OPTIMAL INTEGRATION OF MACHINE LEARNING MODELS:
A LARGE-SCALE DISTRIBUTED LEARNING FRAMEWORK WITH APPLICATION TO SYSTEMATIC PREDICTION OF ADVERSE DRUG REACTIONS

by

Che G. Ngufor

A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Computational Sciences and Informatics

Committee:

_____	Dr. Janusz Wojtusiak, Dissertation Director
_____	Dr. James Gentle, Dissertation Chair
_____	Dr. Kirk Borne, Committee Member
_____	Dr. Igor Griva, Committee Member
_____	Dr. Chi Yang, Interim Director, School of Physics, Astronomy, and Computational Sciences
_____	Dr. Donna M. Fox, Associate Dean, Office of Student Affairs & Special Programs, College of Science
_____	Dr. Peggy Agouris, Dean, The College of Sciences
Date: _____	Fall Semester 2014 George Mason University Fairfax, VA

Optimal Integration of Machine Learning Models: A Large-Scale Distributed Learning Framework with Application to Systematic Prediction of Adverse Drug Reactions

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Che G. Ngufor
Master of Science
Tennessee Technological University, 2008
Bachelor of Science
University of Dschang, 2004

Director: Dr. Janusz Wojtusiak, Professor
Department of Health Administration and Policy

Fall Semester 2014
George Mason University
Fairfax, VA

Dedication

My mother, Neh, barely knew how to read and write but raised ten children for the most part by herself, most of them reaching at least a masters degree level: I am very proud of you mom! I have also sustained unwavering support, motivation and love from my father Ngufor Waji, my sisters Lum, Nchang, Akwa, and Bih, my brothers Nji, Fru, and Cheo, and my beloved, late brother Cheo Ignatius throughout the seemingly endless years of my studies. It is therefore with great pride and affection that I dedicate this dissertation to my family.

Acknowledgments

The pursuit for this advanced degree would not have been possible without the support and prayers of my family, professors, the administrative staff of Mason and friends. This very important section is one of the very few occasions where I have the opportunity to formally acknowledge and thank those responsible for the success of this work.

I would first like to sincerely thank my PhD dissertation director Dr. Janusz Wojtusiak for agreeing to serve in this important capacity. When I first met Janusz, I had blurry ideas of what I wanted to do for my PhD. I was uncertain whether research in machine learning was a good fit for me. However, after just a few months working with him on a medical claims prediction modeling project, it became clear to me the research direction I needed to pursue for my PhD. I could tell he was surprised when I ask him to serve as my dissertation director. Words alone cannot describe how much I appreciate all he has done for me both as my PhD and graduate research assistant supervisor. Janusz has been very supportive of myself and my labmates at the Center for Discovery Science and Health Informatics (of which he is the director), and has always provided me with prompt and valuable feedback. I owe a great part of my exposure to health informatics and expertise in machine learning to him.

Given the experience I have had working with Janusz, I would like to thank Dr. James Gentle for introducing us. But most importantly, I would like to thank Dr. Gentle for accepting to serve as my dissertation chair. For his advice and guidance throughout my PhD work, and above all for his insightful teaching and feedback on the many courses I took with him. Dr. Gentle taught me in mathematical statistics to always look at the "big picture" and to create simple models or prototypes for investigating new problems in what he calls the "easy pieces". This has greatly served me in my modeling approach to complex problems. Additionally, I would like to thank Dr. Kirk Borne and Dr. Igor Griva for agreeing to serve on my dissertation committee. I truly appreciate all of their time, assistance, comments, and suggestions as I worked my way through this process. I must mention that my first big exposure to data science or my Big Data "Aha" Moment was during a class project in Dr. Borne's excellent course on scientific databases and e-science.

A very special thanks is due to Dr. Len Nichols director of the Center for Health Policy Research and Ethics for his encouragement, support, and for including me in a major Health Care Reform modeling project he chaired for the Fairfax County government. I initially started carving out my statistical and machine learning modeling skills while working on this real world complex problem. I also wish to thank Dr. PJ Maddox, chair of the department of Health Administration and Policy (HAP) and the many faculty and staff members of HAP and the School of Physics, Astronomy, and Computational Sciences (SPACS) for providing an excellent environment conducive for research. The development of this thesis has been less stressful thanks to our many exciting conversations, thought provoking seminars, and free pizza and cakes during meetings and birthday parties! A note of thanks is also due to Dr. Jack Hadley for including me in his two year research project

on treatment choices for men with early stage prostate cancer. I don't know of a graduate student who is excited about weekly meetings and presentation of results! But thanks to Jack and the fantastic research group he put together: faculty members, medical doctors, and graduate students, I have always look forward to the weekly meetings so as to present my exciting discoveries. Some of the scalable agorithms presented in this thesis would not have been possible without those meetings.

Many friends have helped me survive and stay sane in graduate school. Many thanks to my SPACS friends: Yang Xu, Arun Vedachalam, Angela Lijue, Alex Koufos, Adam Cadien, Prabal Saxena, Phillip Hess, Brian Curtis, Nishu Karna, Fatena El-Masri, Dr. Greg Byrne, Dr. Fernando Mut, Dr. Susan Wright, Dr. Byeonghwa Park, Talha OZ (Krasnow Institute), and Trevor Crawford for your advise, technical help, jokes, motivation, and ideas shared on class projects. I will never forget the many happy hours, softball games, hiking, gym workouts, and fun activities we've done together. I also wish express my gratitude to Samantha Smiley, administrative assistant to the dean of the Volgenau School of Engineering for proofreading and editing this thesis. Special thanks to my secondary, high school and undergraduate friends: Kpunsah Mcdavidson, Tita Valentine, Dr. Dugah Divine , Sindjui Yves, Jeff Stone, Dr. Abatih Emanuel, Dr. Chi Celestine, Dr. Nyadong Leonard, Dr. Awondo Sebastain, Ayuk Austin, Gadinga Pascal, Franklin Fru, Dr. Gemoh Ernest, Dr. Arrey-Mbi Takor, Dr. Chenwi Ambe, Bertin Nono and all "Mendem". I have been blessed in a unique way to have shared very exciting years with you back home in Cameroon as well as in the USA. I greatly appreciate the general help, advice, gifts, phone calls, visits, rides to and from the airport/bus station as well as our famous annual winter gatherings which is always filled with heated laughter, "mbah" (funny jokes), debates, night outs and games.

Last, but certainly not least, to my family, I say thank you for your unbelievable patience, trust, and encouragement. I am especially grateful for the emotional and financial support you provided me through out my studies even when you had little idea of what exactly I was studying.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Abstract

OPTIMAL INTEGRATION OF MACHINE LEARNING MODELS:
A LARGE-SCALE DISTRIBUTED LEARNING FRAMEWORK WITH APPLICATION TO SYSTEMATIC PREDICTION OF ADVERSE DRUG REACTIONS

Che G. Ngufor, PhD

George Mason University, 2014

Dissertation Director: Dr. Janusz Wojtusiak

Too often in the real world information from multiple sources such as humans, experts, agents, or classifiers need to be integrated to provide support for a decision making system. One popular approach in machine learning is to combine these sources through an ensemble learning method. Ensemble learning has been proven to provide appealing solutions to many complex and challenging problems in machine learning. These include for example learning under non-standard conditions such as learning from large volumes of data, learning in the presence of uncertainties, learning with data streams, or when the concept to be learned drifts over time. Although considerable amount of research work has been done in ensemble learning in recent years, there still remain many open issues and challenges. This thesis explores three major challenges in this research area: First, development of techniques that scale up to large and possibly physically distributed databases. Second, construction of exact or approximately exact global models from distributed heterogeneous datasets with minimal data communication while preserving privacy of the data. Third, how to efficiently learn from modern large-scale datasets which are often characterized by noisy data points, unlabeled or poorly labeled, sample bias, missing values, etc.

These challenges are addressed in this thesis by the introduction of a large-scale parallel efficient optimal Bayesian integration framework. The framework is divided into three main parts: In the first part, two simple, fast and scalable active learning techniques are used to provide the base learners with "desirable" training sets. Then, application of a Bayesian inference and decision making technique allows the computation of several performance measures for ranking and selection of the base learners. The second part presents computationally efficient Bayesian inference and generative models to optimally integrate the outputs of a collection of classifiers optionally selected in the first part. The models improve overall performance by their ability to incorporate highly informative features not available to the classifiers at training time. In addition, the influence of weak classifiers in the final decision can be mitigated while the performances of reliable ones complimented. The last part presents a collective machine learning system for learning large-scale homogeneous and heterogeneous distributed databases through the integration of parts one and two. Each distributed data site modeled as an agent is tightly integrated with a collection of classifiers, a data and algorithm selection model, a classification evidence model and an ensemble model.

Two parallel programming models are explored for collective learning: an efficient single-pass MapReduce programming model is proposed for homogeneous agents while the MPI programming model with minimal communication is proposed for heterogeneous agents. By sharing a small set of highly informative non-sensitive feature vectors hidden from the agents at training time, the system is able to improve classification accuracy compared to traditional methods. A salient feature of the system is that global models generated are approximately exact. Further, since the information shared during learning is non-restrictive, it negates in some cases the need for difficult and computationally intensive privacy-preserving machine learning algorithms.

Three real world applications demonstrates the accuracy and scalability of the proposed integration framework: First, experiments on a series of benchmark datasets demonstrates the superiority of the framework in learning from distributed heterogeneous data sites. Second, the framework is applied to predict with high accuracy flight delays from a large-scale distributed flight arrival, departure and aircraft information modeled as homogeneous agents on a small Hadoop cluster. The last experiment is a case study of the usability of the collective machine learning system. Implemented on low cost cloud computing infrastructures such as Google Compute Engine, the system is used in a novel systematic and structured approach to detect and predict large-scale drug-side effects interactions with very high accuracy.

Chapter 1: Introduction

In many real world application problems, information from multiple sources such as agents, experts or classifiers, be they computational or human, need to be integrated to provide support for a local or global decision making system. Given the explosion of information in the world today, this problem is now even more important. Most often, the various sources are black boxes with no realistic measure of the reliability or confidence with which they make decisions, or on the information they provide (e.g online crowd-sourcing, consumers online rating or preferences to products, spontaneous adverse events reports, citizen science, etc). Bayesian methods have been shown to provide optimal solutions to the problem of integrating multiple sources of information in the presence of uncertainty. This thesis presents scalable, distributed, and computationally efficient Bayesian methodology for learning the performance of a collection of classifiers or agents enabling optimal decision integration. The presented framework has a unique feature in that it improves overall accuracy by allowing the incorporation of additional information from the data not visible to the base classifiers at training time. In addition, the influence of weak classifiers in the final decision can be mitigated while the performances of reliable ones complemented.

The primary case study used to demonstrate the efficiency and usability of the presented framework is predicting adverse drug reaction (ADR) from spontaneous reporting systems (SRS). ADRs are a major cause of morbidity and mortality worldwide. In the United States alone, serious ADRs affects more than two million patients and cause more than 100,000 deaths every year (Giacomini et al., 2007), making them the fourth leading cause of death and disease. Efficient methods capable of detecting and predicting ADR with high accuracy are in great need to reduce the unintended and sometimes fatal harm to patients. SRS databases such as the US Food and Drug Administration (FDA) Adverse Event Reporting System (FAERS), WHO International Drug Monitoring Programme (VigiBase), and

MedEffect Canada are database resources containing large amounts of voluntary reports of suspected ADRs. The availability of such large real world data not available during clinical trials provides a unique opportunity to detect and predict new ADRs. This case study is the first of its kind to learn in a principled, structured and systematic way from large-scale distributed SRS with very different observations and feature spaces. Specifically, supplementary information about each drug such as chemical and biological properties and known side-effects are merged with reported demographic information of "patients" who consumed the drugs in FAERS and MedEffect Canada. The Bayesian integration framework developed in this thesis is then applied to the distributed drug sites to predict ADR for each drug and patient.

1.1 Combining Multiple Classifiers

Combining multiple classifiers in an efficient way has been firmly established as a practical and effective solution to achieving better classification than any of the individual classifiers, even the best one. The idea of a multiple classifier system has appeared under various names: multiple experts, mixture of experts, opinion pools, decision combination, hybrid classifiers, classifier ensemble, etc.

