture the correct meaning of the text. To address the shortcomings of a simple bag of words, we first explore a phrase-based sentiment analysis approach (Socher et al., 2013). However, our experimental results using this method where we represent each sentence in the review with an average word vector and train with Random Forests, are not better than the bag of words approach for feature generation.

Table 2.1: Comparing accuracy performance on Yelp dataset

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BOW</td>
</tr>
<tr>
<td>HT (Hoeffding Trees)</td>
<td>0.92</td>
</tr>
<tr>
<td>ORF (On-line Random Forest)</td>
<td>0.91</td>
</tr>
<tr>
<td>iRF (Our approach)</td>
<td>0.89</td>
</tr>
<tr>
<td>iNB (Incremental Naive Bayes)</td>
<td><strong>0.90</strong></td>
</tr>
</tbody>
</table>

Table 2.2: Comparing accuracy performance on IMDB dataset

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BOW</td>
</tr>
<tr>
<td>HT (Hoeffding Trees)</td>
<td>0.80</td>
</tr>
<tr>
<td>ORF (On-line Random Forest)</td>
<td>0.82</td>
</tr>
<tr>
<td>iRF (Our approach)</td>
<td>0.84</td>
</tr>
<tr>
<td>iNB (Incremental Naive Bayes)</td>
<td>0.68</td>
</tr>
</tbody>
</table>

Table 2.3: Comparing accuracy performance on Amazon dataset

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BOW</td>
</tr>
<tr>
<td>HT (Hoeffding Trees)</td>
<td>0.78</td>
</tr>
<tr>
<td>ORF (On-line Random Forest)</td>
<td>0.80</td>
</tr>
<tr>
<td>iRF (Our approach)</td>
<td>0.83</td>
</tr>
<tr>
<td>iNB (Incremental Naive Bayes)</td>
<td>0.71</td>
</tr>
</tbody>
</table>

As a result, we explore two advanced techniques, i.e., Word2Vec (Mikolov et al., 2013) and FOFE (Fixed-size Ordinally Forgetting Encoding) (Zhang et al., 2015a) for representing
sentences. These two techniques allow us to transform a review text into a uniform format upon which hybrid features are built. These new features are constructed with the following combinations of settings: i) Bag of Words on FOFE (simply as FOFE), and ii) Bag of Words with Word2Vec. We compare the accuracy in order to select the method to generate our features beside statistical summary features. We refer to the two settings as FOFE (instead FOFE + BOW) and Word2vec (Word2Vec + BOW).

Reviews are split into sentences to generate vectors. The FOFE method is applied without using pretrained feature vectors. We do not use the Google pretrained set for a fair comparison. Table 2.1, Table 2.2 and Table 2.3 show the accuracies of selected algorithms on Yelp, IMDB and Amazon review datasets, respectively.

We observe that the Word2vec method provides consistent results, consisting of more than half of the best results. Word2Vec produces better results than FOFE does with the corresponding algorithms. Based on these results, we prefer to use the Word2vec method for the representation of text datasets on our next evaluation task. Figure 2.5 illustrates the AUCs curves in this experiment.

3.2 Challenges in review datasets

Additionally, we use the following additional features.
• Popularity: A review may have a number of up votes (useful votes) and a number of down votes (funny votes). Popularity is defined as the sum of up votes and down votes.

• Confidence-adjusted time: A ratio of the number of reviews posted over time elapsed since the reviewer registered (by month unit).

4 Experimental setup

To evaluate the proposed algorithm, we experiment with state-of-the-art tree-based algorithms. These include Online Random Forests (ORF-Safari) (Saffari et al., 2009), Extremely Randomized Trees (ERT-1, ERT-k) (Geurts et al., 2006), and Random Forests (Breiman RF*) (Breiman, 2001). Our evaluation includes other incremental algorithms, in particular incremental Naive Bayes (Ren et al., 2014), Stochastic Gradient Descent (Bottou, 2010). Stochastic Gradient Descent (SGD) is selected because it has worked well for a sparse data and it has an online learning ability.

We report the result of the Mondrian Online Random Forest model only on numeric datasets because it is unable to work on sparse data which occurs when many words rarely occur in a majority of reviews. Online Mondrian Random Forest also have difficulty with numeric data (e.g., Mnist and Poker datasets). Non-incremental learners are retrained for each incremental size, introducing bias comparison with incremental learners because they allows retraining on entire current available from scratch. To set up a fair comparison, we do not remove the current training example while allowing incoming data to be read in sequence by incremental learning methods. Training sizes are from 10% to 90% with 10% incremental step.
We use three review datasets Yelp, IMDB and Amazon review datasets, as well as three benchmack datasets Mnist, Poker and cover type datasets. In addition, our final feature set for review datasets includes statistical features and Word2vec based features. The features generated include the following:

- numCapital: number of words with all capital letters,
- avgSentLen: average sentence length,
- avgWordLen: average number of words in sentence,
- avgChacLen: average number of characters in word, and
- numUsefull: the number of useful votes.

Table 2.4: Accuracy performance comparison of tree-based models

<table>
<thead>
<tr>
<th>clf</th>
<th>iRF</th>
<th>HT</th>
<th>ERT-k</th>
<th>ERT-1</th>
<th>ORF</th>
<th>iNB</th>
<th>SGD</th>
<th>MF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>0.91</td>
<td>0.82</td>
<td>0.86</td>
<td>0.76</td>
<td>0.85</td>
<td>0.76</td>
<td>0.92</td>
<td>NA</td>
</tr>
<tr>
<td>Yelp</td>
<td>0.95</td>
<td>0.96</td>
<td>0.92</td>
<td>0.89</td>
<td>0.92</td>
<td>0.91</td>
<td>0.96</td>
<td>NA</td>
</tr>
<tr>
<td>IMDB</td>
<td>0.87</td>
<td>0.86</td>
<td>0.67</td>
<td>0.83</td>
<td>0.83</td>
<td>0.73</td>
<td>0.86</td>
<td>NA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>clf</th>
<th>iRF</th>
<th>HT</th>
<th>ERT-k</th>
<th>ERT-1</th>
<th>ORF</th>
<th>iNB</th>
<th>SGD</th>
<th>MF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mnist</td>
<td>0.91</td>
<td>0.86</td>
<td>0.82</td>
<td>0.81</td>
<td>0.86</td>
<td>0.77</td>
<td>0.87</td>
<td>0.86</td>
</tr>
<tr>
<td>Poker</td>
<td>0.93</td>
<td>0.91</td>
<td>0.88</td>
<td>0.84</td>
<td>0.85</td>
<td>0.67</td>
<td>0.88</td>
<td>0.90</td>
</tr>
<tr>
<td>coverType</td>
<td>0.94</td>
<td>0.91</td>
<td>0.88</td>
<td>0.53</td>
<td>0.85</td>
<td>0.62</td>
<td>0.92</td>
<td>0.89</td>
</tr>
</tbody>
</table>

Note: iRF: our model, ERT (Extra Tree), SGD: Stochastic Gradient Descent, ORF: Online Random Forests, iNB: increment Nave Bayes, MF: Mondrian Forests

We conduct experiments that focus on ensembles of tree-based models (refer to Table 2.4) but also include experiments with other incremental learning models, such as incremental Naive Bayes and Stochastic Gradient Descent models.
5 Results and discussion

In the first set of experiments in this thesis, we compare our proposed model with other ensembles of tree-based models such as Hoeffding Trees (HT), Extra Trees (ERT-k, ERT-1) and Online Random Forests (ORF). Table 2.4 indicates that our proposed model outperforms significantly all tree-based models as well as incremental Naive Bayes (iNB). To verify this claim, we next compute Anova one way test at 95% confident level.

- Null hypothesis $H_0$: All classifiers are similar
- Alternative hypothesis $H_a$: There exist differences

The $F = 3.91732939$ and $p$-value $= 0.01327529$ indicate that we can reject the null hypothesis. We further report the result of the paired t-test in Table 2.5 to conclude that our model is better than tree-based incremental learning models. Figure 2.6 illustrates the incremental learning models used in the experiments. Cover Type dataset seems to be the most challenging data when half of the algorithms produce very low performance at initial 10% training size while others generate around 50% to 60%, similar to random guess at the same proportion of training size. The Online Random Forest model yields strong the
Table 2.5: Results of the paired t-tests

<table>
<thead>
<tr>
<th>Classifiers</th>
<th>T value</th>
</tr>
</thead>
<tbody>
<tr>
<td>iRandomForests vs Hoeffding Trees</td>
<td>0.021087</td>
</tr>
<tr>
<td>iRandomForests vs ERT-k</td>
<td>0.0380761</td>
</tr>
<tr>
<td>iRandomForests vs ERT-1</td>
<td>0.0022568</td>
</tr>
<tr>
<td>iRandomForests vs Online Random Forests</td>
<td>0.00481123</td>
</tr>
<tr>
<td>iRandomForests vs incremental NaiveBayes</td>
<td>0.00158587</td>
</tr>
<tr>
<td>iRandomForests vs Mondrian Forest</td>
<td>0.0343152</td>
</tr>
</tbody>
</table>

predictions but is quickly outperformed by remaining classifiers except ETR-1 model. In fact, it makes the least significant improvement in all data in experiments.

For non-tree base classifiers, incremental Naive Bayes illustrate just better than average performances in both numeric and text data. The significant improvement is obtained by Naive Bayes with Yelp dataset. Stochastic Gradient Descent (SGD) outperforms our model with Amazon and Yelp reviews. However, our model shows consistent performance in the experiments. In particular, it performs more consistently than other ensembles of trees. Hoeffding Trees and Online Random Forests are comparable, but increasing the number of trees in Online Random Forest improves significantly its accuracy. Next, we present our test to evaluate the results.

Validating comparing results with statistical tests

We use two-way ANOVA multiple repeated test to validate our comparison results. Two way ANOVA is used for the following two factors: algorithms and different training sizes. Algorithms includes Hoeffding Trees, Extra Trees (ERT-k with k=100), Extra Trees with single tree member, Online Random Forests, incremental Naive Bayes, Online Mondrian Forest and our model (incremental Random Forests) to test of our hypothesis. Our hypotheses are as:

- $H_{01}$: All classifiers are the same, i.e., all classifiers perform similarly.
• $H_{02}$: Different sizes do not have impact on the performance

The result of ANOVA test is $F$-statistic $= 0.00362303$ for the first hypothesis $H_{01}$ and $1e-4$ for the second hypothesis $H_{02}$ with 95% confidence level. As a result, we reject the null hypotheses. Furthermore, we compute a pairwise t-test, whose results we report in Table 2.5. From these results, we conclude that our proposed model is significantly better than the other ensembles of tree-base models.

### 6 Conclusion and contribution

We have proposed an incremental learning algorithm and demonstrated its performance in an incremental settings in the context of numeric data. We conclude that increasing the number of trees can indeed improve the performance of the model. We observe that a low number of the trees (e.g., $n=50$) can severely affect performance of our algorithm. To improve the performance in text data, hashing methods such as murmurHash (Couceiro et al., 2009) is a worthy candidate for the future work.

When data consist of many low informative features, our model’s performance deteriorates requiring feature selection and feature generation. However, the concepts of strong informative and low informative features might need to be investigated more because the correlation among features need to addressed. Furthermore, we want to investigate features dropping which has been successfully applied in neutral networks (Hinton et al., 2012), allowing a reduction in the training time. We believe that when the number of features is reduced, fewer trees will be required to maintain high performance.

The contributions of this work can thus be summarized as follows.
1. Our method outperforms other current ensembles of tree-based models.

2. Our approach allows optimal tree learning on the fly due to the ability of the Mondrian process.

3. Our approach produces nearly identical accuracy with trees built by conventional batch learners given enough examples.

4. Our approach has been implemented optimal execution of algorithms on multi-core system.

In conclusion, this model introduced in thesis chapter can handle a sparse datasets, often found in data mining tasks in the area of natural language processing.
A problem that deals with a lot of data may encounter an unseen class issue when training examples for some classes were not available are not available at the training time. Including unknown classes in prediction is obviously not an easy task, as incomplete knowledge of unknown objects during training hampers classification with the traditional closed set assumption. Unknown classes can be in minority, but may be majority as well. As a result, unbalanced classes in the open set cannot be handled with common techniques such as stratified sampling.

The open set problem was first introduced in computational linguistics for authorship identification (Diederich et al., 2003), which is often also called to authorship attribution. In this problem, a list of authors and their sample styles, which are presented in form of text, articles are known. An unknown text is needed to determine an actual author. It may be the case that the real author is not among them. Schaalje et al. (Schaalje et al., 2011), (Schaalje and Fields, 2012) use the term open set to introduce the concept of openness. A statistical analysis of literary style is the main technique known as stylometry, to determine who is the
Researchers in computer vision domain (Scheirer et al., 2013a), (Bendale and Boult, 2015b) presented their intensively mathematical work that lay out the breakthrough framework to deal with the open set problem in the field of computer vision. By exposing the weaknesses of current data mining techniques which may result in an unreliable solution in a real world application, they establish a new framework for data mining algorithm development. (Scheirer et al., 2013a) quantify the openness of the data space as follows:

\[
openness = 1 - \sqrt{\frac{2 \times trainingclasses}{testingclasses + targetclasses}}.
\]  

(III.1)

where trainingclasses denotes the number of classes in training, testingclasses denote the number of classes in testing, and targetclasses denotes the number of target classes.

In open set recognition, the risk of the unknown can be estimated by an empirical risk function on the training data without sufficient knowledge. Its objective is to minimize what
is defined as the total recognition error (Scheirer et al., 2013a). Studies on open set are found beyond computer vision (for object detection) in computational linguistics (authorship attribution) and in biometric verification (Rattani et al., 2015b). Initial work on the open set problem has focused on an off-line setting where data is considered statically. Our work on the open set problem has been inspired by the published research in an excellent paper (Bendale and Boult, 2015b). These authors give a new life to the use of data mining on evolving data where the closed set assumption has less hold. Like anomaly detection, open world recognition can detect new objects or unknown classes or simply rare classes, e.g., in fraud detection (He et al., 2008). The difference is that open world recognition allows it to update the classifier with these new classes in its multi-class classification system.

Incremental or online learning poses challenges because of finite system resources and unavailability of training data in one place. In such a case, incremental ensemble learning is a natural choice due to its ability to process training data in chunks, and the ability to update the learned models. The logic behind incremental ensemble learning for the open set problem is based on the wisdom of the crowd, which uses the combination of knowledge from many experts to help identify the common uncertainty among them.

