Incremental / Online Learning and its Application to Handwritten Character Recognition

by
Rituraj Kunwar

Submitted to the School of Information and Communication Technology, Griffith Sciences, Griffith University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the
GRIFFITH UNIVERSITY
2016

©Griffith University
In real world scenarios where we use machine learning algorithms, we often have to deal with cases where input data changes its nature with time. In order to maintain the accuracy of the learning algorithm, we frequently have to retrain our learning system, thereby making the system inconvenient and unreliable. This problem can be solved by using learning algorithms which can learn continuously with time (incremental/online learning). Another common problem of real-world learning scenarios that we often have to deal with is acquiring large amounts of data which is expensive and time consuming. Semi-supervised learning is the machine learning paradigm concerned with utilizing unlabeled data to improve the precision of classifier or regressor. Unlabeled data is a powerful and easily available resource and it should be utilized to build an accurate learning system. It has often been observed that there is a vast amount of redundancy in any huge, real-time database and it is not advisable to process every redundant sample to gain the same (already acquired) knowledge. Active learning is the learning setting which can handle this issue.

Therefore in this research we propose an online semi-supervised learning framework which can learn actively. We have proposed an "online semi-supervised Random Naive Bayes (RNB)" classifier and as the name implies it can learn in an online manner and make use of both labeled and unlabeled data to learn. In order to boost accuracy we improved the network structure of NB (using Bayes net) to propose an Augmented Naive Bayes (ANB) classifier and achieved a substantial jump in accuracy. In order to reduce the processing of redundant data and achieve faster convergence of learning, we proposed to conduct incremental semi-supervised learning in active manner. We applied the proposed methods on the "Tamil script handwritten character recognition" problem and have obtained favorable results. Experimental results prove that our proposed online classifiers does as well as and sometimes better than its batch learning counterpart. And active learning helps to achieve learning convergence with much less number of samples.

Since in any machine learning problem there is always a risk of encountering
outliers which tampers the model parameters. Therefore we study robustness to incorporate the capability to handle outliers in the learning systems. In this regard K-Largest norm have been proposed for the learning system and its efficacy has been tested across different applications.

Thesis Supervisor: Michael Blumenstein
Title: Professor

Thesis Supervisor: Umapada Pal
Title: Professor
Acknowledgments

I would like to thank my advisor Prof. Michael Blumenstein and Prof. Umapada Pal for their immense support in all respect, throughout my research work. I would like to thank my family and wife for supporting me during my lows which kept me going.

I would also like to thank all my colleagues, especially Adel and Abhijit for the fruitful discussions. I am also very thankful to the Griffith University’s Higher Degree Research staffs for their seamless support.
Statement of Originality

This work has not previously been submitted for a degree or diploma in any university. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

Signed:
Rituraj Kunwar
24th April, 2016.
## Contents

1 **Introduction** 17  
1.1 Definition of Incremental and Online learning 19  
1.2 Incremental/Online Semi-Supervised Learning 21  
1.3 Active Learning 22  
1.4 Motivation 23  

2 **Literature** 27  
2.1 Incremental / Online learning 27  
2.2 Online Approaches for Stable Concept Learning 31  
2.2.1 Modified Adaptive Boosting (MadaBoost) 32  
2.2.2 Online Bagging 33  
2.2.3 Online Boosting 34  
2.3 Online learning algorithms which can handle concept drifts 34  
2.4 Semi-Supervised incremental / online learning 35  
2.4.1 Self-Training / Learning 35  
2.4.2 Probabilistic generative models 37  
2.4.3 Co-Training / Multi View Learning 40  
2.5 Conclusion 41  

3 **Incremental/Online Semi-supervised Learning Using Bayesian Network** 43  
3.1 Introduction to online Tamil handwritten character recognition and the proposed Naive Bayes Approach 44
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2 Online Random Naive Bayes</td>
<td>48</td>
</tr>
<tr>
<td>3.2.1 Naive Bayes:</td>
<td>48</td>
</tr>
<tr>
<td>3.2.2 Random Naive Bayes:</td>
<td>49</td>
</tr>
<tr>
<td>3.2.3 Online RNB:</td>
<td>50</td>
</tr>
<tr>
<td>3.3 Online Semi-Supervised Learning: Naive Bayes Approach</td>
<td>51</td>
</tr>
<tr>
<td>3.4 Experimental Evaluation</td>
<td>53</td>
</tr>
<tr>
<td>3.4.1 Dataset and Challenges</td>
<td>53</td>
</tr>
<tr>
<td>3.4.2 Experimental settings and experiments:</td>
<td>55</td>
</tr>
<tr>
<td>3.5 Discussion I</td>
<td>55</td>
</tr>
<tr>
<td>3.6 Online Semi-Supervised Learning: The Online Random Bayesian Network Approach</td>
<td>61</td>
</tr>
<tr>
<td>3.6.1 Bayesian Network</td>
<td>62</td>
</tr>
<tr>
<td>3.6.2 Random Bayesian Network</td>
<td>64</td>
</tr>
<tr>
<td>3.6.3 Batch / Offline Learning of RBN</td>
<td>65</td>
</tr>
<tr>
<td>3.6.4 Online RBN:</td>
<td>68</td>
</tr>
<tr>
<td>3.7 Online Semi-Supervised Learning: Bayesian Network approach</td>
<td>70</td>
</tr>
<tr>
<td>3.8 Experimental Evaluation</td>
<td>72</td>
</tr>
<tr>
<td>3.9 Discussion II</td>
<td>73</td>
</tr>
<tr>
<td>3.10 Conclusion</td>
<td>76</td>
</tr>
<tr>
<td>4 Incremental Active Semi-supervised Learning</td>
<td>79</td>
</tr>
<tr>
<td>4.1 Active Learning settings</td>
<td>80</td>
</tr>
<tr>
<td>4.1.1 Membership Query Synthesis</td>
<td>80</td>
</tr>
<tr>
<td>4.1.2 Selective or Sequential Sampling</td>
<td>81</td>
</tr>
<tr>
<td>4.1.3 Pool-based Sampling</td>
<td>81</td>
</tr>
<tr>
<td>4.2 Query Strategies</td>
<td>82</td>
</tr>
<tr>
<td>4.2.1 Uncertainty Sampling</td>
<td>84</td>
</tr>
<tr>
<td>4.2.2 Query-By-Committee</td>
<td>86</td>
</tr>
<tr>
<td>4.2.3 Expected Model Change</td>
<td>88</td>
</tr>
<tr>
<td>4.2.4 Expected Error Reduction</td>
<td>89</td>
</tr>
</tbody>
</table>
5 Robust Prediction with K-Largest Norm

5.1 Introduction

5.2 Dataset

5.3 Introduction to the $L_{\text{lgst,k}}$ norm

5.4 Linear Regression

5.4.1 Objective functions

5.5 Support Vector Regression (SVR)

5.6 Artificial Neural Networks (ANN)

5.7 Applications

5.7.1 High Performance Concrete (HPC) Strength Prediction and Challenges

5.7.2 Forest Fire Prediction and Challenges

5.7.3 Wine Quality Prediction and Challenges

5.8 Experimental Results

5.8.1 Flood prediction

5.8.2 High performance concrete strength prediction

5.8.3 Forest fire prediction

5.8.4 Wine Quality prediction

5.8.5 Robustness of L-Largest

5.9 Discussion and Conclusion
List of Figures

3-1 Block diagram representing online semi-supervised learning . . . . . . 44
3-2 Plotted accuracies of RNB and Online RNB shows that as the number of training samples increases, the gap between the accuracies of RNB and ORNB reduce monotonically. The accuracy gap increases at the beginning showing better learning by the offline RNB with few training samples. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56
3-3 Plotted accuracy of RNB and Online Semi-Supervised RNB shows that unlabeled data boosts the accuracy sharply when there is a smaller number of labelled data samples and as the number of labeled sample increases, the accuracy of online semi-supervised RNB converges to offline RNB. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
3-4 Plotted accuracy shows that the Online Semi-Supervised RNB accuracy increases with unlabeled training samples. The increase in accuracy is a maximum for the classifier, which was pre-trained with a smaller number of labeled samples. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58
3-5 Plotted accuracy shows that as the value of alpha increases (memory of classifier becomes smaller) the generalization capability of the classifier goes down. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 59
3-6 Plotted accuracy shows that if number of unlabeled data is not huge then for better incremental semi supervised learning we should give relatively more weight to unlabeled data but if excessive weight is given then accuracy suffers a small dip. Number of unlabeled data used 10 per class . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
3-7 Plotted accuracy shows that if number of unlabeled data is huge then
for better incremental semi supervised learning we should give relatively less weight to unlabeled data otherwise accuracy suffers a huge dip. Number of unlabeled data used 350 per class.

3-8 Bayes net structure for Naive Bayes.

3-9 Augmented Bayes net structure of Naive Bayes.

3-10 Block diagram representing the batch learning (MLE) of RBN.

3-11 Plotted accuracies of RBN and Online RBN (ORBN) shows that as the number of training samples increases, the ORBN seems to perform better.

3-12 Plotted accuracy of RBN and Online Semi-Supervised RBN shows that unlabeled data boosts the accuracy sharply when there is a smaller number of labeled samples and as the number of labeled sample increases, boost in accuracy reduces.

3-13 Plotted accuracy shows that the Online Semi-Supervised RBN accuracy increases sharply with unlabeled training samples. Increase in accuracy is maximum for the classifier, which was pre-trained with smaller number of labeled samples.

3-14 Plotted accuracy shows the superiority of variable weighting factor over constant weighting factor.

4-1 Comparison of different query strategies accuracy.

4-2 Learning converging much faster by injecting unlabeled data while performing active learning.

4-3 The constraint of processing one unlabeled sample at a time delays the convergence but at the same time it is better than the base SS+RS and plain incremental learning in terms of fast convergence and accuracy.

5-1 Comparison of flood forecasting, focus on historical peak.

5-2 Comparison of flood forecasting methods, focused on the fourth highest peak.
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-3</td>
<td>Comparison of flood forecasting methods, focused on the seventh highest peak</td>
<td>125</td>
</tr>
<tr>
<td>5-4</td>
<td>Comparison of flood forecasting methods, focused on both low and high flows</td>
<td>128</td>
</tr>
<tr>
<td>5-5</td>
<td>Comparison of concrete strength prediction methods</td>
<td>129</td>
</tr>
<tr>
<td>5-6</td>
<td>Comparison of concrete strength prediction methods</td>
<td>130</td>
</tr>
<tr>
<td>5-7</td>
<td>Comparison of concrete strength prediction methods</td>
<td>131</td>
</tr>
<tr>
<td>5-8</td>
<td>Comparison of forest fire prediction methods</td>
<td>132</td>
</tr>
<tr>
<td>5-9</td>
<td>Comparison of wine quality prediction methods</td>
<td>133</td>
</tr>
<tr>
<td>5-10</td>
<td>Prediction error in the presence of zero-mean Gaussian noises of varying standard deviation added to each sample</td>
<td>134</td>
</tr>
<tr>
<td>5-11</td>
<td>Prediction error in the presence of 5 extreme outliers and zero-mean Gaussian noises of varying standard deviation added to each sample</td>
<td>135</td>
</tr>
<tr>
<td>5-12</td>
<td>Prediction error in the presence of outliers of varying strength</td>
<td>136</td>
</tr>
</tbody>
</table>
List of Tables

| 3.1 | Table below show the detail of each classifier. 20 random features were used to create 50 different classifiers to form the ensemble (RNB and ORNB). DTW-based classification use 15 hand-picked samples for each class. | 56 |

| 3.2 | Table below show the detail of each classifier. 20 random features were used to create 50 different classifiers to form the ensemble (RBN and ORBN). In case of SSORBN L and UL stands for labeled and unlabeled respectively. | 74 |
Chapter 1

Introduction

Machine Learning is an area within Artificial Intelligence which involves the study and development of computational models capable of improving their performance with experience and of acquiring knowledge on their own [107]. It is generally used for classification, recognition, prediction, planning etc. There are two main phases of machine learning, namely learning (training) and testing. Usually, learning is conducted in offline mode. They are first trained to perform a particular task and subsequently the trained system is tested by asking it to perform that particular task. The bottleneck of these systems is that, no testing can be performed during the training phase, and no training is possible while the testing is taking place (i.e. the learner cannot improve once its learning phase is complete).

Offline learning works fine in an ideal scenario i.e. where there is no change in the underlying distribution of the input with time. However, this does not often hold in real time problems that we intend to address using machine learning [110]. For instance, consider an information filtering system which predicts the readers reading preferences. Preference of a reader gradually changes with time and the system which does not have the capability to update its learning with time will not be able to perform optimally [137]. Similarly a recommender system for "smart" advertising will not perform well if it is trained offline since the clients choices may change depending on state of the economy or on new products introduced, etc. List of such real time problems (where we need our learner to adapt with time) are endless, few among
them are, spam detection [113], market analysis [137] [3] and computer security [55].

In contrast to offline learning, incremental/online learning can be simultaneously trained and tested. Precisely, it need not stop performing its task (i.e., prediction or classification) if the learner has to update its learning. Learning parameters can be updated as soon as the new training data is available. This leads to the creation of a never ending learning process which can adjust itself even if the environment changes and can perform learning while performing task.

We need robust incremental/online learning systems because in the last few decades information and communication technology has developed by leaps and bounds leading to exponential growth in data generation. The demand for data analysis methods has risen proportionally. As an alternative to the expensive computer cluster frameworks for large-scale analysis [38], incremental/online algorithms that build approximate models on continuous and possibly infinite data streams have been developed. In the real-time domains like finance, medical, web data analysis (or any data streaming domain), it is required to collect huge data to create a reliable predictive model. Since huge data collection is expensive as well as time consuming, we land up collecting data in batches, which incremental and online methods can very efficiently handle whereas offline approach is a debacle in scenarios like these.

The earliest formal description of an algorithm with the ability to learn from new data was the description of learning in the limit by Gold [59], which authors in [144], considers being the ideal case of incremental learning. In the initial description, a learning machine could use all the information seen so far to yield new hypotheses. This ideal scenario has been theoretically compared to more realistic scenarios where the past information is partly accessible or is completely inaccessible [76]. Following this research, Lange [90] concluded that, when noisy data is used, incremental learners with restricted access to data are as powerful as unconstrained learning algorithms.

A naive way to utilize newly generated data would be to create a new learner using both old and recently generated data but this will be very expensive in terms of time and computation. Above solution may be impractical if old data has limited access or no access at all. Other more efficient way to handle this is by making relevant changes
in the present trained learner itself to adapt to the changed conditions. Alternatively researchers have tried retaining a relevant subset of old data. Previous work on partial memory learning [101], boundary methods and maximum margin methods [129], [147] validates the effectiveness of above approach. Several methods have been proposed in the last two decades to equip the learner with incremental learning capability. Various incremental techniques were designed for decision tree learners [134], [158] and [168] that were successfully used on data streams [54]. Similarly numerous incremental learning techniques have been proposed for neural network based learner as well, where the technique is either concerned with adaptation of structure [121] [163] [109] and [42] or weight [159]. Incremental version of principal component analysis has also been proposed in literature where they update the projection matrix incrementally [25]. And on the similar lines incremental Fisher’s Linear Discriminant Analysis has also been developed [79] [73].

The critical assumption on which most of the incremental learning algorithms are based upon is that previous data is completely or partially accessible. Based on this assumption, to handle streaming data they apply the time windowing technique of either fixed or variable size [161] [137] [125]. Others have handled streaming data by weighting models in the ensemble [109] [154] or by weighting the data [81] or by retaining only the relevant subset of previous data [101] [147]. We assume for our research that we have no access to the previous data, thus making the algorithm capable to handle the scenario where old data is inaccessible.

1.1 Definition of Incremental and Online learning

Several definitions have been given by different authors in the past. Many instances can be found in the literature where these two terms have been used interchangeably. And at the same time, in some past work we can find instances where these two terms differ in their meaning. The general characteristics of incremental learning as stated by Polikar et al. [121] are 1) it should be able to learn additional information from new incoming instances. 2) It should not require access to the original data,
used to train the existing classifier. 3) It should preserve [useful] previously acquired knowledge (that is, it should not suffer from catastrophic forgetting). 4) It should be able to accommodate new classes that may be introduced with new data.

Similarly several definitions of online learning can also be found in the literature, few are listed below: ... online learning algorithms take as input a single labeled training instance as well as a hypothesis and output an updated hypothesis. Thus, given a sequence of training instances an online algorithm will produce a sequence of hypotheses. (Fern et al. [45])

Online learning algorithms process each training instance once "on arrival" without the need for storage and reprocessing, and maintain a current hypothesis that reflects all the training instances so far. (Oza et al. [119])

Terms like incremental learning, real-time learning, online learning and continuous learning have been frequently used interchangeably in literature to indicate online learning. Therefore it is necessary to explicitly mention the definition of incremental learning followed in the presented work. It is a combination of definition given by Fern et al. [45] and Oza et al. [119]:

"Online learning algorithms process each training example once "on arrival" without the need for storage and reprocessing [45]. They take as input a single labeled training example as well as a hypothesis and output an updated hypothesis [119] of the function being learnt."

It is worth mentioning the subtle difference between incremental and online learning: we can easily conclude from the above definitions that online learning can be considered as the special case of incremental learning. Most of the criteria for each learning paradigm are common except that incremental learning algorithms are capable of processing a larger number of samples at once (not necessarily one sample at a time as it is in online learning case). It is possible that the incremental learning may process considerably larger chunk of data several times and for which it can even use offline learning method.

Therefore a method may be online according to one definition but it may not satisfy the required conditions of other definition. For example online back propagation
algorithm for MLP in [14] cannot be considered as online learning as per our definition because, even though the weights are updated after processing each training sample, it is possible that same sample will be processed more number of times (as number of epochs are increased). But in case of number of epochs is one or if one sample on arrival is used to update the weights several times and discarded then in that case it will satisfy our online learning definition [118].

The definition highlights the general problem of classification models called the stability-plasticity dilemma [63]. This dilemma reveals that some information may be lost when new information is learned (gradual forgetting) and highlights the difference between stable classifiers and plastic classifiers. On one hand, a completely stable classifier will preserve existing knowledge, but it will not incorporate any new information. On the other hand, a completely plastic classifier will learn any new information without preserving any previous knowledge. The latter case is also known as catastrophic forgetting [103] [124] and it happens when an already trained model learns a new set of patterns completely erasing its previous knowledge. Essentially, the challenge is how to design a learning system that is sensitive to new input without being radically disrupted by such input.

1.2 Incremental/Online Semi-Supervised Learning

A close look at the way we humans learn will make us realize that we learn by example [8]. A child’s initial (as well as further) learning is based on the observation of the surrounding world. And this may happen in two possible ways, either unsupervised or supervised i.e. under some guidance of teacher or parents etc. But on further cerebration it can be easily concluded that supervised and unsupervised are the two extreme ways in which human learns in an environment. It is obvious that human learning process do not adopt any of the above extreme ways to learn instead it is an intelligent combination of both which can be considered as semi-supervised. Mostly when a child learns, the child not only rely on instructions from parents or teachers, but he or she also examines the surrounding world without supervision simultaneously
This observation implies that machines can also learn using labeled samples (supervised) as well as unlabeled samples (unsupervised) at the same time.

Recently, semi-supervised learning (SSL) [169][26], has attracted much research attention. This is mainly because, through SSL, the labeling effort can be kept to a minimum, while one can take advantage of a large number of unlabeled samples, which in turn increases the coverage of the sample space and therefore, potential mismatches between training and testing conditions can be reduced. In this research we focus mainly on online semi-supervised learning because unlabeled data are often collected sequentially. For instance handwritten characters are continuously generated during use, or people usually upload photos after taking couple of shots instead of waiting for thousands of images to get collected.

The main concern for semi-supervised incremental learning is how to judiciously incorporate unlabeled examples into the learning process while enhancing initially learned concept models over time. For e.g. one of the usual approach in past have been to select the unlabeled sample which can be classified with high confidence by the current model. The current model is then further updated with the selected sample assuming the class to be the predicted class label. This approach, called a confidence score based method, is based on the fact that the probability to generate incorrect classification outputs is minimized if such samples are picked. However, we claim that these samples might not be good candidates to reduce modeling error because they are similar to the image samples already seen in the initial training phase. As a result, even after incorporating a large number of unlabeled samples, we might end up with a sub-optimal concept model, which we refer to as the sub-optimality problem of a confidence score based method.

1.3 Active Learning

One of the most intuitive (closest to the way human learns) way to learn would be to learn actively. In active learning paradigm the learner invoke queries to the oracle (user or teacher or human annotator) to provide the missing labels of the unlabeled
data. Given that posing such queries to the oracle is expensive, the main challenge of learning in active learning setting is to minimize the number of queries fired and at the same time achieving the best performance of the learner. Comparison of the active learner performance is made with the passive learner which has all the possible training data. The purpose of the active learning setup is to use only the most informative labeled samples (thus avoid the redundant labeled training data) to conduct the best supervised learning.

Various methods have been proposed in past to perform active learning. Active learning algorithms can be categorized based on its query strategy. Few common query strategies are uncertainty sampling, query by committee, expected model change, expected error reduction and variance reduction etc. The ultimate aim of different strategies is to increase the performance of the system with minimum query. For example one of the earliest proposed active learning strategies was based on uncertainty-based sampling [95]. Here, the system demands class information to user for unlabeled samples that the current model is least confident about.

In order to maximally improve the learning model by using unlabeled data in online setting, we intend to perform online semi-supervised learning actively. The efficacy of conducting online semi-supervised learning actively has been recently demonstrated in [62] [52] [146].

1.4 Motivation

A close observation of the environment around us and the way we humans learn made us realize that there is a huge gap between the way we learn and the way we make machines learn. This research is intended towards bridging some of the gaps between the ways a human learn and methods used for machine learning. Let us consider the way children learn, they do not learn in a batch; instead their learning happens continuously over time. Essentially it is a never ending learning process. In order to make machine learn continuously with time we need to resort to online learning.

A number of authors have addressed this problem of incremental and online learn-
ing in past but most of the work suffer from one or the other kind of constraint, for instance partial access to previous data [81] [161] [54] [154] [137] or catastrophic forgetting or inability to handle one sample at a time etc. In this research we consider that our online learning method will not have any access to the past data and it will not get data in batches instead it will have access to one sample at a time which will be discarded after use.

A further analysis of ways of human learning will make us notice the following striking parallel: children learn concepts from a combination of parental feedback (labeled data) and the unsupervised observations of the world around them (unlabeled data). This manner of learning is known as semi-supervised learning. This style of learning has gained immense popularity in recent past since in this method minimal human intervention (as very few labeled samples are required for training) is required and performance wise it is on par with batch learning methods or sometimes even better on some dataset.

Once again considering children as a learning case we can easily observe that they use both labeled (supervised learning) and unlabeled (unsupervised learning) in sequence (not in batch). So it is easy to conclude that human learning lies somewhere in between pure supervised and unsupervised learning, it can be considered as semi-supervised learning which takes place continuously.

In past semi-supervised learning has been often used in offline mode but in this research we use semi-supervised learning in online mode. Literature indicates that in past researchers have worked on incremental / online learning, semi-supervised but the combination, online semi-supervised has not been studied throughly. Online semi-supervised is an interesting topic to be studied and it seems to be like a step forward towards reducing the gap between human and machine learning.