The motivation behind the growing interest in designing multiple classifier systems has been in part due to the natural requirements of the application problem: for example the need to employ a variety of remote sensor types for environmental and natural resource mapping. For the implemented classifier to produce results that meets the needs of the application domain, it is crucial that the classification algorithm matches the properties of the data. One way to alleviate this algorithm/application match is by combining multiple classifiers.

The other part has arisen due to the fact that numerous applications of data-mining and machine learning processes have shown the validity of the "No Free Lunch Theorem" (Wolpert, 1996) which states that there is no single learning algorithm that is uniformly

the most accurate in all applications. This can be explained by the fact that each learning algorithm determines a model with certain additional assumptions and/or tuning parameters in order to adapt it to the characteristics of a training data. These assumptions may hold in some applications and on some datasets but may completely fail on others. Further, determining the best set of tuning parameters to match specifics of different training data can be a very difficult task and may even require the intervention of human experts. Under these conditions, the machine learning method has not yet removed the need of a human expert and the process is not automatic. The use of multiple classifiers has the potential to alleviate the problem of algorithm/application assumptions, and to some extent, parameter settings.

Even when restricted to a specific application, the classical approach of finding the best classifier has some major drawbacks. The main drawback is that the best classifier for a classification task may be very difficult if not impossible to find. In addition a single classifier cannot exploit the complementary, or discriminatory information that other classifiers may encapsulate.

Other advantages of combining multiple classifiers include:

1. By using multiple classifiers, a complex problem can be decomposed into multiple sub-problems that are easier to solve, understand and interpret.

2. Mathematically, combining multiple classifier may produce a better bias/variance trade-off (Rokach, 2010) than a single classifier. Some classifiers may reduce the bias component while others the variance component of the generalization error. The combination of these classifiers may therefore help reduce the generalization error.

3. The combination method can benefit from prior knowledge about the classifiers, domain of application or some additional information not available to the classifiers to help guide the classification process and improve accuracy.

These and many other advantages of using multiple classifiers have given birth to a new research field called *ensemble learning*. Ensemble learning has been proven to provide

appealing solutions to many complex and challenging problems in machine learning and many other fields. The main idea of ensemble learning is to combine the outputs of a variety of classifiers (either of different types or different instantiations of the same classifier) and make a final decision. The intuition behind this idea is that, it will become much less likely that the ensemble will missclassify a new data point compared to a single classifier. The learning algorithms comprising an ensemble are referred to as *base learners*. When trained, the base learners generate classification models referred to as *base classifiers*.

Combining the base classifiers raises some interesting questions including:

1. How to select the base classifiers?

2. How to combine the outputs of these base classifiers to maximize classification accuracy?

3. How to make the method scalable?

The first step in building an efficient multiclassifier system is the selection of the base classifiers. Intuitively, combining multiple classifiers that agree on all sub-areas of the training set based only on their decisions cannot lead to any improvement in accuracy. Disagreement or diversity among the base classifiers is therefore one key factor in combining multiple classifiers. A detailed discussion on the importance of diversity in ensemble learning can be found in Alpaydin (2004). Diversity methods in ensemble learning can be broadly grouped into two groups:

- Base classifiers generated by different learning algorithms on the same data. This is also known as heterogeneous set of base classifiers. For example in a classification problem, a naïve Bayes classifier, k-nearest neighbors, Support Vector Machines (SVM), and decision trees could be combined.

- Base classifiers generated by the same algorithm, either by running it on different partitions of the training data, or using different parameter settings. Popular ensemble methods using this approach include bagging, boosting and random forest.

While ensemble methods that manipulate the training set or tuning parameters have been well explored and shown to improve classification accuracy in many real applications, methods that combine a set of heterogeneous classifiers have not been given sufficient attention. However, in the real world multiple sources of information often need to be combined when no training data is available, thus manipulating the training set is not possible. In addition these sources are often heterogeneous with different underlying theories and assumptions making it inconvenient or difficult to use classifiers with the same underlying theory and assumption. The ensemble method can greatly benefit from the complementary information provided by heterogeneous classifiers.

Given the advantages of combining heterogeneous classifiers, possible reasons why this approach has not been given much attention can be attributed to: the embarrassingly large number of algorithms available today, the perceived high learning curve required to use some algorithms, high computational cost, and scalability issues. This thesis therefore focuses on methods that allow efficient combination of heterogeneous classifiers with the aim of designing very simple, accurate and scalable combination methods.

Given a diverse set of classifiers, the quality or accuracy of the base classifiers is another key factor for the success of the chosen combination scheme. Overall improvement in accuracy cannot be achieved if the base classifiers are not accurate (Kuncheva, 2004; Brown, 2010). Clearly, diversity and accuracy are two primary factors that must be carefully considered for a successful combination scheme. However, in most combination methods there is little or no principled approach to select the base learners to ensure they fit the characteristics of the data and maximize classification accuracy. These methods either rely on domain experts or on some heuristic approach. Relying on these approaches could be problematic especially in applications where domain experts are not available or in environments where data is limited, large, expensive or unsafe to collect. Therefore there is need for a systematic approach to select appropriate base learners for a given problem.

The second step in building an efficient multiclassifier system is construction of a good combination scheme. A great number of combination methods have been proposed in the

past years. Due to the large number of articles that have been written on these methods, a section cannot be devoted for a comprehensive review, therefore, references will be given throughout this work as necessary.

While most of these combination methods have been extensively studied and produce reliable results one drawback is their inability to correct or mitigate missclassification by using some additional information not available to the base learners at training time. When class decision boundaries are separable or when data points are relatively easy to classify, reliance on the predictions of the base learners alone may be enough to guarantee accurate results. However, when the boundary is mixed or in the presence of ambiguous data points that are difficult to classify, some of these points will inherently be missclassified even by the best base classifier. Under such conditions, a classifier makes assignment to a class with little or no evidence to support class membership. This problem can be mitigated if in addition to the base classifier's decision, the combination method can also make use of additional information such as prior knowledge about the classifiers or some class related hidden information. This additional information can be used to improve the performance of the ensemble by correcting some misclassifications.

Finally, computational cost, complexity, and scalability are important factors to carefully consider for an efficient multiple classifier system. Combining multiple classifiers is not a trick that always increases accuracy; it increases time and space complexity in training and testing. Given today's ever growing dataset sizes this problem cannot be ignored. Unless the base learners are scalable, trained carefully and their decisions combined intelligently no significant benefit can be derived.

1.2 Distributed Learning

In recent years, many organizations and institutions have witnessed an exponential growth in the number and size of their databases. For example, the development of high throughput data acquisition technologies have made it possible for hospitals and other institutions to

generate large volumes of data (e.g Magnetic resonance imaging (MRI) files, Magnetic Resonance Spectroscopy (MRS) files, digital mammographic files); protein sequence and gene expression data are gathered steadily in biomedical sciences; commercial organizations amass huge collection of their daily transactions (products and customer profiles); satellites and other space borne instruments transmit terabytes of data daily. The acquisition of these large, complex (and possibly distributed) data have significantly outpaced the ability to analyze, summarized and extract "knowledge" from them. A major reason for this is that, classical implementation of most machine learning algorithms require all data to be present in main memory. In other words, the traditional approach to classification requires that all data be collected in a centralized repository and a classification algorithm applied to produce a predictive model. However, in the real world data is rarely available in a centralized location. It is often distributed over several nodes of a network or different geographic locations and thus integrating all data from the local nodes to a central node is clearly unrealistic and untenable. In addition there might be other concerns or constraints involved in moving or sharing data. Different institutions may not allow the exchange of data due to privacy, legal restrictions or competitive advantage concerns. Under such conditions, there is thus need for learning systems that can acquire knowledge at the location of the data and where the computational resources are available. The knowledge acquired can be used locally or transmitted and possible combined with knowledge from other locations to establish a global solution to the problem.

The following examples may help highlight the importance of these points;

1. The diagnosis and prognosis of brain tumors from clinical data to aid patient management and treatment (González-Vélez et al., 2009). Individual hospitals do not typically encounter sufficient cases of particular tumor types to constitute a sizable training set for the construction of robust predictive models. However sufficient tumor data with all tumor types may be available in several hospital groups across various countries. Hospitals and countries vary their approach to restricting the mobility of data. Some local centers may completely restrict the movement of patient data while

others may agree to share only specific cases. Thus all data cannot be collected in a centralized location. But through the use of distributed data mining techniques, powerful diagnostic and prognostic support can be facilitated base on individual hospital data. Local classifiers can be constructed that target specific patient population and a global classifier to aggregate appropriate cases in the distributed system.

2. Analyzing customer transaction profiles of a major corporate chain in the US with the hope of developing successful local and central marketing campaigns quickly. However, the chain is composed of several outlets scattered across the country which makes integrating all customer data to a central location time consuming and expensive if not impossible.

3. Detecting and preventing fraudulent activities in financial information systems. Fraudulent patterns from individual financial institutions can help other institutions identify new patterns and implement preemptive measures swiftly. However, financial companies such as banks may avoid sharing data due to legal and competitive reasons. Therefore, combining the databases at a single location is not possible.

4. Drug Discovery: New technologies and research advances in cheminformatics and bioinformatics has led to the generation of enormous amount of data in the pharmaceutical industry. In many cases, these datasets continue to grow in size, making it impossible to store them at a centralized location. Handling these datasets, as well as allowing sophisticated search queries for similarity structures, chemical properties, or molecule keywords necessitates scalable distributed databases and learning systems.

Based on these motivational examples, it is clear that a centralized solution to many real world problems is inapplicable or inconvenient. These problems can be best modeled by multiple cooperating intelligent systems or multiagent systems. Multiagent systems (MAS) is a sub-field of distributed artificial intelligence that studies how autonomous entities or agents interact in a collaborative way to solve a given problem (Stone and Veloso, 2000). By integrating learning in MAS, a decentralized learning, problem solving and decision

framework known as Multiagent Learning Systems (MALS) is derived which can serve as a viable option for modeling many real world problems. In MALS, agents are able to perform inductive learning on their individual data and in addition can interact or collaborate with other agents, learning from each other in such a way that they improve individual performances and hence the overall performance of the system. Due to the inherent parallelism and distributed nature of MAS, the design and implementation of MALS may offer a more powerful, faster, and more robust distributed data mining system compared to a single agent.

The individual agents can also benefit from ensemble methods as well. Since the major goal of classification is to improve overall accuracy, integrating accurate ensemble methods within each agent can produce very accurate local models. Yet, another benefit can be derived through ensemble of agents. When a global solution is required, agents within MALS can be modeled as an ensemble of agents, by which the same ensemble method can be applied to integrate their decisions and thus generate a more accurate global model.

1.3 Research Goals and Contributions

From the simple questions that were raised in combining classifiers discussed in section 1.1, and the discussions on distributed learning in section 1.2 this thesis formulates three related research questions:

1. How to effectively evaluate, compare and rank learning algorithms on modern datasets characterized by noise, poor labeling, class imbalance, sample bias, missing values, and other complex phenomenons.

2. How can a machine learning system identify and correct the mistakes of learning algorithms for distributed homogeneous/heterogeneous datasets. Producing exact models while not compromising the sensitivity of the data.

3. How to solve the two problems above and still remain efficient in both sample complexity and computational complexity.

Based on these research questions, this thesis presents an alternative solution to the problem of learning multiple classifier systems in a centralized or distributed data environment. Specifically, a framework is constructed in which each distributed data site modeled as an agent is endowed with a collection of learning algorithms. Each agent uses these tools to construct an ensemble model and all ensemble models are combined to produce one global model for the system. It should be noted that the term agent is used here simply to describe the independent data sites with a collection of learning algorithms and should not be confused with its classical usage and meaning in artificial intelligence. That is, intrinsic properties of agents such as autonomy, sociability, reactivity, pro-activeness, etc. are not modeled (Stone and Veloso, 2000).

1.3.1 Optimal Bayesian Muticlassifier Integration Framework

To simplify the construction of the framework, the main goal is broken down into three main parts:

1. **Ensemble Learning**.

   (a) *A Data and algorithm selection method:* A crucial step in building efficient ensemble models is the selection of base classifiers. A poor selection could lead to an ensemble that is as worse as the worst classifier in the ensemble. On the other hand, it is not necessarily true that the most accurate classifier contributes the most to the ensemble's final decision. Diversity among the classifiers also plays a major rule. Diversity is the extend to which the individual classifiers disagree in the error they make in classifying a new data point. This property alone can guarantee to produce a good ensemble classifier even with the simplest combination scheme (Sankar K. Pal, 2004).