Classification is often referred to as the task of discriminating one class from others in a given set of classes. Traditionally, classifiers work well assuming that a prior knowledge of all classes is given. This scenario is defined as a closed set in the context of classifying objects from prior knowledge about them. Unfortunately, a presenting of an example from an unknown class during testing can lead to poor performance of even state-of-the-art classifiers due to observed classes being incorrectly identified as one of the known classes. This problem has been reported by (Jain et al., 2014b).
It is normal to see unknown objects during testing in traditional batch learning, in which all classes are presented at training time. It is because we often have insufficient knowledge of the evolving real world. Incremental learning, particularly online learning, complicates this problem when continuous data, known as streaming data may introduce unknown objects at any time. In fact, a large dataset is more likely to include unknown classes because training data is not available all at once.

Without prior knowledge, unknown objects cannot be assumed as belonging to a negative class under 1-vs-rest approach at training time (Wilber et al., 2013). Without a label even as simple as unknown class, a classifier suffers uncertain decisions and poor performance (see Figure 3.1). This problem is ill-defined with unsupervised and semi-supervised learning for two reasons. Unsupervised learning does not take labels into account and often works well to identify the existing groups based on similarity. This is known as clustering. Semi-supervised learning considers both labeled and unlabeled examples and propagates the knowledge gained to improve classification performance.

1 Problem definition

The generation and collection of a large volume of data has become the norm in domains such as computer vision and computational linguistics. A model is built from a set of training data and is evaluated with a test set, with or without a validating set. What if some classes are not observed in training but are presented at testing? It may be a result of bad samples, under-sampling or non-available samples at the time of sampling data for training, e.g., in streaming data. In general, a pattern recognition system, particularly classifiers,
learn to identify a class by discriminating it from other classes. With an unobserved class at training, the performance of such systems is in question because they will classify unknown classes to other known classes.

In general, deteriorated performance of algorithms caused by bad samples or inadequate samples can be dealt with advanced sampling techniques. However, the best sampling techniques cannot tackle the problem of the unknown class, which is present only at testing time due to the dynamic nature of data. This problem is a common weakness of prior knowledge assumption in many data mining algorithms. Intuitively, an unknown class can be treated as an outlier and using a threshold to identify an unknown class is a natural method. However, the question is how a threshold is defined for an unknown.

In many knowledge domains, the acquired knowledge of the data is often far from complete, given the dynamic real world. Implicitly, a multi-class classification model is only able to distinguish one class from the rest, assuming a full knowledge of classes a priori, referred to as a closed set of classes. This assumption may not be guaranteed and cannot be assumed in many real world scenarios, as an unseen class may only be available at the time of prediction and may be impossible to learn using traditional techniques. Some examples are:

a) a self-driving car cannot learn all classes of objects, a priori; b) an authorship attribution system may not have any knowledge of most authors while investigating true authorship of an anonymous text; and c) a biological system may lack knowledge of rare existing species when investigating an underwater living organism. Human experts may have to thoroughly check all known species before making any conclusion about the outlier.

Given a level of uncertainty due to the presence of unknown classes, classification is more subjective due to the openness of the surrounding real world. As a result, a recognition
system deployed in the real world, given the static world assumption, produces unreliable results, known as risk. This motivates a new direction of research seen in recent studies trying to address the limitation of correctness in algorithms’ assumptions based on a closed set conception. Studies on open set recognition such as forensic authorship attribution (Schaalje and Fields, 2012), image processing (Costa et al., 2012), fake fingerprint detection (Rattani et al., 2015a), object detection, (De Rosa et al., 2016), authorship attribution in social media (Rocha et al., 2017) and an open world recognition such as (Bendale and Boult, 2015b), (De Rosa et al., 2016) illustrate a growing interest over a short time.

The remainder of this chapter is organized as follows: Section III.2 discusses related work. Section III.3 illustrates our proposed model. Section III.4 describes the set up of our experiment to compare our algorithm with other state-of-the-art solutions. We present results and discussion in Section III.5. Finally, Section III.6 summarizes up the this chapter with suggestion of future work and our contribution.

2 Related work

A recognition system often deals with an unknown scenario, referred to as openness, when identifying objects of interest. Object detection is a typical scenario where an open set recognition is naturally described. The object of interest, e.g. a human face may be among others, e.g. cars, trees, electric poles where the recognition of one class, referred to as the positive class, is discriminated from others using specific features. This system supports a rejection mechanism. This type of mechanism is widely used for face detection (Jin et al., 2004), document classification (Manevitz and Yousef, 2001), and speech processing (Shen
and Yang, 2007).

Intuitively, the 1-vs-1 classifier (Rocha and Goldenstein, 2009) has a goal similar to open set recognition to distinguish one class from others by training in one class as positive, against all other classes as negative. In the same vein, the 1-vs-all classifier treats examples of non-positive classes as negative. Adapting the idea from the work of (Schölkopf et al., 2001), (Scheirer et al., 2013a) proposed the 1-vs-Set machine and further presented what is called calibrated W-SVM for non-linear classification (Scheirer et al., 2014b).

A data mining application, e.g., forensic authorship attribution (Stolerman et al., 2014), may want a system with an ability to discriminate unknown authors from known authors, or classes when an unknown author referred to as a class, may not be in a set of known authors, referred to as known classes. The openness of this system is due to an incomplete knowledge of the real world. In fact, earlier studies of open set recognition have been conducted on authorship attribution by (Schaalje et al., 2011), (Schaalje and Fields, 2012), a branch of the natural language processing for determining the authorship of unknown documents, referred to as stylometric authorship attribution.

(Fei and Liu, 2016) in their work on text classification address the open set problem in computational linguistics. They propose a variant version of the Rocchio algorithm (Rocchio, 1971) in information retrieval where each class is represented by a ball-shaped boundary. They name their approach as center-based similarity space learning, or CBS learning. The idea of their approach is to transform the original data space of documents into a space representation of centroid classes. This transformation generates a new feature space, one for unigram and one for bi-gram, for each text document. A document, in close proximity to one class, is assigned to this class, and a new center is computed. A drawback
of this approach is that the center of each class may or may not be an actual data point that represents a document. As a result, a close distance from a document of interest to the center of a class to which it should actually belong may be longer than distance to the class it may be assigned to.

Experience based learning without sufficient theoretical support is the downside of these studies. The open set concept presented in (Scheirer et al., 2013b) is the first one that theoretically addresses the closed set assumption in an object recognition system. At first, a single class model (e.g., one class SVM) emerges as a feasible solution for single class detection (implicitly rejecting others). However, one class SVM relies on a closed set assumption because it does not take into account the presence of a negative class.

### 2.1 Distance metric

When an unknown class is introduced, a model should be able to detect it and reject it. This ability allows the system to avoid misclassifying the unknown to a known class. A common way is to measure a distance from an unknown object to a known object. However, a dataset may have features which use different measurement units. For example, consider a subset of IMDB movie dataset with a gross revenue feature (unit in hundred million dollars) and a number of review features as a non-negative integer. The gross feature measures the revenue before tax of a particular movie while the number of reviews indicates the count of reviews for this movie. Here are our rules to detect an outlier by thresholding:

- the gross revenue is greater 187 or less than 150 (unit hundred million dollars), and
- the number of reviews is less than 40 or greater than 75.
In Figure 3.2, the left plot illustrates how outliers can be obtained by using thresholds. However, there are three more outliers left (bold red circles). The reason these red circles are considered outlier is that they are not part of the remaining data points. We see that a movie with gross revenue of $176 hundred million with only 42 reviews and even another with a gross revenue of $185 hundred millions with only 45 reviews are examples of abnormal data examples. Ignoring these outliers will affect the accuracy in both classification and regression tasks. However, thresholds with different units may have difficulty detecting these outliers. It is a downside of using simple thresholds which do not taking into account the fact that the variance in each direction is different. This problem poses a big challenge when we measure distance from a data example to the center of a class to determine whether or not this data point belongs to a class using a distance based approach.

**Mahalanobis distance**: (Mahalanobis, 1936) proposed a measure of distance between two points, considering a distribution of data in a manner that can deal with the covariance among the variables. The Mahalanobis distance between point $x$ and point $y$ is defined as follows:

$$d(x, y) = \sqrt{(x - y)^T C^{-1} (x - y)}$$  \hspace{1cm} (III.2)
where $C$ denotes the covariance matrix among the feature variables.

One interesting characteristic of the Mahalanobis metric is that it corresponds to Euclidean distance in the transformed space by rescaling the axes into unit variance. Mahalanobis distance is unitless and scale invariant. We use Mahalanobis distance for our distanced base method in ensemble learning with unknown classes as well as in combining classifiers.

### 2.2 Open space risk challenge

In their first work on open set recognition, (Scheirer et al., 2013b) introduce their 1-vs-Set model to counter the closed set assumption. The main idea is that there is a risk in class assignment in open space. This is known as open space risk. This risk can be formulated in the Compact Abating Probability Model (CAP) (Scheirer et al., 2013b) for a class assignment which describes an inverse relationship between the probability and the distance of data from a specific class.

Concretely, the probability of class membership is decreasing as the points of interest move toward an open space away from known data. A CAP model can be illustrated by the following abating property s.t.

$$\min_{x_i \in \mathbb{R}} ||x - x_i|| > \tau \Rightarrow M_\tau(x) = 0$$  \hspace{1cm} (III.3)

where $M_\tau$ represents CAP for a given finite distance threshold $\tau$ and $\forall x \in X$.

In the CAP model, any point beyond a given threshold distance $\tau$ from the closest training data has zero probability. However, computing the confidence level of class membership has not been determined.
2.3 Nearest Class Mean and its limitations

To deal with the problem of open space risk, many researchers currently favor distance-based approaches where each class is represented by a closed sphere with its center being computed as the class mean. Figure 3.3 illustrates a data space with two classes where each class may be represented by more than a ball-shaped class boundary. This closed boundary model (referred to as Nearest Class Mean) addresses the problem of open space risk, is implemented as NCM forest by (Ristin et al., 2014). (Bendale and Boult, 2015a) adapt this method and propose their NNO model, which has been further modified by (De Rosa et al., 2016) in their online NNO model.

Intuitively, Bayes theorem (refer to Equation B.1) is a common way to compute the posterior probability of an unknown class given that other classes are taken into account. Prior probability of event describes the probability of the event before any collection of new data. When we known more about a hypothesis H, the adjusted probability is the posterior probability. Since the evidence does not depend on H, Bayes’ theorem can be written as

\[ p(H|D) \propto p(D|H)p(H). \] (III.4)

Inability to determine the probability of an unknown class means undetermining its
posterior probability. One way is to estimate the approximate lower bound for an unknown classes is proposed by (Schaalje and Fields, 2012). This solution relies on experimental thresholds and may be domain specific in practice. If a model generates only scores for class assignment (e.g., SVM), mapping raw decision scores to probability scores is a way to determine the level of uncertainty of an existing unknown class w.r.t. open space risk. These calibrated scores can be estimated by several methods, e.g., (Platt et al., 1999), (Huang et al., 2006), (Bravo et al., 2008), (McCann and Lowe, 2012). Despite the fact that the proposed estimating method in Platt’s work (Platt et al., 1999) can be used to calibrate scores for general non-linear models in addition to SVM, the gap between the threshold selection heuristic and the Gaussian distribution’s theoretical assumption for the support of unknown class has been reported as a critical issue in (Scheirer et al., 2014a).

(Scheirer et al., 2014a) argue that EVT (Extreme Value Theory) (Lindgren et al., 1987) has the theoretical underpinning to address the occurrence of rare events, i.e., the appearance of unknown classes. Because the appearance of unknown classes is rare, they recommend using the Weibull distribution, which is independent from the model regardless of the overall distribution. In fact, sampling the extrema (as the outlier for the unknown class) in the right tail is always guaranteed by the Weibull distribution (Scheirer et al., 2011). These authors introduce the scoring function approach in their Weibull calibrated SVM or W-SVM (a variant of binary SVM) for a linear boundary where the optimization problem of the SVM transforms into an optimization of the risk within a space between two parallel hyperplanes. For a multi-class problem, (Jain et al., 2014a) proposed $P_I$-SVM (Probability of Induction SVM). Their argument indicates that unnormalized posterior probability is sufficient to represent the unknown class probability.
These proposed methods implement a simple approach: assign an “unknown” label if the probability of the unknown class is below threshold $\delta$:

$$\hat{y} = \begin{cases} 
\arg\max_{y_i \in C} P(y_i|x), & \text{if} \ P(\hat{y}|x) \geq \delta \\
"unknown" & \text{otherwise} 
\end{cases}$$  \hspace{1cm} (III.5)

where $C$ denotes known classes, $y$ denotes ground truth, $y_i$ being one of the known classes of $C$, and $\hat{y}$ denotes predicted label w.r.t $x$ as a data point. Threshold has been used as a rejection option in several studies (Platt et al., 1999) (Bartlett and Wegkamp, 2008). Furthermore, the heuristic of selecting a decision threshold is often natural given prior knowledge of unseen classes in testing (or with closed set assumption).

Another related work is the Center Based Similarity (CBS) approach proposed by (Fei and Liu, 2016). The data space in the original problem is transformed into a representation of class means, called a similarity space. Their model, named cbsSVM, is applied to this new space. By limiting the boundary to a ball, they avoid the problem of open space risk in a traditional SVM model, but still support the detection of unknown classes. Their SVM uses the Platt calibrated score method assuming that the Gaussian distribution for feature normalization and center-based positive classes hold true. However, this assumption may not always hold the case in practice (Scheirer et al., 2011). One common weakness of models in (Scheirer et al., 2013b),(Jain et al., 2014a) and (Fei and Liu, 2016) is a static setting design where unseen classes in training may be a problem due to ill-sampling.

Another direction of research is the incremental learning approach where, because the data is continuous, there is no control over the appearance of unknown classes over
time. Incremental learning does not require the system to retrain from the beginning, which is important from a practical point of view. While the open set problem has not been investigated much in natural language processing, we acknowledge that researchers (Scheirer et al., 2013b), (Scheirer et al., 2014a), (Jain et al., 2014a) in the field of computer vision have been laying out solid foundations to address the limitation of the closed set assumption.

Even though an open world concept was introduced by (Stolerman et al., 2013) in stylometric authorship attribution, (Bendale and Boult, 2015b) are the first contributors in incremental learning in his redefined open world recognition approach to the best of our knowledge. Their NNO (Nearest Neighbor Outlier) model extends the NCM model (Nearest Class Mean) (Ristin et al., 2014) to illustrate open world recognition. The main idea is to treat an outlier as an unknown class $y$ using a confidence score, which is computed as follows:

\[
S_y(x, \tau) = Z_\tau(1 - \frac{1}{\tau}d_W(x, \mu_y))
\]  
(III.6)

where $\tau$ denotes a threshold to determine a ball for each class mean, $Z_\tau$ denotes a normalization factor using gamma distribution, computed as $Z_\tau = (\Gamma(\frac{m}{m-1} + 1))/(\pi^{\frac{m}{2}}\tau^m)$.