Another important aspect of human learning is their discretion to demand for label for a particular sample (while learning) which they believe to be most challenging or informative to improve their learning model. This paradigm of learning is referred as to "active learning" in literature. Furthermore, it is natural to incorporate active learning [53] in this setting; upon receiving an unlabeled data point, the learner
may request the label from an oracle. Many real-world learning tasks fit nicely into this framework, such as classifying images collected by a surveillance camera, or categorizing blog posts and tweets in real-time as they emerge on the social Web. This combination of online learning, semi-supervised learning and active learning has seldom been studied in past [60] [146].
Chapter 2

Literature

This chapter covers previous work done in the incremental/online learning domain. This section focuses on the following area related to continuous learning:

1. Incremental and online learning.

2. Online Approaches for Stable Concept Learning.

3. Online learning algorithms which can handle concept drifts.

4. Incremental / online semi supervised learning.

5. Summary.

2.1 Incremental / Online learning

Various methods have been proposed in past to conduct learning in continuous manner, but most of the method require small, big or entire data set to be present for learning instead of processing one sample at a time. Few examples which assume complete or partial availability of old data are [137] [69] [123] [136]. These are ensemble based approaches where a new member of the ensemble is formed at the arrival of new chunk of data and their outputs are weighted according to their accuracy on the recent data. The authors in [55] report, the use of un-weighted ensemble improves
the performance in cases where the new data belongs to the different concept (from the most recent training data).

Approach in [44] for stream mining explicitly considers one of the following four combinations of situations may occur when working with changing environments: new data is or is not sufficient by itself and there is or there is no concept drift. So, instead of always increasing the ensemble size, whenever a new chunk of data is available, Stream Miner explicitly compares the accuracy of a new classifier which learns only the new data chunk, the old classifier updated with the new chunk of data and a new classifier which learns both the new data and a subset of the old data. The problem with this strategy is that it makes it necessary to store an unlimited number of old examples for further training. This number can become very large, increasing the running time and memory requirements. Besides, as old examples are not kept in case they are not beneficial, there is no strategy to benefit from recurrent concepts.

Recurrent concept is explicitly dealt in [123] and [47]. Work in [47] creates a new classifier to learn each new chunk of data and augments the set of features of the new chunk with P features, generated by the predictions that the P previous classifiers would have made. In this way, knowledge of previous base learners can be used for the training of the new base learner. Authors in [123] explicitly maintain a global set of classifiers which can be used if they are accurate enough on the current chunk of data. New classifiers are included only if neither the ensemble formed by the classifiers in the global set nor the classifiers by themselves are considered accurate enough.

One of the most critical questions (can be a bottleneck) in designing an incremental learning system is to what will be a reasonable data chunk size in general to capture the concept drift. It is obvious that if the chunk size is small, a rapid concept drift can also be tackled but the new classifier that is formed using the small chunk of data cannot be reliable for obvious reasons [137]. Another constraint that a typical incremental learning setting by default imposes is that a new classifier cannot be made until full new chunk of data is not received. Precisely continuous learning faces stability-plasticity dilemma [63]. Incremental learning gives more attention to plasticity (as a completely new classifiers is generated for new data chunk) but at the
cost of stability and opposite is true for online learning.

Besides the need to determine the chunk size, the non-continuity in handling drifts and the lack of strategy to deal with periods of stability, incremental learning approaches also suffer from delayed system update (it is necessary to wait for a whole new chunk of data in order to update the system) and higher memory and time requirements (the new chunks of data have to be stored in memory and are usually processed many times). As there are some applications which present inherent incremental environments and allow heavier memory and time requirements, incremental learning is still an important area of study. However, besides the fact that incremental learning approaches present the problems explained above, online approaches can be used to solve both online and incremental problems. So, the setup for the presented experiments will be online learning.

We next give an overview of the different popular incremental/online learning methods proposed in the literature to work in the presence as well as in the absence of concept drifts.

**Fuzzy ARTMAP:** This is a neural network based structure and it is one of the earliest and popular methods used in incremental learning. The fuzzy ARTMAP [24] has two fuzzy ART modules that are linked via an inter-art module known as "map field". The map field is used to form predictive categories for learning class association. Fuzzy ARTMAP will generate new decision clusters in response to new input patterns that are sufficiently different from previously seen instances. The 'sufficiently different' patterns are controlled using a free parameter of ARTMAP known as the vigilance parameter. ARTMAP is sensitive to the vigilance parameter especially in presence of significant noise in the training data. Using stability and match tracking, fuzzy ARTMAP, automatically constructs as many categories as are needed to learn any static training set to 100%. Thus, fuzzy ARTMAP may overfit leading to poor generalization.

**Learn ++:** This is one of the most notable families of incremental learning algorithm which was first introduced by Polikar et al. in [121] and later upgraded by few other authors for e.g. [109]. It creates multiple classifiers to each data chunk
presented to the system. Inspired by AdaBoost [49] [48] for each chunk, the training set for each base learner is created by sampling examples according to a distribution of probability. Like AdaBoost, Learn++ maintains a distribution of instance weights; however Learn++ does not update the weights in the same manner as performed with AdaBoost. In AdaBoost, the distribution of probability is built to give higher priority to instances misclassified by the last previously created classifier whereas Learn++ uses the ensemble decision, rather than the decision of the latest classifier. When a new dataset arrives, the distribution is re-initialized by evaluating the entire ensemble and initializing the distribution. Pros: Learn++ does not have to access the previous data chunks and it demonstrate considerable improvement at generalization when compared with fuzzy ARTMAP on common databases. Cons: Problem is that a new set of classifiers is created for each new data chunk. So, the ensemble size can become extremely large considering lifelong learning.

**Evolutionary Approach:** Authors in [138] developed an approach to incremental learning using evolving neural networks. An evolutionary algorithm is used to evolve some MLP (Multi-Layer Perceptron) parameters, such as the learning rates, initial weight distributions and error tolerance. The evolutionary process aims at evolving the parameters to produce networks with better incremental abilities. In each generation, the neural networks with the parameters codified by the evolutionary algorithms are trained using first the training examples of a particular chunk of data, then using another one and so on. While a neural network is being trained with a particular data chunk, the other chunks are not used in the training. However, each generation uses all the available chunks. The error produced by testing the neural networks with a validation set is used as a fitness measure. Pros: This proposed method can handle addition of new class and it has been shown to outperform Learn++ and MLP on UCI dataset. Cons: The network on its own does satisfy the criterion of incremental learning as it is using only one chunk of data at a time but evolutionary algorithm at the same time is not because it uses all the chunks in a generation of evolution.
2.2 Online Approaches for Stable Concept Learning

Several online methods have been proposed in past which uses ensemble technique to handle online scenario but they don’t have explicit procedure to handle concept drift. As explained in [45], online algorithms which use ensemble classification techniques can be divided into sequential/parallel generation approaches and single/multiple-update approaches. An algorithm is considered to take a sequential generation approach "if it generates the ensemble member one at a time, ceasing to update each member once the next one is started". Otherwise it is considered to take a parallel generation approach. Similarly, an online ensemble algorithm is considered to take a single-update approach "if it updates only one ensemble member for each training instance encountered". Otherwise, it takes a multiple-update approach.

According to [45] sequential generation and single update approaches carry some obvious but fundamental problem. If the ensemble system is a sequential generation then one need to decide when a member of the ensemble should stop being trained, but answer to this question is not straight forward. In case the approach is single update, then it is possible the system convergence to the target concept will be slow since only one member of the ensemble system is updated at a time. On the contrary there are offline learning ensemble based methods (for e.g. boosting and bagging) which use single training example to update all the members of the ensemble. According to authors of [45] this is a necessary property and online system must have this in order enable the ensemble system to converge faster.

There are approaches which were originally developed for offline learning but those might also be used for online learning, e.g., Boosting by Filtering [131] and Adaptive Boosting (AdaBoost) [49], [48]. In order to do so, a certain number of examples could be selected to train a first classifier. After that, a second classifier would be created by filtering new examples based on the previous classifier. Similar process could be applied to create more classifiers. However, these would be sequential-generation single-update approaches and would discard large amounts of data in the process of drawing the desired distribution of data to each ensemble member. So, they would
need a long time to get new training examples that can be used for learning.

2.2.1 Modified Adaptive Boosting (MadaBoost)

MadaBoost [41] is a variant of AdaBoost that can be used for filtering framework with smaller execution time. In order to reduce the execution time weights that are attributed to the training samples were bounded. This reduced the time necessary for the filter to choose an example to be used for learning. However, MadaBoost is still a sequential-generation single-update approach. Among other examples of sequential-generation single-update approaches are Pasting Small Votes [20] and Streaming Random Forests [1].

The accuracy become worse for sequential generation and single update approaches in the presence of concept drifts but the problem can be ameliorated by using some additional drift detection procedure. The ensemble system whose members keep increasing with time will have difficulty in adapting to the new concept and similarly if the system is single update then it is very likely that the system will have difficulty in recovering from the drift as new sample is updating only one member at a time. The authors in [83] emphasizes the use of explicit concept drift detector even in parallel generation and multiple update approach in order to get rid of slow adaptation to new concept.

Parallel generation and multiple update can be further sub divided into two groups depending on whether same or different sequence of training sample is used by all the ensemble members. Methods in which all the ensemble members receive the same sequence, they have different provision to assign different weight to the members in an online manner. Similarly others [85] [92] employ different methods to create different ensemble members in online manner. Above approaches will be parallel generation and multiple update but still they have a constraint that the model diversity have to be built a priori rather than that emerging from the data itself.

Similarly in parallel generation and multiple update there are approaches which send different sequences of samples to the members of ensemble so that model diversity do not have to be built a priori. In this case the major concern is to how to determine
this sequence in an online way. Several works in this regard are various online versions of bagging and boosting proposed by different authors, for e.g. online version of bagging [45] [119] and [118] and online version of boosting [45] [119] [118] [92].

2.2.2 Online Bagging

The principle behind online bagging by authors of [119] [118] is that when the number of training examples tends to infinity, in offline bagging [18] each ensemble member contains a certain number (M) of copies of each original training example. And author of [18] claims that the distribution of M can be approximated by Poisson(1) distribution. Online bagging’s sampling distribution is Multinomial (N, 1/N), for N draws. Online bagging’s sampling distribution is Poisson(N), for N draws. Authors of [118] proves that, as the number of examples N tends to infinity, the probability generating function of online bagging’s sampling distribution converges to the same function as the probability-generating function of online bagging’s sampling distribution. That means that they have the same expectation when N tends to infinity.

Therefore, as soon as a training example is available for each ensemble member a weight M is drawn is from a Poisson(1) distribution and the example is learnt in association to this weight. Poisson(1) produces a discrete value, so according to the algorithm each training example will be used for training each member Poisson(1) time. Instead of using a training sample one with a weight M, it is used M time to train a member of the ensemble. In this way algorithm avoids the use of weights in the algorithm. The classification is in the usual way, by unweighting majority vote, exactly the way it is done in the offline bagging.

Fern et al. in [45] proposed a very similar online bagging algorithm. The only difference is that it allows the choice between either using a Poisson(1) distribution or associating each example to W = 1 with probability Pu and W = 0 with probability (1 - Pu), where Pu is chosen by the user. Lee et al. in [92] proposed a lossless2 online Bayesian bagging algorithm. The algorithm works in a similar way to Oza et al. in [119] [118] and Fern et al. in [37], but it draws weights from a Gamma(1,1) distribution. As the weights provided by Gamma(1,1) are continuous-valued on (0,1),
the base learning algorithms need to be able to cope with weighted training examples, instead of receiving a certain number of repetitions of training examples.

2.2.3 Online Boosting

Similar to online bagging, Oza et al. in [119] [118] has proposed online boosting but here the author uses Poisson($\lambda$) instead of Poisson(1). Parameter $\lambda$ is associated with each training example and its value is increased or decreased when presented to the next ensemble member depending on whether the last ensemble member has misclassified or correctly classified respectively. To classify a sample ensemble uses weighted majority vote method, where weight of a member of the ensemble depends on the accuracy of that particular member.

Fern et al. in [45] has proposed online boosting on the similar line but uses a different method to calculate the weights for each training sample. The method used to weight each training sample is borrowed from the work by Breiman in [19] (i.e. Arc-x4) instead of Adaboost. The weight is calculated as $1 + wt^4$, where $wt$ is the number of previous ensemble member which has misclassified the new training sample. Though the method used to weight a training sample is slightly different from the one used in Adaboost but the basis is same that is to increase the weight of the training sample which has been previously misclassified by an ensemble member.

2.3 Online learning algorithms which can handle concept drifts

In general, learning algorithms use the classification accuracy of the current classifier to handle concept drifts. Researchers in past have either used a dedicated procedure to detect the concept drift [115] [116] [33] [10] and then take appropriate action to handle it, or they implicitly handle it [83] [84]. The dedicated procedure detects drift with some confidence and accordingly either the present system is discarded (if the drift is detected or confirmed) and replaced by the new one or the current system is retained.
The learning system which has explicit drift detection method is recommended for the scenario where sharp drifts are expected and quick response is needed. This approach would have been the best way to perform learning in the presence of concept drift but it sometimes suffers from the non-accurate drift detection. The approaches where the drifts are detected implicitly, generally associates some weight to each ensemble member which is based on their respective accuracy. Some examples of this approach may have provisions to prune or discard some of its members. The limitation of this approach is that it generally suffers from slow recovery rate. For analysis and information about the online learning methods that can operate in the presence of concept drift interested readers can refer \[106\]

2.4 Semi-Supervised incremental / online learning

While the approaches reviewed in the preceding sub-section outline a wide range of solutions to incremental and online learning. All the algorithms assume that the training dataset is completely labeled i.e. no information is missing. However, there are many cases, like in a real life scenario where there are chances that nature of data may change with time (dynamically changing environment) and it is potentially impossible to keep the human annotator available all the time (as its expensive). In such cases we may have to resort to unlabeled data for performing learning. The paradigm, in which learning process uses data with missing information in addition to few or many training samples with no missing information, is known as semi-supervised learning. In our study the missing information would be the class label of each of training sample. Next we discuss few of the popular learning approaches in semi-supervised incremental / online learning.

2.4.1 Self-Training / Learning

This is the most naive way of addressing the issue of conduction learning in semi-supervised manner. In this approach we simply trust the outputs of the current classification model. Precisely learning model uses its own prediction to teach itself,
therefore this model is also known as self-teaching. More formally, suppose $\delta(x; \theta_t)$ is a decision rule for $x$ learned with $L_t$, a labeled set at time $t$. The prediction results given by $\delta(x; \theta_t)$ for all $x \in S_t$ are then treated as clean, ground-truth class labels. But unfortunately, generally the number of initial labeled samples used is very small so consequently the generated labels are often noisy. Therefore when such noisy class labeled augmented data set is used for retraining the model to produce model parameters at time $t+1$, $\theta_{t+1}$, then it is very likely that the model parameters will diverge (i.e. the current model will perform worse than the previous). The chance of model parameters to diverge is higher when the classes are not well separated in the feature space.

In order to avoid possible divergence of model parameters, a measure of confidence score for the current model decision $\delta(x; \theta_t)$ on sample $x$ has been explored in [139]. The logic behind this approach is that it is very likely, that the predictions made with high confidence score would be correct. So, the unlabeled samples with very high confidence score would be merged with the labeled training data with their respective labels as predicted by the model. The drawback of this approach of learning is that it often leads to suboptimal solutions, even after adding a substantial number of unlabeled data. The cause of this suboptimal solution is that all the unlabeled samples added to the training data were very similar to each other ($\theta_t = \theta_0$ for all $t$, $\theta_0 =$ initial model parameter vector). Therefore unlabeled data added could not change the decision boundary significantly. Recent findings in [22] [104] also concluded that confidence based scoring function does produce suboptimal solutions. The chances that the final model created on continuous learning for some time would suffer from class imbalance problem are high. Another very common problem is that initial mistakes in learning would reinforce itself and the model would deteriorate more on further training. Various methods have been proposed in literature to alleviate this problem. Among the few advantages of self-learning is that it is very simple to implement this algorithm. Self-learning is a wrapper method as this approach do not impose any condition on the type of learner to be used. The self-training procedure "wraps" around the learner without changing its inner workings. This is important
for many real-world tasks like natural language processing, where the learners can be complicated black boxes not amenable to changes.

2.4.2 Probabilistic generative models

The previous approach of self-learning suffers from risk of working with incorrect samples which would lead the models to diverge. This risk can be ameliorated by using expectation - maximization (EM) algorithm [98], which is a probabilistic generative model approach.

In semi-supervised learning, the dataset $D$ consists of both labeled and unlabeled data. The likelihood depends on both the labeled and unlabeled data-this is how unlabeled data might help semi-supervised learning in mixture models. It is no longer possible to solve for the Maximum Likelihood Estimation (MLE) analytically. Given the labeled and unlabeled data $D = (x_1, y_1), \ldots, (x_l, y_l), x_{l+1}, \ldots, x_{l+u}$, the log likelihood function (what we are trying to maximize) is defined as

$$\log p(D|\theta) = \log \left( \prod_{i=1}^{l} p(x_i, y_i|\theta) \prod_{i=l+1}^{l+u} p(x_i|\theta) \right)$$

$$= \sum_{i=1}^{l} \log p(y_i|\theta)p(x_i|y_i, \theta) + \sum_{i=l+1}^{l+u} \log p(x_i|\theta)$$

(2.1)

Marginal probability $p(x|\theta)$, the probability of generating $x$ from any class, is defined as:

$$p(x|\theta) = \sum_{y=1}^{C} p(x, y|\theta) = \sum_{y=1}^{C} p(y|\theta)p(x|y, \theta)$$

(2.2)

Where $\theta$ is a set of model parameters (e.g. Gaussian means and variances). The second term (equation 2.1) in the above equation is the term that differentiates between the supervised and above semi-supervised data likelihood function. Solving the MLE estimates would be non-trivial problem as the second term has made the objective function non concave, hence it is hard to optimize. EM algorithm as pro-
posed in [40] is specifically designed to solve this kind of problem. The basic EM as proposed in [40] has an initialization step, where the model parameters are assigned some initial value (various methods are proposed in literature for this step) and then this step is followed by two alternating step, expectation step and maximization step:

\[ \text{E step: model's expected sufficient statistics are computed under the current model parameters (i.e., given some current assignment of the unlabeled instances to classes)} \]

\[ \text{M step: model parameters are updated to maximize the likelihood of observing data with these sufficient statistics (i.e., update the mean and variance of the two class distributions).} \]

\textbf{EM for GMM:} For a given set of data \( X = (x_1, \ldots, x_N) \), and a pre-specified number of mixture components \( K \), the goal of Gaussian Mixture Model (GMM) learning is to find the parameter values for \( \mu, \Sigma, \pi \) that maximize the likelihood of [generating] the training data. If \( X \) is available as a single batch, this optimization can be accomplished using the Expectation-Maximization algorithm [40] which, given a set of initial parameter estimates, works by iterative application of the following steps:

\[ \text{E Step: Estimate the "responsibility" } r(x_n, k) \text{ of the } k_{th} \text{ component for the } n_{th} \text{ data point } x_n: \]

\[ r(x_n, k) = \frac{\pi_k N(x_n | \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_n | \mu_j, \Sigma_j)} \tag{2.3} \]

\[ \text{M Step: calculate } \mu, \Sigma, \pi \text{ for each component } k \text{ according to the estimated re-} \]
Responsibilities:

\[ \pi_k^{\text{new}} = \frac{1}{N} \sum_{n=1}^{N} r(x_n, k) \]

\[ \mu_k^{\text{new}} = \frac{1}{(N, \pi_k^{\text{new}})} \sum_{n=1}^{N} r(x_n, k) . x_n \]

\[ \Sigma_k^{\text{new}} = \frac{1}{(N, \pi_k^{\text{new}})} \sum_{n=1}^{N} r(x_n, k) . (x_n - \mu_k^{\text{new}})(x_n - \mu_k^{\text{new}})^T \] (2.4)

In the case of semi-supervised mixture models, the EM algorithm can be thought of as assigning "soft labels" to the unlabeled data according to the current model \( \theta^t \). Since objective function of EM is non-concave it can only converge to a local optimum, and the specific one depends on the initial parameter \( \theta^0 \). A common choice of \( \theta^0 \) is the MLE on the small labeled training set. Still, similar to confidence score based techniques, EM-based algorithms might result in sub-optimal final models because every EM-based algorithm is subject to the initial parameter \( \theta^0 \).

This sub-optimality problem can be tackled by applying some prior knowledge when creating model \( S^t \). In particular, [167] used a closeness measure in a feature space \( X \) to select samples, and in [148], a performance-driven measure was used instead of confidence scores. The striking parallel between self-learning and EM algorithm is worth noting and i.e. both algorithm assigns label to the unlabeled samples, but EM assigns label with a fractional weight. Self-learning assigns label to only the unlabeled samples about which the current model is most confident.

The above mentioned EM algorithm is for the offline case i.e. for batch learning but there are various online variants of EM have been proposed in literature [148] so that the model parameters can be updated online. GMM parameters can be adjusted on the basis of single items of training data. The key insight is that the parameters \( \pi \) (prior probability), \( \mu \) (mean vector), \( \Sigma \) (covariance matrix) can be represented in terms of the following sufficient statistics (specifically, sums of the contributions from each data point) \( S_{\pi_k}, S_{\mu_k}, and S_{\Sigma_k} \) which can then be updated - in a number of
different ways - for single data points:

\[ S_{\pi_k} = \sum_{n=1}^{N} r(x_n, k) \rightarrow \pi_k = 1/NS_{\pi_k} \]

\[ S_{\mu_k} = \sum_{n=1}^{N} r(x_n, k).x_n \rightarrow \mu_k = \frac{S_{\mu_k}}{S_{\pi_k}} \]

\[ S_{\Sigma_k} = \sum_{n=1}^{N} r(x_n, k).x_n^T \rightarrow \Sigma_k = \frac{S_{\Sigma_k}}{S_{\mu_k}} - \mu_k\mu_k^T \]  

(2.5)

An incremental (not online) variant of EM was proposed by Hinton et al. [111]. But since this is an incremental algorithm it needs full chunk of data at a time. On the similar line many online variant of EM has been proposed in past. Authors in [91], [149] and [39] present a wide range of strategy to update the model parameters using a single new sample. But updating the model parameters is not enough to make a system completely online because the model parameters are updated online assuming that both the number of mixture components and their initial parameters value have been appropriately specified. But the assumption may be wrong sometimes or due to minor change in the input distribution the structure of GMM may need some change. Therefore many researchers have worked on incrementally / online learning the structure of GMM. In [27] author has proposed a method to incrementally learn the structure of GMM, whereas approaches in [7], [149] and [39] suggest some strategy which made online learning of the GMM structure possible.

2.4.3 Co-Training / Multi View Learning

This kind of semi-supervised learning for the cases where there are two distinct and sufficient features set (also referred as view). The underlying assumptions of co-training [15] are: 1) each feature set should be sufficient to achieve perfect classification accuracy when used independently and 2) each feature set should be conditionally independent given a class label. It is worth noting that satisfying both the conditions (especially the second condition) at the same time sometimes happens to be unrealis-
tic so it is possible that co-training may not perform very well. However the authors in [11] have relaxed the second strong condition. The strong conditions include "each view is conditionally independent of the other given the class label (used in [13])", "weak rule dependence [2]". These strong conditions were used by different authors to prove their guarantees. Several online variants co-training algorithms have been proposed recently [23] [58] for handling different applications.