Given the profusion of classification algorithms, it is difficult to make a good selection. Most often the analyst knowledge of the algorithms and the problem domain is required to select a reliable algorithm. A reliable algorithm on one dataset may not remain reliable on another. This problem is even more important on datasets characterized by noise, missing values, labeling issues, sample bias and class imbalance. To further complicate matters, many of these algorithms require tuning parameters to adapt them to the characteristics of the training set. A substantial increase in accuracy can be archived if the right parameters are used.

To overcome these difficulties, a ***Bayes Active Data and Algorithm Selection*** method is proposed to optimally select informative data points and reliable learning algorithms for ensemble learning. Specifically, starting with a set of learning algorithms, the method trains them using a small set of carefully selected learning examples and ranks their performances based on several performance measures. The top k classifiers can then be selected for ensemble learning.

(b) *Classification Evidence Model:* The use of highly informative features extracted from relevant features in the training set has the potential to provide evidence and correct some misclassifications by a well designed ensemble method. A variational Bayesian factor common spatial pattern is proposed for the computation of these hidden feature scores or ***"classification evidence"***.

(c) *Efficient multiple classifier integration schemes:* An optimal integration scheme is required to generate efficient and accurate ensemble. Given a good selection of diverse and accurate committee of classifiers and classification evidence, a poor combination scheme will not guarantee improved results. It is desirable for the combination method to take advantage of the strength of base classifiers while ignoring their weakness.

Based on these observations and desired characteristics, three new ensemble methods are proposed to optimally combine the decisions of multiple classifiers

with supplementary information provided by the classification evidence. The first method uses the ***expectation-maximization (EM)*** to derive point estimates of classifier combination weights while the other two applies ***variational Bayesian (VA)*** methods with automatic relevance determination (ARD) to generate posterior distribution of the weights.

2. **Collective Learning**. At the base of collective learning are data sites or agents tightly integrated with the ensemble learning properties described in the first part. Thus each agent selects a set of relevant base learners, computes classification evidence and construct an ensemble model. In a distributed environment with possibly different observed features, the performance of the agents can be improved by collaboration. Depending on the constraints of the environment, collaboration can be done by;

   (a) *Sharing data:* Where sharing of data is possible, two agents may share their most informative instances.

   (b) *Sharing classification evidence:* Since the classification evidence are hidden variables, it is assumed that they can be shared without any restriction. This simple collaboration technique therefore negates the need for difficult and expensive privacy preserving methods.

   (c) *Sharing models:* Finally the agents can share learned models. Either the ensemble models or the base classifiers can be shared. Sharing base classifiers may be restricted by the format of the database schema, however, no such restriction applies when sharing ensemble models. Another possibility is to share classification evidence models.

3. **Ensemble Agents**. The local ensemble strategy designed within each agent is naturally extended to integrate multiple agents. Each agent simply communicates the local ensemble and classification evidence to one agent who applies the same integration algorithms to combine the local models.

1.3.2 Thesis-Generated Publications

Designing machine learning algorithms that scale up to very large and distributed datasets is a central motivation of the work presented in this thesis. In the course of writing the thesis, a number of large-scale parallel efficient algorithms were developed. The first algorithm (Ngufor and Wojtusiak, 2013) is based on a simple approximation of the logistic regression model leading to an approximate close form solution. The algorithm was then implemented as a single-pass fast and inexpensive MapReduce program on a Hadoop (`http://hadoop.apache.org/`) cluster. Based on this approach and the extreme machine learning theory (Huang et al., 2006), five new simple and very fast algorithms for kernel logistic regression were developed (Ngufor and Wojtusiak, 2014a; Ngufor et al., 2014). Two of these algorithms are used as base algorithms for numerical experiments in this thesis. Finally, a computational and parallel efficient single-pass formula for computing the covariance matrix of distributed datasets is developed in Ngufor and Wojtusiak (2014b). The single-pass covariance formula is used in this work to scale down a large scale dataset on Hadoop for ensemble learning which otherwise would have been a tedious and computationally very expensive exercise. The formula is also used to implement one of the presented ensemble methods on Hadoop.

List of Thesis generated articles:

Che Ngufor and Janusz Wojtusiak. Learning from large-scale distributed health data: An approximate logistic regression approach. *ICML 13: Role of Machine Learning in Transforming Healthcare*, 2013.

Che Ngufor and Janusz Wojtusiak. Extreme logistic regression. *Advances in Data Analysis and Classification (ADAC), Springer (Accepeted with Revisions)*, 2014a.

Che Ngufor and Janusz Wojtusiak. Learning from large distributed data: A scaling down sampling scheme for efficient data processing. *International Journal of Machine Learning and Computing (IJMLC)*, 4(3):216–224, 2014b.

Che Ngufor, Janusz Wojtusiak, Andrea Hooker, Talha Oz, and Jack Hadley. Extreme logistic regression: A large scale learning algorithm with application to prostate cancer mortality prediction. In *The Twenty-Seventh International Flairs Conference*, 2014.

1.4 Thesis Overview

The rest of the thesis is arranged as follows: background material on classification, ensemble learning and distributed learning are discussed in chapter 2. A brief description of some articles generated by the thesis is also presented in this chapter. Chapter 3 presents algorithms that select informative data points for learning, and a subset of learning algorithms for ensemble learning. The performance of the algorithms selection method is illustrated on two benchmark datasets. Algorithms for computing classification evidence and to integrate multiple classifiers is presented in chapter 4. Different learning schemes and parallel programming models for homogeneous and heterogeneous distributed databases implemented as a collective machine learning system are presented in chapter 5. The performances of the various methods are illustrated on 14 small benchmark datasets and a large-scale flight dataset. The main case study of the thesis is presented in chapter 6. Chapter 7 offers conclusions and discusses future work.

Chapter 2: Background and Theory

This chapter presents notations used throughout the thesis, a review of some state-of the art machine learning classification methods, distributed learning and databases, and scalable and parallel efficient learning algorithms developed in the course of writing this thesis.

2.1 Machine Learning and Classification

Machine learning is a scientific field that is concerned with building computer programs able to construct new knowledge or improve on already processed knowledge (Anderson et al., 1986). Machine learning draws on concept and results from many fields including statistics, artificial intelligence, information theory, biology, cognitive science, social science, philosophy and control theory. Thus, machine learning offers an immense diversity of research tasks and testing grounds. Learning can be studied in many different contexts, such as decision making, classification, sensory signal recognition, problem solving, task execution, control and planning.

2.1.1 Classification

The task of classification is an integral part of most human activities. Many daily situations arise where people are faced with making a decision or forecast based on past or currently available information. A classification procedure then, is some formal method for repeatedly making judgments in new situations. Specifically, a classification problem deals with the construction of a procedure that can be applied to a sequence of cases or data points, in which each data point must be assigned to one of pre-defined or unknown classes on the basis of observed attributes or features.

A classification problem falls into one of two main divisions: *supervised classification* and *unsupervised classification*. In supervised classification, given a set of observations or *input* with corresponding class labels, the task is to use these examples to design a classification procedure serving as an automatic means for deriving the class of new observations. In unsupervised classification, the available data is not labeled and the task is to establish the existence of classes, groups or clusters in the data. This thesis is mainly concerned with the machine learning task of supervised classification. Henceforth the term classification will mean the supervised classification in which the class labels are discrete and unordered.

A more formal definition of classification can be presented as follows: suppose a class t_j of $K \geq 2$ possible classes is to be learned. A learning algorithm is given a set of n training examples $\mathcal{D}_l = \{(x_1, t_1), \ldots, (x_n, t_n)\} \subseteq \mathcal{X} \times \mathcal{Y}$ where each input instance x_i are typically p-dimensional vectors: $x_i = \langle x_{i1}, x_{i2}, \ldots, x_{ip} \rangle \in \mathcal{X}$ also called features or attributes is associated with a categorical output or target variable $t_i \in \mathcal{Y} = \{t_1, \ldots, t_K\}$ that denotes its class membership. $\mathcal{X}$ is typically the set of reals $\mathbb{R}^p$. The learning algorithm is then trained using the labeled data to induce a hypothesis $\hat{f}$ from a set of possible hypotheses $\mathcal{F}$ that approximates as closely as possible the true but unknown function $f : \mathcal{X} \to \mathcal{Y}$ that is assumed to correctly map an input $x \in \mathcal{X}$ to its true class $t \in \mathcal{Y}$ without any error.

In the context of decision theory, a classifier is an approximation of the true but unknown decision rule using the training data such that the generalization error or *expected risk*,

$$\varepsilon(\theta) = \mathop{\mathbb{E}}_{\mathcal{D}} \left[\mathcal{L}\left(\hat{f}(x), f(x) \right) \right] \tag{2.1}$$

is minimized, for some loss function $\mathcal{L}$. $\mathcal{D}$ is the true joint distribution of $\mathcal{X} \times \mathcal{Y}$ and θ is a vector of parameters (if any) of the classifier. The loss function $\mathcal{L}\left(\hat{f}(x), f(x) \right) \in [0, \infty)$ measures the quality of the classifier on the example (x, t). During learning, the goal is to find hypotheses $\hat{f}$ and values of θ that minimizes the generalization error. However, this is usually a difficult task since the true distribution $\mathcal{D}$ is not known. The best alternative is

to carry out the minimization using the training set leading to the empirical risk.

$$\hat{\varepsilon}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left[\mathcal{L}\left(\hat{f}(x_i), f(x_i)\right)\right]. \tag{2.2}$$

In summary, a classification problem can be achieved in a sequence of two steps as shown in Figure 2.1:

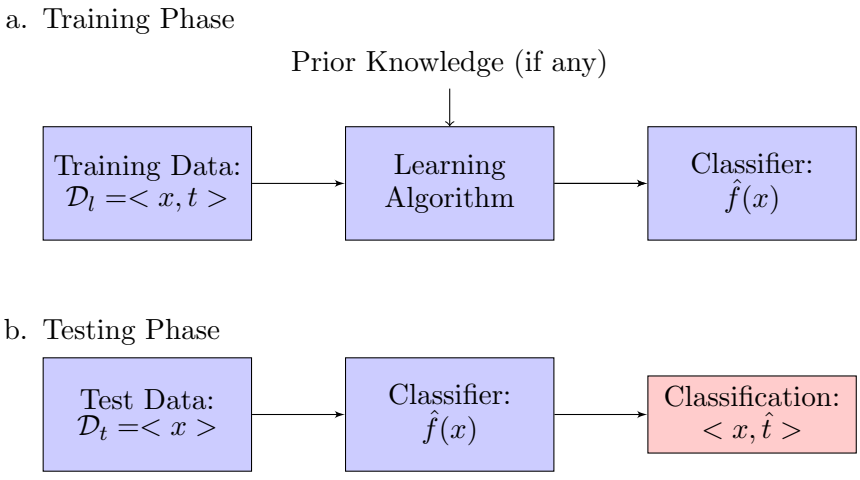


Figure 2.1: Training and Testing phase of classification

Some learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing the performance of the classifier through an intermediate validation step using a subset of the training data or through cross-validation. In cross-validation, the training set is divided into mutually exclusive and equal-sized subsets and the classifier is trained on all but one subset and tested on the left-out subset. The average of the error rate on all subsets gives an estimate of the error rate of the classifier.

2.1.2 Examples of Classifiers

There are by far too many machine learning classification algorithms available to give a brief description of each one. The purpose of this section is to offer a brief review of a few state-of the art methods closely related to is thesis, and to present new scalable algorithms designed in the course of developing this thesis.

2.1.2.1 Support Vector Machine

Support Vector Machine (SVM) has emerged as one of the most powerful machine learning algorithms for solving binary classification and regression problems since its introduction by Vapnik in his statistical learning theory Vapnik (1998). The basic learning principle of SVM is to map the original input space into a high-dimensional feature space through a kernel function. In the new feature space, an optimal separating hyperplane with maximal margin is determined in order to minimize an upper bound of the expected risk instead of the empirical risk.

2.1.2.2 Decision tree and Rule-based methods

These methods induce classification rules in the form of decision trees or rules from a set of given examples. Decision tree (Quinlan, 1993) are constructed top-down: at each node, a split is made based on an impurity measure such as the normalized information gain. The attribute with the highest information gain is used to make the decision. The process is stopped when no improvement is possible.