The NCM model and its variants have limitations similar to those of the class mean approach which is known for linear classifiers. The NNO model also needs to obtain an offline a metric $W$ during training on an initial set of known classes. To tackle this problem, (De Rosa et al., 2016) use the Hoeffding bound method to incrementally estimate a confidence threshold for an unknown class. Hoeffding bound (Hoeffding, 1963) is used in incremental learning (Bifet et al., 2009) to estimate good attributes for streaming data when
current data is not sufficient to determine the best split.

As we apply this incremental updated threshold in (De Rosa et al., 2016), we briefly review the formulation of Hoeffding inequality. Let $\tau$ denote threshold to assign an instance as belonging to an unknown class. The estimated $\bar{\tau}^t$ at the time $t$ is defined by

$$\bar{\tau}^t = \tau^t + \sqrt{\frac{(b - a)^2 \log(1/\delta)}{2t}} \leq \tau^t + \sqrt{\frac{1}{2t} \log\left(\frac{1}{\delta}\right)}$$

(III.7)

given that $a, b \in [0,1]$ and $\delta$ denotes the desired confidence level s.t. $\delta = \frac{1}{t^C}$ where $C$ is the number of current classes and time $t$. The more training data we have, the closer this threshold becomes to $\tau^t$. The estimation of confidence prediction follows in (De Rosa et al., 2016).

However, their proposed online approach (refer to lines 11 and 12 of the algorithm in (De Rosa et al., 2016)) also suffers from a drawback when an updated class ball is created for a class to cover incoming data item overlaps the class ball of a different existing class. In this case, a new class ball may have many incorrect data points because it contains a majority of data points of other classes. Our proposed distance based model uses multiple centroids and local learning similar to online NNO (De Rosa et al., 2016), but differs with two main contributions. First, we address the initial issue of incrementally adding new classes in our solution. Second, we optimize the nearest neighbor search for determining the nearest local balls.
3 Intuition and proposed approach

We are interested in developing an incremental learning approach, which is more suitable for open set recognition especially using the ensemble approach, to facilitate classification when the nature of data collection is continuous. Incremental learning can work well in off-line learning and online learning when new classes may be introduced during a trained model's use. Our first choice is not an incremental SVM model due to the high computational cost for the update and the performance depending on the number of support vectors retained (Laskov et al., 2006).

Our preliminary work focuses on an ensemble of a tree-based model. An issue with splitting data space with axis-align boundaries is highly incorrect predictions when a tree built has a small number of splits, and the overfit problem that prevents the generalization of a solution. A new class can exist at an arbitrary location in the data space. This introduces a big challenge for an ensemble of tree-based models due to the high cost of re-doing any splitting at parent nodes. While Mondrian process supports this ability, it comes at a cost as the data sampling method does not take labels into account. As a result, extending our previous work on incremental learning model is not a feasible solution. Using rotation method to change decision boundaries has a limitation since high computational costs result when dimensions are high.

Using an ensemble of tree-based models, such as Random Forests has been criticized due to its averaging prediction strategy. One feasible solution is to use the closed shape boundary method where each class is determined by a closed boundary. The Rocchio algorithm in Information Retrieval uses this concept, with a class being determined by a round shaped
boundary. The Rocchio algorithm has its own problem since this vector space method may fail to classify classes if two data points belong to two different classes but refer to the same object.

The Nearest Class Mean model addresses this problem by changing its mean to reflect the new examples. It uses a threshold to detect unknown classes. A class may change its boundary by shifting the class mean of each ball to its new class mean to counter each mistaken classification. This strategy has been used efficiently in (Schaalje et al., 2011) for open set authorship attribution. However, we do not use this technique due to the reasons described later in this chapter.

Remember that the goal in open set recognition for the general classification problem is to counterbalance an open space risk with an empirical risk. Our model design uses a set of closest neighbors represented in terms of centroid class. Here, a centroid or a core point $r$ of a ball-shape boundary presents an actual data point instead of a mean of data points which may not represent real data point in ball-shaped boundary. A class is a set of ball clusters where each ball has a minimum number of member points. Unlike class mean, a ball center is a data point instead of a class mean which may not represent an actual data point. We allow reduction of a ball’s radius to take into account errors in label assignment for each ball. Since only the nearest ball’s radius is adjusted, we refer to our model as the Nearest Centroid Class model (NCC).

This design minimizes open space risk since it guarantees CAP property III.3. Let $d_x$ denote the distance from $x$ to its nearest centroid neighbor. For $\forall d_x > \tau$, the probability $p_x$ that $x$ is a member of a particular class is.
(Bendale and Boult, 2015b) define the open space as

\[ \mathcal{O} = S_o - \bigcup_{i \in N} B_r(x_i). \]  

(III.9)

where \( B_r(x_i) \) denotes a closed ball with a radius \( r \) around arbitrary known positive training point \( x_i \in \mathcal{K} \) where \( \mathcal{K} \) denote a number of balls representing for a class, \( i = 1..N \). Open space risk \( R_O(f) \) for a class is defined (Scheirer et al., 2013a) by

\[ R_O(f) = \frac{\int_{\mathcal{O}} f_y(x)dx}{\int_{S_o} f_y(x)dx}. \]  

(III.10)

A membership function to determine if an example belongs to a class is defined in (Bendale and Boult, 2015b), (De Rosa et al., 2016) as

\[ p(y|x) = \frac{\exp(-\frac{1}{2}d_W(x, \mu_y))}{\sum_{y' \in Y} \exp(-\frac{1}{2}d_W(x, \mu_{y'}))} \]  

(III.11)

using a low rank Mahalanobis distance \( d_W(x, \mu) = \sqrt{(x - \mu)^T W^T W (x - \mu)} \) where \( x \) and \( \mu \) are d-dimensional vectors, \( W \in \mathbb{R}^{m \times d} \).

### 3.1 Incremental learning metric

As it is a case of a cold start problem, thresholds for unknown class detection can be established by a pre-training stage (Bendale and Boult, 2015b). A way to update the threshold for unknown class detection is suggested by (De Rosa et al., 2016). We use their proposed update method for better performance. In fact, our model is on par with the extension of NNO by (De Rosa et al., 2016) in which we use a set of centroids instead of
a class mean for representation, but differ on how each centroid is selected. The posterior probability of class $y$ is defined as

$$p(y|x) = \frac{\exp(-\frac{1}{2}d_W(x, \mu_y))}{\sum_{y' \in Y}(-\frac{1}{2}\exp(d_W(x, \mu_{y'})))}$$  \hspace{1cm} (III.12)$$

such that the estimation of prediction confidence for nearest ball $b^*$ is defined by (De Rosa et al., 2016) as

$$p(m_{cj}|x) = \frac{1}{Z}\exp(-\frac{1}{2}d_W(x, m_{cj}))$$  \hspace{1cm} (III.13)$$

where $p(m_{cj}|x)$ denotes the posterior probability of a centroid $m_{cj}$ and $Z = \sum_c \sum_j \exp(-\frac{1}{2}d_W(x, m_{cj}))$ denotes a normalizer.

$$C_y(x_t, b^*) = p(y|x) \times \exp(-\frac{1}{2r_{b^*}}d_W(x_t, c_{b^*}))$$  \hspace{1cm} (III.14)$$

### 3.2 Determine nearest class boundary

Whenever a new incoming data point arrives, its distance to the center of the nearest class is measured. Computing the distances from all current data points to the data point of interest
Figure 3.5: Example of initial values for a new ball in Online NNO

is not practical. In fact, we only need to determine the nearest classes to this data point. Our approach is to use only special data points that are centers of each class. This number is much smaller than the total number of current data points. Then, we can measure the distance to the centroid of the nearest class from the data example to predict a new class boundary or adjust the extent boundary of class.

As we do not know how many balls are needed for one particular class, we adapt the technique, known as DBSCAN, which was introduced in (Ester et al., 1996). Our proposed method is described in Algorithm 6 using an actual data point at the center of a class ball. It is the main distinct feature in our designed algorithm compared to NCM Forest (Ristin et al., 2014) and its variant version NNO (Bendale and Boult, 2015b), both of which rely on K-Means clustering. Their approach uses K-means assuming a fixed number of centroids per class to determine the centroid of each class. Our approach is more flexible as the number of centroids may increase. This approach is also a distinct feature of our method comparing to NCM and its variants. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) technique answer the query whether or not a data point belongs to a cluster of points that shares similar features. This technique does not requires the number of clusters a-prori like the K-Means method, and as a result it can perform well for many real world datasets.
We reuse the following notations from DBSCAN (Ester et al., 1996).

- A core point is a point that has at least a minimum number of points inside a circle of radius $r$ centered at it.

- A point is reachable $q$ from a point $p$ if there exists a path $p_1, p_2, \ldots, p_n$ such that each point $p_{i+1}$ is reachable from $p_i$, as well as $p_1 = p$ and $p_n = q$.

- Any point not reachable is considered outlier or a feasible candidate for a new class.

We use core points to determine if a data point belongs to an existing class or is an outlier. Our main idea is that a data point is rejected by a class if it is an outlier with respect to the class. When this data point is rejected by all known classes, it may be a candidate for a possible new class. The pseudo code is described in Algorithm 6.

---

**Figure 3.6:** Example of initial values for a new ball of the same class in NCC

**Figure 3.7:** Second case of initial new ball of the same class in NCC

We further investigate how an overlapping region can affect the accuracy. In fact, another difference of our NCC model from the online NNO (De Rosa et al., 2016) model is how a
Algorithm 3: Get neighbors

Input: a data point $p$, radius $r$, a list $C$ of core points
Output: a list of closest neighbors of $p$, a list of nearest core points

1. $\text{nearCorepoints} = \arg\min \{ \text{dist}(p, C_i) | C_i \in C \}$
2. $\text{nearNeighbors} = \text{All reachable points from } p \text{ with radius } r$

new ball is initialized as illustrated in Figure 3.6 and Figure 3.7. For a new ball boundary, Online NNO assigns the nearest ball’s radius (referred as to Figure 3.5) to the new ball which may overlap with the existing boundary of another class. This can lead to incorrect label assignment in predicting. To avoid this, we make the new ball’s radius as equal to the radius of the current nearest ball if its distance to a new ball is greater than its radius (see Figure 3.7). When a data point is close the boundary of another existing class, the radius of a new ball is changed to radius $r' = \text{distance(nearest center of ball, data point)}$, the radius of this nearest ball (see Figure 3.6). We refer to this as conservative initialization (referred to Algorithm 4), instead of spanning the full space in (De Rosa et al., 2016).

Algorithm 4: Initial radius for a new class ball

Result: Write here the result
Input: A training example $p$, nearest core point $C_i$
Output: Initialized radius

/* This implements Figure 3.6 and Figure 3.7 */

1. $r_t = \min(\text{dis}(p, C_i) - r_{c_i}, r_{c_i}, \frac{1}{2}\text{dist}(p, C_i))$

In Figure 3.8, we show the case of updating a ball by expanding when the nearest ball gives a correct prediction such that the ball center or a core point remains in its location. We adjust the ball’s radius and update list of core points during training. Figure 3.8 shows another case of creating a new ball that is far away from the nearest ball. Here we initialize the new ball radius to its nearest ball radius instead of distance between the two as proposed by (De Rosa et al., 2016).
3.3 Proposed algorithm

**Algorithm 5:** Create a new Ball

**Input:** A training example $x_t$, a nearest core point $C_{near}$

**Output:** Updated model NCC

1. $dist_{near} = \text{dist}(x_t, C_{near})$ neighbors, $C' = \text{getNeighbor}(x_t)$
2. Initialize radius
3. Update a list of core points

**Algorithm 6:** Outlier detection algorithm

**Input:** a set of data point $D$, a list of core points $C$, a minimum $\text{min}$ of data points

**Output:** updated model

1. for each point $p$ in a set $D$ do
2. neighbors, $C_i \leftarrow \text{getNeighbor}(p, C)$
3. if $p$ is reachable from a core point $C_i$ in $C$ then
4. determine $p$’s class member
5. update class ball
6. end
7. else
8. if number of neighbors $< \text{min}$ then
9. $p$ is a noise
10. end
11. else
12. create a new class ball($p, C_i$)
13. end
14. end
15. end

Our main algorithm works as follows. Each data point is tested with the nearest class ball. A new ball is created if the distance between the training data and the center the nearest ball of the same class is greater than the distance between the training data and the center the nearest ball of different class; otherwise we expand the class ball’s boundary. The new boundary may or may not be updated. For testing, the basic idea is that if a data point is not covered by its nearest class boundary, it is treated as an outlier or noise. When none of the nearest class boundaries covers this outlier, it is treated as unknown.
4 Experiment setup

To validate the proposed model, we experiment with three datasets. The first two are the The 20 newsgroups (Asuncion and Newman, 2007), and Amazon reviews (Jindal and Liu, 2008) datasets which have been used in experiments in (Fei and Liu, 2016) for document classification based on topics using a simple bag of words with TF-IDF computation. The last dataset is the aubset of IMDB movie reviews dataset, which consists of 18 genres of movies including action, animation, comedy, documentary, family, musical, romance, war,
adventure, biography, crime, drama, fantasy, music, mystery, sci-fi, thriller, western.

A brief description of each dataset is as follows:

- The 20 newsgroups dataset consisting of 18828 documents, is divided into 20 classes ranging from religion, science to sports.

- Amazon reviews in (Fei and Liu, 2016) contains 1000 reviews with a total of 50 types of products or domains.

- The subset of IMDB movie reviews is grouped into 3 genre domains which are relevant contents.

To validate classifier models in an open world setting, we use the evaluation protocol proposed in (Bendale and Boult, 2015b). In the experiment, classes (referred to as categories) are injected to the system in incremental steps while unknown test classes are evaluated. Since the results in (Fei and Liu, 2016) are used for comparison, we do not include run time which is hardware dependent.

Two domains of the 20 newsgroups dataset consist of 10 and 20 topics whereas the Amazon product reviews are split into 5 domains. Five domains of the Amazon dataset consist of 10, 20, 30, 40 and 50 classes. For IMDB movies, 3 genre domains consist of 6, 12 and 18 classes. We select a set of movies from year 2010 to year 2016, with 5 reviews for each with a total of 1,977,035 reviews in the three genre domains. The grouping is based on correlated topics, which allows us to maintain a fair balance among three groups.