Authors in [58] have proposed an online co-training approach for audio-visual co-training for vehicle classification which is based on the audio and the video information. They proposed to use an on-line multi-class gradient boosting for visual classification which inherently allows discriminating between cars and trucks as well as the local background. For increased diversity between audio and visual classification, on-line random naive Bayes classifier was used for acoustic classification. Experimentation proved that the gain in accuracy was substantial and it adapted to the traffic scenes without any additional human annotation. Their work shows that even for a very small amount of labeled data, we can gain reasonable classification accuracy by combining heterogeneous classifiers and improve the classifier during runtime by on-line co-training.

On the other hand authors in [23] have used deep belief network and data association classifier on the co-training framework to perform online learning. The on-line nature of algorithm implies that the distribution that generates test samples is different from the original distribution of training samples, which complicates the functionality of the co-training algorithm. This issue was handled by imposing a criterion on how to select un-annotated images into the new training sets.

2.5 Conclusion

In this chapter we have presented a brief review of different approaches present in the literature for performing incremental and online learning in supervised and semi-supervised learning paradigm. It is evident from the review that not many works has been done on semi-supervised incremental/online learning front. Essentially the
probabilistic graphical model has not been explored thoroughly for conducting online learning in semi-supervised manner. Looking into the literature we observed that one of most natural way to learn would be to conduct learning is to learn incrementally/online in active manner using both labeled and unlabeled data (semi-supervised learning). Very few works present in literature [61] [62] actually deals with this learning setting "semi-supervised online active learning". Therefore in present work we explore this promising mode of learning using randomized Bayesian Network. In the next chapters we give the detail of the experiments done for increment/online semi-supervised learning.
Chapter 3

Incremental/Online Semi-supervised Learning Using Bayesian Network

As described in detail in the previous chapter, a number of approaches have been used in the past in the area of incremental machine learning, however gaps still exist. We in this chapter present the detail of preliminary experiment conducted in order to make semi-supervised incremental/online learning possible. To begin with, our approach makes use of simplest probabilistic graphical model structure i.e. Naive Bayes (later we augment the structure of NB) and adapt that so that it could be used in the semi-supervised incremental/online learning setting. The Fig.3-1 shows the basic steps of online semi-supervised learning for RNB classifier.

In this section of the report, we address the problem of Semi-supervised incremental/online learning which makes use of both labeled and unlabeled data to learn continuously over time. The proposed method makes learning possible from a continuous inflow of a potentially unlimited amount of data without the requirement for storage. We introduce an incremental/online Randomized Naive Bayes (RNB) classifier which is used to learn in an online supervised manner. Later we introduce an incremental variant of the Expectation Maximization (EM) algorithm which makes incremental semi-supervised learning possible. A weighting factor is used to modulate the contribution of unlabeled data which makes parameter estimates stable. An empirical evaluation of the proposed method on Tamil handwritten base character
recognition proves its efficacy. It shows that the accuracy of the proposed incremental RNB and incremental semi-supervised RNB converges to their batch counterpart. The IWFHR 2006 competition dataset was used for the evaluation process.

3.1 Introduction to online Tamil handwritten character recognition and the proposed Naive Bayes Approach

We investigate the problem of semi-supervised incremental learning using Naive Bayes classifier. In the process we propose a novel Incremental Naive Bayes classifier and an incremental EM algorithm which makes the semi-supervised incremental learning possible. In order to demonstrate the efficacy of the proposed scheme we have applied our method on a real time multi-class (30 class) pattern recognition problem i.e. online Tamil handwritten base character recognition.
Given the high popularity of tablet computers, we know there is a substantial amount of handwritten data already available and massive amounts of data are created every day. If we look at the popular classification methods in practice, then we will find numerous statistical methods which can very efficiently address this problem. There is a key restriction with most of the classification methods used currently for handwritten character recognition, which is addressed with our presented approach. The common constraint of most of the current classification algorithms is that they require prohibitively large amounts of labeled data to learn precisely. Labeling data is a painstaking job which involves human effort. Hence, this labeling process can be tremendously expensive and also time consuming.

Given that in most of the scripts, the number of classes to be handled is very high, thus manually labeling many samples for each class is very tedious. Therefore it would be much more preferable if a classification method required significantly less manually-labeled data instead of a large amount in order to learn and classify accurately. Another major constraint of most of the contemporary classification methods for handwritten characters is that classifiers need to be trained in an online mode i.e. the entire training data is given in advance and the training and testing phases are separated. However, if we consider a real-time problem such as handwriting recognition, where there is a continuous inflow of data, (may be with slight variation in nature, as user may change) and prediction is required on the fly, then in that case, online training fails to perform adequately. If we intend to exploit the sequential inflow of data, then our learning method should have the ability to learn incrementally, where training and testing occurs in parallel. Few advantages of incremental learning are: a) very little memory requirements as it does not require the storage of training samples for future reference, b) huge amount of data can be used for better training, c) incremental methods are usually very fast, d) can handle data generated continuously with time e) can adapt if the underlying distribution of input data changes with time.

To make online handwritten character recognizer we make use of very scarce number of labeled samples. These few labeled samples were augmented by plenty of low-
cost and readily-available unlabeled data for better estimation of learning parameters in an incremental manner. In order to learn online we propose a novel Online Random Naive Bayes (RNB) classifier. To perform learning in an incremental and semi-supervised manner we also propose an incremental variant of the Expectation Maximization (EM) algorithm for the parameter estimation of a univariate Gaussian, which is in a way, inspired by the work of Lee in [91] but it is different in terms of: a) the learning rate estimation step, which helps moderate the contribution of unlabeled data in the M step and hence prevents model parameters from generating improper estimates, b) estimation of variance in the M step, c) the capability to handle both labeled and unlabeled data together, d) its application to a single Gaussian, e) the way it is implemented to suit the problem.

We use the proposed semi-supervised incremental variant of the EM algorithm to train (RNB) that takes advantage of both labeled and unlabeled data. EM is an iterative algorithm [40] for maximum likelihood or maximum a posteriori estimation of the incomplete data problem. Data is considered incomplete because it lacks sample class information. The algorithm first trains classifiers incrementally from scratch using the first few labeled samples and then this trained classifier is used to classify each element of unlabeled data with some posterior probability. This posterior value is used as a weight to update the learning parameters of every class. This iterative process is repeated for every unlabeled and labeled sample. In the above process, we use a weighting factor that dynamically controls the contribution of unlabeled data for the parameter estimation in EM.

Motivations

(1) We use online semi-supervised learning because it is a more natural learning setting (closer to the way humans learn, leveraging from both labeled and unlabeled data).

(2) To ensure the learning system capability is adaptive to change (e.g. changes in handwriting style) in the input data. This is achieved by making the learning process online.

(3) To make the learning system capable of taking advantage of immense unlabeled
data, which are cheaply available. This is achieved by making the online learning system semi-supervised.

(4) The Naive Bayes (NB) is an elegant and simple classification method which in spite of being very simple, performs quite accurately for many applications (e.g. text classification). Therefore all those applications can benefit from the proposed online version of NB with slight required modifications.

(5) The NB classifier is simple but it has much scope for further improvement by applying Bayesian network techniques.

Our Contributions

(1) To the best of our knowledge, this is the first time an online RNB has been proposed. Though it appears similar to the RNB proposed in [57], our method uses different method to update learning parameters online (eq. 3.9, 3.10 and 3.11). Besides, our method uses online bagging [118] for random selection of input and they use randomized threshold selection. Moreover, unlike [57] our method takes advantage of unlabeled data.

(2) Our proposed online EM is novel. Though it is inspired by the growing GMM proposed in [91], there exist the following differences: a) the learning rate estimation step, which helps moderate the contribution of unlabeled data in the M step and hence prevents model parameters from generating improper estimates, b) estimation of variance in the M step, c) the capability to handle both labeled and unlabeled data together, d) its application to a single Gaussian.

(3) To the best of our knowledge this is the first time online learning of handwritten characters has been performed in a semi-supervised setting using a Naive Bayes classifier.

Through our experimentation we prove that (1) our proposed Online RNB does converge to the offline RNB, (Fig. 3-2), (2) our proposed incremental semi-supervised learning method, with comparatively very few labeled samples augmented with many of unlabeled samples, does converge to the batch learning method (Fig. 3-3). (3) unlabeled data can significantly increase the accuracy of the classifier, especially when the available labeled data is sparse (Fig. 3-3) and (4) by tuning parameters of
the incremental EM, we can increase or decrease the memory of the online learning system.

This chapter is organized as follows. In section 3.2 we introduce Online Random Naive Bayes (RNB) and examine how to make an online version of RNB. In section 3.3, semi-supervised incremental RNB is presented. In section 3.4 we introduce Augmented Naive Bayes which is based on probabilistic graphical model approach, precisely it is a Bayes net. In section 3.4 we describe in detail the learning process, both for batch learning and online learning of the proposed model parameters.

3.2 Online Random Naive Bayes

3.2.1 Naive Bayes:

Naive Bayes [114] is well known probabilistic classifier. It assumes the conditional independence among features which greatly reduce the number of features to be estimated, and hence tremendously reduces the training time. Let us introduce some notation to describe the data. Training dataset \( \chi = (X^1, Y^1)(X^N, Y^N) \) where \( X^i = x^i_1...x^i_D \), \( X^i \in R^D \) are the samples in a D dimensional feature space and \( Y \in \{1...K\} \) are the corresponding labels for a K-class classification problem.

Using the Bayes rule we can write:

\[
P(Y = y_k|x_1...x_D) = \frac{P(Y = y_k)P(x_1...x_D|Y = y_k)}{\sum_j P(Y = y_j)P(x_1...x_D|Y = y_j)} \quad (3.1)
\]

where \( P(Y = y_k) \) is the class prior probability \( \pi_k \) for each class k, and \( P(Y = y_k|x_1...x_D) \) is the unknown conditional probability that we must determine. Naive Bayes assumes conditional independence among the features \( x_i \)'s in the feature vector \( X \), hence we can write:

\[
P(Y = y_k|x_1...x_D) = \frac{P(Y = y_k)\Pi_i P(x_i|Y = y_k)}{\sum_j P(Y = y_j)P(x_1...x_D|Y = y_j)} \quad (3.2)
\]

Therefore to find \( P(Y = y_k|x_1...x_D) \) we need to estimate the value of \( P(x_i|Y = y_k) \)
for each feature \( x_i \) and this estimation is the only step we need to take in order to train our Naive Bayes classifier. So to train our classifier we fit a Gaussian \( \mathcal{N}(x_i; \hat{\mu}_{ik}, \hat{\sigma}^2_{ik}) \) to each \( P(x_i|Y = y_k) \), and we estimate mean and variance for the same using the training data. We perform Maximum Likelihood Estimation (MLE) to find the mean \( \hat{\mu}_{ik} \) and variance \( \hat{\sigma}^2_{ik} \) of \( P(x_i|Y = y_k) \) for each \( x_i \).

**Maximum Likelihood Estimation is given as:**

\[
\hat{\mu}_{ik} = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j x^j_i \delta(Y^j = y_k)
\]

\[
\hat{\sigma}^2_{ik} = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j (x^j_i - \hat{\mu}_{ik})^2 \delta(Y^j = y_k)
\]

where: \( x^j_i \) is \( i^{th} \) feature of \( j^{th} \) sample, \( Y^j \) is class no. of \( j^{th} \) sample, \( \delta(Y^j = y_k) = 1 \) if \( Y^j = y_k \) else 0. \( \hat{\mu}_{ik} \) = estimated mean of the \( i^{th} \) feature of the \( k^{th} \) class, \( \hat{\sigma}^2_{ik} \) = estimated variance of the \( i^{th} \) feature of the \( k^{th} \) class.

The classification rule for a new sample \( X^{new} = < x_1 \ldots x_D > \) is:

\[
Y^{new} = \arg \max_{y_k} P(Y = y_k) \prod_i P(x_i|Y = y_k)
\]

\[
Y^{new} = \arg \max_{y_k} \pi_k \prod_i \mathcal{N}(x^i_{new}; \hat{\mu}_{ik}, \hat{\sigma}^2_{ik})
\]

### 3.2.2 Random Naive Bayes:

RNB is an ensemble of Naive Bayes classifiers where each classifier is trained offline on different sets of training samples (Bagging) [21] and feature sets (random subspace method). For training with different sets of features we pseudo-randomly select subsets of components of the feature vector, that is, Naive Bayes is constructed on randomly chosen subspaces. To achieve this we use the Random Subspace method as proposed by T. K. Ho in [70]. Bagging and the random subspace method helps to make a diverse classifier ensemble. Classifiers made using this randomization technique are very fast (in both training and testing), easily parallelized, inherently multi-class and
also stable in the sense that classifiers have a very low variance (therefore less chance of over-fitting).

Instead of using trees as classifiers with the randomization technique, we use the Naive Bayes classifier. It has been shown by Prinzie et al. [122] that other learning methods such as Naive Bayes can replace tree structures without reducing the performance. The low computational and memory cost of this classifier makes it the right choice for online learning where both computational power and memory usage is reduced.

We create each classifier in the ensemble of B classifiers by repeating the following steps B times:

1. Perform bagging to create bootstrapped training data for each classifier.

2. Randomly select F features from the pool of D features.

3. Estimate the learning parameters for the classifier.

A test sample will be classified by each classifier in the ensemble and the class which gets the majority vote by the ensemble will get assigned to the test sample.

3.2.3 Online RNB:

Among one of the previous attempts to make Random Naive Bayes (RNB) [122] online was by Godec et al. [57], which uses the equally-binned histogram technique. We propose a different method to update the learning parameters in an online manner. It processes a sample only once to update the learning parameters and discard that afterwards.

We fit a univariate Gaussian $\mathcal{N}(x_i; \hat{\mu}_{ik}, \hat{\sigma}^2_{ik})$ to each $P(x_i|Y = y_k)$. Learning parameters (mean and variance) are updated in an online manner as shown below.

Initialization (by just one labelled training sample)

$$\hat{\mu}_{ik} = x^1_{ik}, \hat{\sigma}^2_{ik} = \sigma_0.$$ (3.7)
\[ \pi_k = \frac{1}{\text{no.of classes}}; c_k = 1; \alpha = \alpha_0. \quad (3.8) \]

where \( c_k \) = no. of samples used so far for training, \( \alpha \) = decides the length of memory of the classifier \((\alpha < 1)\)

Repeat below steps for all the available labeled training samples:

\[ c_k = c_k + 1; \eta_k = \left( \frac{1 - \alpha}{c_k} + \alpha \right) \quad (3.9) \]

where \( \eta_k \) is learning rate for class \( k \).

\[ \mu_{ik}(t) = (1 - \eta_k)\mu_{ik}(t - 1) + \eta_k x^j_i \delta(Y^j = y_k) \quad (3.10) \]

\[ \sigma^2_{ik}(t) = (1 - \eta_k)\sigma^2_{ik}(t - 1) + \eta_k (x^j_i - \mu_{ik}(t))^2 \delta(Y^j = y_k) \quad (3.11) \]

The classification step is the same as it is for the off-line case. The Fig. 3-2 and the result Table:3.1 show that online training does converge to the off-line method.

### 3.3 Online Semi-Supervised Learning: Naive Bayes Approach

In a typical case of semi-supervised learning, we have missing information in the data and the task is to estimate the value of that missing information. Usual solutions to this would be to use Expectation Maximization (EM) \([40][114]\). In the E-step, the posterior probability of data points is computed, while in M-step, parameters of the learning model are computed. An iterative algorithm like this is not suitable in our context of online learning since our goal is to process training samples one at a time and to avoid their storage. This is a real-time scenario when we need to deal with streams of data or a huge dataset (which cannot be processed fully in a single step). Hence a method that updates the model in a continual and an evolutionary
manner can be used in the present study. This problem has been addressed by a few researchers such as Stauffer et al. [153], Radford et al. [112], Bouchachia et al. [16] and Lee [91]. In this study we use some findings of Lee [91].

We have incorporated a regulating constant $\lambda$ in the proposed method. This parameter moderates the contribution made by the unlabeled data by reducing the learning rate ($\eta$) and hence the weight of the unlabeled samples during step M. The best value of $\lambda$ in practice is decided using a validation set, which was found to be 0.01 in our case. $\lambda$ reduces the learning rate for all unlabeled data equally but the value of $\eta$ increases for the correct class sample because of the higher posterior value $q_k$. It is a slow hill-climbing process in which the correct class sample makes the hill-climbing step greater towards the hill top, whereas the incorrect class sample makes the step small and in the opposite direction, so effectively we get closer to the hill top after processing many unlabeled data samples.

Details of the proposed incremental (online) EM algorithm for creating an incremental semi-supervised learning system are as shown below:

1. Train RNB online with a few labeled training samples. Repeat the following steps for every unlabeled or labeled data sample for better estimation of learning parameters.

2. **E Step (of proposed online EM):**

   If incoming new sample is unlabeled then use trained classifier to find the posterior $q_k = P(Y = y_k|X)$ corresponding to all k (class). Else $q_k = 1$.

3. **M Step (of proposed online EM):**

   $$c_k = c_k + q_k \lambda; \eta_k = q_k \left( \frac{1 - \alpha}{c_k} + \alpha \right) \lambda$$  \hspace{1cm} (3.12)

   where $\lambda = $ weight factor applied to moderate the contribution of unlabeled data in the parameter estimation step ($\lambda = 1$ if the sample is Labelled) and $\eta_k$ is a
learning rate parameter.

\[ \mu_{ik}(t) = (1 - \eta_k)\mu_{ik}(t - 1) + \eta_k x_i^j \delta(Y^j = y_k) \]  \hspace{1cm} (3.13)

\[ \sigma^2_{ik}(t) = (1 - \eta_k)\sigma^2_{ik}(t - 1) + \eta_k (x_i^j - \mu_{ik}(t))^2 I \]  \hspace{1cm} (3.14)

where \( I = \delta(Y^j = y_k) \)

4. Repeat step E and M (just once) for each labeled or unlabeled sample left.

The above algorithm is capable of handling both labeled and unlabeled data at the same time to update the parameters of the learning system. Experiments indicate that this online semi-supervised learning approach does converge Fig. 3-3 to the accuracy achieved by the offline supervised RNB with far less labeled samples. The gain in accuracy is remarkable when the labeled data used is less which is evident from Fig. 3-4.

3.4 Experimental Evaluation

In this section we discuss Tamil handwritten character recognition, experimental settings, the database, issues with the recognition problem at hand and results. We demonstrate our proposed classification method performance on the recognition of Tamil (one of the Indian scripts) handwritten base characters. Base characters constitute only vowels and consonants. It is a 30 class problem. The IWFHR 2006 competition handwritten online dataset [75] was used for experimental results reported in this paper. Each character in the above online dataset is represented by a discrete variable number of points, and each point is described by its x and y co-ordinates.

3.4.1 Dataset and Challenges

The dataset used in this chapter for experiments is IWFHR 2006 competition handwritten online dataset [75]. It is a Tamil script character dataset. For experimenta-
tions base characters were used which constituted 30 classes. The character recog-
nition problem comes with usual challenges of learning the immense intra-class vari-
ability. Huge intra-class variability occurs because the data was collected from many
users (different individuals). Large intra-class variability becomes more grave when
the number of classes of characters to be learnt are high, which is because many a
times a character may be derived from another character, e.g. compound characters.
Though large inter-class variability (which occurs because the data is collected from
large number of users) makes the learning task difficult but most of the time it proves
to be a blessing in disguise because it improves the generality of the learner which
is good for real-time scenario. Another challenge which is very typical to charac-
ter recognition is that of a confusing classes. Unfortunately there are many sets of
confusing classes in Tamil scripts. Confusing classes problem become more grave in
case of handwritten character recognition because of very casual or carefree way of
handwriting that people generally practice.

The above challenges can be best handled by training a very strong learner using
a huge dataset. In case when the dataset is not very big to train a strong learner, for
example a deep neural network, could not be optimally trained then researchers have
tried to resolve the confusing classes separately. Separate methods have been tried to
solve this problem in literature: 1) by using heuristics, checking the structural details
of the confusing classes 2) using a special separate learner which has been trained
using some special discriminating features (to resolve the confusion) thus classifying
the confusion classes more accurately.

On the first part of the experimentations with incremental Naive Bayes learner,
which is relatively a weak learner, we boosted the accuracy of the classification using
a Dynamic Time Warping (DTW) based pattern matching. Confusing classes are the
classes which gets mixed up with each other during the classification process. The set
of such classes are found using the confusion matrix. DTW base pattern matching
was done on a limited number of confusing classes as a post processing method.
Results are indicated in the result Table:3.1. When the Naive Bayes structure was
augmented to turn it into a stronger learner by itself, therefore the gain in accuracy

54
after post processing using DTW is not very substantial as evident from the result Table:3.2. Therefore it shows that as the learner got stronger the confusion among classes reduces to a great extent. Though there are more sophisticated post processing methods are available in literature which when applied will boost the accuracy even further, but since that is a separate and different research topic which is not explored in this thesis.

3.4.2 Experimental settings and experiments:

For all the experiments, the data was pre-processed. For pre-processing, we normalized each character, maintaining their aspect ratio and also we resampled each character into 60 equidistant points. The five features extracted per point are a) x-coordinate b) y-coordinate c) x-derivative d) y-derivative as proposed in [77] and e) pen-direction angle as described in [145].

In all the Naive Bayes experiments we fitted a single Gaussian for each feature for each class i.e. \( P(x_i|Y = y_k) \), where \( i \) and \( k \) represent the \( i^{th} \) feature and the \( k^{th} \) class respectively. For the offline RNB using the validation set (of size 100 samples), we decided upon the best number of features and classifiers in the RNB ensemble which was found to be 20 and 50 respectively. So we performed online bagging [118] to create 50 different datasets of size 350 each, chosen from a pool of 350 samples. Consequently, 50 different classifiers were created by using a set of 20 random features out of the pool of 300 and then training a classifier using a bootstrapped dataset. Classification of a test sample is performed using the majority vote. All the results reported are on a separate test set of 100 samples. Results in Table:3.1 show that RNB produced better recognition accuracy when compared to Naive Bayes.

3.5 Discussion I

The conditional independence assumption on which Naive Bayes is based upon is readily violated in any real world application and such is the case for handwritten character recognition. Despite that, it manages to give good accuracy as we know that
Figure 3-2: Plotted accuracies of RNB and Online RNB shows that as the number of training samples increases, the gap between the accuracies of RNB and ORNB reduce monotonically. The accuracy gap increases at the beginning showing better learning by the offline RNB with few training samples.

Table 3.1: Table below show the detail of each classifier. 20 random features were used to create 50 different classifiers to form the ensemble (RNB and ORNB). DTW-based classification use 15 hand-picked samples for each class.

<table>
<thead>
<tr>
<th>Classifier Name</th>
<th>Training/Testing</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve Bayes (NB)</td>
<td>350/100</td>
<td>78.26</td>
</tr>
<tr>
<td>Random NB (RNB)</td>
<td>350/100</td>
<td>81.35</td>
</tr>
<tr>
<td>Online RNB (ORNB)</td>
<td>(Initial 2 + 350)/100</td>
<td>78.77</td>
</tr>
<tr>
<td>ORNB + DTW</td>
<td>(Initial 2 + 350)/100</td>
<td>91.81</td>
</tr>
<tr>
<td>SVM</td>
<td>350/100</td>
<td>90.68</td>
</tr>
<tr>
<td>HMM</td>
<td>350/100</td>
<td>87.82</td>
</tr>
</tbody>
</table>

a classification estimate is only a function of sign (in a 2 class problem) of function estimation. Though the estimations are poor, classification accuracy can still be high. RNB is used to improve accuracy further.