Rule induction is perhaps one of the most important techniques of knowledge representation in machine learning. The general learning problem of the AQ learning system as implemented in AQ21 (Wojtusiak et al., 2006), for example, is to generate general hypotheses $\mathcal{H} = \{h_1, \ldots, h_k\}$ about target labels or classes $\mathcal{T} = \{t_1, \cdots, t_K\}$ respectively from the training set drawn from these classes. The hypotheses are generated in the form

of attributional rulesets that optimize a given multi-criterion measure of hypotheses utility. An attributional ruleset is a set of attributional rules describing the same target class or concept.

2.1.2.3 Neural Networks

An Artificial Neural Network (ANN) is an information processing device that loosely models the way biological nervous systems such as the brain processes information. The key component of the device is the information processing unit which is composed of a large number of highly interconnected processing elements, or neurons, working in unison to solve specific problems. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. For example, most ANNs contain some form of "learning rule" which modifies weights assigned to the connections according to the input patterns that it is presented with. Thus, just like SVM and Decision trees, ANNs learn by example as do their biological counterparts; a child learns to recognize dogs from examples of dogs. A review of ANN from a statistical perspective can be found in Cheng and Titterington (1994).

2.1.2.4 Naive Bayes

Naive Bayes is a simple probabilistic classifier based on applying Bayes' theorem with strong assumption of independence between the features. That is, it assumes that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature, given the class variable. Despite this strong and not necessarily true assumption, naive Bayes classifiers have worked quite well in many complex real world situations (Zhang, 2004).

2.1.2.5 k-nearest neighbours

k-nearest neighbours finds a group of k instances in the training set, which are closest to the test pattern. The predicted class label is based on the predominance of a particular class in this neighbourhood. The distance and the number of neighbours are key elements of the algorithm.

2.1.3 Approximate Logistic Regression

This section briefly describes a few large-scale learning algorithms based on simple approximations to the logistic regression (LR) model. First a simple approximation converts classical LR into a least squares algorithm. Then the approximation is applied to kernel logistic regression (KLR). Finally, extreme learning method is extended to KLR and its approximate least squares version to derive a number of very fast, scalable and accurate algorithms. See Ngufor and Wojtusiak (2013, 2014a); Ngufor et al. (2014) for a full discussion of the methods. Based on the approach in Ngufor et al. (2014), another large-scale algorithm for KLR is also presented.

2.1.3.1 Least-Squares Logistic Regression

Logistic Regression (LR) is a learning algorithm with sound statistical background and widely used in machine learning, data mining, and statistics. The popularity of LR can be attributed to its simplicity and interpretability of model parameters.

The fundamental assumption of the model is that the log-odds of the class posterior probability $\pi(x) = \mathbf{Pr}(t = 1|x)$ is a linear combination of the independent variables i.e

$$\eta(x) = \log\left(\frac{\pi(x)}{1-\pi(x)}\right) = \boldsymbol{\beta}^T x \tag{2.3}$$

where $t \in \{0, 1\}$ is a binary response variable, $x = (x_0, \ldots, x_p)$ are the independent variables

and $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_p)$ are the unknown parameters of the model. The regularized negative log-likelihood of the model can be written as

$$l(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 - \frac{\gamma}{2}\sum_{i=1}^{n}\Big[y_i\boldsymbol{\beta}^T x_i - \log\left(1 + \exp(\boldsymbol{\beta}^T x)\right)\Big]$$

where the parameter γ reflects the strength of regularization.

The maximum likelihood method is commonly used to estimate the model parameters. The log-likelihood equation is differentiated with respect to $\boldsymbol{\beta}$, set to zero and solve

$$\frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} - \frac{\gamma}{2}\sum_{i=1}^{n} x_i(y_i - \pi_i) = 0. \tag{2.4}$$

Unfortunately, equation 2.4 is nonlinear and there is no close form solution. The traditional approach is to approximately solve it using iterative methods.

There are many numerical optimization methods that can provide efficient solutions to equation 2.4. The Newton-Raphson method is perhaps the first goto off-the-shelf method to use. The method takes the first degree Taylor series approximation of equation 2.4 at a point β^{old} (initial guess), sets this to zero and solve for a new approximate solution β^{new}. The update process is repeated until convergence. A full treatment of the Newton-Raphson algorithm and other equivalent maximization techniques can be found in standard statistics text such as Hastie et al. (2001).

However, for convergence, most of these techniques require several passes over the data, and in addition some may require computing the inverse of the Hessian matrix. This procedure can be computationally expensive especially for large datasets. To overcome this problem, in Ngufor and Wojtusiak (2013), a simple approximation to the logistic function was performed to transform the iterative solution into a closed-form "least-squares" solution

for the MLE of the logistic model stated as:

$$\left(\frac{1}{4}\mathbf{X}^T\mathbf{X} + \frac{\gamma}{2}\mathbf{I}_{d+1}\right)\boldsymbol{\beta} = \mathbf{X}^T\left(\mathbf{t} - \frac{1}{2}\mathbf{1}_n\right) \tag{2.5}$$

were $\mathbf{t}$ is the $n\times 1$ response vector, $\mathbf{X}$ is the $n\times(p+1)$ model matrix, $\mathbf{I}_{p+1}$ is a $(p+1)\times(p+1)$ identity matrix and $\mathbf{1}_n$ is a $n \times 1$ vector of ones. This simple least-squares algorithm for logistic regression, or LS-LR, was shown to have excellent generalization and scalabilty properties compared to popular gradient based methods such as the stochastic gradient descent.

2.1.3.2 Extreme Logistic Regression

One major disadvantage of classical LR (and LS-LR) is that it is a linear classifier. However, most real world classification problems are non-linear, and so LR cannot capture any non-linearity that may exist in the data. Using the "kernel trick", a kernelized version of LR, or Kernel Logistic Regression (KLR) can be constructed. A mapping function $\boldsymbol{\phi} : x \in \mathbb{R}^d \rightarrow \boldsymbol{\phi}(x) \in \mathbb{R}^{d_f}$ is chosen to convert the non-linear relationship between the response and the independent variable into a linear relationship in a higher (and possibly infinite) dimensional feature space. The map function is however usually unknown, but dot products in the feature space can be expressed in terms of the input vector through the kernel function: $\mathbf{K}(x, y) = \boldsymbol{\phi}(x) \cdot \boldsymbol{\phi}(y)$.

Estimation of model parameters can be performed just as in classical logistic regression. In Ngufor and Wojtusiak (2014a) for example, KLR was cast into a constrained optimization

problem leading to an iterative re-weighted least squares algorithm or IRLS-KLR stated as:

$$\begin{pmatrix} \mathbf{K} + \frac{1}{\gamma}\mathbf{W}^{-1} & \mathbf{1}_n \\ \\ \mathbf{1}_n^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ \\ b \end{pmatrix} = \begin{pmatrix} \mathbf{z} \\ \\ 0 \end{pmatrix} \tag{2.6}$$

$$\mathbf{z} = \mathbf{K}\boldsymbol{\alpha} + b\mathbf{1}_n + \mathbf{W}^{-1}(\mathbf{t} - \boldsymbol{\pi}). \tag{2.7}$$

$\mathbf{W}$ is an $n \times n$ diagonal matrix of weights with i'th element $\pi_i(1-\pi_i)$, $\boldsymbol{\pi} = (\pi_1, \cdots, \pi_n)$, $\boldsymbol{\alpha} = (a_1, \ldots, \alpha_n) \in \mathbb{R}^n$ are Lagrange multipliers and b is the bias term.

The IRLS-KLR algorithm proceeds iteratively, updating $\boldsymbol{\alpha}$ and b according to equation 2.6 and then updating $\mathbf{z}$ according to equation 2.7. As demonstrated in Ngufor and Wojtusiak (2014a), this recursive training can be unbearably slow for small to medium size datasets. Coupled with the expensive computation of the kernel matrix, IRLS-KLR can quickly become infeasible for large datasets. Using a similar approximation technique as introduced in Ngufor and Wojtusiak (2013), a least square version of KLR or LS-KLR can be derived as:

$$\begin{pmatrix} \mathbf{K} + \frac{4}{\gamma}\mathbf{I}_n & \mathbf{1}_n \\ \\ \mathbf{1}_n^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ \\ b \end{pmatrix} = \begin{pmatrix} 4(\mathbf{t} - \frac{1}{2}\mathbf{1}_n) \\ \\ 0 \end{pmatrix}. \tag{2.8}$$

Though LS-KLR significantly reduces the computational cost of IRLS-KLR, computing the kernel matrix can still be a bottle neck for very large datasets. Motivated by the Extreme Learning Machine theory of Huang et al. (2012), Ngufor and Wojtusiak (2014a) extended the theory to IRLS-KLR and LS-KLR to derive a simple, computationally efficient and accurate KLR called Extreme Logistic Regression (ELR). For LS-KLR, the ELR linear

system is given by

$$\begin{pmatrix} \boldsymbol{\phi}^T\boldsymbol{\phi} + \frac{4}{\gamma}\mathbf{I}_N & \boldsymbol{\phi}^T\mathbf{1}_n \\ \\ \mathbf{1}_n^T\boldsymbol{\phi}(\boldsymbol{\phi}^T\boldsymbol{\phi})^{-1} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\beta} \\ \\ b \end{pmatrix} = \begin{pmatrix} 4\boldsymbol{\phi}^T(\mathbf{t} - \frac{1}{2}\mathbf{1}_n) \\ \\ 0 \end{pmatrix}. \tag{2.9}$$

where $\boldsymbol{\phi} = (\phi_1(x; w_1), \ldots, \phi_N(x; w_N))^T$ are N hidden node feature mapping with respect to the input x. $\phi_i(x; w_i)$ is the activation function of the i'th hidden node. The weights of the hidden nodes $\mathbf{w} = (w_1, \ldots, w_N)$ can be randomly generated from any continuous probability distribution. After solving for $\boldsymbol{\beta}$ and b, the class posterior probabilities can be computed as:

$$\boldsymbol{\pi} = \mathbf{Pr}(t = 1|x; \boldsymbol{\beta}) = \frac{1}{1 + \exp(-\boldsymbol{\phi}(x; \mathbf{w}) \cdot \boldsymbol{\beta} - b)}$$

Based on the approached in Ngufor et al. (2014) used to derive equation 2.9, a similar fast iterative system for IRLS-KLR can be written as

$$\begin{pmatrix} \boldsymbol{\phi}^T\boldsymbol{\phi} + \frac{1}{\gamma}\boldsymbol{\phi}^T\mathbf{W}^{-1}\boldsymbol{\phi}(\boldsymbol{\phi}^T\boldsymbol{\phi})^{-1} & \boldsymbol{\phi}^T\mathbf{1}_n \\ \\ \mathbf{1}_n^T\boldsymbol{\phi}(\boldsymbol{\phi}^T\boldsymbol{\phi})^{-1} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\beta} \\ \\ b \end{pmatrix} = \begin{pmatrix} \boldsymbol{\phi}^T\mathbf{z} \\ \\ 0 \end{pmatrix} \tag{2.10}$$

with

$$\mathbf{z} = \boldsymbol{\phi}\boldsymbol{\beta} + b\mathbf{1}_n + \mathbf{W}^{-1}(\mathbf{t} - \boldsymbol{\pi}) \tag{2.11}$$

2.2 Ensemble Methods

The idea of ensemble learning is to integrate multiple classification models in order to obtain a better representative model, with more accurate and reliable estimates or decisions than

can be obtained from a single model. Experimental studies have shown that combining the outputs from multiple classifiers can significantly reduce the generalization error (Rokach, 2005). Given the potential usefulness of ensemble methods, it is not surprising that a plethora of methods exist in the machine learning literature. This section describes some of the most popular approaches which allow the combination of a set of heterogeneous classifiers.

They are three major types of classifier output: *categorical, rank*, and *measurement or continuous* levels (Kuncheva, 2004). In most cases, the categorical level are the discrete class predictions while the continuous are the class conditional probabilities. This work will mainly consider classifiers that produce class predictions and/or class probabilities.

Let $x \in \mathbb{R}^p$ be a feature vector and $T = \{t_1, \cdots, t_K\}$ be the mutually exclusive set of classes. Let $\mathcal{F} = \{f_1, \cdots, f_L\}$ be a collection of L classifiers.