The correct class assignment is referred to as TP (True Positive: correct class assignment for a known class), or the TN (True Negative: correct assignment for unknown class). Incorrect class assignment is either denoted as False Positive (FP: incorrect assignment of unknown to known) or False Negative (FN: incorrect assignment known as unknown). We
Table 3.1: F-score results on Amazon product review dataset with 10, 20 domains

<table>
<thead>
<tr>
<th>Method</th>
<th>Amazon data</th>
<th>10 domains</th>
<th>20 domains</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.25</td>
<td>0.5</td>
<td>0.75</td>
</tr>
<tr>
<td>1-vs-Set*</td>
<td>0.59</td>
<td>0.69</td>
<td>0.7</td>
</tr>
<tr>
<td>W-svm*(linear)</td>
<td>0.60</td>
<td>0.69</td>
<td>0.69</td>
</tr>
<tr>
<td>W-svm*(rbf)</td>
<td>0.25</td>
<td>0.59</td>
<td>0.70</td>
</tr>
<tr>
<td>PI-svm*(linear)</td>
<td>0.6</td>
<td>0.7</td>
<td>0.70</td>
</tr>
<tr>
<td>PI-svm*(rbf)</td>
<td>0.25</td>
<td>0.59</td>
<td>0.72</td>
</tr>
<tr>
<td>cbsSVM</td>
<td>0.45</td>
<td>0.72</td>
<td>0.78</td>
</tr>
<tr>
<td>onlineNNO</td>
<td>0.61</td>
<td>0.71</td>
<td>0.77</td>
</tr>
<tr>
<td>our model</td>
<td>0.61</td>
<td>0.72</td>
<td>0.78</td>
</tr>
<tr>
<td>Experiment on</td>
<td>10 domains</td>
<td>20 domains</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.2: F-score results on Amazon product review dataset with 30 and 40 domains

<table>
<thead>
<tr>
<th>Method</th>
<th>Amazon data</th>
<th>30 domains</th>
<th>40 domains</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.25</td>
<td>0.5</td>
<td>0.75</td>
</tr>
<tr>
<td>1-vs-Set*</td>
<td>0.46</td>
<td>0.51</td>
<td>0.54</td>
</tr>
<tr>
<td>W-svm*(linear)</td>
<td>0.52</td>
<td>0.57</td>
<td>0.58</td>
</tr>
<tr>
<td>W-svm*(rbf)</td>
<td>0.37</td>
<td>0.44</td>
<td>0.50</td>
</tr>
<tr>
<td>PI-svm*(linear)</td>
<td>0.52</td>
<td>0.58</td>
<td>0.58</td>
</tr>
<tr>
<td>PI-svm*(rbf)</td>
<td>0.38</td>
<td>0.52</td>
<td>0.63</td>
</tr>
<tr>
<td>cbsSVM</td>
<td>0.56</td>
<td>0.65</td>
<td>0.63</td>
</tr>
<tr>
<td>onlineNNO</td>
<td>0.59</td>
<td>0.64</td>
<td>0.63</td>
</tr>
<tr>
<td>our model</td>
<td>0.59</td>
<td>0.62</td>
<td>0.64</td>
</tr>
<tr>
<td>Experiment on</td>
<td>30 domains</td>
<td>40 domains</td>
<td></td>
</tr>
</tbody>
</table>

report the F-score, suggested by (Scheirer et al., 2013b),(De Rosa et al., 2016), which is also used in (Fei and Liu, 2016) by computing

\[
F-score = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} 
\]

where \( \text{Precision} = \frac{TP}{TP + FP} \) and \( \text{Recall} = \frac{TP}{TP + FN} \).

We include 1-vs-Set (Scheirer et al., 2013b), \( P_I \)-SVM (Jain et al., 2014a) and online NNO (De Rosa et al., 2016) for comparison. In fact, the online NNO model, a modified NNO, can generate the same performance as NNO but does not need a training phase to
Table 3.3: F-score results on Amazon product review on dataset with 50 domains

<table>
<thead>
<tr>
<th>Experiment on Amazon</th>
<th>0.25</th>
<th>0.5</th>
<th>0.75</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-vs-Set*</td>
<td>0.42</td>
<td>0.48</td>
<td>0.51</td>
<td>0.55</td>
</tr>
<tr>
<td>W-svm*(linear)</td>
<td>0.49</td>
<td>0.54</td>
<td>0.55</td>
<td>0.56</td>
</tr>
<tr>
<td>W-svm*(rbf)</td>
<td>0.32</td>
<td>0.37</td>
<td>0.44</td>
<td>0.58</td>
</tr>
<tr>
<td>PI-svm*(linear)</td>
<td>0.49</td>
<td>0.55</td>
<td>0.55</td>
<td>0.56</td>
</tr>
<tr>
<td>PI-svm*(rbf)</td>
<td>0.36</td>
<td>0.51</td>
<td>0.63</td>
<td>0.63</td>
</tr>
<tr>
<td>cbsSVM</td>
<td>0.56</td>
<td>0.62</td>
<td>0.59</td>
<td>0.63</td>
</tr>
<tr>
<td>onlineNNO</td>
<td>0.55</td>
<td>0.62</td>
<td>0.65</td>
<td>0.72</td>
</tr>
<tr>
<td>our model</td>
<td>0.55</td>
<td>0.62</td>
<td>0.68</td>
<td>0.75</td>
</tr>
</tbody>
</table>

50 domains

Table 3.4: F-score results on 20 newsgroup dataset

<table>
<thead>
<tr>
<th>20 newsgroup</th>
<th>0.25</th>
<th>0.5</th>
<th>0.75</th>
<th>1</th>
<th>0.25</th>
<th>0.5</th>
<th>0.75</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-vs-Set*</td>
<td>0.68</td>
<td>0.67</td>
<td>0.66</td>
<td>0.57</td>
<td>0.49</td>
<td>0.56</td>
<td>0.55</td>
<td>0.58</td>
</tr>
<tr>
<td>W-svm* linear</td>
<td>0.67</td>
<td>0.67</td>
<td>0.67</td>
<td>0.68</td>
<td>0.56</td>
<td>0.59</td>
<td>0.60</td>
<td>0.68</td>
</tr>
<tr>
<td>W-svm* rbf</td>
<td>0.32</td>
<td>0.52</td>
<td>0.68</td>
<td>0.77</td>
<td>0.37</td>
<td>0.47</td>
<td>0.61</td>
<td>0.77</td>
</tr>
<tr>
<td>PI-svm* linear</td>
<td>0.67</td>
<td>0.67</td>
<td>0.67</td>
<td>0.68</td>
<td>0.56</td>
<td>0.60</td>
<td>0.60</td>
<td>0.68</td>
</tr>
<tr>
<td>PI-svm* rbf</td>
<td>0.32</td>
<td>0.54</td>
<td>0.70</td>
<td>0.75</td>
<td>0.37</td>
<td>0.49</td>
<td>0.68</td>
<td>0.77</td>
</tr>
<tr>
<td>cbsSVM</td>
<td>0.42</td>
<td>0.77</td>
<td>0.80</td>
<td>0.86</td>
<td>0.59</td>
<td>0.70</td>
<td>0.72</td>
<td>0.85</td>
</tr>
<tr>
<td>onlineNNO</td>
<td>0.65</td>
<td>0.78</td>
<td>0.81</td>
<td>0.87</td>
<td>0.64</td>
<td>0.72</td>
<td>0.77</td>
<td>0.85</td>
</tr>
<tr>
<td>our model</td>
<td>0.65</td>
<td>0.79</td>
<td>0.82</td>
<td>0.88</td>
<td>0.64</td>
<td>0.73</td>
<td>0.79</td>
<td>0.86</td>
</tr>
<tr>
<td>Experiment on</td>
<td>10 domains</td>
<td>20 domains</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

establish the initial threshold. We were not able to obtain the implementation of cbsSVM in (Fei and Liu, 2016) to which our proposed solution’s results are compared. We report the results of online NNO and NCC (our model) and compare the results reported by (Fei and Liu, 2016). We extend our experiment with the IMDB dataset 3.4 for which we reimplement cbsSVM (similar to Rochio algorithm). Table 3.1 and Table 3.2 illustrate the experimental results for the Amazon dataset, and compare the performance of our model to online NNO (De Rosa et al., 2016). Table 3.4 reports the result for the 20 newsgroup dataset.

Settings in the experiment are
Table 3.5: F-score on IMDB dataset with 6 and 12 domains

<table>
<thead>
<tr>
<th>Experiment on IMDB</th>
<th>0.25</th>
<th>0.5</th>
<th>0.75</th>
<th>1</th>
<th>0.25</th>
<th>0.5</th>
<th>0.75</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-vs-set</td>
<td>0.50</td>
<td>0.57</td>
<td>0.69</td>
<td>0.77</td>
<td>0.49</td>
<td>0.56</td>
<td>0.55</td>
<td>0.58</td>
</tr>
<tr>
<td>W-SVM linear</td>
<td>0.51</td>
<td><strong>0.63</strong></td>
<td>0.69</td>
<td>0.77</td>
<td><strong>0.50</strong></td>
<td>0.55</td>
<td>0.67</td>
<td>0.78</td>
</tr>
<tr>
<td>W-SVM rbf</td>
<td>0.49</td>
<td>0.55</td>
<td>0.67</td>
<td>0.75</td>
<td>0.49</td>
<td>0.51</td>
<td>0.62</td>
<td>0.77</td>
</tr>
<tr>
<td>PI-SVM linear</td>
<td>0.53</td>
<td>0.61</td>
<td>0.69</td>
<td>0.78</td>
<td>0.5</td>
<td>0.53</td>
<td>0.64</td>
<td>0.76</td>
</tr>
<tr>
<td>PI-SVM rbf</td>
<td>0.46</td>
<td>0.53</td>
<td>0.66</td>
<td>0.75</td>
<td>0.46</td>
<td>0.54</td>
<td>0.63</td>
<td>0.72</td>
</tr>
<tr>
<td>cbsSVM</td>
<td>0.49</td>
<td>0.55</td>
<td>0.68</td>
<td>0.78</td>
<td>0.49</td>
<td>0.54</td>
<td>0.62</td>
<td>0.76</td>
</tr>
<tr>
<td>Online NNO</td>
<td>0.52</td>
<td>0.62</td>
<td>0.75</td>
<td>0.81</td>
<td><strong>0.50</strong></td>
<td>0.6</td>
<td>0.67</td>
<td>0.78</td>
</tr>
<tr>
<td>NCC (Our model)</td>
<td>0.52</td>
<td><strong>0.63</strong></td>
<td><strong>0.76</strong></td>
<td><strong>0.82</strong></td>
<td>0.50</td>
<td>0.62</td>
<td>0.74</td>
<td>0.80</td>
</tr>
</tbody>
</table>

6 classes | 12 classes

Table 3.6: F-score on IMDB dataset with 18 domains

<table>
<thead>
<tr>
<th>Experiment on IMDB</th>
<th>0.25</th>
<th>0.5</th>
<th>0.75</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-vs-set</td>
<td>0.43</td>
<td>0.51</td>
<td>0.55</td>
<td>0.57</td>
</tr>
<tr>
<td>W-SVM linear</td>
<td>0.46</td>
<td>0.57</td>
<td>0.64</td>
<td>0.73</td>
</tr>
<tr>
<td>W-SVM rbf</td>
<td>0.44</td>
<td>0.56</td>
<td>0.61</td>
<td>0.70</td>
</tr>
<tr>
<td>PI-SVM linear</td>
<td><strong>0.47</strong></td>
<td>0.56</td>
<td>0.65</td>
<td>0.74</td>
</tr>
<tr>
<td>PI-SVM rbf</td>
<td>0.45</td>
<td>0.52</td>
<td>0.64</td>
<td>0.69</td>
</tr>
<tr>
<td>cbsSVM (imitate)</td>
<td>0.46</td>
<td>0.57</td>
<td>0.65</td>
<td>0.73</td>
</tr>
<tr>
<td>Online NNO</td>
<td>0.46</td>
<td>0.57</td>
<td>0.65</td>
<td>0.75</td>
</tr>
<tr>
<td>NCC (Our model)</td>
<td><strong>0.47</strong></td>
<td><strong>0.60</strong></td>
<td><strong>0.68</strong></td>
<td><strong>0.77</strong></td>
</tr>
</tbody>
</table>

18 classes

- For 1-vs-Set model: $p_A = 1.6, p_\omega = 4, \lambda_r = 1$
- For W-SVM: $\delta_r = 0.001$, $\delta_R$ computed as equation III.1
- For rbf (kernel) SVM, (C=5, $\gamma = 0.2$) for Amazon, (C=5, $\gamma = 0.5$) for IMDB, and (C=10, $\gamma = 0.5$) for 20 newsgroup.

5 Results and discussion

By withholding a test class during training, we evaluate the performance of algorithms with unknown classes in different scenarios. For example, when we say training is on 25% (or
.25) it corresponds the case of an experiment where we train on examples from 8 classes and test on examples from 10 class domains in case of Amazon reviews dataset.

The F-score performances of 1-vs-Set, W-SVM, $P_f$-SVM and cbsSVM in Table 3.1, Table 3.2, Table 3.3 and Table 3.4 are from (Fei and Liu, 2016) (marked as *) followed by online NNO and our model (NCC).

Table 3.1, Table 3.2, and Table 3.3 report the F-score performance with subsets of data domains in 10, 20, 30, 40 and 50 domains for the Amazon review dataset. Similarly, Table 3.4 reports the F-score performance for subsets of 10 and 20 of newsgroups dataset while Table 3.5 and Table 3.6 report F-score performance for the IMDB movie genre dataset. In general, F-score performance (in short, performance) reduces when the number of unseen classes in training increases. Overall, our proposed model performs well in all three experiments, especially with the IMDB dataset with 0.5, 0.75 and 1 settings. We obtain similar performance with the 20 newsgroups datasets and the Amazon dataset. The 0.25 setting is the most challenging since it represents using only 25% of the testing classes during training. Both our model and online NNO place in top 80% of the time. Our model
shows more improvement when the number of the testing classes increases. Our NCC model, and online NNO (as well as NNO) are still better than random guess, albeit the number of unobserved classes in training is high. With a small number of categories (domains), we observe that linear models such as 1-vs-Set and W-SVM (linear) are very close to our performance, even outperforming in the case of the 20 newsgroups dataset. This has been demonstrated with the closed world assumption, where linear classifiers often perform well in text classification (Yuan and et al., 2012).

The cbsSVM model makes the most gain when only half the test classes are presented during training with the Amazon dataset, especially with 50 domains. However, we believe this is an unusual result since the performance of an algorithm is often low when the number of categories (domains) is high, given the high number of unobserved classes in this case, 38 out of 50 domains or .25 observed test classes. Our model performs well across the entire set of experiments, 70% or 14 out of 20 with the Amazon dataset and 87.5% or 7 out of 8 the time with the 20 newsgroups dataset.

To experiment with the IMDB genre dataset, we simulate the cbSVM model using the pseudocode in (Fei and Liu, 2016). First, we process the IMDB dataset with uni-gram and bi-gram representation. The Rocchio algorithm is used to query the document to the center of each class. Finally, the SVM model is used as the final classifier with Platt’s calibration option. Table 3.5 and Table 3.6 reveal the result of F-measure performance in this experiment. 1-vs-Set shows poor performance for text dataset in general. We observe that W-svm does better than cbsSVM model. Our model demonstrates good performance over all in three domains, especially compared to cbsSVM.