A method to learn online in a supervised manner using RNB is proposed. $c_k$ is one of the sufficient statistics which is updated at every step of incremental learning,
Figure 3-3: Plotted accuracy of RNB and Online Semi-Supervised RNB shows that unlabeled data boosts the accuracy sharply when there is a smaller number of labelled data samples and as the number of labeled sample increases, the accuracy of online semi-supervised RNB converges to offline RNB.

and is just the count of samples from each class. $c_k$ is one of the parameters that decides the learning rate $\eta_k$. It depends on the learning rate at a particular instance of time i.e. what weight will be given to the new sample. Initially the $\eta_k$ value will be high, as the sample count would be small, thus giving more weight to the new incoming sample, but as the count grows with time, $\eta_k$ reduces and finally as time tends to infinity, it converges to $\alpha < 1$. This convergence to $\alpha$ makes sure that learning does not stop ever thus making the incremental supervised learning scheme adaptable. With this understanding it is possible to make an application (based on this learning scheme) where the user has the liberty to change this single parameter to increase or decrease the learning rate. For example, let us consider a device that has a handwriting recognition system and a user wants to customize that according to his handwriting style quickly, then the user has to just decrease the value of a single parameter $c_k$ to increase the learning rate so that it adapts faster by giving
Figure 3-4: Plotted accuracy shows that the Online Semi-Supervised RNB accuracy increases with unlabeled training samples. The increase in accuracy is a maximum for the classifier, which was pre-trained with a smaller number of labeled samples.

Along a similar line, the proposed incremental EM makes incremental semi-supervised learning possible. Unlike the usual EM algorithm (iterative), incremental EM is executed only once for each labeled or unlabeled sample. This makes the procedure very fast. It is well understood from the literature that if our statistical model is close to the data generating model then the EM algorithm can help to extract information from the unlabeled data and improve the estimates of the parameters, else it can degrade the estimates. In the present case, since the assumption of independence among features does not hold in handwritten characters, it degrades the estimates and hence the accuracy. This is evident from Fig. 3-7 where it is noted that as we increase the weight of an unlabeled data contribution (i.e. as the value of $\lambda$ increases) the accuracy goes down. This phenomenon is less severe in cases when we use less unlabeled data as shown in Fig. 3-6. From Fig. 3-6 and Fig. 3-7 we can easily ascer-
Figure 3-5: Plotted accuracy shows that as the value of alpha increases (memory of classifier becomes smaller) the generalization capability of the classifier goes down.

tain that if we have more unlabeled data we should give less weight to the unlabeled data for a better estimate (in our case $\lambda = 0.01$) as per Fig. 3-7. And in case where we have less unlabeled data then we should give more weight to the unlabeled data (best accuracy is achieved around $\lambda = 0.3$), as in Fig. 3-6. Eventually we observed that with the increase in the number of unlabeled data, estimates of the parameters improved, which manifested in terms of higher accuracy. Thus it is evident from Fig. 3-7 that if parameter $\lambda$ had not been used ($\lambda = 1$) then the accuracy would have suffered a very sharp dip. As expected, the degradation in accuracy is sharper when the number of labeled data used is high.

Another crucial parameter used in incremental EM is $\alpha << 1$, which can be thought of as the one which decides length of the memory of the learner. If the value of $\alpha$ is large then (learning rate $\eta$ increases) the learner has a smaller memory and gives more weight to the most recently encountered data and forgets quickly. Thus it is can be easily understood that $\alpha$ decides the generalization capacity of the
Figure 3-6: Plotted accuracy shows that if number of unlabeled data is not huge then for better incremental semi supervised learning we should give relatively more weight to unlabeled data but if excessive weight is given then accuracy suffers a small dip. Number of unlabeled data used 10 per class.
Figure 3-7: Plotted accuracy shows that if number of unlabeled data is huge then for better incremental semi supervised learning we should give relatively less weight to unlabeled data otherwise accuracy suffers a huge dip. Number of unlabeled data used 350 per class

3.6 Online Semi-Supervised Learning: The Online
Random Bayesian Network Approach

Bayesian network is based on probabilistic graphical model (PGM) approaches. Bayesian network builds on the same intuition as Naive Bayes (NB) model by exploiting the conditional independence properties of the distribution in order to allow compact and natural representation. Bayesian network approach allows creating our own network structure which represents the data distribution (which embeds the reasonable conditional independence properties specific to a particular case or setting). Bayesian network provides the flexibility by which the unwarranted (by the data) independence assumptions can be avoided. Bayesian network provides an appropriate language and efficient machinery to represent and manipulate independence assertions.
3.6.1 Bayesian Network

More formally Bayesian Network representation can be defined as an annotated Directed Acyclic Graph (DAG) G, whose nodes represent different random variables (RVs) \( U' = \{X_1, X_2, \ldots, X_N\} \) and an edge correspond to dependency (direct influence) between the two connected nodes (i.e. RVs). The graph G represents following independence assumption: "each variable \( X_i \) is independent of its non-descendants given its immediate parents \( P_a(X_i) \) in G". A Bayes Net "G" represents a joint probability distribution over a collection of RVs. For e.g. NB can be considered as the special case of Bayes net where all the RV has just one parent (dependency) (which is common among all the RVs), which means that, given that particular common parent, all other RVs are conditionally independent of each other. This is a very strong assumption. The RVs in case of NB are \( U = \{Y, X_1, X_2, \ldots, X_N\} = \{Y, U'\} \). Where \( X_i \) and \( Y \) represents attributes and class labels respectively. Bayes Net structure of NB is shown in Fig 3-8.

The joint probability distribution over \( U \) with respect to Naive Bayes BN is represented as follows:

\[
P(U) = P(Y, X_1, X_2, \ldots, X_N) = P(Y) \prod_{i} P(X_i | P_a(X_i)) \quad (3.15)
\]

Figure 3-8: Bayes net structure for Naive Bayes.

Figure 3-9: Augmented Bayes net structure of Naive Bayes.
Where $P_a(X_i)$ is the set of RVs which are the immediate parents of $X_i$. With respect to NB, $P_a(X_i) = Y$. Conditional probability for each node $X_i$ is represented by $P(X_i|P_a(X_i))$.

The facet that the assumption used for constructing NB classifier is too restrictive and strong, this raises an obvious question that whether a classifier with less restrictive assumption can perform better. It has been proved in literature that by augmenting the structure of naive Bayes, classifier’s performance improves at the cost of some computational overhead [50]. Therefore to increase the accuracy of the classifier we assume that maximum number of parents of each RV is 2 (except RV "Y"). In order to capture the dependence between the attributes we augment the network structure of NB and we use that BN as classifier. This improved network is more plausible and corresponds to a probability distribution which is closer to the actual distribution (which we are trying to approximated through Bayesian modeling). In order to augment the structure we identify the highest correlation between any two attributes and subsequently connecting them. Highest correlation is determined by finding the highest mutual information between every two possible pair of attributes which do not have second parent in NB Network. In literature, network like this is also referred as Tree Augmented Naive Bayes (TAN) network [50].

For experimentation we have used the Bayesian networks similar to as shown in Fig 3-9. It is generally easy to handle parents which take discrete value but when RV takes continuous values then parameters estimation is not trivial. Network shown in Fig 3-9, RVs take different type of data ($X_i$ takes continuous values whereas $Y$ takes discrete values (class labels)). We use Gaussian distribution to parameterize the distribution of $P(X_i|X_{ipc}, Y)$ (for all values of $i$ and $X_{ipc} =$ continuous parent of node $i$). Each $X_i$ (child) has both continuous ($X_{ipc}$) and discrete parents ($Y$). Therefore $P(X_i|X_{ipc}, Y)$ for each particular discrete value of $Y$ can be expressed as shown below [82].

$$P(X_i|X_{ipc}, Y) = N(\beta_0 + \beta_1 X_{ipc}, \sigma^2) \quad (3.16)$$
The joint probability distribution (using equations above) over the RVs in the above network Fig. 3-9 can be written as:

\[ P(Y, U) = P(Y, X_1, X_2, \ldots, X_N) = P(Y)P(X_1|Y)P(X_2|Y, X_1)\ldots P(X_N|Y, X_{N-1}) \]

we know that:

\[ P(Y, U) = P(Y)P(U|Y) = P(U)P(Y|U) \]

from above two equations we can write:

\[ P(Y|U) = \frac{P(Y)P(X_1|Y)P(X_2|Y, X_1)\ldots P(X_N|Y, X_{N-1})}{P(U)} \]

\[ P(Y|U) \propto P(Y) \prod_{i}^{N} P(X_i|P_a(X_i)) \]

For classification equation 3.20 can be used, provided the estimation of right hand side terms are completed corresponding to all possible values of i and Y.

3.6.2 Random Bayesian Network

Random Bayesian Network (RBN) is an ensemble of Bayesian network classifiers where each classifier is trained offline on a different set of training samples (Bagging) and feature sets (random subspace method). For training with different sets of features we pseudo-randomly select subsets of components of the feature vector, that is, Bayesian network is constructed on randomly chosen subspaces. To achieve this we use the Random Subspace method as proposed in [70]. Bagging and the random subspace method helps to make a diverse classifier ensemble. Classifiers made using this randomization technique are very fast (in both training and testing), easily parallelized, inherently multi-class and also stable in the sense that classifiers have a very low variance (therefore less chance of over-fitting).

In literature we can easily find tree and naive Bayes classifiers being used in conjunction with randomization technique to boost the accuracy but in the present
work, unlike the usual trend we have used Bayes net and experimental results proves the efficacy of the approach. The low computational and memory cost of this approach makes it the right choice for online learning where both computational power and memory usage are limited.

We create each classifier in the ensemble of B classifiers by repeating the following steps B times:

1. Perform bagging to create bootstrapped training data for each classifier.
2. Randomly select F features from the pool of N features.
3. Estimate the learning parameters for the classifier.

A test sample will be classified by each classifier in the ensemble and the class which gets the majority vote by the ensemble will get assigned to the test sample.

### 3.6.3 Batch / Offline Learning of RBN

We use random feature selection and bagging to create the ensemble of Bayes net classifiers. For batch or offline learning of each classifier we use Maximum Likelihood
Estimation (MLE) of learning parameters parameters $\beta_0, \beta_1$ and $\sigma^2$ as indicated in the block diagram. The basic steps of MLE are shown in Fig.3-10. Training dataset $D = \{(X[1], Y[1]), \ldots, (X[M], Y[M])\}$, where $X[m] \in \mathbb{R}^N$, are the samples in a $N$ dimensional feature space and $Y \in \{1, \ldots K\}$ are the corresponding labels for a $K$-class classification problem.

Log likelihood of the data can be written as (assumption is that samples are independently and identically distributed (IID)):

$$
\ln(L(D : \theta)) = \ln\{\prod_{m=1}^{M} P(X[m], Y[m] : \theta)\} \quad (3.21)
$$

$$
\ln(L(D : \theta)) = \sum_{m=1}^{M} \ln(P(X[m], Y[m] : \theta)) \quad (3.22)
$$

By Bayes net chain rule we can write RHS of above equation as follows:

$$
\ln(L(D : \theta)) = \sum_{m=1}^{M} \ln\{P(Y[m] : \theta_Y)P(X_1[m]|Y[m])I\} \quad (3.23)
$$

where $I = \prod_{i=2}^{N} P(X_i[m]|Y[m], X_{ipc}[m] : \theta_{X_i|Y,X_{ipc}})$

$$
\ln(L(D : \theta)) = \sum_{m=1}^{M} \ln(P(Y[m] : \theta_Y)P(X_1[m]|Y[m])) + I \quad (3.24)
$$

where $I = \sum_{m=1}^{M} \sum_{i=2}^{N} P(X_i[m]|Y[m], X_{ipc}[m] : \theta_{X_i|Y,X_{ipc}})$

In order to maximize log data likelihood we need to maximize each summation term. First term $P(Y[m] : \theta_Y)$ in the above equation is prior probability for each class data and has been parameterized by multinomial distribution. $P(X_1[m]|Y[m])$ is parameterized by univariate Gaussian distribution which can be easily maximized. Where as term in the second summation, $P(X_i[m]|Y[m], X_{ipc}[m] : \theta_{X_i|Y,X_{ipc}})$ is parameterized by a Gaussian distribution of the form $N(\beta_0 + \beta_1 X_{ipc}, \sigma^2)$.

We can rewrite second term (let that be represented by $L_2$), using equation (24)
and changing the order of the summation:

\[ L_2 = \sum_{i=1}^{N} \sum_{m=1}^{M} \ln \left\{ \frac{1}{(2\pi\sigma^2)^\frac{3}{2}} e^{-\frac{(X_i[m]-(\beta_0+\beta_1 X_{ipc}[m]))^2}{2\sigma^2}} \right\} \]  

(3.25)

\[ L_2 = \sum_{i=1}^{N} \sum_{m=1}^{M} \left\{ -0.5 \ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}((\beta_0 + \beta_1 X_{ipc}[m]) - X_i[m])^2 \right\} \]  

(3.26)

Therefore in order to maximize \( L_2 \) we need to partially differentiate above equation with respect to \( \beta_0 \) and equate to 0:

\[ \frac{\partial L_2}{\partial \beta_0} = \sum_{i=1}^{N} \sum_{m=1}^{M} \left[ 0 + \frac{1}{2\sigma^2} 2((\beta_0 + \beta_1 X_{ipc}[m]) - X_i[m]) \right] = 0 \]  

(3.27)

On simplification we get:

\[ \sum_{i=1}^{m} X_i[m] = \sum_{i=1}^{m} \beta_0 + \beta_1 \sum_{i=1}^{m} X_{ipc}[m] = M\beta_0 + \beta_1 \sum_{i=1}^{m} X_{ipc}[m] \]  

(3.28)

Finally on dividing both sides by \( M \) we get:

\[ E_D X_i = \beta_0 + \beta_1 E_D [X_{ipc}] \]  

(3.29)

Similarly, partially differentiating equation (32) with respect to \( \beta_1 \) and equating to 0 we get:

\[ E_D [X_i X_{ipc}] = \beta_0 E_D [X_{ipc}] + \beta_1 E_D [X_{ipc} X_{ipc}] \]  

(3.30)

Above two system of equations can be represented in matrix form \((AX = B)\)

\[
\begin{bmatrix}
1 & E_D [X_{ipc}] \\
E_D [X_{ipc}] & E_D [X_{ipc} X_{ipc}]
\end{bmatrix}
\begin{bmatrix}
\beta_0 \\
\beta_1
\end{bmatrix}
= 
\begin{bmatrix}
E_D [X_i] \\
E_D [X_i X_{ipc}]
\end{bmatrix}
\]

Solving above system of equations we get:

\[ \beta_1 = \frac{E_D [X_i X_{ipc}] - E_D [X_i]E_D [X_{ipc}]}{E_D [X_{ipc} X_{ipc}] - E_D [X_{ipc}]^2} = \Sigma_{X_{ipc} X_{ipc}}^{-1} \Sigma_{X_i X_{ipc}} \]  

(3.31)
\[
\beta_0 = E_D[X_i] - \left( \frac{E_D[X_iX_{ipc}] - E_D[X_i]E_D[X_{ipc}]}{E_D[X_{ipc}^2] - E_D[X_{ipc}]^2} \right) E_D[X_{ipc}]
\] (3.32)

\[
\beta_0 = E_D[X_i] - \beta_1 E_D[X_{ipc}]
\] (3.33)

Similarly, to obtain the value of \( \sigma^2 \) that maximizes \( L_2 \), we partially differentiate equation (33) with respect to \( \sigma^2 \) and equate to 0 to obtain \( \sigma^2 \):

\[
\sigma^2 = (E_D[X_iX_i] - (E_D[X_i])^2) - \left( \frac{E_D[X_iX_{ipc}] - E_D[X_i]E_D[X_{ipc}]}{E_D[X_{ipc}^2] - E_D[X_{ipc}]^2} \right) \left( E_D[X_iX_{ipc}] - E_D[X_i]E_D[X_{ipc}] \right)
\] (3.34)

This can also be expressed in terms of covariance (between parents \( X_{ipc} \) and child \( X_i \)) and variances (of child and parents):

\[
\sigma^2 = \Sigma_{X_iX_i} - \Sigma_{X_iX_{ipc}} \Sigma_{X_{ipc}X_{ipc}}^{-1} \Sigma_{X_iX_{ipc}}
\] (3.35)

This concludes the maximum likelihood estimation of the learning parameters \( \beta_0, \beta_1, \sigma^2 \) for each \( n_{ith} \) RV (an attribute or feature). This way we learn \( \theta_{X_i|Y,X_{ipc}} \) for each value of \( i \). For each value of \( i \) we get a set of \( \beta_0, \beta_1, \sigma^2 \), which describe the univariate Gaussian distribution (equation 3.16) i.e. \( \theta_{X_i|Y,X_{ipc}} \).

### 3.6.4 Online RBN:

To the best of our knowledge this is first time a randomized ensemble of Bayes net classifier has been used in an online manner. Here we assume the structure of the network to be fixed initially and later learning the parameters online. Among one of the previous attempts to use randomization with naive Bayes (the most basic version of BN) was Prinzie et al. [122]. Where as fair amount of attempts have been made to make the online learning of learning parameters [57] [87]. We propose a method to update the values of learning parameters of BN in online fashion. This is a generic method and hence versatile. It processes a sample only once to update the learning.
parameters and discards it later.

We fit a Gaussian distribution of the form $N(\beta_0 + \beta_1 X_{ipc}, \sigma^2)$. In order to make the learning online of the networks similar to as shown in Fig. 3-9, we need to update learning parameters as each sample $X$ comes in. The parameters that need to be evaluated in online manner are (as they are the building block of the learning parameters $\beta_0, \beta_1$ and $\sigma^2$, as described in equations (3.31), (3.33) and (3.35)): $E_D[X_i], E_D[X_{ipc}], E_D[X_{ipc}X_{ipc}]$ and $E_D[X_iX_{ipc}]$. We propose to update these parameters in the following manner:

Initialization of the learning parameters is done by 1 sample as given in [87]. Repeat, steps below for all the available labeled training samples:

$$c_k = c_k + 1; \eta_k = \left(1 - \frac{\alpha}{c_k} + \alpha\right)$$ (3.36)

where $\eta_k$ is learning rate for class $k$, where $c_k =$ no. of samples used so far for training class $k$, $\alpha$ = decides the length of memory of the classifier ($\alpha < 1$).

$$E_D[X_i](t) = (1 - \eta_k)E_D[X_i](t - 1) + \eta_kX_i$$ (3.37)

$$E_D[X_{ipc}](t) = (1 - \eta_k)E_D[X_{ipc}](t - 1) + \eta_kX_{ipc}$$ (3.38)

$$\sigma^2_{X_{ipc}}(t) = (1 - \eta_k)\sigma^2_{X_{ipc}}(t - 1) + \eta_k(X_{ipc} - E_D[X_{ipc}](t))^2$$ (3.39)

We know that covariance between two RVs, A and B is given by:

$$\sigma_{A,B} = E[AB] - E[A]E[B]$$ (3.40)

$$E_D[X_{ipc}X_{ipc}](t) = \sigma^2_{X_{ipc}}(t) + \{E_D[X_{ipc}](t)\}^2$$ (3.41)

Similarly:

$$\sigma_{X_iX_{ipc}}(t) = (1 - \eta_k)\sigma_{X_iX_{ipc}}(t - 1) + I$$ (3.42)
where \( I = (X_i - E_D[X_i](t))(X_{ipc} - E_D[X_{ipc}](t)) \) using equations 3.40 and 3.42 we can write

\[
E_D[X_iX_{ipc}](t) = \sigma_{X_iX_{ipc}}(t) + E_D[X_i](t)E_D[X_{ipc}](t) \tag{3.43}
\]

Experiments conducted suggest that proposed Randomized BN (augmented structure of NB) is much more accurate than RNB as shown in the result table. Proposed online parameter learning method does converge with its batch learning counterpart in fact, most of the time it is more accurate than its offline counterpart Fig. 3-11. This approach does have potential to improve classification accuracy further as the structure chosen is based on mutual information between the attributes but there are more sophisticated methods present in the literature which can be used.

### 3.7 Online Semi-Supervised Learning: Bayesian Network approach

In a typical case of semi-supervised learning, we have missing information in the data and the task is to estimate the value of that missing information. Usual solutions to this would be to use Expectation Maximization (EM) [40] [114]. In the E-step, the posterior probability of data points is computed, while in M-step, parameters of the learning model are computed. An iterative algorithm like this is not suitable in our context of online learning since our goal is to process training samples one at a time and to avoid their storage. This is a real-time scenario when we need to deal with streams of data or a huge dataset (which cannot be processed fully in a single step). Hence a method that updates the model in a continual and an evolutionary manner can be used in the present study. This problem has been addressed by a few researchers such as Stauffer et al. [153], Radford et al. [112], Bouchachia et al. [16] and Lee [91]. In this study we use some findings of Lee [91].

We have incorporated a regulating variable \( \lambda \) in the proposed method. This parameter moderates the contribution made by the unlabeled data by reducing (by different amount which depends on time) the learning rate (\( \eta \)) and hence the weight
of the unlabeled samples during step M. The definition of variable $\lambda$ is given in equation 3.44. The best value of parameters which decides the value of $\lambda$ in practice is decided using a validation set $\lambda_k$ reduces the learning rate for all unlabeled data (by different amount which depends on at what time instance the sample is coming) but the value of $\eta_k$ increases for the correct class sample because of the higher posterior value $q_k$. It is a slow hill-climbing process in which the correct class sample makes the hill-climbing step greater towards the hill top, whereas the incorrect class sample makes the step small and in the opposite direction, so effectively we get closer to the hill top after processing many unlabeled data samples. Details of the proposed incremental (online) EM algorithm for creating an incremental semi-supervised learning system are as shown below:

1) Train RNB online with a few labelled training samples. Repeat the following steps for every unlabelled or labelled data sample for better estimation of learning parameters.

2) E Step (of proposed online EM):

If incoming new sample is unlabelled then use trained classifier to find the posterior $q_k = P(Y = y_k | X)$ corresponding to all $k$ (class). Else $q_k = 1$.

3) M Step (of proposed online EM):

$$c_k = c_k + q_k \lambda; \eta_k = q_k \left( \frac{1 - \alpha}{c_k} + \alpha \right) \lambda_k$$

and

$$\lambda_k = \left( \frac{1}{(c_k)^a + b} \right) + c$$

(3.44)

(3.45)

where $\lambda_k$ = variable weight factor applied to moderate the contribution of unlabeled data in the parameter estimation step ($\lambda_k = 1$ if the sample is labeled) and $\eta_k$ is a learning rate parameter. And the estimation of the learning parameters $E_D[X_i], E_D[X_{ipc}], E_D[X_{ipc}X_{ipc}]$ can be done using the same equations (3.37-3.43) but using the new learning rate $\eta_k$ as shown in equation 3.44. The value of three constants $a$, $b$, $c$ are typically set (which are usually small) using validation set of 100 samples per class. In our case the values were $a=1.5$, $b=20$ and $c=0.003$. 

71
4) Repeat step E and M (just once) for each labeled or unlabeled sample left.

The above algorithm is capable of handling both labeled and unlabeled data at the same time to update the parameters of the learning system. Experiments indicate that this online semi-supervised learning approach does converge Fig. 3-12 to the accuracy achieved by the offline supervised RBN with far less labeled samples. The gain in accuracy is remarkable when the labeled data used is less which is evident from Fig. 3-13.