2.2.1 Bayesian Model Averaging

Bayesian Model Averaging (BMA) is a well known technique ideally suited for combining multiple machine learning models through Bayes Theorem. BMA starts with a prior distribution over $\mathcal{F}$, $p(f_i)$ expressing the belief that the "right" classifier for the problem prior to observing the data is f_i. After observing the data $\mathcal{D}$, the prior belief is updated by computing the posterior probability that classifier f_i is the right classifier for the problem through the application of the Bayes Theorem:

$$p(f_i|\mathcal{D}) = \frac{p(f_i)p(\mathcal{D}|f_i)}{p(\mathcal{D})} \tag{2.12}$$

where $p(\mathcal{D}|f_i)$ is the likelihood of the data given that the right classifier is f_i

Given a new data point x, let $p(t_k|x, \mathcal{D}, f_i)$ be the class posterior probability for x by classifier f_i based on the training data $\mathcal{D}$. BMA computes the single predicted class posterior probability for x using the classifier posterior probability to weight the class posterior

probability across all classifiers i.e the predicted class conditional probability is the expected probability $p(t_k|x, \mathcal{D}, f_i)$ over $\mathcal{F}$:

$$p(t_k|x, \mathcal{D}) = E_{\mathcal{F}}\Big[p(t_k|x, \mathcal{D}, f_i)\Big]$$
$$= \sum_{i=1}^{L} p(t_k|x, \mathcal{D}, f_i)p(f_i|\mathcal{D}) \tag{2.13}$$

The classification decision can then be taken as

$$t^* = \underset{t_k \in C}{\operatorname{argmax}}\, p(t_k|x, \mathcal{D})$$

One of the limitations of the BMA is that, it is often difficult to and sometimes impossible to compute the likelihood $p(\mathcal{D}|f_i)$.

2.2.2 Stacked Generalization

Stacked generalization, as introduced in Wolpert (1992), is a multi-classifier system in which the output of base classifiers serves as new input data for the combining function or *meta-learner*. Typically the approach consists of two steps: in the first step, different based classifiers are trained and the output of each classifier is collected into a new data set. The new data along with the true class of each instance in the original data is then used to train the meta-learner in the second step.

The original data and generated base models are called *level-0* data and *level -0 model* while the new data and model generated from the new data are referred to as *level-1 data* and *level-1 generalizer*.

The effectiveness of stacked generalization in improving accuracy depends on the level-1 attributes and the type of level-1 generalizer. Ting and Witten (2011) performed a series of experiments with stacked generalization using different base classifiers whose predictions

are probability distributions over the set of class values. They concluded that, the use of Multi-response linear regression as a level-1 generalizer was more suitable than any of the non-linear algorithms attempted in their experiments. Džeroski and Ženko (2004b) extend the method of Ting and Witten (2011) to include additional level-1 attributes and to use a multi-response model tree for the level-1 generalizer. For the additional level-1 attributes, they proposed multiplying the class probability distribution by the maximum probability and to include the entropy of the probabilities. The use of class probability distribution as level-1 data instead of unique class values has the advantage of capturing the confidence of the classifiers predictions.

Stacking Framework

Given a training data $\mathcal{D} = \{(x_1, t_1), \cdots, (x_n, t_n)\}$ and L classifiers, randomly split the data into J almost equal parts $\mathcal{D}_1, \cdots, \mathcal{D}_J$. For each $j = 1, \cdots, J$, train all base classifiers on the training set $\mathcal{D} \backslash \mathcal{D}_j$ (all training data except the $j'th$ fold training data) to induce a level-0 model M_{lj} for each classifier $l = 1, \cdots, L$. For each instance x_i in $\mathcal{D}_j$ use model M_{lj} to predict the class conditional probabilities $P_l(x_i) = (P(t_1|x_i), P(t_2|x_i), \cdots, P(t_K|x_i))$ of x_i. At the end of the cross-validation process, the new data set assembled from the predictions of the L level-0 models is given by

$$\mathcal{D}_{CV} = \Big\{ t_i, P_1(x_i), P_2(x_i), \cdots, P_L(x_i) \Big\}, \quad i = 1, \cdots, n$$

$\mathcal{D}_{CV}$ forms the level-1 data set. Then use a learning algorithm (could be any of the base algorithms) to induce a level-1 model M for t as a function of the input $(P_1(x), P_2(x), \cdots, P_L(x))$

To predict the class of a new instance x_{new}, each base classifier predicts the class probabilities $\langle P(t_1|x_{new}), P(t_2|x_{new}), \cdots, P(t_K|x_{new}) \rangle$ of x_{new} which is then passed to the level-1

model for a final decision.

2.2.3 Ensemble Decision Rules

One of the most simple way of combining multiple classifiers is through non-trainable or algebraic combiners. Fixed rules such as minimum, maximum, sum, mean, product, median, etc are defined as functions that receive as inputs the outputs of a set of base classifiers and combine them to produce a unique output.

Let $p_{l_k}(x) = \mathbf{Pr}(t_k|x)$ be the normalized output scores of classifier l in class k for the data point x, then the predicted class t^* for x by some of the most popular algebraic combination rules is defined as:

1. Product Rule: $t^* = \underset{t_k \in T}{\operatorname{argmax}} \prod_{l=1}^{L} p_{l_k}(x)$

2. Mean Rule: $t^* = \underset{t_k \in T}{\operatorname{argmax}} \frac{1}{L} \sum_{l=1}^{L} p_{l_k}(x)$

3. Sum Rule: $t^* = \underset{t_k \in T}{\operatorname{argmax}} \sum_{l=1}^{L} p_{l_k}(x)$

4. Maximum Rule: $t^* = \underset{t_k \in K}{\operatorname{argmax}} \max_{l \in \{1,\ldots,L\}} \left\{ p_{l_k}(x) \right\}$

5. Minimum Rule: $t^* = \underset{t_k \in K}{\operatorname{argmax}} \min_{l \in \{1,\ldots,L\}} \left\{ p_{l_k}(x) \right\}$

2.2.3.1 Majority Vote

The simplest and oldest decision rule for combining multiple classifiers is by voting. Voting methods are applicable to classifiers whose output are unique class labels. Continuous outputs need to be converted to discrete outputs for majority vote to work. The preferred class of the ensemble using the majority (plurality) vote is the class $t^* \in T$ that receives

the largest total vote. Thus

$$t^* = \underset{t_k \in T}{\operatorname{argmax}} \sum_{l=1}^{L} \delta_{lk} \tag{2.14}$$

where

$$\delta_{lk} = \begin{cases} 1 & \text{if classifier } f_l \text{ outputs class } t_k \text{ (or } p_{l_k} = \max\limits_{t_i \in T}\{p_{l_i}\} \text{)} \\ 0 & \text{otherwise} \end{cases}$$

The above voting procedure assumes errors from individual classifiers are equally important. To account for varying errors, weights can be assigned to the more competent classifiers, that is, weak classifiers can be down-weighted. The preferred class t^* using the weighted majority method is thus

$$t^* = \underset{t_k \in T}{\operatorname{argmax}} \sum_{k=1}^{L} w_l \delta_{lk}$$

where w_l is a reliability weight for classifier f_l. However, in most applications the weights are rarely known.

2.3 Distributed Machine Learning

The amount of data in the world has been growing at an exponential rate. Businesses, healthcare, academic and government institutions amass petabytes of information on their customers, patients, suppliers and other operations continuously. The healthcare system for example is made up of multiple stakeholders including patients, providers, pharmaceutical companies, insurers, and government entities. Each of these groups generate bigger and bigger pools of data on a daily basis. Multimedia and the vast use of smartphones and social networks sites continue to fuel this exponential growth of data. "Very large" data,

or "Big data", as it is commonly referred to is now part of every sector and function of the global economy (Manyika et al., 2011).

Big data simply refers to data sets so large and complex that it becomes difficult for conventional database management systems to capture, store and process efficiently in a reasonable amount of time. Besides the share volume of big data, it has many economic value if used efficiently. Mckinsey Global Institute (Manyika et al., 2011) estimated that big data when used creatively and efficiently by the U.S healthcare industry can unlock more than $300 billion every year in additional value throughout the sector. In addition, its efficient use has the potential to revolutionize the way various stakeholders operate and most importantly, the way patients are treated.

2.3.1 Machine learning and Big Data

This unstoppable growth in big data opens the way for new application of machine learning. More advanced, scalable, and efficient automatic data analytic tools are needed to gain insight from big data. One popular approach to gaining insight from data is to deploy decision support systems that combine machine learning, optimization and visualization techniques to assist decision making. For example, machine learning assisted decision support systems are helping to reduce adverse drug reactions, reduce treatment error rates, and hence liability claims, automatic discovery of drug treatment patterns in electronic medical records (EMRs), improve the diagnosis of rare diseases, predicting complications for patients undergoing various treatment options. However, most of these machine learning assisted decision support systems are designed and work efficiently for centralized systems with relatively small data sizes. Most existing learning algorithms assume that the entire data set fits into main memory, which is not possible for big data.

A promising approach to deal with the big data crisis is to partition the data into subsets and distribute them to different computing workstations. This approach works well, because allocating learning tasks to several computing workstations is a natural way to scale up a learning algorithm. In addition, if optimized for speed then the combination can produce

efficient distributed machine learning systems.

The focus of this thesis is to design efficient distributed machine learning systems in which the data may be distributed artificially between different computing nodes, such as the distributed file system offered by Hadoop (`http://hadoop.apache.org/`) or where the data is naturally distributed, but moving data between nodes is not permitted. A good example of the latter is the situation of learning from two or more hospitals or financial institutions such as banks where data may be private. High performance computing frameworks with message passing can efficiently handle such restrictive environments. This topic is picked up again in chapter 5, where these technologies are used to implement the ensemble learning methods presented in this thesis.

This chapter ends by presenting a scalable algorithm to efficiently compute the covariance matrix of a distributed dataset in a single-pass fashion. The algorithm will find useful application in chapter 5. The algorithm was first presented in Ngufor and Wojtusiak (2014b), one of the articles written during the development of this thesis.

2.3.2 Single-Pass Parallel Statistics

Accurate computation of statistics such as the mean, variance/covariance matrix and the correlation coefficient/matrix is critical for the deployment of many machine learning applications. For example, the performance of discriminant analysis, principal component analysis, outlier detection, *etc.* depends on accurate estimation of these statistics. However, the computation of these statistics, particularly the variance or covariance matrix can be expensive on large datasets and potentially unstable when the magnitude is very small. The standard approach consists of calculating the sum of squares deviation from the mean. This involves passing through the data twice, first to compute the mean and second the deviations from the mean. This naive two-pass algorithm is known to be numerically stable, but may become too costly especially in the context of learning from large distributed data.

To overcome these difficulties, several alternative single-pass algorithms have been proposed. These include the pairwise and incremental updating formula for the variance by

Chan et al. (1979) and its extension by Bennett et al. (2009) to the covariance matrix. The draw back of these methods lies in the pairwise incremental updating steps. At each round of the computation, the data is split into two and the formula is applied recursively. Some special data structures and bookkeeping may be required to handle the tree-like structure of the algorithm. Moreover, it is not readily amenable to the distributed frameworks like Hadoop as some communications between processes may be required.

A general single-pass non-recursive covariance formula is presented next, which avoids the tree-like combination structure of previous approaches.

2.3.2.1 A Single-Pass covariance matrix formula

Given a large distributed data set D that can be partitioned into $k \geq 2$ finite blocks $D = \bigcup_{i=1}^{k} D_i$ with $D_i \cap D_j = \emptyset$, $\forall\, i \neq j$. Each D_i is typically a set of n_i multivariate random samples $D_i = \{X_1, \ldots, X_{n_i}\}$ where each X_i is a $p \times 1$ random vector: $X_i = (X_{i1}, \ldots, X_{ip})^T$. The scatter matrix for each block is given by

$$S_i = \sum_{X \in D_i} (X - \bar{X}_i)(X - \bar{X}_i)^T,$$

where $\bar{X}_i = \frac{1}{n_i} \sum_{X \in D_i} X$ is the sample mean of each block. Unbiased estimate of the covariance matrix of each block is given by $\hat{\Sigma}_i = \frac{1}{n_i - 1} S_i$. The main goal is to compute the covariance matrix of the complete data D.