Both distance-based approaches, online NNO and our model NCC, perform better than
The accuracy of all algorithms, computed based on formula [13] in (Scheirer et al., 2013b), decreases as the number of unknown classes is introduced into testing increases (refer to Figures 3.10 and 3.12) with some exceptions for the 20 newsgroup dataset (see Figure 3.11).

However, accuracy losses vary in two distinct ways. Our model and online NNO gradually lose accuracy as more unknown classes appear in testing. 1-vs-Set, W-SVM and $P_1$-SVM perform well in our experiment with the IMDB dataset but not as well in (Fei and Liu, 2016).

Furthermore, we address the issues of initializing values when a new ball is added, and we perform a nearest neighbor search as described in the DBSCAN algorithm (Ester et al., 1996). The results demonstrate that our proposed model does not suffer from the issue of poor radius approximation $r_0$ in (Fei and Liu, 2016) when positive and negative training examples are far away.

Both Figure 3.10 and Figure 3.12 show similar trends in accuracy loss as the number of
unseen classes or domains increases. It shows how the classification problem becomes a hard problem when unknown classes exist only in testing. While the performance of the linear models reduces gradually, the RBF kernel suffers significant losses. Our model shows gradual loss in performance.

We do not experiment with the Exploratory EM model as presented in (Fei and Liu, 2016) because it is not suitable in the context of open set recognition. Last but not least, the cbsSVM model is not an incremental learning model, indicating that this model is similar to 1-vs-Set, W-SVM and $P_I$-SVM. They all retrain whenever unknown classes are introduced into the system. The reason for unexpected performance of cbsSVM (actually a simulation of csbSVM) in the IMDB dataset may be due to two reasons: a single class representation and features used. We observe that the additional of tri-grams may further improve the result.

Our proposed model seems to work better than online NNO. We believe there are two main reasons for the success of our proposed algorithms: the use of a cluster based
approach, as DBSCAN performs better when we have knowledge of the number of ball-shaped components for each class, and the new way of initializing values in our proposed model is more effective than the way it is performed in the Online NNO model.

**Evaluating the results with statistical tests**

In order to validate the results of our experiment, we compute the two-way ANOVA multiple repeated test. There are two main factors inte ANOVA test in our study: algorithms and varying training sizes. The algorithms include 1-vs-set, W-svm-linear, W-svm-rbf, PI-svm-linear, PI-svm-rbf, cbsSVM, online NNO and our proposed model NCC. The training sizes are 25%, 50% and 75% and 1. ANOVA also allows testing whether or not there are interactions between the first factor and the second factor. Interaction testing can indicate the independence of each other, for example, the effect of training size on the performance of the algorithm. For this purpose, the hypotheses we test are

- $H_{01}$: All the classifiers are the same, i.e, they produce the same results.
- $H_{02}$: Different testing class sizes during training do not impact on the performance, and
- $H_{03}$: There is no interaction effect

The result of the two way ANOVA test is F-statistic= 1.32e-06 for the first hypothesis $H_{01}$, and 2e-16 for the second hypothesis $H_{02}$, and 0.052 for the third hypothesis, all with 95% confidence levels. As a result we reject the null hypothesis. We conclude that there are differences among these classifiers and there is an impact on existing testing class during training on the performances. However, we cannot reject the first hypothesis that indicates the existing interaction among testing class and the performance.

Further, we compute pairwise t-tests for the top pairwise classifiers. We compute these
tests with the 25% open set setting which is considered the most challenging task because only 25% testing classes introduced in training.

Table 3.7: Pairwise t-test comparison

<table>
<thead>
<tr>
<th>Classifiers</th>
<th>p-values</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCC (our model) vs i-vs-set</td>
<td>0.96134312</td>
</tr>
<tr>
<td>NCC (our model) vs W_{svm-linear}</td>
<td>1.</td>
</tr>
<tr>
<td>NCC (our model) vs W_{svm_rbf}</td>
<td>0.91062365</td>
</tr>
<tr>
<td>NCC (our model) vs PI_{svm}</td>
<td>0.02635109</td>
</tr>
<tr>
<td>NCC (our model) vs PI_{svm}</td>
<td>0.04138188</td>
</tr>
<tr>
<td>NCC (our model) vs online NNO</td>
<td>1.</td>
</tr>
<tr>
<td>NCC (our model) vs cbsSVM</td>
<td>0.03224211</td>
</tr>
</tbody>
</table>

Table 3.7 indicates that our proposed model can be considered better than W-SVM rbf, PI-SVM rbf and cbsSVM. We fail to reject the null hypotheses that our proposed model is better than 1-vs-set, W-SVM linear, PI-SVM linear and Online NNO.

6 Conclusion and contributions

In this chapter, we have presented our proposed solution that attemps to overcome the deficiencies of online NNO in initializing and updating information for a class ball. While our proposed model works well in the text domain, we see room for the improvement since we do not focus on replacing old concepts with the new one in this design to make the algorithm more effective working in an online fashion.

One direction of future work is to adapt eXtreme Gradient Boosting by allowing it to create a new model as new data examples arrive and use a modifies the combination strategy. We investigate these ideas in the final chapter.

The contributions of this work can be summarized as follows.
1. proposed NCC model is comparable to state-of-the-art open set solutions,

2. incremental learning design has advantages than 1-vs-set, W-SVM and PI-SVM

3. NCC adapts better in different data domains due to its use clustering technique.
CHAPTER IV

A SCALABLE ENSEMBLE MODEL

An individual often has only partial knowledge required to solve a particular problem. A committee is a common way to address this issue because committee members can pool their expertise in order to make a better decision. Similarly, an ensemble learning can combine models trained in different sub-regions of a data space to provide scalability to data mining algorithms. We hypothesize that an ensemble model can achieve scalability if the combining method also improves the predicted results of its classifier members operating on the different regions of original data space. For example, if there is no single classifier that excels in the entire data space, a solution is to split data into smaller regions and to generate the final prediction. However, if the majority of classifiers are not trained in the same region where testing occurs, the majority voting method will fail to generate proper results. Herein, we propose a novel method to overcome the challenge of efficiently assigning and adjusting weights for highly confident classifiers.

(Kuncheva, 2002) illustrate that an ensemble model may perform better than any base classifier as long as all the classifiers are complimentary. In other words, the errors made by
the ensemble classifier members must be independent (Zupanski and Zupanski, 2006). For example, consider an ensemble model \( D = D_1, \ldots, D_L \) where \( D_i \) denotes the \( i^{th} \) member of the ensemble. Let \( K \) denote the number of sub-regions, refer to as regions of competence, into which the original feature space is decomposed. Let \( R_1, \ldots, R_K \) denote the \( K \) sub-regions. Let \( D^* \in D \) denote the ensemble member with the highest average accuracy over the entire feature space. Let \( P(D_i|R_j) \) denote the probability of correct classification by \( D_i \) in sub-region \( R_j \). Let \( D_{ij} \) denote the classifier responsible for sub-region \( R_j, j = 1, \ldots, K \). The probability of correct classification by the aforementioned classifier selection system is computed as

\[
P(\text{correct}) = \sum_{j=1}^{k} P(R_j)P(D_{ij}|R_j) \tag{IV.1}
\]

where \( P(R_j) \) denotes the probability that an input is drawn from region \( R_j \). To maximize \( P(\text{correct}) \), we assign \( D_{ij} \) such that \( P(D_{ij}|R_j) \geq P(D_i|R_j) \) where \( i \in [1, \ldots, L] \). As a result,

\[
P(\text{correct}) \geq \sum_{j=1}^{K} P(D^*|R_j) = P(D^*). \tag{IV.2}
\]

Assuming that \( D_{ij} \) is the best classifier of the ensemble model \( D \) in region \( R_j \), it simply indicates that the ensemble model is as good as the best classifier \( D^* \), regardless of the decomposition method used.

1 Common techniques for ensemble models

Bagging and Random Subspace are two techniques often used in an ensemble model to generate different subsets of data for ensemble members.
• Bagging method or Bootstrap Aggregation (Breiman, 1996) produces diversity for ensemble models using sampling with replacement. Bagging efficiently reduces an ensemble model’s variance.

• Random Subspace (Ho, 1998) splits a feature space into smaller regions that may overlap. This is an efficient way to reduce a high dimensionality domain, particularly when the number of features is larger than the number of examples.

Another variation of Bagging is the Pasting method (Breiman, 1999) where subsets are randomly generated from a dataset. However, the ensemble model is reported to have higher accuracy (Louppe and Geurts, 2012) using both methods. In other words, to aggregate the results, an ensemble model can implement the following methods:

• The boosting method focuses on fixing situations where the previous classifiers makes wrong prediction.

• The combining method, commonly used with bagging, aggregates the outcomes from ensemble members.

A boosting model such as AdaBoost (Freund et al., 1996) can be built on top any classifier by assigning a weight to each training example. At each round, the weight to the misclassified examples is increased to correct the problem. However, Boosting gives more weight to misclassified instances for the next iteration based on previous predictions of a classifier in a sequential process. Examples of boosting include AdaBoosting and eXtreme Gradient Boosting.

One variant of boosting is stacking, which is defined as a two layer model. Stacking can tackle both variance and bias problems by using another classifier, also known as the combiner to reduce generation error. The Stacking method or the Stacked generalization model (Wolpert, 1992), works on a three layer model.

• The first classifier is trained on a random subset of the available training data, similar to bagging.
Figure 4.1: Illustration of common ensemble approach

- The second classifier is trained on a dataset where half the examples from different subsets than the origin subset and half the examples come from misclassified instances.

- The third classifier is trained where with instances where the first and the second disagree.

A modified approach to Stacked generalization, named Blender uses a hold-out subset of data for training instead of out-of-fold prediction whereas the combining method, which can be applied in Bagging and Boosting approaches, focuses on how to fix incorrect predictions with correct outcomes. We present a brief review in the following section.

2 Combining methods in ensemble learning

Given different predicted outcomes of ensemble members, an ensemble model can use a combining method to aggregate a final prediction. A simple solution is the majority vote method which considers all classifiers as being equal. Majority vote was first introduced in Bagging by (Breiman, 1996), Extra Trees (Geurts et al., 2006) and in Random Forests (Breiman, 2001) where an ensemble model was initially comprised of homogeneous classifiers. For an ensemble of \( \mathcal{L} \) classifiers, Majority vote (Battiti and Colla, 1994) will generate a correct prediction if at least \( \lceil \mathcal{L}/2 \rceil + 1 \) classifiers obtain the correct answers. Let us consider an
ensemble of three classifiers $C_1$, $C_2$, and $C_3$ with accuracy 0.6, 0.6 and 0.8, respectively. An ensemble model with majority vote, corresponding to two classifiers that classify correctly, generates the final accuracy: 

$$P_{majority} = 0.6^2 \ast 0.2 + 2 \ast 0.4 \ast 0.6 \ast 0.8 + 0.6^2 \ast 0.8 = 0.774$$

<0.8: the best accuracy of a single classifier from this model. If we drop the two poor performers $C_1$ and $C_2$, our ensemble model achieves an accuracy of 0.8 as a single best classifier. This pitfall of Majority vote can be rectified by adding the coefficients or weights $w_1, w_2$ and $w_3$ such that $w_1 = w_2 = 0$ and $w_3=1$. We see that Weighted Majority vote can improve Unweighted Majority vote.

**Algorithm 7: Majority Vote system**

**Input**: A number of experts N

**Output**: A final prediction

1. **for each member i in ensemble model do**
2. Assign class labels $s_1, \cdots, s_n$ by n classifier members.
3. Tally the number of votes for each class $c_i$, $i=1..m$, $m < n$
4. 
   
   $$P(k) = \sum I(s_i, c_i) \text{ where } I(a,b =1) \text{ if } a=b \text{ and } 0 \text{ otherwise}$$
5. Assign label $l^*$ to the example such that $l^* = \arg\max_n P(k)$
6. **end**
7. **return** prediction from majority vote

The Majority vote is effective so long as all the committee members have identical predictive power. However, many studies have criticized this assumption (Ruta and Gabrys, 2001) and have pointed out the limitation of the Majority vote (Kuncheva et al., 2003). In the introduction to this thesis, we illustrate that the final prediction can be improved if and only if a diversity of classifiers is guaranteed (Kuncheva and Whitaker, 2003). The presence of distinguished committee members leads to less assimilated classifiers, and as a result a best classifier member needs to weigh more than a weak classifier member.

As sampling techniques are commonly used to create the different samples of data, each
subset of data may be generated from a different portion of the data space. Many researchers question the use of invariant weights (Kim et al., 2011) in ensemble models because there is no guarantee of a uniform distribution in the sampling process. In fact, each member of an ensemble model may have different expertise on a given data domain. As a result, we will need to adjust for members who give incorrect advice. Suppose we construct an ensemble model for a binary problem where each member’s (expert) decision is either correct (1) or incorrect (0). Let \( W = \{w_1, w_2, \ldots, w_n\} \) a weight vector of ensemble members where \( i \in [1, n] \).

**Algorithm 8: Weighted Majority vote system**

```
Input : Each ensemble member initially has weight \( w_i = 1 \)
Output : A final predict decision

for each round do
  1  Sum all weights for 0 decision vote
  2  Sum all weights for 1 decision vote
  3  Predict based on (Weighted) Majority Vote
  4  Validate the prediction with a ground truth
  5  Penalize mistakes by lowering corresponding weights by half,
     a new weight \( w_i \leftarrow \frac{1}{2} \cdot w_i \)
end
return prediction
```

Weighted Majority voting has been implemented in many state-of-the-art ensemble models such as Adaboost, eXtreme Gradient Boosting (XGB), and Random Forests. While Adaboost assigns fixed weights to each ensemble member, XGB allows updating weights of members in each training step to address incorrect predictions. The Dynamic Weight method allows adjustment of weights for classifiers that generate correct prediction. In particular, the Dynamic Weight method often uses an extra classifier, for example, a single layer of neural network to learn the weights. This combining method has demonstrated
high accuracy in many reports (Jin et al., 2003), (Karmaker et al., 2007), (Valdovinos and Sanchez, 2009).

3 Related work

The Weighted Majority vote algorithm was first studied by (Littlestone and Warmuth, 1989) where it was shown that it can generalize by adding a term quota where a quota is a minimum number of votes needed to achieve a majority. In the Weighted Majority vote, we reduce the current weight in half when the ensemble member or expert makes an incorrect prediction. Let \( n \) denote the number of ensemble members (referred to as experts or members) and \( W \) denotes the total weight of all members.