3.8 Experimental Evaluation

In this section we discuss Tamil handwritten character recognition, experimental settings, the database, issues with the recognition problem at hand and results. We demonstrate our proposed classification method performance on the recognition of Tamil (one of the Indian scripts) handwritten base characters. Base characters constitute only vowels and consonants. It is a 30 class problem. The IWFHR 2006 competition handwritten online dataset [75] was used for experimental results reported in this paper. Each character in the above online dataset is represented by a discrete variable number of points, and each point is described by its x and y co-ordinates.

For all the experiments, the data was pre-processed. For pre-processing, we normalized each character, maintaining their aspect ratio and also resampled each character into 60 equidistant points. The five features extracted per point are a) x-coordinate b) y-coordinate c) x-derivative d) y-derivative [77] and e) pen-direction angle as described in [145].

In all the Bayes net experiments we fitted a Gaussian of the form \( P(X_i|X_{ipc}, Y) = N(\beta_0 + \beta_1 X_{ipc}, \sigma^2) \) for each feature, for each class i.e. \( P(X_i|X_{ipc}, Y = y_k) \), where i and k represent the \( i^{th} \) feature and \( k^{th} \) class respectively where as ipc denote continuous parent of node i. For the offline RBN using the validation set (of size 100 samples per class), we decided upon the best number of features and classifiers in the RBN ensemble which was found to be 20 and 50 respectively. So we performed online bagging [118] to create 50 different datasets of size 350 each, chosen from a pool
Figure 3-11: Plotted accuracies of RBN and Online RBN (ORBN) shows that as the number of training samples increases, the ORBN seems to perform better.

of 350 samples. Consequently, 50 different classifiers were created by using a set of 20 random features out of the pool of 300 and then training a classifier using a bootstrapped dataset. Classification of a test sample is performed using the majority vote. All the results reported are on a separate test set of 100 samples per class. Results in TABLE 1 show that RBN produced better recognition accuracy when compared to BN or Naive Bayes.

3.9 Discussion II

The conditional independence assumption on which Naive Bayes is based upon is readily violated in any real world application and such is the case for handwritten character recognition. Despite that, it manages to give good accuracy as we know that a classification estimate is only a function of sign (in a 2 class problem) of function estimation. Though the estimations are poor, classification accuracy can still be high. RBN is used to improve accuracy further.
Online−SS−RBN and offline RBN

Figure 3-12: Plotted accuracy of RBN and Online Semi-Supervised RBN shows that unlabeled data boosts the accuracy sharply when there is a smaller number of labeled samples and as the number of labeled sample increases, boost in accuracy reduces.

Table 3.2: Table below show the detail of each classifier. 20 random features were used to create 50 different classifiers to form the ensemble (RBN and ORBN). In case of SSORBN L and UL stands for labeled and unlabeled respectively.

<table>
<thead>
<tr>
<th>Classifier Name</th>
<th>Training/Testing</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayes Net (BN)</td>
<td>350/100</td>
<td>83.85</td>
</tr>
<tr>
<td>Random BN (RBN)</td>
<td>350/100</td>
<td>86.10</td>
</tr>
<tr>
<td>Online RBN (ORBN)</td>
<td>(Initial 2 + 350)/100</td>
<td>87.80</td>
</tr>
<tr>
<td>Semi-supervised ORBN (SSORBN)</td>
<td>(100L+200UL)/100</td>
<td>88.48</td>
</tr>
<tr>
<td>SSORBN + DTW</td>
<td>(100L+200UL)/100</td>
<td>92.34</td>
</tr>
<tr>
<td>Naive Bayes (NB)</td>
<td>350/100</td>
<td>78.26</td>
</tr>
<tr>
<td>SVM</td>
<td>350/100</td>
<td>90.68</td>
</tr>
<tr>
<td>HMM</td>
<td>350/100</td>
<td>87.82</td>
</tr>
</tbody>
</table>

A method to learn online in a semi-supervised manner using RBN is proposed. $c_k$ is one of the sufficient statistics which is updated at every step of incremental learning, and is just the count of samples from each class. $c_k$ is one of the parameters
Figure 3-13: Plotted accuracy shows that the Online Semi-Supervised RBN accuracy increases sharply with unlabeled training samples. Increase in accuracy is maximum for the classifier, which was pre-trained with smaller number of labeled samples.

that decides the learning rate $\eta_k$. It depends on the learning rate at a particular instance of time i.e. what weight will be given to the new sample. Initially the $\eta_k$ value will be high, as the sample count would be small, thus giving more weight to the new incoming sample, but as the count grows with time, $\eta_k$ reduces and finally as time tends to infinity, it converges to $\alpha << 1$. This convergence to $\alpha$ makes sure that learning does not stop ever thus making the incremental supervised learning scheme adaptable.

Analysis has been done over the static $\lambda$ in [87] and results have shown that, how its large value can otherwise hurt the accuracy of the classifier. On the similar line we have studied the dynamic $\lambda_k$. This variable $\lambda$ changes with time and finally converges to a small value as time tends to infinity so that learning rate never turns to zero. In the initial phase of learning the count of samples $c_k$ is small, therefore the value of $\lambda_k$ will be more, there by giving more weight to the contribution of unlabeled data (that means more information is extracted from the unlabeled data
Figure 3-14: Plotted accuracy shows the superiority of variable weighting factor over constant weighting factor.

in the initial semi supervised learning phase). The experimental results proves that variable $\lambda_k$ better extracts information from the unlabeled data which manifests in form of higher accuracy Fig. [3-14].

Another crucial parameter used in incremental EM is $\alpha << 1$, which can be thought of as the one which decides length of the memory of the learner. If the value of $\alpha$ is large then (learning rate $\eta$ increases) the learner has a smaller memory and gives more weight to the most recently encountered data and forgets quickly. Thus it can be easily understood that $\alpha$ decides the generalization capacity of the learner.

3.10 Conclusion

The goal of the present work was to make a Semisupervised online NB and BN classifier which should be more accurate than its supervised and offline counterpart even after using less labeled samples. This was successfully demonstrated on Tamil handwritten character recognition using a standard dataset. Table 1 shows that the
The proposed method was almost as accurate as SVMs and HMM.
Chapter 4

Incremental Active Semi-supervised Learning

In this chapter we perform incremental learning in active fashion. This combination of active and incremental learning is very natural. It is so because we know that incremental learning is most suitable for the cases where the data availability is massive and there are chances of presence of redundancy in the dataset. The size of dataset is huge because of the continuous inflow of unlabeled data and labeling all of them manually would be too cumbersome and expensive to be done for supervised learning. Therefore, in order to make use of unlabeled data, various semi-supervised techniques are used. However, alternatively active learning can also be used to make a better use of unlabeled data. The learning is faster and more accurate, using way less sample when compared to semi-supervised learning.

The key hypothesis is that if the learning algorithm is allowed to choose the data from which it learns, it will perform better with less training. It is generally a desirable property of any learning system since labeling the data is expensive, time consuming and in some cases very difficult to obtain, as is the case with annotating dataset for speech recognition for a language with different dialects. Active learning is sometimes called query learning and it is a subfield of machine learning. An active learning system overcomes the problem of labeling huge dataset by asking queries in the form of unlabeled instances to be labeled by a human annotator. In this way,
the active learner aims to build an accurate learner using as few labeled instances as possible, thereby minimizing the cost of obtaining labeled data. The key is to choose relatively the most informative data samples and subsequently pose query to the human annotator for their labels. There are several different query strategies that have been used to decide which instances are most informative.

4.1 Active Learning settings

There are several kind of active learning/problem settings but primarily there are three, (a) membership query synthesis, (b) stream-based selective sampling, and (c) pool-based sampling. These settings assume that queries take the form of unlabeled instances to be labeled by the human expert.

4.1.1 Membership Query Synthesis

This active learning setting is one of the oldest active learning scenario that was studied [6]. In this case, the learner may request labels for any unlabeled instance in the input space, including queries that the learner generates from the beginning, rather than those sampled from some underlying natural distribution. The key here is that the learner may actually construct instances from the underlying space, which may not be a part of any actual pre-existing data. However, this may lead to challenges in the sense that the constructed examples may not be meaningful and labeling such instances may be awkward for the human annotator. For example, in case of handwritten character recognition, the query instances may not closely relate to any of the character classes and similarly in case of natural language processing the streams of text or speech generated may sound complete gibberish. However, the usability of the approach clearly depends upon the underlying scenario. It is demonstrated by the authors in [80] the utility of this approach. They employ a robot scientist which can execute a series of autonomous biological experiments to discover metabolic pathways in the yeast Saccharomyces cerevisiae. The stream-based and pool-based scenarios have been proposed to address limitations of membership query based active learning.
4.1.2 Selective or Sequential Sampling

Learner decide whether to choose a particular instance which is generated from the underlying data distribution, as a query or not [9]. It is a viable method to be used in cases where the data is abundant and cheap. For example, data that is continuously flowing, the learner can afford to discard the redundant ones and select only the most informative ones for querying, based on some active learning query strategy. Querying strategy can be anything that depicts the informative measure of an instance. The information measure can be related to whether the sample is from the "uncertain region" or whether the informative measure has crossed a particular threshold or one can select the samples just randomly with the hope to grab the most informative one with greater chances. Selective/Sequential sampling is synonymous with the streaming scenario, since the decisions about querying an instance need to be made in real time. However, "Selective sampling" term is overloaded, because authors of many works such as those in [135] [108] use it to refer "pool-based sampling".

4.1.3 Pool-based Sampling

There is a boom in the digital world which creates immense amount of data for majority of the real world learning problems. And assuming the labeling of data to be expensive affair, generally we have a relatively smaller chunk as labeled whereas the unlabeled data mass is huge. This is the ideal scenario for pool-based sampling [96]. The learner initially learns using the labeled data and later uses some sampling technique based on certain querying strategy to select most informative instances and ask the human annotator for their labels. Later those labeled samples are added to the previously labeled dataset and the process to update the learning model is initiated. Typically, instances are queried in a greedy fashion, according to an informative measure used to evaluate all instances or may be a sub-sample of the huge unlabeled data mass. There is a very subtle difference between selective/sequential and pool-based sampling, i.e. in the former case informativeness of the instances are checked one by one sequentially whereas in the latter, informativeness of all the samples are checked,
graded and later query instance is selected. This is the sampling technique which appears to be the most intuitive and consequently have been widely studied in various real-world learning applications, for example text classification [96] [157] [72], information extractions [141], image classification and retrieval [166], video classification and retrieval [67].

Though the active learning seems to be very intuitive, it does not always make the model more accurate because of certain challenges related to the algorithmic aspects like selection bias [13] or it may also happen because of the inherent nature of the data. Active learning certainly improves the sample complexity of the learning system. Authors in [36] found that the number of labels needed to learn actively can be logarithmic in the usual sample complexity of passive learning.

Pool-based sampling happens to be the natural choice in many learning applications but there are several other scenarios where sequential sampling would be more appropriate. For example, learning setting where there is a memory and processing power constraints as is the case with the mobile and embedded device based applications (with time this bottleneck appears to be disappearing). Incremental learning would be an ideal scenario for sequential sampling where we do not have huge dataset to start with, instead there is a continuous flow of data.

4.2 Query Strategies

Beyond the scenario of active learning lies its key aspect and that is the strategy to choose important query instances and ask the annotator for its label. The goal is to make the learning more accurate by better learning the underlying distribution of the data. Essentially we need to map the contour of separation between the classes precisely. Boundary region can be considered to be the most uncertain region i.e. the classifiers most disagree over the samples lying along the boundary. Therefore one intuitive query strategy could be to select data samples from this uncertain region i.e. boundary between the classes for training the learner. But sometimes this strategy does not work, especially when we over-sample from this region. It is so because
samples from either sides of the boundary do not completely represent the respective classes and thus the learner’s accuracy deteriorates. This problem occurs more when the dataset is very noisy. This has led the researchers to design query strategy that would focus directly on error minimization or finding a better representative of different classes within the dataset.

Due to this reason the authors of [3] have broadly classified the query strategies into three separate classes: (1) Heterogeneity-based models under which comes three most prevalent querying strategies and they are; uncertainty sampling, query-by-committee and expected model change. These models only look at the heterogeneity behavior of the queried instances rather than the effect of its addition on the performance of a classifier on the remaining unlabeled dataset. (2) Performance based model: strategies under this section directly try to reduce either the variance or error (i.e. optimizes the final outcome) of the learner. (3) Representativeness-based models: These models attempt to create data that is as representative as possible of the underlying population of training instances. For example, density-based models.

In general there is no universal querying strategy to actively learn therefore, one has to deal with tradeoff depending on the nature of the data, aim of machine learning application and also the learning approach used for modeling. For instance, it is very easy to use query-by-committee sampling strategy with ensemble based learning models and in case the classifiers in the ensemble are probabilistic models, then uncertainty sampling can also be easy to use. In this work we use uncertainty and query-by-committee sampling along with ensemble based learner to study the effect of active learning when used in incremental learning scenario. Therefore we shall discuss only these two sampling techniques in detail next.

It is not often that one finds work which deals with this combination of learning i.e. incremental active semi-supervised learning [60] in the context of ensemble of randomized Bayesian Network. The authors in [93] [64] do address the combination of semi-supervised and active learning using SVM but in batch mode. This is the most natural kind of learning setting, which makes use of both labeled and unlabeled data (semi-supervised learning) and asks for label from the annotator for some samples
which may improve the learning (Active learning). It also is not allowed to stop and train from the scratch or store lot of data (incremental/active) learning. We have tried to work in this setting where we first we assume that we are allowed to access the whole unlabeled dataset and also have access to the data that have been labeled actively. Later we also address the scenario where we don’t have access to whole unlabeled dataset and all that is allowed is to process one unlabeled instance at a time.

4.2.1 Uncertainty Sampling

This belongs to the Heterogeneity-based models and perhaps the most commonly used query strategy [96]. The key idea behind this strategy is to query the data sample which the classifier is least sure about how to label. It is easy to implement this strategy with a probabilistic classifier. A typical probabilistic classification model output is the posterior probability \( P(y|x) \) i.e. given the sample \( x \), it belongs to class \( y \) with probability \( P(y|x) \). For an instance, when using a probabilistic model for binary classification, uncertainty sampling simply queries the instance whose posterior probability of being positive is nearest 0.5 [96]. In general we can say that the strategy queries the samples about which the classifier is least confident, which can be expressed as below [141]:

\[
x^*_{LC} = \arg \max x \ 1 - P_\theta(\hat{y}|x)
\]  

where \( \hat{y} = \arg \max_y P_\theta(y|x) \) i.e. the class label with the highest posterior probability under the model \( \theta \). The probabilities predicted by the classifier should be normalized so that they sum up to 1. This is important, since many classifiers such as the unnormalized naive Bayes classifier often predict probabilities that do not sum to 1. Uncertainty Sampling has been successfully tried out for sequence modeling for information extraction [141] [33] [96]. But the authors in [132] observed that the above scheme only considers information about the most probable label and neglects rest of the label distribution and hence in order to make use of it, proposed margin
based sampling defined as:

\[
x^*_M = \arg\max_x P_\theta(\hat{y}_1|x) - P_\theta(\hat{y}_2|x)
\]

(4.2)

Where \(\hat{y}_1\) and \(\hat{y}_2\) are the first and second most probable labels. Essentially it prompts to pick the instance which has the least difference between the probabilities to belong to 2 classes. Data samples with small margins are more ambiguous and vice versa, thus knowing the true label would help the model discriminate more effectively between them. Overall this scheme also neglects the most part of the label distribution, hence the popularity of the entropy based uncertainty measure:

\[
x^*_H = \arg\max_x - \sum_i P_\theta(y_i|x) \log P_\theta(y_i|x)
\]

(4.3)

where \(y_i\) refers to all the class labels. This entropy based uncertainty measure is an information theoretic measure which can be used to represent the level of impurity or uncertainty in the learning applications. On the similar line Gini-Index is defined which is directly proportional to the entropy based measure:

\[
x^*_G = \arg\max_x 1 - \sum_i P^2_\theta(y_i|x)
\]

(4.4)

In practice the entropy-based approach generalizes easily to probabilistic multi-label classifiers and probabilistic models representing structurally more complex data samples like sequences, therefore it is more commonly used. Though it is theoretically sound as well as very intuitive, it is not a universal query strategy for all machine learning problems. Empirical comparisons [133] [141] of these measures have yielded mixed results, suggesting that the best strategy may be application dependent. It is found in [140] that entropy seems appropriate if the objective function is to minimize log-loss. And margin based measure is more appropriate if we aim to reduce classification errors, as they prefer instances that would help the model better discriminate among specific classes.

It should be pointed out that the uncertainty based sampling has also been used
with non-probabilistic or non-Bayesian models that do not output probabilities. It is sufficient that the model outputs some kind of confidence for each class and later apply methods/heuristics to convert them into probabilistic values for each class. Uncertainty sampling was first used for tree-based classifier [94], later similar approaches of active learning was also used in nearest neighbor (i.e. instance or memory based) classifier [98] by allowing each neighbor to vote on the class label of x, with the proportion of these votes representing the posterior label probability. Authors in [157] applied active learning with Support vector Machines (SVM) by querying the samples close to the margin area for the purpose of text classification. Beyond the applications of classification it is also possible to apply uncertainty sampling in the application of regression. Using the fact that under the gaussian assumption, the entropy of a random variable is a monotonic function of its variance [151], those samples can be queried for which the predictor has the highest output variance in its predictions. Therefore it can be considered that, in spirit, this approach is very similar to entropy-based uncertainty sampling for classification.

4.2.2 Query-By-Committee

In this approach a set of models are trained using a labeled dataset $L$ to form a committee $C = \theta^{(1)}, \theta^{(2)}..., \theta^{(C)}$ of competing hypotheses [143]. In order to decide which unlabeled data instance is most informative, all the committee members are allowed to vote for its class. The sample for which the committee most disagrees upon is chosen to be the most informative and therefore chosen as the query sample. What QBC achieves by getting the committee vote is the version space minimization, so this querying framework can be considered to be theoretically more motivated. In a machine learning problem we search for the best hypothesis in the version space. And in active learning we constrain the size of the version space using as few sample as possible. Using QBC strategy we query the most controversial samples and this constrains the search space precisely.

A disagreement criterion and diverse (so that it represents different regions of the version space) committee of models are required for implementing QBC. Immense
amount of work has been done to make an ensemble of classifiers diverse, for both
generative and discriminative prediction models. For achieving diversity in generative
model committee, a random sampling from the posterior distribution \( P(\theta|L), \) of
the model parameter is often used \cite{102} \cite{34}. And for other non probabilistic and
discriminative models; boosting, bagging and random sampling of features from the
feature space and their variants have been commonly used in literature to increase
diversity among the committee members. There is no general agreement in the litera-
ture on the appropriate committee size to use, which may in fact vary by model
class or application. However, evidences show that even small committee sizes of two
or three suffice \cite{143} \cite{102}.

Measurement of disagreement can be done by various methods as proposed in
the literature which suits the application at hand. Essentially there are two main
measurement approaches that have been proposed, one is vote Entropy \cite{34}:

\[
x^*_{VE} = \arg\max_x - \sum_i \frac{V(y_i)}{C} \log \frac{V(y_i)}{C}
\]

(4.5)

where \( y_i \) varies over all possible labels, \( V(y_i) \) is the number of votes that a label
receives from the committee member’s predictions and \( C \) is the number of mem-
bers in committee. Another disagreement measure is average Kullback-Leibler (KL)
divergence as proposed in \cite{102}:

\[
x^*_{KL} = \arg\max_x \frac{1}{C} \sum_{i=1}^{C} D(P_{\theta^{(i)}}||P_C)
\]

(4.6)

where:

\[
D(P_{\theta^{(i)}}||P_C) = \sum_i P_{\theta^{(i)}}(y_i|x) \log \frac{P_{\theta^{(i)}}(y_i|x)}{P_C(y_i|x)}; \text{ and}
\]

(4.7)

\[
P_C(y_i|x) = \frac{1}{C} \sum_{i=1}^{C} P_{\theta^{(i)}}(y_i|x)
\]

(4.8)

\( \theta^{(i)} \) is the model parameter of a particular member of the committee and \( C \) re-
present the committee as a whole. \( P_C(y_i|x) \) is the consensus probability that sample
\( x \) belongs to class \( y_i \). KL divergence \cite{86} is an information-theoretic measure of the
difference between two probability distributions. So this disagreement measure considers the most informative query to be the one with the largest average difference between the label distributions of any one committee member and the consensus. Similarly other information theoretic approach like Jensen-Shannon divergence have also been applied to measure the degree of disagreement [105]. Apart from QBC there are other strategy too which tries to achieve the same goal of minimizing the size of version space, for e.g. using selection sampling with a pair of very differently trained neural network [28] and using pool-based margin strategy with SVM [157]. Authors in [68] concludes that the version space can grow exponentially with the size of dataset, therefore version space of any model class cannot be explicitly represented in practice. Hence The QBC framework, rather, uses a committee to serve as a subset approximation.

4.2.3 Expected Model Change

This approach of query selection is only applicable to the discriminative probabilistic models. It relies on the premise that if the label of an instance is known then what will be the change in the gradient during the learning process and consequently select the data sample which creates the greatest gradient change. Therefore it can be applied if the learning system uses gradient-based optimization algorithm. It was proposed in [142] in conjunction with multiple instance learning and was again applied with a probabilistic discriminative learning model namely conditional random field (CRF) in [141] for sequence labeling task.

Let $\nabla \ell(L; \theta)$ be the gradient of the gradient function $\ell$ with respect to the model parameter $\theta$ [155]. Let $\nabla \ell(L \cup (x, y); \theta)$ be the new gradient that would be obtained by adding the training tuple $(x, y)$, to $L$. Since the query algorithm does not know the true label sequence $y$ in advance, we instead calculate the expected gradient length (EGL) [141]:

$$x_{EGL}^* = \arg\max_x \sum_i P_\theta(y_i | x) \| \nabla \ell_\theta(L \cup (x, y_i)) \|$$ (4.9)
\[ || \cdot || \text{ above represent the euclidian norm of each resulting gradient vector. The following approximation can be used: } ||\nabla \ell_\theta(L \cup (x, y_i))|| \approx ||\nabla \ell_\theta(x, y_i)|| \text{ because } \ell \text{ converged in the previous round of training therefore } ||\nabla \ell_\theta(L)|| \text{ should be nearly zero. This approximation also increases the computational efficiency. This approach generally works well but the computation gets very tasking if the feature dimension and number of classes are very high. Proper scaling of the feature vector is important or else the informativeness of a sample may get over estimated because of few high feature values. This issue can be handled by regularizing the parameters.}

4.2.4 Expected Error Reduction

It is possible to draw a parallel between uncertainty sampling and expected error reduction as realized in [3]. The two approaches are complementary to each other in a sense that, uncertainty sampling maximizes the label uncertainty of the queried sample whereas expected error reduction minimizes the expected label uncertainty (i.e. expected future error) of the remaining instances \(U\) (remaining unlabeled instances), when the queried instance \(x\) is added to the labeled data (i.e. \(L \cup (x, y)\)). Finally, query the sample which produced the least future expected error. Since the class of the queried sample is unknown, true future error cannot be found, therefore in order to approximate it, we evaluate the expectation over all possible labels under the current model \(\theta\) [3].