Proposition 1. The sample mean vector and scatter matrix of the distributed data D

partitioned into $k \geq 2$ disjoint data-blocks $\{D_i\}_{i=1}^{k}$ is given by

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{k} n_i \bar{X}_i \tag{2.15}$$

$$S = \sum_{i=1}^{k} S_i + \frac{1}{n}\sum_{\pi} n_{\pi_1} n_{\pi_2} (\bar{X}_{\pi_1} - \bar{X}_{\pi_2})(\bar{X}_{\pi_1} - \bar{X}_{\pi_2})^T \tag{2.16}$$

where $n = \sum_{i=1}^{k} n_i$ and $\sum_{\pi}$ denotes the summation over $\binom{k}{2}$ combinations of distinct pairs (π_1, π_2) from $(1, \ldots, k)$.

Due to space limitation in Ngufor and Wojtusiak (2014b), the proof of this proposition could not be given. The full proof is given in appendix A. Experimental results comparing the performance of the single-pass formula with the standard two-pass algorithm are presented in the article.

Chapter 3: Bayes Active Data and Algorithm Selection

This chapter deals with the problem of data and algorithm selection. An active learning and decision making method is proposed for this task. The introduced methods employs two simple, fast and efficient sampling schemes based on uncertainties in the linear discriminant score and the logistic regression model to select informative data points for training a set of learning algorithms. The prediction probabilities from each classifier are then modeled by beta distributions from which a Bayesian decision procedure allows easy computation of the probability of error, ROC analysis and Precision-Recall (PR) analysis. The active learning methods for data selection were previously introduced in Ngufor and Wojtusiak (2014b), an article written in the course of developing this thesis. Thus, most of the discussion will be drawn if not replicated from that article. However, in Ngufor and Wojtusiak (2014b) the focus was on sample size reduction and not on algorithm selection. As such, the active learning algorithms are modified so that the base classifiers can be fully trained on carefully selected training examples.

3.1 Introduction

Substantial progress has been made in machine learning over the past decades in the development of learning algorithms, ranging from those based on stochastic models to those based purely on symbolic representations. Given the profusion of these learning algorithms, it is often difficult for researchers to determine which algorithm will perform most effectively on any given application problem. Usually, the researcher's understanding of the algorithms and detailed knowledge of the problem is required to select a reliable algorithm. In addition to this already difficult task, many of these algorithms have parameters that must be tuned

to adapt them to the specifics of the dataset. A substantial increase in the performance of an algorithm on a given dataset can be obtained if the right parameters are used.

A näive approach to solve this problem is to evaluate the performance of all algorithms on the dataset and select the best one. In practice this is not possible, because they are too many algorithms available and the computational complexity of some algorithms may be too expensive especially when very large datasets are involved. An alternative is to pre-select a small number of alternatives based on knowledge about the data and algorithms. Without some improvements this approach is feasible, but it may still require considerable computing time.

Traditionally, the problem of comparing and/or selecting learning algorithms has been approached in two ways. The easiest and most popular approach is to train all algorithms on a collection of datasets and compare their performances. The second approach involves constructing a predictive model or a meta-learner. In the latter approach, a meta-learner is contructed from a large collection of datasets with inputs or *meta-features* being some statistical or information theoretic summaries of the datasets. Examples of such meta-features include sample sizes, number of attributes, mean, standard deviation, skewness of numeric features, class distributions, class entropies, etc. (Brazdil et al., 2008). The meta-learner then use as dependent variable some performance measure of the algorithms on the datasets. The meta model can then be used to predict the most appropriate algorithm or tuning parameters for a given dataset.

While these approaches may be feasible in most cases, there are some major limitations. The success of meta-learning algorithm selection methods depends on the extraction of informative features able to discriminate between the performances of the base algorithms. It is therefore imperative to compute appropriate measures from the data that are good predictors of the relative performances of the algorithms. Many decisions have to be made in this process and sometimes expert knowledge may be required. A poor choice of meta-features may have a negative effect on the predictive power of the model. Meta-learning algorithm selection methods require a number of datasets to build its database. When

access to several datasets is not possible or when only one dataset is available, a meta-learner cannot be generated.

With today's ever and steadily growing dataset sizes, these methods may be computationally expensive to deploy. Without some special sampling techniques it may not be possible to get reliable estimates of performance measures of the base algorithms. Equally, it might be the case that the majority of available data is unlabeled and only a small fraction is labeled. Getting estimates of performance from this small fraction may be unreliable. In addition, the implementation of these methods requires that all data be collected at a central location. This might be unrealistic and unattainable due to band width and storage limitations. Other reasons such as privacy and legal restrictions may forbid the sharing of data. Under such conditions, it might not be possible or feasible to inspect all the datasets at one processing unit in order to train the algorithms or build a meta-learner predictive model. There is thus need for selection methods that scale up to large and possible physically distributed datasets.

This chapter presents an alternative solution to the problem of algorithm selection to help researchers make informed decisions about the algorithm most suitable for a given dataset. The selection process requires only the decisions of the base classifiers in the form of prediction probabilities. Thus, it is possible for the base algorithms to have been trained on different datasets but modeling the same input-output relationship. This is beneficial in that, it can be advantageously applied in distributed learning environments or in situations where data cannot be shared or moved around easily. The algorithms can be trained independently on each data site and their predictions collected at a central location for comparison.

Training of the base algorithms at each site is carried out by simple, fast and accurate active learning algorithms based on interval estimates of the linear discriminant score and the logistic regression model. Uncertainties in these statistics are used to select only the most informative data points for training. The presented method has the following major advantages:

1. Active learning can drastically reduce computational time by reducing the amount of data required to train an algorithm (Settles, 2010).

2. By selecting only the most informative or uncertain examples for learning, each algorithm is adapted in a data driven way to the unique characteristics of the data.

3. Each algorithm may be trained on a different subset of the input space, making them more diverse. This is an important property in ensemble learning where the purpose is to combine the selected algorithms to improve accuracy.

4. A learning algorithm may have one or more parameters that need to be tuned so as to adapt it to the specifics of the training set, such as the location of class boundary. However, by providing the learning algorithm with the "desired" input/output pairs such as those examples close to the class boundary, the problem of tuning parameters can be seen as the problem of acquiring the desired training set. Algorithms trained on their desired training set can thus be transparently compared.

5. The algorithms are compared using several performance measures instead of a single measure as commonly done for meta-learner algorithm selection methods. Since different data points may carry different misclassification error cost and some classifiers may attach high cost to false positive rates while others are more tolerant, by using several performance measures, these different qualities can be captured and the classifiers that score high across the majority of measures are suitable for ensemble learning.

6. There are many real learning problems where data is very expensive or difficult to obtain, complex and non-standard such as non stationary environments, noisy, missing values and sample bias. Under such circumstances, a method to optimally select relevant data points for training could minimize or eliminate these learning difficulties.

7. Conversely there are situations where data is available in abundance, often for free but obtaining class labels for the training set may be very expensive (e.g examining a

huge collection of breast cancer biopsy images to determine presence or non presence of breast cancer). In this case, if no optimal method to intelligently select specific data points from the pool of unlabeled data is available, the analyst is limited to any existing labeled data or the amount of data she can afford to label.

8. Finally, active learning is capable of solving the imbalance classification problem by providing the algorithms with a more balanced training set (Ertekin et al., 2007).

The data and algorithm selection method presented in this chapter first actively trains a collection of learning algorithms and extracts prediction probabilities [1] on an independent test set for further analysis by a *decision-maker* (DM). The DM then uses parametric methods to model the distribution of the prediction probabilities. Specifically, the prediction probabilities are modeled as beta parametric models from which Bayesian decision theory enables the computation of the probability of error, ROC analysis, PR analysis and the Rényi-α divergence measures for comparing the performance of the classifiers.

3.2 Active Learning

The ultimate goal of machine learning is to create systems that can improve their performance with minimal cost at some task as they acquire experience and data. In other words, the learner is able to improve its generalization as it acquire more knowledge.

In most natural learning tasks, however, the generalization problem is studied only with respect to selecting or randomly drawing from an underlying distribution independent and identically distributed training samples for labeling. The learning is simply a "passive" learner, i.e, it acts as a passive recipient of data to be processed. Passive learning is not universally applicable, it ignores the fact that in many learning situations, the most powerful tool of the learner lies in its ability to make use of at least some form of action, or do some data gathering so as to influence the environment it is trying to understand. Learning

[1] The prediction probabilities of all classifiers are assumed to be well-calibrated, that is the probabilities are reliable estimates of the true probabilities of class membership

systems that try to make effective use of this ability are referred to as *active* learners. Formally, active learning is a form of learning in which the learner has some control over the inputs on which it learns. That is, the learner has the ability to make queries that influence what data are added to its training set.

With the desirable properties of active learning, it is therefore not surprising to see the recent growth of interest in this field with quite a large number of algorithms and heuristics proposed. Some of key works on active learning can be found in Lewis and Gale (1994); Seung et al. (1992); Roy and McCallum (2001).

The key idea behind active learning is that, if the learning algorithm can optimally choose the most informative data points to be labeled, it may perform better with fewer data points and at a lower cost. Precisely, the learning algorithm itself is responsible for acquiring its training set. It can optimally select a new input x^* from a pool or a stream of unlabeled data, ask an oracle/teacher for its label t^* and then incorporate the new example (x^*, t^*) into its training set. The oracle is often a human user or domain expert or any other appropriate information source.

To formalize the discussion on active learning, assume that the active learner has access to a labeled training set $\mathcal{D}_l = \{(x_1, t_1), \ldots, (x_n, t_n)\} \in \mathcal{X} \times \mathcal{Y}$ and a set $\mathcal{D}_u = \{x_1, \ldots, x_u\} \subseteq \mathcal{X}$ of unlabeled examples. The labeled training set $\mathcal{D}_l$ is usually very small and may contain only a single data point. On the other hand the unlabeled set $\mathcal{D}_u$ may be a stream of incoming data points or a very large pool of unlabeled data. Given the training set $\mathcal{D}_l \cup \mathcal{D}_u$, the active learning task is therefore to select the "best" query instance $x^* \in \mathcal{D}_u$ and ask an oracle $\mathcal{O}$ for its label $t^* = \mathcal{O}(x^*)$ and add to $\mathcal{D}_l$: $\mathcal{D}_l \leftarrow \mathcal{D}_l \cup \{(x^*, t^*)\}$. Algorithm 1 summaries the basic active learning structure.

For active learning to be successful, there must be some method to efficiently query and evaluate the informativeness of new unlabeled examples. Let $\phi(x)$ be some query strategy to select and evaluate the usefulness of a new unlabeled example $x \in \mathcal{D}_u$. The question then is how to design $\phi(x)$ to optimally select the "best" x^*? Some of the most popular active

learning query strategies include *Uncertainty Sampling* (Lewis and Gale, 1994), *Query-by-Committee* (Seung et al., 1992), *Expected Error Reduction* (Roy and McCallum, 2001), *Variance Reduction* (Schein, 2005), minimum distance or cluster based (Fu et al., 2012; Bodó et al., 2011) etc.

Algorithm 1: Generic Active Learning Structure

Input : Training dataset $\mathcal{D}_l \cup \mathcal{D}_u$, query measure ϕ, oracle $\mathcal{O}$, stopping criteria

Output: Classified examples

1 **repeat**

2 Obtain query: $x^* \leftarrow \phi(x \in \mathcal{D}_u)$;

3 Label query: $t^* \leftarrow \mathcal{O}(x^*)$;

4 Add example to labeled training set: $\mathcal{D}_l \leftarrow \mathcal{D}_l \cup \{(x^*, t^*)\}$;

5 Induce new classifier from training dataset: $\hat{f}(x) : \mathcal{D}_l \rightarrow \mathcal{T}$;

6 Evaluate classifier/stopping criteria;

7 **until** *stopping criteria*;

3.3 Linear Discriminant Uncertainty Based Active Learning

Linear discriminant analysis (LDA) aims at discriminating between two multivariate normal populations with a common covariance matrix $H_0 = \mathcal{N}_p(\boldsymbol{\mu}_0, \boldsymbol{\Sigma})$ and $H_1 = \mathcal{N}_p(\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$ say, on the basis of independent random samples of sizes n_0 and n_1. Fisher's linear discriminant rule assigns a new test example x into population H_0 if the discriminant score $\theta(x)$ satisfies

$$\theta(x) = \lambda_0 + \boldsymbol{\lambda}^T x \geq 0 \tag{3.1}$$

where $\lambda_0 = \log(\pi_1/\pi_0) - \frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_0)$, $\boldsymbol{\lambda} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$ and π_i is the probability that x belongs to population $H_i, i = 0, 1$ (Ngufor and Wojtusiak, 2014b).