Assuming that weights are initialized as \( w_i = 1 \, \forall i \in n \). Let \( m \) and \( M \) be the numbers of mistakes made by the best members and made by Weighted Majority vote algorithm, respectively. If at least half of the total weights of ensemble members produce incorrect prediction results, it results in an incorrect prediction by the Weighted Majority vote algorithm. Because the penalty reduce the weight by half for each corresponding weight for incorrect prediction, the total weight \( W \) will be reduced by at least \( 1/4 \). Repeating this predicting process \( n \) times, the final weight eventually becomes \( W \leq n(3/4)^M \). On the other hand, the best ensemble members makes \( m \) mistakes and so its weight is \( (1/2)^m \). The best ensemble member weight is \( (1/2)^m \leq W \). From \( W \leq n(3/4)^M \) and \( (1/2)^m \leq W \), we derive \( M \leq 2.41(m + lgn) \) or the number of mistakes of the algorithm depends on the number of mistakes made by the best expert. Since this algorithm guarantees an upper bound of 2.41, it may not always be better than random guess. To illustrate this, consider an
ensemble of 10 members where the best member makes 20% mistakes. The upper bound is $2.41(20 + \log_2 10) \approx 56\%$ mistakes, which is far from a good upper bound because the smaller percentage of mistake the better is the ensemble classifier as a whole.

An improved version, Random Weighted Majority (RWM) vote, normalizes each weight of member by the inverse of the weight total. The RWM method, proposed by (Littlestone and Warmuth, 1989) views the weights as probabilities. The predictions are generated proportionally to the weights such that the probability of the $i^{th}$ ensemble member becomes $\frac{w_i}{W}$ where $W = \sum_i w_i$. Such that, we replace the weight of the $i^{th}$ classifier (as expert) with the ratio $w_i/W$ where $W = \sum_i w_i$. In other words we choose an expert $i$ with probability $w_i/W$ and predict what this expert says. Here $\epsilon$ is a fraction of weight reduction. Where $\epsilon$ is

\begin{algorithm}
\textbf{Algorithm 9: Randomized Weighted Majority vote system}
\begin{itemize}
\item \textbf{Input} : Each ensemble member initially has weight $w_i = 1$
\item \textbf{Output} : A final prediction decision
\item \textbf{for each round do}
\item \hspace{1em} Given a set of predictions $w_1, w_2, \ldots, w_n$ by experts
\item \hspace{1em} Produce $x_i$ with the probability $p_i = w_i/W$ where $W = \sum_i w_i$
\item \hspace{1em} Validate and predict and compare to the ground truth
\item \hspace{1em} Penalize each mistake made by expert s.t. $w_i \leftarrow (1 - \epsilon) \cdot w_i$
\item \textbf{end}
\end{itemize}
\end{algorithm}

a fraction of weight reduction.

(Littlestone and Warmuth, 1989) show that the expected number of mistakes ($M$) satisfies the upper bound

\[ M \leq \frac{mln(1/\beta) + ln(n)}{1 - \beta} \]  \hspace{1em} (IV.3)

(Polikar et al., 2001) adapted the Weighted Majority vote, which was introduced in the AdaBoost model, to address the problem of aggregating predictions by a majority of
ensemble members with incorrect predictions. This situation arises when the testing data come from different regions of training data. The idea is to update weights of each classifier member and retrain the model. Another modified version of the Weighted Majority vote approach is introduced in (Muhlbaier et al., 2004), known as Dynamic Weighted voting, and another by (Muhlbaier et al., 2009), known as Dynamically Weighted Consult and Vote. These combining methods use error rates in predicting as indicators to adjust weights and historical performance to determine the confidence level of votes.

Generally, combining methods can obtain the final predictions by focusing on the best classifier of the ensemble as we see in the example of implementing a single classifier to aggregate the results. The simplest way to determine the best classifier is to ignore other ensemble members with low performance. On the other hand, (Zamani et al., 2014) recommend that the number of best classifiers be equal to the number of classes plus one to improve the prediction. In their ensemble model, classifiers at the output form a cascade where output of one is the input to other classifiers. The correct label of an example is an input for the classifier that produces the output label. This reduces the weights for the remaining classifiers by a factor. (Zamani et al., 2014) demonstrate that mistakes of an ensemble model can be lowered below the mistakes generated by Randomized Weighted Majority vote when the training size is increased. However, their proposed ensemble model requires high running time and produces poor performance on unbalanced classes.

In this thesis, we overcome the problems in handling a large dataset by proposing a novel combining method for an ensemble model, where each classifier may be trained in a sequence, producing aspects of the classification knowledge for the individual classifiers. According to our knowledge, our proposed method is the first investigation into developing
a combining method for an ensemble of different classifiers.

4 Limitation of current combination methods

Randomized Weighted Majority vote may not generalize well in dealing with classifiers in an ensemble model which, allows participation of distinct type of classifiers as committee members. Even though this approach is common in enhancing expertise by diversity, the problem is that when a majority of ensemble members are not trained in the same region of the testing data, a combining methods may often incorrectly reject the correct predicted outcomes. Figure 4.2 illustrates another example of two groups of classifiers. Here,

Figure 4.2: Problem of ensemble combination given different training.

members of group 1 have been trained on classes 1 and 2 while all members of group 2 have been trained on classes 1, 2 and 3. Note that at this point, the ensemble model includes two groups of members which have acquired different aspects of the classification knowledge necessary. Now, consider a testing stage where an object from class 3, is introduced as shown in Figure 4.3. If a simple majority vote is taken, we see that three members without
knowledge of class 3 outvote a minority group of two and continue doing so until the minority group gains sufficient votes to outvote the remaining members, which may only be after be several misclassifications. Later while a modified weight for the minority may be a promised direction to explore, it is clear in this example that a simple Weighted Majority vote does not provide a mechanism to solve this problem. Herein, we demonstrate precisely what we solve in this dissertation.

![Figure 4.3: An ensemble member not trained on a certain class, during testing](image)

It is important to reiterate that simply coming up with a good voting process does not always guarantee the correct solution unless the knowledge of training is taken into account. Since the training process can be used to obtain the confidence levels of votes to be used in aggregating results, we can use this confidence level as the weight of a classifier’s decision. Decision making in ensemble learning represents a core weakness that needs to be resolved. The dilemmas between exploration and exploitation arises when an ensemble model is forced to make decisions under uncertainty. We seek to obtain more information to ensure a higher level of confidence in decision making. A popular example is a multi armed bandit, in which when we pull the lever of a slot machine, we may get lucky and get a reward. Some slot machines reward us more than others, but it is a secret that someone may not let others win. Each machine has an unknown probability of generating a prize which corresponds to the fact that each ensemble classifier has a different weight of its confidence level in
In academic literature, this problem is referred to as a multi-armed bandit problem (Kuleshov and Precup, 2014). Multi-armed bandit methodologies often talk about a one-arm bandit, a term for slot machine. For ensemble learning, a multi-arm bandit illustrates multiple classifiers in an ensemble model. We use the Multi-armed bandit analogy to formulate a solution for adjusting weights of an ensemble model. Initially, the independent assumption of ensemble members is guaranteed as each classifier represents a slot machine which is independent of all others. This thesis allows the overlap of regions when sampling the data. In our proposed solution, we update weights for the ensemble classifiers based on prior knowledge of historical training data.

5 Proposed method

Intuitively, we want to have more confidence on a classifier which is trained on the region from which a tested example comes. A simple solution is to determine the best classifier and give it a larger weight and give other classifiers proportional weights. These weights can be determined based on the correctness of predictions of corresponding classifier in previous step. The difficulty of this approach occurs when two classifiers generate different prediction but have proximity for training regions. To counter this issue, we exploit the training history of each classifier. In this thesis, we use Bayesian inference to formulate the posterior belief and compute the weight of each classifier. We denote the information known up to time $t$ such that a classifier $C$ maps data $\mathcal{D}$ to distributions over actions $a$. The multi-armed bandit algorithm can describe as follows.
Initially, our belief for each classifier is drawn from a uniform distribution,

At each time step \( t \) we choose action \( x_t \in \chi \),

We compute a new belief (referred to as weight) factor \( r_i \) based on our prior belief factor,

We determine the weight factor for remaining arms (classifier).

We use a Bayesian approach to combine label outputs as follows. Let \( s = s_1, \ldots, s_L \) denote class labels, and \( P(c_k | s) \), \( k=1..n \) denote the probability that class \( k \) is the true label given class label \( s \). We assume that all classifiers are independent such that \( P(s_1, s_2, \ldots, s_L | c_k) = P(s_1 | c_k)P(s_2 | c_k) \ldots P(s_L | c_k) \). Therefore

\[
P(c_k | s) = \frac{P(c_k)}{P(s)} \prod_{k=1}^{L} P(s_i | c_k). \tag{IV.4}
\]

Let \( I^+_k \) denote the set of indices of classifiers vote for class \( c_k \) and \( I^-_k \) denote the set of indices of classifiers vote for other class label. The probability can be written as

\[
P(c_k | s) = \frac{P(c_k)}{P(s)} \prod_{i \in I^+_k} P(s_i | c_k) \times \prod_{i \in I^-_k} P(s_i | c_k) \tag{IV.5}
\]

This formula describes the optimality condition for several combination rules (Kuncheva and Rodríguez, 2014). Our goal is to determine the probability of correct prediction of each classifier in order to determine a strong estimation of the probability of correct prediction for the combination method. Keeping track of the history which examples worked well, discovered for which classifier during training may be helpful, but our research showed that this method suffers from memory loss as (Muhlbaier et al., 2009). We apply Bayesian inference to compute the posterior probability, given prior knowledge in adapting the multi-
arm bandit algorithm to improve the performance of the Randomized Weighted Majority vote solution.

During training, we have a set of action $a$ and a context $x$ where a reward $r$ is observed. The set of historical observation is computed using a likelihood function $P(r|a, x, \theta)$ providing on some parameters $\theta$. For some prior distribution $P(\theta)$ on these parameters, the posterior distribution is given by Bayes rule $P(\theta|D) \propto P(r|a, x, \theta) P(\theta)$.

In our proposed solution, we use the Mahalanobis metric to measure the distance from testing examples to the center of distribution of data points in each region of the original data set. If testing data points are in the same region of the data space where the classifier is trained, we found that this classifier had more confidence in predicting. The classifier assigned to the region with the smallest Mahalanobis distance to the testing data point will have a larger weight. Our method which stores the mean and the covariance matrices of the training set, is more efficient than a classifier that stores the entire history. In addition, an upper bound in the bandit problem using Bayesian inference has been obtained for the theoretical support in (Kaufmann et al., 2012).

Algorithm 10: Bayesian Weighted Majority Vote system

| Input : Initialize all weight $w_i = 1$ |
| Output : A final predict decision |

1. for each round do
2. Given a set of predictions $w_1, w_2, \cdots, w_n$ by experts
3. $p_i \leftarrow \text{BayesianMultiArmedBandit}(i,W)$
4. Update history prediction
5. Penalize each mistake made by expert s.t. $w_i \leftarrow (1 - \epsilon) \cdot w_i$
6. end

where $\epsilon$ is a fraction for weight reduction. Since our proposed solution is based on Bayesian inference, an assumption is required that the ensemble classifiers (referred to as
arms) are independent. Fortunately, a diversity of classifiers in the ensemble model can be achieved by independently selecting the classifiers. It in turn guarantees that the samples can be drawn from each classifier in an independently and identically distributed manner.

**Algorithm 11:** Bayesian Multi armed bandit

**Input:** \( i^{th} \) index of a committee member, \( W \)

**Output:** A probability for confidence weight of \( i^{th} \) member

1. **for each prediction of committee with data \( p \) do**
   2. Compute Mahalanobis distance from \( p \) to center of data distribution
   3. \( h_i \leftarrow \): inverse Mahalanobis distance
   4. Compute posterior distribution of weight \( p[R|h_i] \)

---

### 6 Experimental setup

We used the IMDB movie reviews and Amazon product reviews datasets to evaluate our proposed combining method in two experiments. First we test under the open set setting to create an ensemble of different classifiers. For the Amazon dataset, we use 50 domains and we used 3 group domains in the IMDB movie reviews dataset. The testing classes are introduced into training by 0.25 (25%), 0.5 (50%), 0.75 (75%) and 1 (100% of the testing classes available in training.

In our prior work, the reported in Chapter III, the proposed NCC model revealed a limitation in difficult test cases when there was were fewer than 25% testing classes existing during training. In other words, when the number of untrained classes is high, NCC does not perform as well as other open set solutions. We hypothesize that a combining method can improve this weakness if there exists strong classifier members in the ensemble model. In the experimental setting, our choice of one strong classifier may be picked up from one of
the proposed models for open set classification. This intentionally designed ensemble model allows traditional classifiers as ensemble members to work well under the open set scenario.

While the NCC classifier has been designed for working with open set, other classifiers can be selected from a pool of classifiers. Alternative, the selection of remaining classifiers can be formulated on the algorithm selection problem. In a the second experiment, the IMDB movie review and Amazon product review datasets are used to predict a negative or positive sentiment in a classification task. Sentiment analysis is a popular research topic to explore what users think about the services they have paid for (Pang et al., 2002), (Agarwal et al., 2011). We choose a word embedding method Word2Vec (Mikolov et al., 2013), and use statistical features to combine classifiers. We apply 65%, 20%, 15% splits for training, testing and validation.

Other classifiers include Naive Bayes (NB), eXtreme Gradient Boosting (XGB), Logistic Regression (LR), Stochastic Gradient Descent (SGD), and Linear SVM. In addition, we use neural network models implemented in the Keras library, a Convolutional Neural Network (CNN) (LeCun et al., 1995) and a recurrent neural network (RNN)(Sak et al., 2014). The RNN model is known as a Long Short Term Memory (LSTM) network using

![Diagram of an ensemble model for IMDB sentiment with BMB Random Weighted Majority Vote](image)

Figure 4.4: An ensemble model diagram.
Back-propagation Through Time.

Figure 4.4 illustrates the architecture of our approach. In the processing step, we extract data, remove corrupt words and numbers. The next step is to generate feature sets for review datasets.