This querying strategy has been successful in different contexts with various classification methods like Naive Bayes [126], Logistic regression [65] and SVM [108]. This approach is very intuitive and it works well as a query strategy but in most of the cases the computation cost drastically increases. This strategy has to calculate the expected error for each query from the unlabeled dataset and in addition to that the model need to be retrained every time a query is labeled. This process iterates over the whole unlabeled dataset.

It turns out that for the non parametric learning models like Gaussian Random field [170], incremental learning procedure is efficient and exact. But it is not so for many other kind of learning models, for example; simple binary logistic regression’s
time complexity to choose next query is $O(U, L, G)$. $U$ indicates the number of unlabeled training instances, whereas $L$ and $G$ denote current labeled training set strength and number of gradient computations required by the optimization procedure (until convergence) respectively. Things get worse if the classification problem is multi-class instead of binary. Moreover if the unlabeled data is not available at once, instead if the data comes in continuous flow (which is typically the case with any online and some incremental learning scenario), it then gets hard to implement this query strategy in that framework. Consequently this query framework is more often used in binary classification problems or it is used with monte-carlo sampling trick to bring down the size of unlabeled dataset.

### 4.2.5 Expected Variance Reduction

Note that, as the error of an instance set reduces, the corresponding variance also goes down. Typically the generalization error can be expressed as the sum of the true label noise, model bias and variance [56] [140]. It is noted that minimizing models’s variance, then, is guaranteed to minimize the future generalization error of the model (since the learner itself can do nothing about the noise or bias components) [140].

Advantage of using this querying framework is that it sometime has closed form solution (i.e. greater computational efficiency) for many classification methods like neural networks, mixture models, or linear regression [28][29][99]. Despite that, sometimes this strategy finds itself on the wrong side. Estimating output variance requires inverting a $K \times K$ matrix for each new instance, where $K$ is the number of parameters in the model $\theta$, resulting in a time complexity of $O(UK^3)$, where $U$ is the size of the query pool (i.e. number of unlabeled samples). For some machine learning application like natural language processing, this parameter value is high which makes the matrix inversion less efficient. But there are ways to handle matrix inversion better, for example use of principal component to reduce dimensionality of parameter space [72].
4.2.6 Density-Weighted Methods

The basic difference between Heterogeneity-based models and Performance-based models is: while the former evaluates uncertainty or disagreement of every instance individually, latter considers the model error as an aggregate. This makes the performance based model more robust towards the querying an outlier sample. Outliers are the nonrepresentative of the data. In order to avoid querying the nonrepresentative elements, Density-Weighted methods give more weight to instances which represent the data of a concerned class more. This is achieved by weighting dense regions of the input space to a higher degree during the querying process. It is believed, elements in the dense region in the feature space would share similar class label and characteristics. For example in a binary classification problem, instances which lie on the boundary (area where learner are least certain) or close to it, are not necessarily the representative of either classes, if they are located very far from their respective class’s high density area (where most of their class members are present). Therefore querying such instances does not always make sense. On the contrary, performance based models, while estimating future errors and output variances (utilizing the unlabeled pool $U$), the strategies implicitly avoid querying possible outliers.

In order to overcome this problem, intuitively the data distribution should be modeled explicitly during query selection. Based on this logic authors in [141] have recommended Information Density framework. This method combines the heterogeneity behavior (like uncertainty or a QBC based) of the queried instance with a representativeness function from the unlabeled set $U$ in order to decide on the queried instance. Crux of this procedure is that for an instance to be chosen as an informative query sample, it should not only be located in the uncertain zone but also be located in the dense region of the input space (i.e. be a representative of the underlying distribution). A query is fired in the following manner [141]:

$$x_{ID}^* = \arg\max_x \phi_A(x) \times \left( \frac{1}{U} \sum_{u=1}^{U} \text{sim}(x, x^{(u)}) \right)^\beta$$  \hspace{1cm} (4.10)
\[
\text{sim}_{\cos}(x, x^{(u)}) = \frac{x \cdot x}{\|x\| \times \|x\|} \tag{4.11}
\]

where \(\phi_A(x)\) is an informativeness function based on any query strategy (uncertainty or QBC or margin etc.). The second term is a weight to the informativeness of \(x\) by the average similarity that the query sample holds with the samples in the unlabeled dataset set. Parameter \(\beta\) controls the relative importance of the density terms. Function \(\text{sim}(x, x^{(u)})\) calculates the similarity between the query sample \(x^{(u)}\) and any other unlabeled sample \(x\). A common choice is cosine similarity. First introduced in [141] but different variants (density weighted QBC) can be found in [102] in the context of text classification with Naive Bayes classifier. There are several other instances where density weighted approach have been used in various contexts [162] [113] etc. All the query strategy discussed above improves the performances of the learner even after processing much lesser number of instances when compared to the batch learning mode.

### 4.2.7 Diversity-Weighted Methods

This approach is very similar to Density weighted. But the fundamental difference is that, in the density based approach it gives more importance to the similarity between the query instance and rest of the unlabeled data where as, in the diversity approach it gives importance to diverseness of the query among the previously queried instances. It assumes that, a sample could be representative of the unlabeled data but it may be very similar to the samples already labeled and used for training, therefore, it would contribute less and should be avoided. This argument seems to be plausible and have been shown to work well in clinical text classification, sequence labeling contexts [46] [78] etc.. A query is chosen in the following manner:

\[
x_{IDiv}^* = \arg \max_x \phi_A(x) \times \left(1 - \frac{1}{L} \sum_{l=1}^{L} \text{sim}(x, x^{(l)})\right)^\beta \tag{4.12}
\]

As in the density weighted case \(\phi\) is any arbitrary informativeness function, \(L\) is
the strength of labeled samples that has already been used for training the ensemble. Value of $\beta$ is generally taken as one. One can find a better value for it using validation set. On the similar line authors in [78] have also created a combination of both diversity and density weighted method and found that to be working on the par with the state-of-the-art methods or sometimes better.

4.3 Empirical Evaluation

In this section we outline the details of the experiments conducted related to Active learning. Experiments have been performed to study the behavior of active learning strategies in the incremental learning paradigm. In other words we want to study what benefits in terms of performance can be yielded by carrying out semi-supervised incremental learning in the active fashion. Considering the fact that in this thesis we are using Bayesian Networks to build a battery of diverse learners that makes few active learning query strategies stand out from the rest. It is so because of the ease with which they can be implemented, moreover, efficiency and performance-wise they are among the best.

4.3.1 Data used

We have used the Tamil language HP data set. Its description and preparation (i.e. preprocessing and feature extraction) can be found in the first chapter.

4.3.2 Classification Method

In our case, the learning paradigm is incremental, where we update the learning parameter in the online/incremental fashion. We construct the Bayes net as done in the previous chapter i.e. by augmenting the Naive Bayes structure by connecting the two feature nodes based on the mutual information. The constraint is that every feature node has exactly two parents (except the first node which has one parent, which is the class node $Y$) where one parent of every feature node is essentially the
class node. Class node does not have any parent. The way we create a diverse ensemble of classifiers is explained in the previous chapter i.e. by online bagging and randomly selecting the features. The only difference is that here we use active learning techniques to select the query instance, ask the human annotator for its label and use it to update the learning parameters of every member classifier (only of the class concerned). Repeat the process until the jump in accuracy is substantial.

4.3.3 Active Learning methods

We use an incremental, pool-based, active learning framework. In the incremental approach, a model is not trained from scratch at each iteration: instead, its parameters are updated in successive iterations. The two key factors about the learner used that makes the choice of querying strategy easy: (1) member learner of the ensemble are probabilistic models (2) it is ensemble of members of diverse (because of the way they have been trained) characteristics. The strategies that stands out and we will be working with are: (1) Uncertainty based: (a) Least Confident (LC) (b) Minimum Margin (MM) (2) Query-By-Committee (QBC): (a) Vote Entropy (VE) (b) Kullback-Leibler (KL). (3) Density-weighted methods: (a) Information Density (ID) (b) Information Diversity (IDiv): As discussed above it is combination of any base query strategy (for e.g. LC, MM, VE or KL etc.) and a similarity function. We have chosen \( LC \) as the base informativeness measure. Similarity function checks whether the query sample is representative of the unlabeled data concerned or an outlier.

We train the classifier initially with as few as 2 or 5 randomly chosen labeled samples from each class. Rest of the training of the models will take place in different manner (different permutation of incremental, semi-supervised and active learning) depending on the kind of analysis we want to carry out. We discuss the different experimental scenarios in the Result and Analysis section.
4.4 Results and Analysis

We first study the performance of different query strategies. We initially train the ensemble classifier with 5 samples for each class (training details in previous chapter). Subsequently using different querying techniques we choose a query. Note: while in real settings AL would use human annotators to label informative samples at each iteration, here we simulate this process by using the annotations provided in the training set of the Tamil HP dataset. The query instance and its label is used to update the learning parameters of the concerned class of all the member classifiers of the ensemble in online supervised fashion (as described in the previous chapter). Results show that the training using active learning converges the learning process very fast irrespective of the querying framework used when compared to the base sampling method i.e. random sampling. From the last result, for the learning to converge, the Random sampling method takes approximately 10500 (350 (no. of samples per class) * 30 (no. of classes)) labeled instances to reach the max accuracy of 87.8%. Where as even the worst performing (i.e. Minimum Margin) among all the active learning techniques used converges by 4000 with marginally better accuracy of 88.1%. Some of the better query framework like Information density or Information diversity or Least confident converges after 3000 queries with remarkably better accuracy of 89.2%, 91.6% and 89.9% respectively. From Fig.4-1 we can see that best they achieve is still higher.

Overall from the Fig.4-1 it can be concluded that Information Diversity performs the best. The noticeable point is that it converges fastest and one plausible reason could be that, as the classifier becomes more confident it needs more samples which are dissimilar to the samples that have already been used in the past than the samples which stress more on representing the unlabeled data. This way it stresses more importance on the diversity of the samples used for training, thus reducing the redundancy (which adds less value in the later part of learning). Eventually, if more queries are fired further (which comes at some extra cost in any real-time applications), ID, IDiv and LC turns out to perform equivalently. There is a disadvantage with the ID
Figure 4-1: Comparison of different query strategies accuracy.

approach and i.e. the computational cost explodes as the size of the unlabeled training data increases. The number of required similarity calculations grow quadratically with the number of instances. It can be handled to some extent by pre-computing the similarity and storing it in cache. It generally works well for a small to moderately sized dataset. But this method does not help if the data keeps growing with time which is generally the case in many machine learning applications. By the same argument IDiv suffers from the same problem but one can find a way about it. The labeled samples will also tend to grow but it can be controlled by pruning using various techniques and discarding the old and redundant samples thereby maintaining a compact size representative of each class.

If one can afford to make minor compromise on accuracy then Least confidence (LC) approach is simple and best to choose as it is relatively cheaper on computation. Incremental classifier need not be trained from scratch every time a query is chosen thus saving some computation further.

In the second analysis, we check how does the classification accuracies behave
when some amount of unlabeled data are injected while the active learning is happening. Ensemble is initially trained in batch mode using 10 samples/class, in total 300 samples, since there are 30 classes. Subsequently it is alternately trained in incremental semi-supervised and active manner using 300 samples in each learning mode. From the Fig. 4-2 we can see that semi-supervised learning and active learning are complementing each other very well. After initial off-line batch training we start by using 300 random unlabeled samples for incremental semi-supervised learning which boosts the accuracy from 35% to 68% which is a massive jump. From the dedicated semi-supervised curve (green) we can see that feeding more unlabeled data beyond this point is not useful. Therefore we introduce active learning at this point and fire 300 queries one after another and incrementally updating the concerned class parameter of all the member classifier. This gives a jump of around 6%. Subsequently we repeat the process with another 300 unlabeled sampled which yielded around 10% gain in accuracy. On carrying out the process further, it converges the learning process very fast. This alternating training technique gives a superior accuracy of 92% approximately which is better than the stand alone incremental active learning or incremental semi-supervised learning with best accuracies of 91% (AL using LC) and 88% (when pre-training with 100 labeled samples/class (result from last chapter)) respectively.

Batch and online/incremental (Random Sampling (RS)) learning takes much longer time to converge. From last chapter results, both methods takes around 10500 (i.e.350 samples per class) to converge. Active learning can produce an accuracy of 90 to 91% but it needs to fire around 3300 queries to achieve that, whereas, alternating training technique takes around 1800 samples (including both labeled queries and unlabeled instances) that is just 60 samples per class. This alternating technique saves lot of human annotating effort which means lot of money and time is saved. We stop training when improvement in accuracy after an iteration is less than 0.5%. An iteration in our case is one training session with 300 samples which can be either incremental active or incremental semi-supervised learning.

The third scenario that we analyze is the case when we don’t have all the unla-
Superiority of AL with incremental SS (initial training: 10 samples/class)

Figure 4-2: Learning converging much faster by injecting unlabeled data while performing active learning.

We propose two approaches for this problem: (1) Set a threshold (which is empirically based on the validation set) over any uncertainty measure. For example the threshold could be put over posterior probability (confidence) of the new incoming unlabeled instance or it could be applied over the margin of first two most probable class posterior values. It could also be applied over the query by committee measures like vote entropy (VE) or Kullback-Leibler (KL) measure. Another way out is: (2) Though this strategy would take us out of the realm of active learning, it still works so we would discuss it briefly. The trick is that instead of actively choosing the best query sample we choose them randomly.

First Way: Training by using threshold over query strategy measure: Initially we
train the classifier with 10 labeled samples for each class. Subsequently as the new unlabeled samples keep coming in one by one, check if the posterior value of the most probable class is less than the threshold, then that instance is a potential candidate to be a query. Same strategy can be applied with Margin, VE and KL measure. In this method there are chances that we may end up over feeding with either query or unlabeled sample at a stretch. Therefore we set a cap (to switch over) over number of unlabeled samples or queries fed continuously, i.e. either of unlabeled or queries cannot be fed to the classifier continuously for more than the cap value. Margin method performs the best, rest gets stuck and therefore converges to a low accuracy. Probably a better heuristic can be designed.

Second way: We conduct the experiment by training the ensemble classifier by:

(1) Initially training it with 10 labeled samples for each class. Subsequently we train alternatively, starting with 300 unlabeled samples followed by 300 random queries and repeat until gain in accuracy becomes lower than a threshold (low) value which is generally a sign of convergence.

Fig.4-3 shows that availability of unlabeled data has strong influence on accuracy and rapid convergence of learning, since it gives the opportunity to optimize and choose the most informative sample in lot of unlabeled instances. That is why training alternatively in semi-supervised and active learning mode, using least confidence query strategy is the best in terms of both accuracy and rapid convergence. The semi-supervised Margin threshold method converges but rather slowly. It needed approximately 3000 samples to converge and in the process it used around 1700 queries which is close to 60 samples each class. It uses more queries when compared to the best method but it is still slightly better performing than SS + RS and much better when compared to the stand-alone online learning. One concern with this approach is that it was overfeeding (more than the cap of 300 unlabeled samples continuously) unlabeled data twice and had to be pulled out given that it was violating the cap condition.
Figure 4-3: The constraint of processing one unlabeled sample at a time delays the convergence but at the same time it is better than the base SS+RS and plain incremental learning in terms of fast convergence and accuracy.

4.5 Conclusion

In this chapter we briefly introduced what active learning is and what are the advantages involved. Various query strategies have been discussed with their merits and limitations. We built an ensemble of randomized Bayesian Network and trained it in incremental active semi-supervised manner. We used it for character classification of Tamil language and achieved some very competitive results. We achieved to produce better accuracy making use of much lesser training samples. To accomplish this we made use of both labeled and unlabeled data. And by making use of active learning we reduced the heavy reliance on labeled data which reduces the cost (both in terms of time and effort) of data annotation. Carrying out the whole learning process in incremental fashion saves a huge computational cost. To the best of our knowledge this is the first time this particular combination of learning model (i.e. Ensemble of randomized Bayesian Network) and learning technique (i.e. incremental active semi-
supervised learning) has been ever used in literature in the context of Tamil language handwritten character recognition.
Chapter 5

Robust Prediction with K-Largest Norm

In any machine learning problem there is always a risk of encountering instances which one would not like to train on since they may tamper the learning model parameters. Instances of this kind are referred as outliers in literature. It is easy to point out the most outrageous outliers but not the subtle ones. Therefore it becomes imperative to take care of this issue while developing a machine learning system. We study robustness to incorporate the capability to handle outliers in the learning systems.

There have been many methods in literature to handle this issue for example pruning the data based on some criterion or incorporating capability of robustness in the objective function that is optimized for learning the model. In order to build an objective that is robust towards outliers, researchers have used various metrics or norms that do not get effected by outliers rapidly for example $L_1$ – penalty or huber – penalty there are various other available as well which we shall discuss in detail in this chapter. It is easier to try a different kind of metric or norm into a regression based prediction problem to check its efficacy at handling the outliers. And once it is successful one can move forward towards designing a probability density function which can be used to approximate the actual underlying distribution of the data at hand. Depending on the characteristics of the norm function it can be easy or a difficult task to design a probability density function. Subsequently a classification
system can be designed using that probability density function. And later one can
design an incremental method to carry out the parameter update in incremental
fashion. We study various robust methods to build a robust objective function and
check its efficacy in different regression problems.

Therefore in that regard, in this chapter we proposed to use and study the utility
and efficacy of K-Largest norm in machine learning applications. As our primary
application we have chosen to focus on the problem of robust flood prediction. Flood
forecast problem differ from generic regression problems due to the higher importance
placed on peaks compared to the remaining portion of data as well as the existence of
a larger numbers of outliers. We attempt to embed these domain specific constraints
into the regression formulation. The relative greater importance of peaks can be em-
bedded into the regression penalty by employing the $L_\infty$ as the norm over the residual
terms. However, such modeling is prone to existing outliers which diminishes the per-
formance of the regression model. To address this drawback, utilization of $L_{lgst,k}$ is
proposed because it is less vulnerable to outliers while having the flexibility of retain-
ing the desirable high emphasis on the peaks. A comparative study between standard
regression techniques such as regression with different residual and penalty functions,
Neural Network-based regression and Support Vector Regression (SVR) with differ-
ent kernels is conducted. The results suggest that, the proposed solution provides
competitive performance in terms of speed and accuracy. Besides flood prediction,
we have tested the generality of K-Largest norm with other regression problems such
as concrete strength, forest fire and wine quality prediction. Robustness of the norm
is also tested under various noisy settings. We have got promising and comparable
outcomes when compared to the state-of-the-art regression methods suggesting that
$L_{lgst,k}$ can play a vital role in creating robust solutions for a variety of applications.

5.1 Introduction

A flood forecasting system can bring measurable benefit to a community. This paper
discusses the development of a decision support system that assists in estimating flood
level. Machine learning methods have been extensively used in flood forecast modeling. Machine learning is a computing approach that optimizes performance criteria using example data or past experience. In the context of flood forecasting however, a quantitative criterion that reflects on all important aspects such as flood magnitude and timing are not currently in place [12]. If such criteria were in place, a flood forecasting agent would ideally use it as the optimization criterion. In the data-driven flood forecast literature, the sum of the Least Square Criteria are the only suggested criteria for parameter optimization. It is shown that the most important aspects of flood forecasting as mentioned before are not well reflected in root mean square error (RMSE) [4]. A flood forecasting problem poses a unique challenge in two ways compared to a generic regression problem. The first issue is that a flood forecasting agent is primarily used to forecast the incoming peak flood levels (peaks) whereas in generic problems, the emphasis is equal across all samples. Another difference is the existence of a relatively large number of outliers in many studied datasets. Hence the modeler needs to design a cost criterion that places emphasis on peaks while being robust to outliers. This problem has been identified in similar domains (to be discussed later), yet has not been widely explored in the flood forecast literature. Linear regression models are popular due to their speed, independency from initial conditions and having predictable behavior (Bounded output) and optimization. This paper discusses the issue of designing criteria that makes a linear regression model robust to outliers while maintaining high accuracy in the peak regions. A brief overview of the design of regression models that address the above challenges is given next.

Ridge regression [71] is one the most popular regression techniques Eq (5.24), due to its computational efficiency however, heavily affected by outliers. In order to mitigate this issue, [156] proposed Lasso criteria. Lasso criteria replaces the $L_2$ penalty over regression coefficient by $L_1$ penalty Eq (5.25). The benefits are that it produces sparse solutions thereby increasing interpretability of the solution and also increases robustness to outliers, though at the cost of accuracy. Interpretability is defined as a solution showing the importance of the different features used whilst also increasing the efficiency during prediction. It was found that the Lasso Criteria produces an
estimator which suffer from bias [43]. To address this shortcoming [172] proposed an adaptive Lasso, which included a multiplication of each regression coefficient in the penalty term by a weight factor (learned during the optimization process) producing the sparse solution more effectively than Lasso. Later [160] used an absolute value function on both residuals and the regression coefficient as the criterion, along with the weight applied over each coefficient in the penalty term. This is shown that for clean data the method is suboptimal. Along similar lines [74] designed a criterion aiming to handle the heavier tailed distributions. It describes a robust criterion Eq (5.23) employing a loss function that is less affected by large residual values. [117] designed a convex objective function which uses the huber function as a residual function and reverse-huber function as the penalty over coefficients. The reverse-huber penalty function is chosen to behave like the absolute value function at small values of coefficient making sparse solutions possible while behaving like squared residual function for large value to capture the coefficient sharing property of ridge regression. Meanwhile, the huber error function provides robustness towards noisy samples.

[150] had carried out a mathematical analysis proving that $L_\infty - norm$ which is generally considered to be vulnerable to outliers can be made robust and accurate using a strategy. The strategy iteratively remove training samples with large residuals from the pool and retrains with the remaining data. Every iteration is believed to remove noise samples. This strategy works well with most quasiconvex optimization problems which are solved by SOCP, as is the case with $L_\infty - norm$ minimization problems. [97] demonstrated that along with outliers many inliners were also discarded making it unsuitable when data is scarce.

Here we investigate the behavior of the norm $L_{lgst,k}$ defined by Eq (5.3) for a problem explained earlier. By definition $L_1 - norm$ and $L_\infty - norm$ are special cases of $L_{lgst,k} - norm$ by appropriate substitution. It is well established that $L_\infty - norm$ is susceptible to outliers while L1 attempts to fit to all points equally. The proof of convexity of the $L_{lgst,k} - norm$ is given at equations (3-6). And since, generally the objective function is a sum of two (residual and penalty) convex functions, it is convex and hence the global minimum is guaranteed. To the best of our knowledge
this $L_{lgst,k} - norm$ has not been employed for the flood forecasting problem. The aim of this paper is to study the efficacy of this norm in the flood prediction application and check how effectively it is able to embed the desirable properties of a flood forecasting machine. Effectiveness is checked by conducting an exhaustive comparative study with some state-of-the-art regression techniques including a Support Vector Regressor with different kernels, Neural Network-based regression and variants of Linear regression. Besides flood prediction, we have tested the generality of $L_{lgst,k} - norm$ on other regression problems such as concrete strength, forest fire and wine quality prediction. We have obtained promising and comparable outcomes when compared to the state-of-the-art regression methods suggesting that $L_{lgst,k}$ can play a vital role in creating robust solutions for various applications.

The contribution of this paper are as follows: 1) Formulation of a flood forecasting problem as a linear program (LP) and enforcing the known constraints as the optimization goal. 2) Proving why the function is a norm and how it is a convex function which leads to a convex objective function thus avoiding the multiple minima problem. 3) Formulating $L_{lgst,k} - norm$ of a constant vector as an optimal solution of a Linear Program (in its primal form). Subsequently formulating the Dual problem of the above LP, where the optimization is performed with respect to the regression coefficient to obtain its optimal value. 4) Generality of the $L_{lgst,k} - norm$ is tested with other applications. Besides generality, the robustness of the norm is tested. 5) Popular designs of the objective functions used in the literature considered for the comparative study are discussed and analyzed.