The decision boundary is defined by points satisfying $\theta(x) = 0$. If the true value of $\theta(x)$

is known, then these points can be easily determined. However $\theta(x)$ is not known and is usually estimated using unbiased sample version, such as, the minimum variance unbiased estimator (Critchley and Ford, 1985) given as

$$\hat{\theta}(x) = \frac{1}{2}\hat{\alpha}_1^2(x) - \frac{1}{2}\hat{\alpha}_0^2(x) + \log(n_1/n_0) \tag{3.2}$$

where $\hat{\alpha}_i^2(x) = (n_0 + n_1 - p - 3)(x - \bar{X}_i)^T \mathbf{S}_p^{-1}(x - \bar{X}_i) - p/n_i$ with $\bar{X}_i$ the sample mean and $\mathbf{S}_p$ the pooled covariance matrix. If the probability distribution of $\hat{\theta}(x)$ is known, then one can easily find the likelihood that the true value of the score is within some specified range for each test point x. For example if $\hat{\theta}(x)$ is assumed to be normally distributed, then a 95% confidence interval centered at 0 will correspond to data points close to the decision boundary.

The derivation of the LDA active learning scheme is based on an approximate distribution of $\hat{\theta}(x)$ derived in Critchley and Ford (1985) under the assumption of equal priors i.e $\pi_0 = \pi_1$. The case of unequal priors can be adjusted accordingly. Letting $\Delta_i^2(x) = (x - \boldsymbol{\mu}_i)^T \Sigma^{-1}(x - \boldsymbol{\mu}_i)$ be the squared Mahalanobis distance between x and the population center $\boldsymbol{\mu}_i$ and $\phi(x) = \frac{1}{2}\Delta_1^2(x) + \frac{1}{2}\Delta_0^2(x)$, it is shown in Critchley and Ford (1985) that $\hat{\theta}(x)$ is asymptotically normally distributed with mean $\theta(x)$ and variance $var(\hat{\theta}(x))$ given (in a simplified form) as:

$$\begin{aligned} var(\hat{\theta}(x)) = {} & \frac{a}{d}(\theta(x) - cM/2N)^2 \\ & + \frac{b}{d}(\phi(x)(cn/N + \Delta_1^2(\boldsymbol{\mu}_0)) - \Delta_1^4(\boldsymbol{\mu}_0)/4) \\ & + \frac{bc}{d}\big(2p(n+M)/N^2 - (c+1)M^2/N^2\big) \end{aligned} \tag{3.3}$$

where $n = n_0 + n_1,\ M = n_1 - n_0, N = n_0 n_1,\ a = N - p - 1,\ b = a - 2, c = N - 2$, and

$d = (a-1)(b-2)$.

With the approximate distribution of $\hat{\theta}(x)$, uncertainties in classifying each data point x can be estimated by computing confidence intervals about the mean value $\theta(x)$. In particular, the $(1-\delta)100\%$ confidence interval about the decision boundary $\theta(x) = 0$ is given by

$$-Z_{1-\delta/2} \leq \frac{\hat{\theta}(x)}{\sqrt{var(\hat{\theta}(x))}} \leq Z_{1-\delta/2} \tag{3.4}$$

where $Z_{1-\delta/2}$ is the Z-score at confidence level δ. Data points within this interval represent points for which the classifier has high uncertainty about class memberships and are the most informative for learning. A large confidence interval will select more points while a tight interval will return fewer points. Equation 3.4 therefore presents an efficient principled query strategy for uncertainty base active learning (Settles, 2010) that can be used for sample size reduction.

In the standard pooled based active learning parlance, a small labeled training set $\mathcal{D}_l$ and a large "unlabeled" pool $\mathcal{D}_u$ are assumed to be available. The task of the active learner is to use the information in $\mathcal{D}_l$ in a smart way to select the best query point $x^* \in \mathcal{D}_u$ and ask the user or an oracle for its label and then add it to $\mathcal{D}_l$. This process continues until the desired training set size or accuracy of the learner has been archived. For the presented data selection schemes, both $\mathcal{D}_l$ and $\mathcal{D}_u$ are labeled training sets and the goal is to select the most informative data points and their labels from $\mathcal{D}_u$. Algorithm 2 presents the LDA sampling scheme. The stopping criterion can be set equal to the required sample size of the reduce training set.

When Algorithm 2 is applied to train a number of algorithms such as in ensemble learning, a different training set may be returned by each classifier. Though the selection of examples is based on the linear discriminant score, however the selection is guided by the predictions of each classifier. Each selected data point for training can therefore be considered as informative to that classifier. The active training of the classifiers therefore

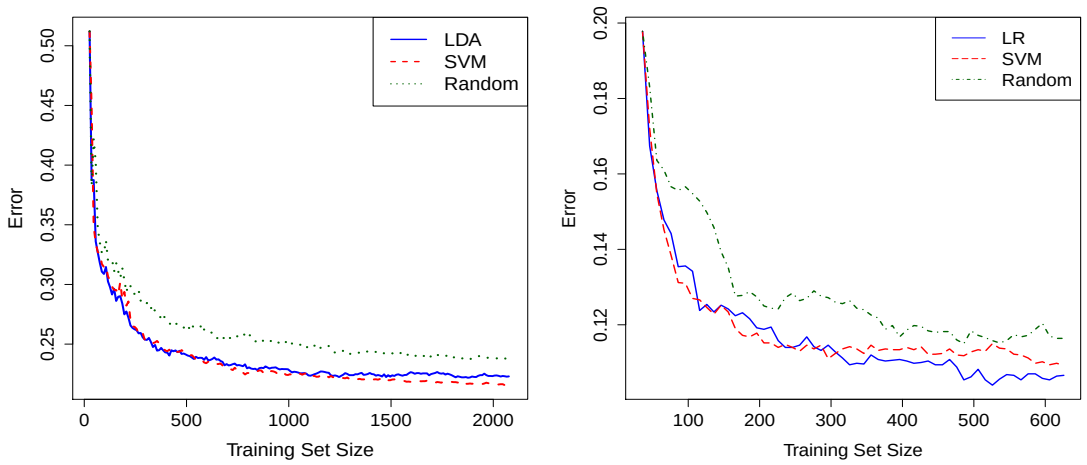
Algorithm 2: LDA Active Data Selection Algorithm

Input : $\mathcal{D}_l$, $\mathcal{D}_u$, learning algorithm f, confidence level δ, stopping criterion γ
Output: Training set $\mathcal{D}_f$, classifier f

Set $\mathcal{D}_f \leftarrow \mathcal{D}_l$ `// most informative instances according to classifier` f
repeat
  Compute $\bar{X}_0, \bar{X}_1$ and $\mathbf{S}_p$ using $\mathcal{D}_l$;
  Train classifier f on $\mathcal{D}_f$;
  $\forall\, x \in \mathcal{D}_u$, predict class label: $\hat{t} \leftarrow f(x)$;
  $\forall\, x \in \mathcal{D}_u$, compute $\hat{\theta}(x)$ and $var\{\hat{\theta}(x)\}$;
  select x^* using equation 3.4;
  Set $\mathcal{D}_l \leftarrow \mathcal{D}_l \cup \{x^*, \hat{t}\}$;
  Set $\mathcal{D}_f \leftarrow \mathcal{D}_f \cup \{x^*, t\}$; `//` t `is the true class of` x^*
  Evaluate stopping criteria;
until γ;

makes the base classifiers more diverse. Optionally, after the active learning process, the informative data points from each classifier can be aggregated and all the algorithms retrain on the aggregated data.

Figure. 3.1 (a) shows a comparison of the performance of logistic regression trained actively using: data points selected by LDA active learning technique, the support vectors from a SVM active learning (Tong and Koller, 2002), and random sampling (Random). The Forest Covertype data from the UCI Machine learning repository (Frank and Asuncion, 2010) was used for training. The classification problem represented by this data is to discriminate between 7 forest cover types using 54 cartographic variables. The data was converted to binary by combining the two majority forest cover types (Spruce-Fir with n = 211840, and Lodgepole Pine with n = 283301) to one class and the rest (n = 85871) to the second class. The data was split into 75% training and 25% testing. The sampling schemes were stopped once 0.7% of the training set has been queried for learning. The results showed that for the LDA sampling scheme to archive a reduction in error of about 22% (approximately where the algorithm stabilizes) only about 0.23% carefully selected training data points where needed whereas random sampling method uses all 0.7% of the training data and still achieved only 24 % reduction in error. The performance of LDA



(a) Comparing LDA, SVM and Random sampling (b) Comparing LR, SVM and Random sampling

Figure 3.1: Performance of Different Sampling Schemes

and SVM sampling schemes are very similar, however LDA took by far a smaller time to converge compared to SVM. Precisely in this example, the time ratio of SVM to LDA was about 140 averaged over ten fold cross-validation.

3.4 Logistic Regression Uncertainty Based Active Learning

Let $\mathcal{D}_l = \{(x_i, t_i)\}_{i=1}^n$ be a set of training examples where the random variables $Y_i = t_i \in \{0, 1\}$ are binary and $X_i = x_i \in \mathbb{R}^p$ are p-dimensional feature vectors. The fundamental assumption of the logistic regression (LR) model is that the log-odds or "logit" transformation of the posterior probability $\pi(\boldsymbol{\beta}; x) = p(t = 1|x; \boldsymbol{\beta})$ is linear i.e

$$\log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \boldsymbol{\beta}^T x \tag{3.5}$$

where β_0, and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$ are the unknown parameters of the model.

Typically, the method of maximum likelihood is used to estimate the unknown parameters $(\beta_0, \boldsymbol{\beta})$. By setting $x = (1, x)$ and $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_p)$, the regularized log-likelihood function is given by

$$l(\boldsymbol{\beta}) = \boldsymbol{\beta}^T \mathbf{X}^T \mathbf{Y} - \sum_{i=1}^{n} \log\left(1 + e^{x_i^T \boldsymbol{\beta}}\right) - \gamma \|\boldsymbol{\beta}\|^2/2$$

where $\mathbf{X}$ is the design matrix, $\mathbf{Y}$ the response vector and γ reflects the strength of regularization. Iterative methods such as gradient based methods or the Newton-Raphson method are commonly used to compute the maximum likelihood estimates (MLE) $\hat{\boldsymbol{\beta}}$ of $\boldsymbol{\beta}$. For example, the one step training of L_2 regularized stochastic gradient descent (SGD) ia given by

$$\boldsymbol{\beta}^{new} = \boldsymbol{\beta}^{old} + \alpha \left[\left(t_i - \pi_i\left(\boldsymbol{\beta}^{old}\right)\right) x_i - \gamma \boldsymbol{\beta}^{old}\right] \tag{3.6}$$

where $\alpha > 0$ is the learning rate. Each iteration of SGD consist of choosing an example (x_i, t_i) at random from the training set and updating the parameter $\boldsymbol{\beta}$.

One important feature of the LR parameters is that the parameter estimates are consistent. It can be shown that the MLE of LR are asymptotically normally distributed i.e

$$\sqrt{n}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \longrightarrow \mathcal{N}\left(\mathbf{O}, \mathbf{I}(\boldsymbol{\beta})^{-1}\right)$$

where $\mathbf{I}(\boldsymbol{\beta}) = \mathbf{X}^T \mathbf{W} \mathbf{X}$ is the Fisher information matrix with $\mathbf{W} = \mathbf{diag}\{\pi_i(1 - \pi_i)\}, i = 1, \ldots n$ (see for example Bickel and Li (1977) Section 6.5).

Based on the distribution of $\hat{\boldsymbol{\beta}}$, the asymptotic distribution of the MLE of the logistic function can be derive by application of the delta method. Specifically, for any real valued

function g with the property that $\frac{\partial g(\beta)}{\partial \beta} \neq 0$ one has

$$\sqrt{n}(g(\hat{\boldsymbol{\beta}}) - g(\boldsymbol{\beta})) \longrightarrow \mathcal{N}\Big(\mathbf{O}, \nabla g(\boldsymbol{\beta})^T \mathbf{I}(\boldsymbol{\beta})^{-1} \nabla g(\boldsymbol{\beta})\Big)$$

where ∇ is the gradient operator. Taking $g(\boldsymbol{\beta}) = \pi(\boldsymbol{\beta}; x)$, the posterior probabilities of the LR model are asymptotically normally distributed with mean $\pi(\boldsymbol{\beta}; x)$ and variance $Var(\pi(\hat{\boldsymbol{\beta}}; x)) = \nabla \pi(\boldsymbol{\beta}; x)^T \mathbf{I}(\boldsymbol{\beta})^{-1} \nabla \pi(\boldsymbol{\beta}; x)$.