Table 4.1: F-scores in Experiment 1a on Amazon product review dataset

<table>
<thead>
<tr>
<th>Experiment on Amazon</th>
<th>0.25</th>
<th>0.5</th>
<th>0.75</th>
<th>1</th>
<th>0.25</th>
<th>0.5</th>
<th>0.75</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-vs-set</td>
<td>0.59</td>
<td>0.68</td>
<td>0.74</td>
<td>0.78</td>
<td>0.56</td>
<td>0.62</td>
<td>0.68</td>
<td>0.72</td>
</tr>
<tr>
<td>W-SVM linear</td>
<td>0.61</td>
<td>0.66</td>
<td>0.76</td>
<td>0.79</td>
<td>0.58</td>
<td>0.64</td>
<td>0.70</td>
<td>0.75</td>
</tr>
<tr>
<td>PI-SVM linear</td>
<td>0.60</td>
<td>0.69</td>
<td>0.78</td>
<td>0.84</td>
<td>0.57</td>
<td>0.66</td>
<td>0.69</td>
<td>0.73</td>
</tr>
<tr>
<td>Online NNO</td>
<td>0.60</td>
<td>0.71</td>
<td>0.77</td>
<td>0.83</td>
<td>0.58</td>
<td>0.66</td>
<td>0.70</td>
<td>0.76</td>
</tr>
<tr>
<td>Our ensemble model</td>
<td>0.61</td>
<td>0.71</td>
<td>0.79</td>
<td>0.84</td>
<td>0.59</td>
<td>0.67</td>
<td>0.72</td>
<td>0.78</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>10 domains</th>
<th>20 domains</th>
</tr>
</thead>
</table>

Table 4.2: F-scores in Experiment 1b on Amazon product review dataset

<table>
<thead>
<tr>
<th>Experiment on Amazon</th>
<th>0.25</th>
<th>0.5</th>
<th>0.75</th>
<th>1</th>
<th>0.25</th>
<th>0.5</th>
<th>0.75</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-vs-set</td>
<td>0.50</td>
<td>0.58</td>
<td>0.68</td>
<td>0.70</td>
<td>0.53</td>
<td>0.55</td>
<td>0.58</td>
<td>0.62</td>
</tr>
<tr>
<td>W-SVM linear</td>
<td>0.55</td>
<td>0.60</td>
<td>0.65</td>
<td>0.72</td>
<td>0.55</td>
<td>0.56</td>
<td>0.63</td>
<td>0.67</td>
</tr>
<tr>
<td>PI-SVM linear</td>
<td>0.56</td>
<td>0.59</td>
<td>0.64</td>
<td>0.71</td>
<td>0.56</td>
<td>0.58</td>
<td>0.64</td>
<td>0.68</td>
</tr>
<tr>
<td>Online NNO</td>
<td>0.57</td>
<td>0.60</td>
<td>0.63</td>
<td>0.69</td>
<td>0.55</td>
<td>0.60</td>
<td>0.63</td>
<td>0.67</td>
</tr>
<tr>
<td>Our ensemble model</td>
<td>0.58</td>
<td>0.63</td>
<td>0.67</td>
<td>0.72</td>
<td>0.57</td>
<td>0.61</td>
<td>0.65</td>
<td>0.68</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>30 domains</th>
<th>40 domains</th>
</tr>
</thead>
</table>

Table 4.3: F-scores in Experiment 1c on Amazon product review dataset

<table>
<thead>
<tr>
<th>Experiment on Amazon</th>
<th>0.25</th>
<th>0.5</th>
<th>0.75</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-vs-set</td>
<td>0.50</td>
<td>0.53</td>
<td>0.57</td>
<td>0.59</td>
</tr>
<tr>
<td>W-SVM-linear</td>
<td>0.54</td>
<td>0.62</td>
<td>0.65</td>
<td>0.68</td>
</tr>
<tr>
<td>PI-SVM linear</td>
<td>0.54</td>
<td>0.58</td>
<td>0.64</td>
<td>0.67</td>
</tr>
<tr>
<td>Online NNO</td>
<td>0.55</td>
<td>0.62</td>
<td>0.66</td>
<td>0.73</td>
</tr>
<tr>
<td>Our ensemble model</td>
<td>0.56</td>
<td>0.62</td>
<td>0.67</td>
<td>0.73</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>50 domains</th>
</tr>
</thead>
</table>

For the CNN model, we use the settings:
• 5 layers: (embedding, 1D convolution, max pooling, 2 dense network layers,

• pre-trained word vectors as input.

• the ADAM optimizer algorithm (Kingma and Ba, 2014)

For LSTM recurrent network, the following settings are use:

• 5 layers

• two dropout layers between embedding and LSTM, LSTM and dense layer

• optimizer ADAM algorithm (Kingma and Ba, 2014)

7 Results and discussion

To obtain the best model for each algorithm, we use hyperopt which is a distributed asynchronous hyperparameter optimization utility developed by Bergstra et al. (2013) for hyper-parameter search using random search option. The Random search can run fast and its results are often comparable with the results from grid search. Noticeably, we limit our algorithms to 1-vs-set, W-SVM linear and PI-SVM because the experiments are highly time consuming. The F-scores of algorithms in Experiment 1 are shown in Table 4.1, Table 4.2 and in Table 4.2. We observe that there are significant improvements of our model as well as 1-vs-set, W-SVM, $P_I$-SVM which rely on parameter tuning. An ensemble model with our proposed combining method demonstrates significant improvements in the hard scenario when only 25% testing classes available in training.

We observe a similar trend with the IMDB movie review dataset. We also notice that the homogeneous ensemble models make large gains when hyper-parameter search is uaws.
Our model also obtains a higher performance when half of the testing classes are presented at training.

Table 4.4: F-scores in Experiment 1a on IMDB movie review

<table>
<thead>
<tr>
<th>Experiment on IMDB</th>
<th>0.25</th>
<th>0.5</th>
<th>0.75</th>
<th>1</th>
<th>0.25</th>
<th>0.5</th>
<th>0.75</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-vs-set</td>
<td>0.50</td>
<td>0.57</td>
<td>0.68</td>
<td>0.78</td>
<td>0.49</td>
<td>0.55</td>
<td>0.63</td>
<td>0.73</td>
</tr>
<tr>
<td>W-SVM linear</td>
<td>0.52</td>
<td>0.63</td>
<td>0.69</td>
<td>0.81</td>
<td>0.51</td>
<td>0.64</td>
<td>0.72</td>
<td>0.79</td>
</tr>
<tr>
<td>PI-SVM linear</td>
<td>0.53</td>
<td>0.61</td>
<td>0.71</td>
<td>0.81</td>
<td>0.5</td>
<td>0.62</td>
<td>0.64</td>
<td>0.80</td>
</tr>
<tr>
<td>Online NNO</td>
<td>0.52</td>
<td>0.62</td>
<td>0.75</td>
<td>0.81</td>
<td>0.5</td>
<td>0.6</td>
<td>0.63</td>
<td>0.78</td>
</tr>
<tr>
<td>Our ensemble model</td>
<td>0.53</td>
<td>0.64</td>
<td>0.76</td>
<td>0.82</td>
<td>0.52</td>
<td>0.64</td>
<td>0.74</td>
<td>0.80</td>
</tr>
</tbody>
</table>

6 classes 12 classes

Table 4.5: F-scores in Experiment 1b on IMDB movie review

<table>
<thead>
<tr>
<th>Experiment on IMDB</th>
<th>0.25</th>
<th>0.5</th>
<th>0.75</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-vs-set</td>
<td>0.44</td>
<td>0.53</td>
<td>0.56</td>
<td>0.62</td>
</tr>
<tr>
<td>W-SVM linear</td>
<td>0.46</td>
<td>0.57</td>
<td>0.64</td>
<td>0.75</td>
</tr>
<tr>
<td>PI-SVM linear</td>
<td>0.47</td>
<td>0.56</td>
<td>0.67</td>
<td>0.78</td>
</tr>
<tr>
<td>Online NNO</td>
<td>0.46</td>
<td>0.57</td>
<td>0.65</td>
<td>0.75</td>
</tr>
<tr>
<td>Our ensemble model</td>
<td>0.47</td>
<td>0.59</td>
<td>0.68</td>
<td>0.78</td>
</tr>
</tbody>
</table>

18 classes

Table 4.6: Accuracy of classifiers in Experiment 2

<table>
<thead>
<tr>
<th>Clf</th>
<th>NB</th>
<th>LR</th>
<th>SGD</th>
<th>XGB</th>
<th>FM</th>
<th>Linear SVM</th>
<th>CNN</th>
<th>RNN</th>
<th>our model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yelp</td>
<td>88.4</td>
<td>93.8</td>
<td>89.34</td>
<td>84.56</td>
<td>96.2</td>
<td>91.35</td>
<td>90.4</td>
<td>93.5</td>
<td>98.2</td>
</tr>
<tr>
<td>IMDB</td>
<td>81.9</td>
<td>95.54</td>
<td>95.67</td>
<td>94.52</td>
<td>96.34</td>
<td>83.4</td>
<td>88.28</td>
<td>86.36</td>
<td>97.5</td>
</tr>
</tbody>
</table>

We perform Experiment 2 assuming that we have prior knowledge of all classes. Table 4.6 illustrates the accuracy of ensemble models with different combining methods compared to a single algorithm solution. In Table 4.7, we show that our proposed method provides comparable results with common combining methods. Table 4.6 illustrates the performance of our ensemble model and several state-of-the-art models including Logistic Regression, eXtreme Gradient Boosting, Stochastic Gradient Descent, Factorization Machine, and
Table 4.7: Accuracy of combining methods in Experiment 2

<table>
<thead>
<tr>
<th>Ensemble model</th>
<th>our method</th>
<th>Dynamic Weighted</th>
<th>Random Weighted</th>
<th>Majority Vote</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yelp</td>
<td>98.2</td>
<td>97.4</td>
<td>93.6</td>
<td>83.4</td>
</tr>
<tr>
<td>IMDB</td>
<td>97.5</td>
<td>95.8</td>
<td>94.52</td>
<td>82.3</td>
</tr>
</tbody>
</table>

LinearSVMm with a closed set assumption. Our ensemble model outperforms all single classifiers in both tasks. We also report the accuracy of the ensemble model with different combination methods in Table 4.7.

**Validating the comparison results in the study** We use the two way ANOVA test with a 95% confidence level. In the open set experiment, our hypotheses are

- $H_{01}$: All the classifiers are the same, and
- $H_{02}$: Unknown classes do not impact the performance.

Table 4.8: t-tests in Experiment 2

<table>
<thead>
<tr>
<th>Combining method</th>
<th>p-values</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMB Weight Majority vs. Dynamic Weighted Majority vote</td>
<td>0.10692227</td>
</tr>
<tr>
<td>BMB Weight Majority vs. Random Weighted Majority vote</td>
<td>0.02247811</td>
</tr>
<tr>
<td>BMB Weight Majority vs Majority vote</td>
<td>0.00188355</td>
</tr>
</tbody>
</table>

The F-statistic from ANOVA test is 2.69E-05 for the first hypothesis $H_{01}$ and 2.64E-08 for the second hypothesis $H_{02}$. As a result, we reject the null hypothesis that says that all classifiers are different and unknown classes impact the performance in the first experiment.

We also compute pairwise t-test for our proposed ensemble model and 1-vs-set classifier to evaluate whether or not our ensemble model with the proposed combining method make any significant improvement. The $p-value = 2.1E - 4$ indicates our model leads to improvement.

Table 4.8 indicates that our proposed method is significantly better than Random Weighted Majority vote and Majority vote but not Dynamic Weighted Majority vote. How-
ever, we think that we have demonstrated a solution to combine the outcome of an ensembles model to improve the performance of our algorithm in the open-set environment using our proposed method. Two questions we want to explore further: Does the performance improve because we include open set assumption in the ensemble model? How does the performance of the proposed method change if it is used in a homogeneous ensemble model?

8 Conclusion and contributions

Our proposed combining method illustrates a promising solution for both closed-set and open set scenarios, although our proposed method is not significantly different in performance compare to Dynamic Weighted Majority vote. However, it is not a fair comparison because of two reasons: i) Dynamic Weighted Majority vote requires keeping historical training data and 2) the experiment was designed to work with current combining methods. Our method needs less memory and is able to work in the open-set scenario. The contributions of this work can be summarized as follows.

- Ability to deal with a combination of different classifiers in open-set scenario.
- An expert’s opinion is weighted with a probability proportional to its performance in history.

Our combining method is an important contribution to the scientific community because in order to aggregate the predicted result in an ensemble model this work has proposed a novel for future research. Experimental results show that this proposed combining method is significant. One reason is that our ensemble model does not need expense time consuming
hyper-parameter search in both mixed open set and traditional situation. We believe there is much room to explore this work with Stacked Generalization.
CHAPTER V

CURRENT AND FUTURE WORK

1 Summary

It has long been held that data mining algorithms have been successfully implemented into variety of domain applications with success. The growing volume of data poses different challenges that have not been addressed by traditional mining algorithms. In this thesis we studied several challenges that are not well studied or documented in Natural Language Processing.

This thesis focuses on ensemble learning methodologies which has recently gained significant attention among data mining communities. A number of popular ensemble learning models often utilize a single classifier to model members generated by different data sets and different parameter subsets. Accordingly these model members contain a common denominator, in that the same data space of the original data set is shared between both members.

In this thesis we focus on an approach where by the data space of each classifier is
distinct. We created this space by ensuring that the data would be available in an incremental means, which means that the model needs to be trained in an incremental fashion as well. As mentioned the drawback of current online NNOs are overcome with our proposed model. In essence overcoming this challenge has proven to solve the problem of text classification when compared to state of the art open set solutions.

We overcome the challenge of dealing with unknown classes during testing that exists when data arrives in different time lines by invoking the idea of online Nearest Non Outlier models using open set problems, that we adapt into clustering methodologies (DBSCAN). We apply our model to solve the problem of text classification and compare with other open set solution.

As mentioned, overcoming the challenges of continuous data means that a solution needs to overcome various smaller sub challenges as will be addressed in this thesis. one of these is that the likely different data spaces during training and testing of each classifier in a domain makes the combination of an ensemble model complex. this is evident when one views state of the art ensemble models such as random forest, extreme gradient boosting which have unique data spaces for both training and testing. However this in not the case for open set models such as NNO and online NNOs that consists of the base classifiers .

The current combination method for the ensemble model, such as Majority vote, Weighted Majority vote suffer low accuracy due to a majority of untrained classifiers. Using historic training may be a solution but it wastes memory. Conversely the logic of the solution presented in this thesis does not waste as many resources and uses the Bayesian method with exploitation and exploration approaches.
2 Future work

While our work tackles several issues in data mining with an ensemble learning, it is not complete. We show that incremental learning plays an important role in real world data mining problems. However, the number of incremental learning classifiers is relative small number compared to current pool of data mining classifiers even with state-of-the-art classifiers. Our belief is that ensemble learning is the best candidate for the task. Our future work is to look at another ensemble of tree base model in two direction: 1) using ensemble of tree base classifier, especially eXtreme Gradient Boosting (XGB) to adapt the need of incremental learning. Particularly, we want to explore our proposed combining method with XGB to give it an ability of dealing with open set problem. 2) studying the strategy of partitioning data space.