5.2 Dataset

The rainfall-runoff (water level) dataset from Tallebudgera Creek Road, Australia with a forecasting horizon equal to 2 hours is considered for this study. The study area is the Tallebudgera catchment on the Gold Coast, Australia. This station measures both water level and rainfall. Two more rainfall station upstream of this station are included in this study. The catchment of the Tallebudgera road station has an area
of approximately 64 square kilometers. The following data set is used in this study:

- Water level measurements at the Tallebudgera Creek road gauging station at 15-minute intervals.
- Measured rainfall at the three gauging stations, i.e. Tallebudgera Road, Tallebudgera Dam and Springbrook. In this study, water level less than 1 meter Australian Height Datum (AHD) are regarded as non-flood events and filtered out.

For simplification, rainfall is assumed as a known quantity with arbitrary temporal and spatial resolution. Mean hourly rainfall from 3 hours prior to the event, until the forecasting horizon were given along the last available recording of water level and one hour prior the event as input variables. In mathematical notations, the experimental setup can be described as:

\[
W(t) = f(W(t-h), W(t-h-4), W(t-h-8), R(t-1:t-4), R(t-5:t-8), R(t-5:t-12))
\]  

(5.1)

where \(W()\) denotes the water level, \(R()\) is rainfall, \(t\) as time, \(h\) as the forecasting time and the overline sign denotes the average over an interval.

5.3 Introduction to the \(L_{lgst,k}\) norm

The norm used is believed to be esoteric and it is hard to find reference [17] [37] to it and its applications in machine learning. Therefore in order to keep the manuscript self-contained and to maintain the mathematical rigor, we add some minor theoretical details. Normed space / normed vector space [128] is a vector space \(X\) together with a function (the norm) defined on it.

\[
\| \cdot \| : X \to \mathbb{R}^+ 
\]  

(5.2)

Then \((X, \| \cdot \|)\) is a normed space. In this paper we deal with the \(\mathbb{R}^d\) space which
is a vector space and when combined with a function qualified to be a norm, forms a normed space. In fact it is a Banach space, i.e. a metric $d_{lgst}$ can be defined using this norm and that would lead to a complete metric space $(X, d_{lgst})$ because it can be proved that all the cauchy sequences \cite{127} in this space are convergent in the metric $d_{lgst}$ defined by this norm. Thus we get a Banach space $(X, \| \cdot \|)$ \cite{128}.

A function cannot qualify to be a norm if it does not satisfy the following 4 conditions \cite{127}:

1. $\|x\| \geq 0$ (nonnegativity)
2. $\|x\| = 0 \iff x = 0$ (identically zero vector)
3. $\|\alpha x\| = |\alpha|\|x\|$ for all $x \in X$ and $\alpha$ is a scalar (nonnegative homogeneity)
4. $\|x + y\| \leq \|x\| + \|y\|$ for all $x, y \in X$ (triangle inequality)

In general the least square criterion is suited for approximation of a variable with a Gaussian distribution but it can give poor performance with heavier-tailed distribution data. In order to overcome the drawbacks of $L_1$ and $L_\infty$, the $L_{lgst,k}$ norm is proposed. The definition of the $L_{lgst,k}$ \cite{17} norm is as follows:

$L_{lgst,k}(k - \text{largest norm}) - \text{norm}$:

$$\|z\|_{lgst,k} = |z|_{[1]} + \ldots + |z|_{[k]}$$ (5.3)

where $|z|_{[i]}$ denotes the $i_{th}$ largest element of the absolute values of the entries of $|z|$, where $z \in \mathbb{R}^d$. It is indeed a norm and it can be verified that it satisfies the necessary four conditions listed above. Very briefly we prove it below:

1. Proof: From Eq (5.3) we know that every term in the summation is an absolute values of an entry of $z$, therefore by definition the norm value of a vector $z$ would be non-negative, i.e. $\|z\|_{lgst,k} \geq 0$.

2. Proof: By definition, Eq (5.3) $\|z\|_{lgst,k}$ is the sum of absolute terms and the sum of absolute terms can only be zero if all the terms of the vector $z$ are identically 0. Similarly it can be proven the other way round as well.
3. Proof: By definition we know: $\|z\|_{lgst,k} = |z|[1] + \ldots + |z|[k]$

$\Rightarrow \|\alpha z\|_{lgst,k} = |\alpha z|[1] + \ldots + |\alpha z|[k]$

since $\alpha$ is a scalar,

$\Rightarrow \|\alpha z\|_{lgst,k} = |\alpha| (|z|[1] + \ldots + |z|[k]) = |\alpha| \|z\|_{lgst,k}$

4. Proof: We know vector $z = (z_1, \ldots, z_k, \ldots, z_d)$, $y = (y_1, \ldots, y_k, \ldots, y_d)$

$\Rightarrow z + y = (z_1 + y_1, \ldots, z_k + y_k, \ldots, z_d + y_d)$

$\Rightarrow \|z + y\|_{lgst,k} = |z + y|[1] + \ldots + |z + y|[k]$

and we know that

$\|z\|_{lgst,k} + \|y\|_{lgst,k} = |z|[1] + \ldots + |z|[k] + |y|[1] + \ldots + |y|[k]$

from above two it is obvious that

$\|z + y\|_{lgst,k} \leq \|z\|_{lgst,k} + \|y\|_{lgst,k}$

with equality holding only when the elements of the vector $z$ and $y$ are originally in increasing order or decreasing order and all the elements of both vectors has the same sign i.e. either only positive or only negative.

When $k = 1$, it is reduced to the $L_\infty$ norm; when $k = d$, the dimension of $z$, it reduces to the $L_1$ norm. The convexity of the above norm function follows Convexity of the function can be proven by the following equation:

$$|z|[1] \geq |z|[2] \geq \ldots \geq |z|[d] \equiv |z_1|, |z_2|, \ldots |z_d| \text{ sorted non-increasing.} \quad (5.4)$$

where the terms above are the absolute value of the components of $z$. Therefore the function

$$f(z) = \|z\|_{lgst,k} = \sum_{i=1}^{k} |z|[i] \quad (5.5)$$

the sum of the $k$ largest elements of $z$, is a convex function. This can be seen by writing it as:

$$\sum_{i=1}^{k} |z|[i] = \max\{|z|_{i_1} + \ldots + |z|_{i_k}| 1 \leq i_1 < i_2 < \ldots < i_k \leq d\} \quad (5.6)$$
that is the maximum of all possible sums of k different components of z. Since it is the
pointwise maximum of n!/r!(n - r)! linear functions, it is convex. For 1 ≤ k ≤ d this
norm produces novel results, so we carry out linear regression using the \( L_{lgst,k} - norm \)
and the norm minimization problem associated to it is of the form:

\[
\minimize_x : \|Ax - b\|_{lgst,k}
\]

The data are pairs \((x_i, y_i)\) for \(i = 1, \ldots, n\). \(y_i \in \mathbb{R}^1\) and the input feature vector
\(x \in \mathbb{R}^d\). The above minimization problem can be cast as a linear program and
the transformation process follows from here on. We first represent \(\|z\|_{lgst,k}\) in the
form of an LP and then form an LP for Eq (5.7). Given that \(z \in \mathbb{R}^d\), we arrange
the absolute values of \(z\) in non-increasing order Eq (5.4) and form a new vector
\(z = |z|_1, |z|_2, \ldots |z|_d\). We use this new definition of vector \(z\) from here on. Then we
can easily find out that optimal value of \(\|z\|_{lgst,k}\), which is:

\[
|z|_1 + |z|_2 + \ldots + |z|_k
\]

\(\|z\|_{lgst,k}\) can be cast in the form of a Linear Program (LP) and the above solution can
be obtained by solving the following LP.

\[
\begin{align*}
\maximize_\eta : & \quad |z|^T \eta \\
\text{subject to :} & \quad 0 \leq \eta \leq 1; \quad \eta \in \mathbb{R}^d \\
\text{subject to :} & \quad 0 \leq \eta \leq 1; \quad \eta \in \mathbb{R}^d
\end{align*}
\]

The two LPs in primal form are equivalent and the solution \(|z|_1 + |z|_2 + \ldots + |z|_k\),
is valid for \(\eta_1 = \eta_2 = \ldots = \eta_k = 1\) and \(\eta_{k+1} = \eta_{k+2} = \ldots = \eta_d = 0\). In the above
linear program we treat \(z\) as a constant. Since our ultimate goal is to optimize the
objective function in Eq (5.7), we need to consider \(z\) as a variable and optimize the
above LP with respect to both \(\eta\) and \(z\) which is difficult to perform in this primal
form of LP. Therefore we form its \textit{DualProblem} in the following manner.
We introduce the Lagrange multipliers $\lambda, v$ for the lower and upper bound of the inequality constraint in the above LP respectively and $q$ for the equality constraint. Therefore the Lagrangian is:

\[
L(z, \eta, \lambda, v, q) = -z^T \eta + \lambda^T (-\eta) + v^T (\eta - 1) + q(1^T \eta - k))
\]
\[
= (-z^T - \lambda^T + v^T + 1^T q)\eta - 1^T v - qk \\
= (-z - \lambda + v + q)1^T \eta - (1^T v + q)
\] (5.10)

In order to get the dual function we minimize the above lagrangian with respect to $\eta$ and get the following:

\[
g(\lambda, v, q) = \begin{cases} 
-1^T v - qk, & \text{if } -z - \lambda + v + q \geq 0; \\
-\infty, & \text{otherwise} 
\end{cases}
\] (5.11)

And we obtain the corresponding dual problem by maximizing the above dual function, hence we get:

\[
\begin{align*}
\text{maximize}_{v, q} : & \quad -1^T v - qk \\
\text{subject to :} & \quad -z - \lambda + v + q \geq 0 \\
& \quad \lambda \geq 0, v \geq 0 
\end{align*}
\] (5.12)

Changing the sign, we convert it into a minimization dual problem and recognizing that $\lambda$ is acting as a slack variable we eliminate that and get the following dual problem:

\[
\begin{align*}
\text{minimize}_{v, q} : & \quad 1^T v + qk \\
\text{subject to :} & \quad z \leq v + q \mathbf{1} \\
& \quad v \geq 0 
\end{align*}
\] (5.13)

The above is a much better LP where we can treat $z$ as a variable. We can
transform the above LP into:

\[
\text{minimize}_{v,q} : \sum_{i=1}^{d} v_i + qk \\
\text{subject to: } |z_i| \leq v_i + q, \quad i = 1, \ldots, d \\
\quad v_i \geq 0, \quad i = 1, \ldots, d
\] (5.14)

Since our goal is to minimize the objective in Eq (5.7), we set \( z_i = a_i^tx_i - b_i \) and treat \( x \) as a variable, one over which the objective Eq (5.7) is optimized besides \( q \) and \( v \). Hence the following LP is derived:

\[
\text{minimize}_{q,v,x} : \sum_{i=1}^{d} v_i + qk \\
\text{subject to: } |a_i^tx_i - b_i| \leq v_i + q, \quad i = 1, \ldots, d \\
\quad v_i \geq 0, \quad i = 1, \ldots, d
\] (5.15)

The \( L_{\text{gst,k}} - \text{norm} \) minimization linear program is further simplified by removing the modulus sign:

\[
\text{minimize}_{q,v,x} : \sum_{i=1}^{d} v_i + qk \\
\text{subject to: } -v_i - q \leq a_i^tx_i - b_i \leq v_i + q, \\
\quad v_i \geq 0, \quad i = 1, \ldots, d
\] (5.16)

\section*{5.4 Linear Regression}

In this section we introduce different linear regression models. The predictor for \( y \) is linear in the features and we can express the structural model using the equation

\[
f(x_i) = \beta_0 + \beta_1 x_i \] (5.17)
where $\beta_0$ is a scalar, intercept parameter or it is also considered as the bias term, $\beta_1 \in \mathbb{R}^d$ is the slope parameter which are also referred to as the learning regression coefficient corresponding to the explanatory variable $x$. It is also possible to transform the features in the original space to some higher dimensional space using some transformation function. This is done in the hope that regression can be performed in that higher dimensional space with less error. The famous Support Vector Regressors (SVR) and Support Vector Machines (SVM) kernel trick is based on this assumption.

In linear regression we aspire to predict the value $f(x_i)$ assumed to be close to the actual observation $y_i$. An objective function containing two terms, namely an error function and a penalty function as shown below, which has to be optimized:

$$
\sum_{i=1}^{n} \phi(y_i - \beta_0 - \beta_1 x_i) + \lambda \sum_{j=1}^{d} \psi(\beta_j) \quad (5.18)
$$

Adding the penalty term reduces variance of the estimate $\beta_i$ while introducing bias. The choice of function for error $\phi$ and penalty $\psi$ may vary from problem to problem as long as they remains convex. This caution ensures the optimization process of the objective function remains reliable and accurate. There are many popular combinations of different error and penalty functions for example Ridge and Lasso.

We have applied different functions such as $L_1 - norm$, $L_2 - norm$, $L_\infty - norm$, $L_{gst,k} - norm$, deadzone and huber functions to build our objective function. The functions above are all convex hence any choice of $\Phi$ and $\psi$ from the above set of functions will result in a convex objective. As it is well known that there is no free lunch, this sparsity comes at some cost and that is some loss in accuracy when compared to the $L_2$ penalty. There are ways out of this conundrum suggested in literature for example imposing a composite absolute penalty is one such approach. Different convex functions for $\phi$ and $\psi$ are tested and the results are compared later. The following equations are the possible choice for $\phi$ and $\psi$ that are investigated herein.
\[ L_1(Taxicab) - \text{norm} : \]
\[
\|z\|_1 = \sum_{i=1}^{d} |z_i| \tag{5.19}
\]

\[ L_2(Euclidean) - \text{norm}: \]
\[
\|z\|_2 = \sqrt{z^t z} \tag{5.20}
\]

\[ L_\infty(Chebyshev) - \text{norm}: \]
\[
\|z\|_\infty = \max\{|z_1|, \ldots, |z_d|\} \tag{5.21}
\]

Deadzone:
\[
H(z, \epsilon) = \begin{cases} 
0, & \text{if } |z| < \epsilon; \\
|z| - \epsilon, & \text{otherwise}
\end{cases} \tag{5.22}
\]

Huber:
\[
H(z, \epsilon) = \begin{cases} 
|z|^2, & \text{if } |z| \leq \epsilon; \\
2\epsilon|z| - \epsilon^2, & \text{otherwise}
\end{cases} \tag{5.23}
\]

Using the above functions we tailored the following criterion functions which have their respective tradeoffs. We optimize these objectives to build the predictor.

5.4.1 Objective functions

Ridge regression (\(L_2\) error and \(L_2\) penalty [71])
\[
\sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2 + \lambda \sum_{j=1}^{d} \beta_j^2 \tag{5.24}
\]

Adding then \(L_2\) penalty will reduce the variance of the estimate of \(\beta\) at the cost of bias. Therefore we need to tune \(\lambda\) such that the predicted output is sensible and useful in the context of the current dataset.
Lasso ($L_2$ error and $L_1$ penalty [156]) is given by:

$$
\sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2 + \lambda \sum_{j=1}^{d} |\beta_j| \quad (5.25)
$$

Replacing $L_2$ with the $L_1$ penalty will increase the sparsity and interpretability of the solution, as the strength of the learnt regression coefficients show the importance of the different features during the prediction process. And it has also been observed at several instances that sparsity comes at the cost of accuracy of the prediction when compared to the $L_2$ penalty.

$L_\infty$ error and penalty function

$$
\sum_{i=1}^{n} \|y_i - \beta_0 - \beta_1 x_i\|_\infty + \lambda \sum_{j=1}^{d} \|\beta_j\|_\infty \quad (5.26)
$$

Both error function and penalty functions uses $L_\infty$-norm. One of the advantages of the $L_\infty$ norm cost function is that it has a single minimum unlike the $L_2$ cost function which may have multiple local minima. On the contrary it has a drawback i.e. it is susceptible to outliers. So the general approach is to remove the extreme outliers from the data set before using the infinity norm.

$L_{gst,k}$ error and $L_\infty$ penalty function

$$
\sum_{i=1}^{n} \|y_i - \beta_0 - \beta_1 x_i\|_{gst,k} + \lambda \sum_{j=1}^{d} \|\beta_j\|_{gst,k} \quad (5.27)
$$

The above objective uses norm-largest both for the residual and penalty function. The function norm-largest is a convex function as shown above and the sum of two convex functions i.e. the above objective function is by default convex. The above criterion can be cast into a linear program and solved. Intuitively $L_\infty$ and $L_{gst,k}$ converge as the value of $k$ tends to $d$ (dimension of the feature space).
Deadzone ($\epsilon$-insensitive) error and $L_2$ penalty function:

$$
\sum_{i=1}^{n} H_{\text{Deadzone}}(y_i - \beta_0 - \beta_1 x_i) + \lambda \sum_{j=1}^{d} \|\beta_j\|_2 
$$

(5.28)

The deadzone is used as the error function and $L_\infty$ norm is used for the penalty function. Such a function is robust to outliers and the noise in the training data. The possible disadvantage could be lower accuracy.

We can re-express the optimization problem by introducing slack variables. For each data point $x_i$, we now need two slack variables $\xi_i \geq 0$ and $\hat{\xi}_i \geq 0$, where $\xi_i > 0$ corresponds to a point for which $y_i > f(x_i) + \epsilon$ and $\hat{\xi}_i > 0$ corresponds to a point for which $y_i < f(x_i) - \epsilon$. Therefore the condition for the $f(x_i)$ to lie in the $\epsilon$-insensitive zone is $f(x_i) - \epsilon \leq y_i \leq f(x_i) + \epsilon$. And the conditions for $f(x_i)$ to lie outside the $\epsilon$-insensitive zone are:

$$
y_i \leq f(x_i) + \epsilon + \xi_i 
$$

(5.29)

$$
y_i \geq f(x_i) - \epsilon - \hat{\xi}_i 
$$

(5.30)

We can finally write the objective function as:

$$
C \sum_{i=1}^{n} (\xi + \hat{\xi}) + 0.5\|w\|^2 
$$

(5.31)

And the above criterion is minimized subject to the constraints $\xi \geq 0$ and $\hat{\xi} \geq 0$ as well as the conditions in the equations Eq (5.29) and Eq (5.30).

Huber error function Huber and penalty function $L_2$:

$$
\sum_{i=1}^{n} H_{\text{Huber}}(y_i - \beta_0 - \beta_1 x_i) + \lambda \sum_{j=1}^{d} \|\beta_j\|_2 
$$

(5.32)
In the above case the error function is a Huber whereas the penalty over the regression coefficient is the $L_1$ penalty. In the context of regression, an outlier is a measurement whereby the noise is relatively large which is generally associated with the data being faulty or there being flawed measurements. When outliers occur, any estimate of $x$ will be associated with a residual vector with some large components. Ideally we would like to guess which measurements are outliers, and either remove them from the estimation process or greatly lower their weight in forming the estimate (but assigning a 0 value for large errors will push the optimal point towards making all residual large so that the total penalty over the error sums up to 0). It could be achieved by the following error function:

$$ H(z, \epsilon) = \begin{cases} 
|z|^2, & \text{if } |z| \leq \epsilon; \\
\epsilon^2, & \text{otherwise}
\end{cases} \quad (5.33) $$

We can conclude from this definition that an error more than $\epsilon$ is completely ignored irrespective of its magnitude. If the error magnitude is unusually large corresponding to a particular sample then it is often found that there is bad data or a flawed sample. But unfortunately the above error function is not convex and therefore the associated objective/criterion function optimization becomes a hard combinatorial problem. Therefore the above definition has been simplified to Eq (5.23). And that is a convex error function which when used, creates a convex objective. Penalty functions as in Eq (5.23) are sometimes called robust, since the associated penalty function approximation methods are less sensitive to outliers or large errors than, for example, least-squares.

### 5.5 Support Vector Regression (SVR)

In this section we give a brief description about the SVR and its implementation. In a simple linear regression approach we perform ridge regression and one minimizes the objective shown in Eq (5.24). In order to produce a sparse solution an objective function is built (5.31) using the dead-zone Eq ($\epsilon$-insensitive) function as the residual
function and the square function as the penalty over the regression coefficient. The final primal problem is:

\[
\begin{align*}
\text{minimize} & \quad C \sum_{i=1}^{n} (\xi + \hat{\xi}) + 0.5\|w\|^2 \\
\text{subject to:} & \quad y_i \leq f(x_i) + \epsilon + \xi_i \\
& \quad y_i \geq f(x_i) - \epsilon - \hat{\xi}_i \\
& \quad \xi \geq 0 \\
& \quad \hat{\xi} \geq 0
\end{align*}
\] (5.34)

The above is a typical example problem of quadratic programming and it can be solved in its primal form to obtain the regression weights to perform linear regression. Any of the standard quadratic programming solver by default solves the above problem in both primal and dual form. The equal value of the primal and dual at the optimal point provides a certificate of the correctness of the solution. Despite the availability of the dual solution of the concerned primal problem, it is not easy to apply the famous kernel trick. Dual solutions of above problem can be used to construct a linear SVR, though since the dual solution of the quadratic solver does provide weights corresponding to each training sample, we can sort the weight and find the support vectors. Beyond the linear kernel if we want to use any other kernel then we need to formulate the dual problem of the above primal problem by introducing lagrange multipliers for each constraint. Introducing lagrange multipliers and
by further simplification we obtain the following dual problem:

\[ \text{minimize : } L(a, \hat{a}) = 0.5 \sum_{i=1}^{n} \sum_{j=1}^{n} (a_i - \hat{a}_i)(a_j - \hat{a}_j)k(x_i, x_j) \]

\[ + \epsilon \sum_{i=1}^{n} (a_i + \hat{a}_i) - \sum_{i=1}^{n} (a_i - \hat{a}_i) \]

subject to: \( a_i \geq 0 \)
\( \hat{a}_i \geq 0 \)
\( 0 \leq a_i \leq C \)
\( 0 \leq \hat{a}_i \leq C \) \hfill (5.35)

where we have introduced the kernel \( k(x_i, x_j) = \phi(x_i)^t\phi(x_j) \). Where \( \phi(x) \) is a transformation function which transforms the features from the original feature space to higher dimensional feature space. \( \phi(x) \) is not used anywhere while performing SVR, so the knowledge of the kernel function \( k(x, x') \) would suffice. There are numerous proposed kernel functions. Predictions for new inputs can be made using:

\[ y(x) = \sum_{i=1}^{N} (a_i - \hat{a}_i)k(x, x_i) + b \] \hfill (5.36)

Which is expressed in terms of the kernel function. \( N \) represents the number of support vectors that have been chosen. Support vectors are the samples which have either high \( a_i \) or \( \hat{a}_i \) values and the rest of the training samples which have values close to zero can be discarded without a loss of information. The parameter \( b \) can be found by considering a data point for which \( 0 < a_m < C \) or \( 0 < \hat{a}_m < C \). In practice, it is better to average over all such estimates of \( b \).

\[ b = y_m - \epsilon - \sum_{i=1}^{N} (a_i - \hat{a}_i)k(x_m, x_i) \] \hfill (5.37)
5.6 Artificial Neural Networks (ANN)

ANNs are among the most common tools used in the flood forecasting literature [100]. ANNs are biologically inspired models that map input variables to the target output variable, layer by layer through synaptic weights. The first layer constitutes the measurements or features (inputs). For each neuron in the Hidden layer the output is calculated by the following formula:

$$y^1_j(n) = \phi\left(\sum_{l=0}^{L} w_{lj}^0 I_l(n)\right)$$

where $L$ is the dimension of features and $I_0 = 1$ and $\phi()$ is a user-defined nonlinear function. The same formulation maps the Hidden layer(s) to the output variable. This provides a nonlinear, global function approximating tool given sufficient training examples are available. When information reaches the output layer, the output of the network is calculated and compared against a known desired output. This in turn generates an error signal as in:

$$e(n) = o(n) - y(n)$$

where $o(n)$ is the output of the network and $y(n)$ is the desired output. Synaptic weights are adjusted subsequent to exposure to training examples, in a process known as training. The aim of training is to minimize a user-specific cost function denoted as $J$. In common optimization algorithms the derivative of the cost function with respect to each of the weights is evaluated $\frac{J}{w_k}$. Subsequent to this evaluation, at each of the iterations, a small jump is made towards the direction that minimizes the objective function. In this work, Matlab’s 2012 implementation of the Levenberg-Marquardt optimization technique is used where $J$ is set to the $RMSE$. The convergence/performance of the method is dependent on the initial conditions of the weights. In the present work, the Nguyen-Widrow technique is used. Since ANNs are used as a comparison, the settings are similar to those found in the related flood forecasting literature. A network with a single hidden layer is used here with input to
hidden transfer functions of \( \text{tanh}() \) and linear transfers connecting hidden to output layers. To identify the number of neurons, 3-fold cross-validation was conducted by varying between 5-30 hidden neurons; little difference was found between performance (in terms of RMSE) and 15 was selected and results are given on that basis. Three-fold cross validation is carried out to maximize utility from available dataset. The available training examples were randomized, divided into three parts. The direction of the gradient was chosen based on two thirds and one third was dedicated to testing. Training is stopped upon 20 successive failures to improve the performance of the network over the testing set. The procedure is repeated by changing the testing set to another portion, resulting in three networks. The resulting networks are averaged to produce the final output.