To tackle a large scale data problem, we will need to split data space into several subspaces. First, how many subspace is considered optimal that is an open question. Can a number of subspaces fixed or varied? Our intuition is that the number of subsets may be equal to the number of target classes. For example, if our task is to classifier a data with two classes, it may be reason to split into 2. However, there is no guarantee to support this intuition as our study of scalablility ensemble. We think that each data domain may have specific characteristics. Exploring a study of partition on domain specific may be help.

On the other hands, current search of algorithm for each sub region has not been discussed in this thesis even though we provide our published paper related to this problem in publication section. Our current solution has a limitation such that it only if the original dataset has at least 4 features. We want a more general solution for algorithm selection. By
looking into a learning curve may provide alternative solutions.

While most of our work experiments on text documents (NLP domain), we want to study the performance of our proposed in other fields such as computer vision, security, commercial which shows high interest in increment learning approach as well as open set solution. On the other hand, we also see that a feasible work for ensemble learning in distributed environment which is not our focus in this thesis. We hope that your future work will look at distributed system which is in favor to dealing with big data.
Bibliography


Arthur Asuncion and David Newman. Uci machine learning repository, 2007. 74


Robert Cattral, Franz Oppacher, and Dwight Deugo. Evolutionary data mining with auto-


IEEE, 1996. 6, 88


Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2, 2009. 16

Robert Neumayer. *Clustering based ensemble classification for spam filtering*. Citeseer, 2006. 2


Anderson Rocha, Walter J Scheirer, Christopher Forstall, Thiago Cavalcante, Antonio Theophilo, Bingyu Shen, Ariadne Carvalho, and Efstatios Stamatatos. Authorship


Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult.


Xiao Yu, Xiang Ren, Yizhou Sun, Bradley Sturt, Urvashi Khandelwal, Quanquan Gu, Brandon Norick, and Jiawei Han. Recommendation in heterogeneous information networks.


Appendix A

DATASETS OVERVIEW

Businesses often compete with each other during their operation. Businesses need customers and they want customers to remember them as their first choice. Advertising in media is often considered a high expense where a limited budget of medium and small businesses are strained. Having their own website helps businesses reach out to their customers and receive the constructive feedback. Businesses have their own setbacks as the information can spread to large communities. They cannot be used as a start up when they have not actually had any customers.

Businesses often want to know how customers think about the quality of their services in order to improve and make more profits. Businesses often spend a large amount of money in advertisements to promote their services and build their image. In fact, successful businesses can be determined by their profit and their customers’ satisfaction. Many social networks (e.g., Yelp) support a platform for customer feedback to express feelings and thoughts to provide guidelines for other customers to decide on a business.

![Number of Yelp Reviews by number of Stars for 2,225,213 Reviews](image)

Figure 0.1: The trend of reviews posted on Yelp by years
1 Yelp review dataset

![Figure 1.2: Star vs useful votes between elite and non-elite users](image)

Why Yelp? Yelp promotes interaction and the sharing of information among communities where one end is the user and other is the business. Yelp provides a public forum for its registered members to share their thoughts or feeling about the service they pay for (see Figure 1.2). This online forum has also allowed businesses to get customers’ feedback. The number of reviews has dramatically increased from 2006. By getting a rating (more stars), businesses can have a channel for promoting their business to potential customers or can improve their service to keep their customers and achieve their goals. Other social networking is also available in this study, such as Amazon reviews, where the number of review texts are over a million.

In Yelp datasets, an elite feature defines a confident level of reviewers. Elite reviewers usually receive higher “useful” votes on their reviews based on the rating other Yelp users give them. Elite users often post at least 20 reviews and they give 3-4 stars for the service presenting in review. Non-elite users have few reviews and tend to spread their votes in the rating scale (see Figure 1.2). Word usage is distributed around common 3 and 4 stars in the reviews (see Figure 1.3). For example, restaurant goers may want to learn from others’ experiences using a variety of criteria such as food quality, service, ambiance, discounts and worthiness. Food and service play the most important role in customer satisfaction. Ambiance reflects the look and feel of the place. While discounts indicate the promotion strategy to attract customers, worthiness demonstrates the level of satisfaction on the services the customers pay for, as well as the cost for these services.

Yelp users may post their reviews and ratings on businesses and services or simply may express their thoughts on other reviews. Negative reviews from one’s perspective may have an effect on potential customers in making decisions, e.g., a potential customer may cancel a service and persuade others do the same. A rating often presents an overall experience for a specific time of used service, but it may change overtime. In addition, star rating does not explain why customers rate a specific business as a thumbs up or a thumbs down. Analyzing the customers’ comments allows for a better understanding of how a number of individual rating stars can make or break a business.

There is no lack of studies on the datasets from different viewpoints that unveil valuable
Figure 1.3: Distribution of votes

information on a wide range of topics such as the effect of promotional strategies (Byers et al., 2012), the benefit of retrieving knowledge from implicit user feedback (Yu et al., 2013), or the important role of local reviewers (Jindal, 2015).

Surprisingly, most published studies for sentiment analysis on these datasets are using the traditional data mining methods where time is not a main factor. Figure 1.5 illustrates that consumers posted more negative reviews in the earlier time of the Yelp operation (year 2006 -2007) but give more positive feedback from year 2010. Ignoring the time-line may affect the reliability of the results since it considers outdated information as important as the current information. These proposed methods lack the ability to capture rapid changes in knowledge so that the findings are valid only under a static view of business activities.

In the real world, an evolving data source, such as Yelp, may be better modeled with a dynamic approach to accurately reflect continuous changes in business activities. Incremental learning, an example of such a dynamic approach, has the ability to learn a new concept without retraining on the entire dataset. The question is to quantify how customers and businesses are influenced and how business ratings change in response to recent feedback with an incremental learning approach.

We obtain over 2.2 million examples of reviews from the Yelp Data challenge. In the first study, we focus on a subset of businesses in the restaurant category (25071 records contain the ‘restaurant’ word), while in the last study, we consider all businesses with all information. As a result, our first dataset for sentiment reviews includes both chain restaurants (such as McDonald’s, Wendy’s, Steak n Shake, etc.) and independent restaurants. The chain restaurants often share similar ambiance, menu and service but each of its restaurants still use the review system for feedback to attract more customers. In fact, the quality of public reviews have been used as a main metric to measure the actual quality of the independent restaurants.

Yelp reviews contain information in raw text format, indicating a posted date and a number of rating stars. Particularly, Yelp allows the user to vote on each review to indicate how they think of a particular review fit in three categories (useful, cool and funny). The
most number of votes implies a useful rating, while the least number of votes imply a funny rating (see Figure 1.6). Useful votes express how people think about the review on the business. They agree with what a reviewer says. A funny vote makes a review less negative instead of saying “it sucks”. These votes allow consumers to tell other consumers which review they like and why.

We found that the reviews posted by elite users with high useful votes often have 3.5 to 4 average stars and 2.5 to 5 stars for low useful votes (refer to Figure 1.2). For non-elite users, the same phenomenon is observed for high useful values while less than 1000 useful votes are seen as inconsistent. A low number of useful votes may either indicate a positive review (more stars) or a negative review (a few stars), while a high number of useful votes is considered an unbiased review without explicit user status. Funny votes, in fact, are more ambiguous and have a tendency of being an indicator for negative sentiment.

Ratings ranging from 1 (worst) to 5 (best) stars can be given by a reviewer while an average rating star in the same range is generated by Yelp. While 3 stars can be considered neutral, we observe that implicit negative sentiment has become more common over the years so that rating star (Bstar) by Yelp and its own reviewers (UStar) have been moving in the opposite direction. Rating stars by elite users are often close to Yelp ratings while non-elite users often have a different rating with Yelp’s ratings.

When there is a large difference between two ratings, the low number of votes for a corresponding review may represent a vote from an unfair competitor or a disgruntled employee. For example, a reviewer gives 2 stars (refer to 9th row in Table ??) while the business for this review received 4.5 stars from Yelp. To avoid this problem, we exclude
any review from a non-elite user who currently has fewer useful votes than the number of reviews. In addition, these non-elite users should post a rating to more than 1 restaurant. Finally, we retain a total of 113,197 users with 424,521 reviews for 12437 restaurants for this study. We assign a label of positive sentiment for 4 and 5 stars and negative sentiment for 1 and 2 stars for each review. Generating features is a challenge in incremental learning because we do not know all values in advance. In this study, we assume that we have the entire corpus for the purpose of extracting features.

2 IMDB movie review dataset

IMDB is a large site that provides details on a large variety of movie entertainment. Registered users of this site can post their reviews in range of 1 to 10. The IMDB site will assign different credential weights for each reviewer member and compute weighted average scores. We put 18 selected movie genres into three genre domains. We use a total of 14,762 movies of 8 genres from the year of 2000 till 2017. We transform the rating of IMDB into
the same scale (5 stars) as Yelp review dataset such that:

- a new 5 star represents origin 9 and 10 stars
- a new 4 star represents origin 7 and 8 stars
- a new 3 star represents origin 5 and 6 stars
- a new 2 star represents origin 3 and 4 stars
- a new 1 star represents origin 1 and 2 stars

An example of IMDB review is illustrated in 2.7

![Figure 2.7: IMDB’s review of Kong Island 2017](image)

3 Other datasets used in this study

In addition, we also use the following datasets (available in UCI repository)

- **MNIST**: This dataset (LeCun et al., 1998) is a subset of handwritten digits available from NIST. It includes 60,000 training and 10,000 test images for 10 classes. Each image is represented by a 784 dimensional feature vector.

- **Forest Cover Type**: This dataset (Blackard and Dean, 1999) includes 581,012 examples where each is represented by 54 features as belonging to one of 7 classes. Each example gives the forest cover type for 30 x 30 meter cells on the ground collected by US Geological Survey and US Service.

- **Poker**: This dataset (Cattral et al., 2002) includes 1,025,010 examples where each example is represented by 10 features and belong to one of 10 classes. Each card is described by two attributes: suit and rank. The class denotes a value of a poker hand. After removing duplicates, there are 1,000,000 instances.
Appendix B

BAYESIAN INFERENCE

Figure 0.1: Bayesian framework for data mining

In data mining, the task of predicting a future data, given knowledge about data, is known as a regress. In a similar fashion, Bayesian inference exploits a prior probability as a knowledge to predict a posterior probability (a future event) using Bayes’ theorem.

\[
p(H|D) = \frac{p(D|H)p(H)}{p(D)}
\]

(B.1)

where

- \( p(H) \): Prior probability
- \( p(D|H) \): Likelihood of the evidence D if the Hypothesis H is true
- \( p(H) \): Posterior probability of H given the evidence
- \( p(D) \): Evidence (known as margin likelihood)

Intuitively, Bayes’ theorem (refers formula in in Equation B.1) is a common way to compute the posterior probability of an unknown class given that other classes are taken into
account. Prior probability of an event describes the probability of the event before a new
collection of new data. When we know about H, the adjusted probability is the posterior
probability. Since the evidence does not depend on H, Bayes’ theorem can be written as
\[ p(H|D) \propto p(D|H)p(H) \].

Several data mining algorithms are developed in the field of statistics, particularly regres-
sion. These probabilistic models are also known as Bayesian models (Barber, 2012) because
it illustrates the relationship between variables and data in the presence of uncertainty in a
real world using probabilities. Without Bayesian approach, any generating data process is
formulated as a stochastic model with noise. Our objective is to maximize the probability of
observing data. Using a Bayesian approach, a data mining model (classifier) can be defined
as a hypothesis given input data from some distributions (refer Figure 0.1).

A Bayesian inference framework studies the distribution of model parameters and finds
parameters that maximize the probability of observing a given data. We can select the best
model from the family of models for a given dataset. It illustrates a learn to learn concept
that we present in our first topic. We address the challenge of unknown objects by taking into
account of the uncertainty of a real world problem with the help of Bayesian theorem. We
further demonstrate Bayesian inference to optimize the predicting method of an ensemble
model.

Bayesian inference framework provides its support to an ensemble learning approach
with a mixture model. A real world data are more complicated for describing by a single
distribution. Bayesian framework uses a mixture model when data is described by a mixture
of distributions as the way we use ensemble learning to tackle a large data problem. For
example, we can have a student dataset at university A that includes gender, and age. We
may have two distributions of male and female students or traditional and non-traditional
students. While these groups may be represented by Gaussian distribution, the population
of exchange students or non-credit enrolled students does not follow. Since the model,
represented in one such group, has its own parameters, a collection of different models may
provide the best predictive power rather than a single model.

1 Pearson correlation coefficient

Correlation describes a relationship between two random variables, or models. We can mea-
sure the dependence between two quantities by computing Pearson correlation coefficient as
follows:

\[ \rho_{x,y} = corr(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \]  \hspace{1cm} (B.2)

where cov denotes covariance, \( \sigma_X, \sigma_Y \) denotes standard deviation of X, Y respectively.
2 Statistical tests

In an comparative experiment, we are going to make a conclusion that algorithm A is better than algorithm B. The conclusion should be supported from a statistical test, such as Wilcoxon signed test, Anova test, etc. Several statistical tools are used to verify our claims in order to validate the performances. Throughout this study, we compare the performances of several algorithms on multiple datasets. To avoid any conclusion on the comparative results by chance, we follow the recommendation by Demsar et al. (Demšar, 2006) in their study (Demšar, 2006). We use a non-parametric Friedman test to evaluate the results. One advantage of the Friedman test is that there are no required assumptions about the distribution of the data. The classifiers are ranked on N datasets where a first rank indicates the best model. It tests the hypotheses for the comparison as follows:

$H_0$: All classifiers have similar performance vs $H_a$: There is a significant difference in the performance of the classifiers.

Each algorithm is ranked for each dataset separately based on the performance measure. We compute the mean rank of a particular classifier $R_j$ on all datasets as $R_j = \frac{1}{n} \sum_{i=1}^{n} R_{ij}$. Let $S_{\text{total}}$ and $S_{\text{error}}$ be the sum of the squared totals (the variations in the ranks) and the sum of the squared errors, respectively. The Friedman statistic test is defined as $\chi^2 = \frac{S_{\text{total}}}{S_{\text{error}}}$. The test used is a Chi-square with 1 degree of freedom. If the p-value results from this test is small ($<0.05$), it is sufficient to reject the null hypothesis.

If $H_0$ is rejected. The post-hoc pairwise Nemenyi test is recommended (Demšar, 2006) to find where the differences locate. If the average ranks $R_i$ and $R_j$ of two classifiers differ than a threshold. The statistical test is computed as

$$z = \frac{R_i - R_j}{\sqrt{\frac{M(M+1)}{6N}}}$$  \hspace{1cm} (B.3)

where the level of significance is determined by the number of pairwise comparisons $M(M-1)/2$. Let $\alpha$ denotes this level of significance, then the difference is considered significant as computed p-value is less than $\frac{2\alpha}{M(M-1)}$.

Alternative, ANOVA test can be used if we can guarantee the distribution is normal.