5.7 Applications

The loss or error functions are functions which are inherently generic and hence can be applied to any suitable application. It is possible that a loss function may work better for a particular application and worse for another. The reason could be the particular nature of the application or error function to name a couple. Therefore the utility of an error function can be very different for different application depending on the priorities of the concerned application. Considering the fact that we are claiming the proposed k-largest norm as a generic error function, we have to evaluate its performance on regression problems other than flood prediction. Therefore we study the performance of the k-largest norm on three other regression problems namely, Concrete strength prediction, Forest fire prediction and predicting Wine quality.

Every application has its own priorities, as in the case of flood prediction it is essential to chase the peaks precisely whereas minor inaccuracy in the prediction of non peaks is acceptable. Similarly, in the case of concrete strength prediction it would be essential to predict accurately irrespective whether it is a peak or not. In the case of Forest fire prediction, identification of fire occurrences would be highly important and helpful to handle the situation. In this application exactly predicting the area
burnt is less important than identifying the fire eruption at each instance.

5.7.1 High Performance Concrete (HPC) Strength Prediction and Challenges

As explained and studied in [165], the strength of concrete does not only depend on water to cement ratio but also depends on the other ingredients of the concrete. HPC is very complex material and its behavior is considered to be difficult to model to high precision. In addition to the three basic ingredients in conventional concrete, i.e., Portland cement, fine and coarse aggregates, water, the making of HPC needs to incorporate supplementary cementation materials, such as fly ash and blast furnace slag, and chemical admixture, such as superplasticizer. The compressive strength of concrete is a function of the following eight input features: 1. Cement (kg/m$^3$) 2. Fly ash (kg/m$^3$) 3. Blast furnace slag (kg/m$^3$) 4. Water (kg/m$^3$) 5. Superplasticizer (kg/m$^3$) 6. Coarse aggregate (kg/m$^3$) 7. Fine aggregate (kg/m$^3$) 8. Age of testing (days). There is lot of other hidden information, to name a few, chemical components of the cement and its degree of fineness, chemical components of the superplasticizer etc., which are known to influence the strength of concrete. Therefore it is a complex
Figure 5-2: Comparison of flood forecasting methods, focused on the fourth highest peak task to model a system to predict the concrete strength precisely.

Challenges associated with the dataset used as discussed in [165] is that, it has many samples which has larger size aggregates (larger than 20 mm), special curing conditions, etc. For experimentations authors in [165] deleted those sample which can be considered as outliers, whereas in our case we did not remove any sample. Experimentation results in this thesis prove that in presence of outliers k-largest norm does produce a very comparable and at many occasions more accurate prediction than SVR.

In the literature several attempts have been made to model its behavior using linear regression techniques and later neural networks have been tried as well. From post experimentations, it was found that the accuracy of linear regression using the usual error function was not on par with more advanced prediction techniques such as ANNs. In this paper we create a linear regression model using the L-Largest norm and compare it with one of the state-of-the-art regression technique, i.e. Support Vector Regression (SVR). The dataset used for the experiments is same as [164]. The performance comparison is discussed in the result section.
Figure 5-3: Comparison of flood forecasting methods, focused on the seventh highest peak

5.7.2 Forest Fire Prediction and Challenges

One of the major environmental concern is the occurrences of forest fires which are also known as wild fires whose aftermath is immense, majorly effecting the economy, the ecology and human lives. This can happen for various reasons such as human negligence or natural causes e.g. lightening. Each year millions of hectares of forest are destroyed all over the world. Detecting this phenomenon early is the key to effective fire-fighting. Traditional human surveillance is one option but it is expensive and affected by subjective factors. Therefore there have been attempts to automate this process, and the approach generally fall into three major categories, namely satellite-based, infrared/smoke scanners and local sensors (e.g. meteorological). Bottlenecks with a satellite-based approach include acquisition cost, localization delays and quality of resolution i.e. not being enough in some cases. Moreover, the issue with the infrared scanner is its high maintenance and equipment cost. Considering the fact that meteorological stations are readily found, it is easy to collect data concerning natural factors (e.g. humidity, temperature, wind and rain.) that are known to cause forest fires. The meteorological data is collected using relatively cheap local sensors at weather stations.
In literature, attempts have been made to automate the process of fire prediction using several data mining techniques. For example support vector regression is used in [32] to model a forest fire prediction system which produces acceptable results. In the present paper, we create a linear regression model using a K-Largest norm and make a performance comparison with SVR. The linear regression model developed produces very comparable results to SVR. Modeling a natural phenomenon such as forest fire is a very complex task [32]. Both SVR and linear regression models with the k-largest norm produce comparable and acceptable results. But their drawback is their lower predictive accuracy for large fires. Though they under-predict the burnt area, it still identifies most of the fire occurrences. In order to improve the accuracy, besides the meteorological data, other information such as the type of vegetation and fire fighting intervention (e.g. time elapsed and fire fighting strategy) need to be procured. The dataset used for the experiments is same as [31].

Apart from the application being challenging, challenge with this particular dataset is that, it is a right-skew and to prevent it from reducing the accuracy of the prediction we take the logarithm of the burned area after adding 1 to it. Similarly we take the logarithm of the other input features as well. It is a common transformation made which tends to improve the prediction accuracy for the right-skewed targets. Apart from that there are presence of outliers due to some problems in the data collection. The predicted output shows that proposes regression method does handle the challenges as well as SVR and at many occasions predicts more precisely.

5.7.3 Wine Quality Prediction and Challenges

Wine has now become part of our usual life, therefore much emphasis has been given to the production of fine wine. In order to boost growth, the wine industry have been investing in new technologies to help the making and selling process. The wine certification process takes care of the wine quality that goes to the market. Quality assessment relies heavily on human experts and it is a known fact that taste is the least understood of the human senses. Therefore it is evident that it is a challenging task to model human preferences of wine.
In the literature, researchers have used various data modeling techniques to model the human preferences of wine [30]. SVR and Neural Networks (NNs) were the state-of-the-art techniques used in the literature to model it. In the current work, we use linear regression using the K-Largest norm as the error function. The results obtained are competitive when compared to SVR. Moreover at some sections of the data samples, Linear regression is relatively more accurate. The dataset used for the experiments is explained in [120]. Details of the experiments can be found in the result section.

5.8 Experimental Results

In this section we discuss the experimental results corresponding to each application. Observed and the predicted output are displayed in the figures so that the predicted output at critical points (depending on some application specific priorities) can be compared visually with the observed values. For all the applications, the reported SVR results were obtained using the RBF kernel. The predictions of neural network and SVR were very close to each other therefore we displayed only the SVR results to avoid further cluttering in figures. All optimization tasks, for example, solving the linear programs associated with each objective functions corresponding to linear regression with a different error functions and SVR were performed using the Gurobi optimization package.

5.8.1 Flood prediction

Objective assessment of performance of a flood forecasting agent is rather difficult. In a scenario where the agent erroneously predicts upcoming floods earlier than they occur in reality have no consequences associated with the decision whilst under-prediction at a particular event may be costly. Hence results are given in a time series format with the graph of observed against the prediction made by the investigated methods. The results focusing on the largest magnitude flood events are shown at Figs 5-1, 5-2, 5-3 and 5-4. As a reminder, the forecasting horizon is two hours.
equivalent to eight points on the horizontal axis.

Figure 5-4: Comparison of flood forecasting methods, focused on both low and high flows

As can be seen in the figures, the famous Ridge regression model and linear regression with Huber penalty terms fail to provide satisfactory response at the historical peaks while having acceptable agreement on the remaining test cases. Notably, even during the training phase, these models fail to provide satisfactory results. The ANN model employed, Gaussian SVR and regression models using the $L_{\text{lgst},k}$ norm provide very accurate results and are categorically similar while $L_\infty$ provides suboptimal results comparatively. It is notable that the regression models are very fast compared to the other methods. The running time for the ANN and SVR methods were in order of minutes while the regression models had a running time of around 1 second computation time on a regular 2.6Ghz processor. An interesting observation is the behavior of the models demonstrated in Fig. 5-4. As can be seen, the Ridge Regression models performs well during the low flows and comparatively poorly during the high flows. While ANNs and SVMs provide similar results and both perform equally well. Since the performance of ANNs and SVMs were very similar, ANN results have not been displayed to avoid confusion. $L_{\text{lgst},k}$ and $L_\infty$ norms both allocate their computational cost at high flows and as can be seen perform poorly during low flows.
As decision-making tools, inaccurate forecast during the low flows has no particular consequences, hence the proposed adaptation provide a clear advantage over Ridge Regression.

5.8.2 High performance concrete strength prediction

The dataset [164] has 1023 samples in total and each sample has 8 features, as discussed in the previous section. We used random 400 samples for training, 250 random samples from the remaining as a validation set and the rest approximately 350 as test samples. The validation set was used to tune the value of K of the k-largest norm and the parameters of SVR. We have not pruned the dataset to remove outliers. The point that we want to make is that even in the presence of some extreme elements in the dataset (presence of outliers were claimed and removed in [165] for experimentation), the proposed method produces a very comparable result to a state-of-the-art SVR. Fig. 5-5 shows that both SVR and linear regression with the k-largest norm predicts very similar values. Fig. 5-5 we can see that at many occasions SVR predictions are more precise but at the same time for the samples in Fig. 5-6 essentially for all the samples regression with k-largest norm is relatively more accurate. Fig. 5-7 gives a very nice overall picture of the prediction along the ideal prediction (red) line, which
again shows that the proposed method’s prediction, precision and robustness towards outliers is on par with the SVR.

5.8.3 Forest fire prediction

The dataset [31] has 517 samples in total and each sample has 12 features. The attributes used are meteorological attributes only, namely relative humidity, temperature, wind speed and rain. The data is originally skewed, therefore to reduce skew and improve symmetry, the logarithm function \( y = \ln(x + 1) \), which is a common transformation that tends to improve regression results for right-skewed targets, was applied to both input (attribute) and output (burnt area). Since the log function is a monotonically increasing function hence it does not change the proportional nature of the sequence, therefore we plot all the graphs in the log scale itself.

For training 250 samples were used, for validation 150 samples were used to set the learning parameters values of both SVR and regression with the k-largest norm. Subsequently the remaining 117 samples were used for unbiased testing. We have not pruned the dataset in order to remove the outliers. The results in Fig. 5-8 shows that even in the presence of some outliers, the proposed error function for the regression is as robust as SVR. As can be seen in Fig. 5-8 on many occasions, regression with
proposed error function performs better than SVR. The experiments were done in 3 folds, similar results were obtained in all 3 cases; one of the sample output is shown in Fig. 5-8. The results seem to be on par with the state-of-the-art SVR method, which may be further improved using some extra information such as fire fighting intervention and types of vegetation.

5.8.4 Wine Quality prediction

The dataset [120] has 1599 samples and each sample has 11 attributes. The attributes are: 1.) Fixed acidity \( (g/\text{tartaric acid})/dm^3 \) 2.) Volatile acidity \( (g/\text{acetic acid})/dm^3 \) 3.) Citric acid \( (g/dm^3) \) 4.) Residual Sugar \( (g/dm^3) \) 5.) Chlorides \( (g/\text{sodium chloride})/dm^3 \) 6.) Free sulphur dioxide \( (mg/dm^3) \) 7.) Total sulphur dioxide \( (mg/dm^3) \) 8.) Density \( (mg/dm^3) \) 9.) pH 10.) Sulphates \( (g/\text{potassium sulphate})/dm^3 \) 11.) Alcohol \( (% \text{ vol}) \).

All 11 features have been used for data modeling. We did not prune the dataset, in order to demonstrate that the robustness of the proposed error function in linear regression is comparable to SVR.

For training 800 samples were used and for validation we used 400 samples. The validation set was used to find the value of the learning parameters for both linear regression and SVR. The remaining 399 samples were used for testing. The experi-
Figure 5-8: Comparison of forest fire prediction methods

ments were performed in 3-folds and very similar prediction was achieved across all three. Fig. 5-9 shows a small portion of the test data predictions. As can be seen, on most occasions the predictions by SVR and linear regression were very close to each other. But at the same time one can find many occasions where linear regression predictions are more precise. There are many samples where both prediction are far from the observed value, this also reflects the complexity of the problem.

5.8.5 Robustness of L-Largest

In order to test the robustness of the K-Largest norm, we use the real-time flood prediction dataset. The robustness has been tested in three different scenarios. The first scenario being how the regression based predictive model, using the k-largest norm, performs when Gaussian noise is added to each sample of varying variances. The results obtained has been compared with commonly used ridge regression which employs the L-2 norm as the error function, L-infinity and SVR-based regression models. In the second scenario we test how the model performs in the presence of both Gaussian noises and a few extreme outliers. And in last case we compare the accuracy of the concerned models with increasing strengths of the outliers. The findings of the three different cases have been depicted in the Figs. 5-10, 5-11 and
The model error has been normalized by:

$$\text{Error} = \frac{\|y_{pred} - y_{obs}\|_2}{\|y_{observed}\|_2}$$  \hfill (5.40)

Where $y_{pred}$ and $y_{obs}$ are model prediction and observed values, respectively. From Fig. 10 we can see that ridge regression performs poorly in the presence of Gaussian noise and its performance deteriorates quickly with the increase in the variance of the noise, whereas models based on L-infinity, k-largest and SVR are relatively resistant to controlled zero-mean Gaussian noise. The performance of the SVR and regression using k-largest norm as an error function are very close and the least perturbed by adding the noise. We added 5 outliers, besides zero-mean Gaussian noise and evaluated all the models. The results as shown in Fig. 11 are consistent with the previous case, i.e. SVR and regression with k-largest performing the best. In the final experiment we tested all the models in the presence of the increasing strength of the outliers. Fig. 12 clearly shows that the accuracy of the model using L-infinity as the error function quickly deteriorates with the increasing strength of outliers, whereas the SVR and regression with the k-largest norm proved to be the most robust. We
observed that the model using L-infinity is determined by the outliers; its performance depends on the number of outliers present in the dataset. In the case of the K-largest norm in Fig. 12, even when the outliers are as high as 40, the error is still acceptable. This indicates that in real scenario, it should produce acceptable results as it has done in the case of concrete strength, wine quality and forest fire predictions.

By definition, L-infinity and L-1 are the special cases of the K-largest norm. As we know that L-1 is very robust to the presence of outliers but this has some cost in terms of precision. At the same time, L-infinity is more precise at capturing the peaks but is prone to outliers (as they are themselves extreme samples pretending to be peaks). Therefore, by choosing the right value of K in the k-largest norm using a validation process, this lets us strike the right balance between precision and robustness.

5.9 Discussion and Conclusion

Tackling a real-world problem from machine learning perspective poses a notable dilemma. While the so-called best machine learning agents are developed independently by computer scientists/mathematicians as all-encompassing software tools, the
constraints deduced from a specific application domain are often not included in their design. In the present work, adaptation of a machine learning paradigm of linear programming for a specific flood forecasting problem was presented. The limited yet convincing results suggested that the proposed method has competitive performance with widely-cited state-of-the-art machine learning models used for similar studies while substantially outperforming them in terms of computational complexity. The present study demonstrates that for successful adaptation, understanding of prior information about the role of machine learning as well as restrictions induced by the dataset are required. The two restrictions identified so far by the authors were the existence of outliers as well as the importance of peaks. An interesting observation of the results is that negligence of the later, as in the usage of $L_{\infty}$, will result in significant deterioration of the results. Meanwhile using $L_{lgst,k}$ that is less sensitive to outliers while being a natural candidate for regression problems that target peaks provides consistent and competitive results. And considering the fact that $L_{lgst,k}$ norm does produce acceptable and comparable results in other applications, demonstrates its generality and acceptance in the broader applications. This potential to be accepted in the broader application domain is exciting for future developments.

Figure 5-11: Prediction error in the presence of 5 extreme outliers and zero-mean Gaussian noises of varying standard deviation added to each sample.
Figure 5-12: Prediction error in the presence of outliers of varying strength

because it has the flexibility to tune its robustness (it comes at some precision cost) and the precision level by choosing the correct value of k in the norm function.

Beyond this we are currently in the process to design a probability density function using $L_{lgst,k} - norm$ so that it can be used to model a classifiers for various classification problems in batch mode and subsequently incremental methods can be used to perform the classification in online manner. But considering the atypical nature of $L_{lgst,k} - norm$ we are not been able to complete this work yet. We hope to solve the issues soon to finish the work.
Chapter 6

Conclusions and future research

In this thesis we study a incremental semi-supervised active learning paradigm. We used an ensemble of Bayesian network as a classifier which was trained in incremental semi-supervised active manner. We proposed a novel randomized Naive Bayes classifier along with its incremental and incremental semi-supervised version. The Naive Bayes structure was further updated to better approximate the data distribution and subsequently the classifier’s incremental semi-supervised learning version was proposed. A new variable learning rate was proposed which carries out learning in a realistic manner. It makes sure that maximum knowledge is absorbed in the early phase (when the learner is weak) of the learning. Learning rate makes sure that information extracted is proportional to the confidence the current learner has in the sample. It was shown that the incremental version do converges with the batch version. Importance of unlabeled data has studied through the experiments emphasizing the boost in accuracy it yield.

Next we study various active learning querying technique in the context of the proposed ensemble of randomized Bayes net classifier. We experiment in different scenarios: First we decide which is the best performing query system with out learner. Second we study the efficacy of alternating training, i.e. initially training the classifier with labeled samples and later training the classifier alternately with unlabeled (semi-supervised) and query (active learning). Training alternately helps the learning converge rapidly. Learning converges much quicker in incremental semi-supervised
active learning when compared to incremental active learning or batch learning or incremental learning. The first two scenario was studied under the assumption that all the unlabeled data is available at once to decide the query instance. In the third scenarios we assume that data comes in sequence, one by one and we are allowed to one data at once. A threshold method is suggested which works best with uncertainty based, minimum margin query strategy. Though this method has some glitches it works. The drawback of this method is that sometimes it tends to over feed the unlabeled data or query. Over feeding the classifier with either query or unlabeled data can be checked by designing a suitable heuristic.

Finally we introduce k-largest norm as the error function to handle the outliers in the dataset. This norm is used to build predictors for various regression applications and results are very convincing when compared to other commonly used error functions. By definition this norm lies somewhere in between L1 and L-infinity norm. Therefore it has the flexibility to tune its robustness (it comes at some precision cost) and the precision level by choosing the correct value of k in the norm function.

Contributions:

In the context of Randomized Naive Bayes (RNB) (published in ACPR 2013 [88]):

(1) To the best of our knowledge, this is the first time an online RNB has been proposed. Though it appears similar to the RNB proposed in [57], our method uses different method to update learning parameters online. Besides, our method uses online bagging [118] for random selection of input and they use randomized threshold selection. Moreover, unlike [57] our method takes advantage of unlabeled data.

(2) Our proposed online EM is novel. Though it is inspired by the growing GMM proposed in [91], there exist the following differences: a) the learning rate estimation step, which helps moderate the contribution of unlabeled data in the M step and hence prevents model parameters from generating improper estimates, b) estimation of variance in the M step, c) the capability to handle both labeled and unlabeled data together, d) its application to a single Gaussian.

(3) To the best of our knowledge this is the first time online learning of handwritten characters has been performed in a semi-supervised setting using a Naive Bayes
classifier.

In the context of randomized Bayesian network (RBN) (published in ICPR 2014 [89]):

(1) To the best of our knowledge, this is the first time an online RBN and its semi-supervised counterpart has been proposed. The most close work present in the literature is by Kunwar et al. [88] but there exist some striking differences for eg. the classifier used is different, the proposed online variant of Expectation Maximization (EM) used is also different.

(2) Our proposed online EM is novel. Though it is inspired by the growing GMM proposed in [91] or online EM proposed in [88], there exist the following differences: a) the learning rate estimation step, in the present work we use a varying weighting factor (learning rate) which helps moderate the contribution of unlabeled data with time in the M step and hence prevents model parameters from generating improper estimates, b) estimation of variance in the M step, c) the capability to handle both labeled and unlabeled data together.

(3) To the best of our knowledge this is the first time online learning of handwritten characters has been performed in a semi-supervised setting using a Bayesian Network (BN) classifier. (4) this approach has also been applied in building a biometric system [35].

In the context of Active learning in conjunction with incremental semi-supervised learning:

This is the first time semi-supervised incremental learning of a randomized Bayes net is performed in Active manner. Study have been conducted proving that Active learning helps immensely in speeding up the convergence of learning and also improving the accuracy of the classifier. Different learning scenarios have been studied, including the possibility of constraint of pure online learning i.e. one sample should be processed at a time.

In the context of robust learning:(under review in Machine Learning (journal by Springer))

In order to make robust learning possible, we have proposed K-largest norm to be
used as the error norm. Its efficacy have demonstrated primarily with robust flood forecast problem and it has been further verified with other regression applications. Currently we are in the process of designing a probability density function using this norm so that it could be used to build classification models. And later its incremental version could be designed (part of our future work).

**Avenues for future work**

An incremental Bayesian network structure learning can be developed to make the whole system complete incremental. It would better if incremental structure learning could be achieved by processing just one data sample at a time i.e. in an online fashion.

A concept drift detection method could be helpful to move to \( N+1 \) class system from \( N \) class learner.

Making the active semi-supervised learning online, i.e. one samples should be handled at a time to learn further.

K-largest norm could be used to design a probability density function which can be used to build a classifier.
Bibliography


143