The decision boundary for the LR model is defined by $\beta_0 + \boldsymbol{\beta}^T x = 0$, i.e where $\pi(\boldsymbol{\beta}; x) = 0.5$. This shows that points on the boundary have equal chances of being assigned to either population. The uncertainties in $\pi(\hat{\boldsymbol{\beta}}; x)$ for the boundary points can therefore be statistically captured by a $(1-\delta)100\%$ confidence interval about 0.5.

Confidence intervals for parameter estimates of LR can be calculated from critical values of the student t-distribution. By following a similar calculation presented in Rencher and Schaalje (2008) section 8.6.3 for computing the confidence interval of a linear function $\mathbf{a}^T\boldsymbol{\beta}$ of parameters of a linear regression model, one obtain for $\pi(\hat{\boldsymbol{\beta}}; x)$ the statistics

$$t = \frac{\pi(\hat{\boldsymbol{\beta}}; x) - \pi(\boldsymbol{\beta}; x)}{\sqrt{Var(\pi(\hat{\boldsymbol{\beta}}; x)}} \sim t(n-p-1)$$

which has a student t-distribution with $n-p-1$ degrees of freedom. Uncertainties about the true decision boundary $\pi(\boldsymbol{\beta}; x) = 0.5$ can then be determined by computing confidence intervals. In particular, the $(1-\delta)100\%$ confidence interval about the decision boundary is given by

$$-t_{\frac{\delta}{2}, n-p-1} \leq \frac{\pi(\hat{\boldsymbol{\beta}}; x) - 0.5}{\sqrt{Var(\pi(\hat{\boldsymbol{\beta}}; x))}} \leq t_{\frac{\delta}{2}, n-p-1} \tag{3.7}$$

A similar algorithm for sample size reduction using the logistic regression model is presented in Algorithm 3.

Algorithm 3: LR Active Learning

Input : $\mathcal{D}_l$, $\mathcal{D}_u$, learning algorithm f, confidence level δ, stopping criterion γ
Output: Training set $\mathcal{D}_f$, classifier f

Set $\mathcal{D}_f \leftarrow \mathcal{D}_l$ `// most informative instances according to classifier` f
repeat
  Estimate $\hat{\boldsymbol{\beta}}$ using $\mathcal{D}_l$;
  Train classifier f on $\mathcal{D}_f$;
  $\forall\ x \in \mathcal{D}_u$, predict class label: $\hat{t} \leftarrow f(x)$;
  $\forall\ x \in \mathcal{D}_u$, compute $\pi(\hat{\boldsymbol{\beta}}; x)$ and $Var(\pi(\hat{\boldsymbol{\beta}}; x))$;
  select x^* using equation 3.7;
  Set $\mathcal{D}_l \leftarrow \mathcal{D}_l \cup \{x^*, \hat{t}\}$;
  Set $\mathcal{D}_f \leftarrow \mathcal{D}_f \cup \{x^*, t\}$; `//` t `is the true class of` x^*
  Evaluate stopping criteria;
until γ;

Figure 3.1 (b) shows the error curve for LR trained actively using: data points selected by the LR active learning technique, the support vectors from a SVM active learning technique, and random sampling. The Waveform dataset from the UCI machine learning repository was used for this example. There are a total of 5000 records in this data with 40 attributes, 75% was used for training and 25% for testing. All the sampling schemes were stopped when 16% of the training set has been queried for learning. Clearly, the LR sampling scheme outperforms both the SVM and Random schemes.

3.5 Hausman Specification Test

Under the normal assumption, LDA and LR estimators are both known to be consistent but LDA is asymptotically more efficient (Efron, 1975). Though the LDA and LR models are very similar in form, there are significantly different in model assumptions. LR makes no assumptions about the distribution of the independent variables while LDA explicitly assumes a normal distribution. An excellent discussion of the two learning techniques can be found in Cox and Snell (1989). Specifically, LR is more applicable to a wider class of distributions of the input than the normal LDA. However, as illustrated in Efron (1975),

when the normality assumption holds, LDA is more efficient than LR. Under non-normal conditions, LDA is generally inconsistent whereas LR maintains its consistency. Since LDA may perform poorly on non-normal data, an important criterion for choosing between the LDA and LR active learning techniques is to check whether the assumption of normality is satisfied before querying a new data point. The Hausman specification test (Lo, 1986) can be applied to test for these distributional assumptions by comparing the two estimators.

The Hausman's specification test is an asymptotic chi-square test based on the quadratic form obtained from the difference between a consistent estimator under the alternative hypothesis and an efficient estimator under the null hypothesis. Under the null hypothesis of normality, both LDA and LR estimators should be numerically close, implying that for large samples sizes, the difference between them converges to zero. However under the alternative hypothesis of non-normality the two estimators may differ and sometimes by a large margin. Naturally then, if the null hypothesis is true, one should use the more efficient estimator, which is the LDA estimator and LR estimator otherwise.

Let $\hat{\mathbf{\Sigma}}_{LDA}$ and $\hat{\mathbf{\Sigma}}_{LR}$ be the estimated asymptotic covariance matrices of $\hat{\boldsymbol{\lambda}}$ and $\hat{\boldsymbol{\beta}}$; the estimators of LDA and LR respectively ($\hat{\mathbf{\Sigma}}_{LR}$ in this case is the observed Fisher information matrix $\mathbf{I}(\hat{\boldsymbol{\beta}})$). Letting $\hat{\mathbf{Q}} = \hat{\boldsymbol{\lambda}} - \hat{\boldsymbol{\beta}}$, the Hausman chi-squared statistic is defined by

$$\mathcal{H} = \hat{\mathbf{Q}}^T [\hat{\mathbf{\Sigma}}_{LDA} - \hat{\mathbf{\Sigma}}_{LR}]^{\dagger} \hat{\mathbf{Q}} \sim \chi^2_p \tag{3.8}$$

where † denotes the generalized inverse.

During training, $\hat{\mathbf{\Sigma}}_{LR}$ is readily available through the Fisher information matrix. Therefore, the main difficulty in computing the Hausman statistic is how to compute $\hat{\mathbf{\Sigma}}_{LDA}$. Several methods have been proposed in the literature to compute $\hat{\mathbf{\Sigma}}_{LDA}$ (Lo, 1986; Efron, 1975). These methods are however too complex to implement on large and/or distributed datasets. A much simpler approach can be pursued by considering the asymptotic distribution $\hat{\mathbf{\Sigma}}_{LDA}$ given by the following proposition.

Proposition 2. Given the training set $\mathcal{D}_l = \{(x_i, t_i),\ i = 1, \ldots n\}$ where $t_i = j = 0, 1$ indicates the multivariate normal $\mathcal{N}_p(\boldsymbol{\mu}_j, \boldsymbol{\Sigma})$ that x_i comes from. The limiting distribution of

$$\sqrt{n}(\boldsymbol{\lambda}^* - \boldsymbol{\lambda}) \sim \mathcal{N}_p(\mathbf{O}, \boldsymbol{\Gamma})$$

where $\boldsymbol{\lambda}^* = (n-2)\hat{\boldsymbol{\lambda}}$, $\boldsymbol{\lambda} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)$ and $\boldsymbol{\Gamma} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1} + \left(\frac{n}{n_2 n_1} + \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right)\boldsymbol{\Sigma}^{-1}$ with $\boldsymbol{\mu} = \boldsymbol{\mu}_1 - \boldsymbol{\mu}_0$.

The proof of this proposition can be obtained by noting that $\sqrt{n_0 n_1/n}(\bar{x}_1 - \bar{x}_0) \sim \mathcal{N}_p(\sqrt{n_0 n_1/n}\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Using this and the fact that the pooled sample covariance matrix $(n-2)\mathbf{S}_p \sim \mathcal{W}(\boldsymbol{\Sigma}; N, p)$ has the Wishart distribution with $N = n_1 + n_0 - 2$ degrees of freedom along with some simplifying algebraic transformations of random variables from these distributions, the result follows by application of Theorem 3.1 in Gupta (1968).

3.6 Comparing Machine Learning Classification Models

As noted earlier, the comparison and selection of classifiers is an important component in many machine learning applications. In the literature, comparing the generalization error of learning algorithms is generally performed through statistical tests (Alpaydm, 1999). Typically, this procedure is carried out through a cross-validation set-up. For example in Bradley (1997), the analysis of variance (ANOVA) is used to test for statistical significance of any difference in accuracy and the area under the receiver operating characteristics (ROC).

This section presents a number of performance measures for comparing the performances of binary classification models presented in the form of posterior class probabilities. By approximating the distribution of the posterior probabilities using parametric models, a Bayesian decision framework allow easy computation of the Bayes risk, ROC analysis and Precision-Recall (PR) analysis for the evaluation of a decision rule for a given classification problem.

The comparison framework proposed in this work is that of a decision-marker (DM) who consults multiple experts regarding an outcome or event that was modeled by the experts. The experts or base-learners express their opinion about the event in the form of probabilities. No information about the training set is assumed to be available.

The opinions of the base-learners will obviously differ since they may have been trained on different subsets of the input space or due to the fact that they make different assumptions or because they may be built from different underlying theories. Thus the comparative measures presented here aim to rank the base-learner according to the superiority of their opinions.

3.6.1 A Bayes Decision Theoretic Framework

For generality, the Bayes decision framework will first be presented for a multiclass classification problem. Later, the results will be restricted to the binary case for which most of the analysis in this thesis are base upon.

Consider a set of L base-learners $\mathcal{H} = \{h_1, \ldots, h_L\}$ each modeling one of $\theta \in \Theta = \{1, 2, \ldots K\}$ possible events or classes represented by probabilities $x_j = (x_{j1}, x_{j2}, \ldots, x_{jK})$, where $x_{jk} = p(\theta = k|h_j), j = 1, \ldots L, k = 1, \ldots K$ is learner h_j's opinion regarding the chance that $\theta = k$. When $\theta = k$, x_{jk} should tend to be large and small otherwise. For clarity of presentation, the j subscript will be dropped with the understanding that there are L different probabilities x to be modelled independently.

Given that the DM has a prior probability π_θ of the events before receiving the probabilities x. After observing the data x, based on its prior, the DM must then assign a probability to the particular value x as indicative of the event $\theta = k$. That is, the DM must express its believe that the value x assigned by h is indicative of the event $\theta = k$ in the form of probability densities $\pi(\theta = k|x), k = 1, \ldots k$ (or simply $\pi(k|x)$). By Bayes theorem, the

DM posterior probability of the event $\theta = k$ given x is given by

$$\pi(k|x) = \frac{f(x|k)\pi_k}{m(x)},\ k = 1, \dots K \tag{3.9}$$

where $f(x|k)$ is the likelihood of the base-learner's opinion given that the event is k and $m(x) = \sum_{i=1}^{K} f(x|i)\pi_i$ is the marginal probability density function of x. Having obtained the class posterior probabilities, the DM can now make decisions about the true representative event expressed in each x.

Many problems in statistics and machine learning involve making decisions under a state of uncertainty. The Bayesian approach to statistical decision theory is a formal way of incorporating prior beliefs with knowledge available from observed data with the goal of deriving optimal decision rules.

Consider the problem of selecting a decision $a \in \mathcal{A} = \Theta = \{1, 2, \dots K\}$ when the uncertainty about an event $\theta \in \Theta$ is expressed by probabilities x supplied by multiple base-learners. No decision can be made without potential losses, thus for each decision-event pair (a, θ) there is an associated loss (or cost) that can be quantified in terms of a loss function $L(a, \theta) : \mathcal{A} \times \Theta \to [0, \infty)$. Generally, no loss is associated with a correct decision, that is $L(a, \theta) = 0$ if $a = \theta$. The objective of the DM is to select a decision a that minimizes the Bayes conditional risk defined by

$$R(a|x) = \mathop{\mathbb{E}}_{\Theta}[L(a, \theta)|x] = \sum_{k=1}^{K} L(a, k)\pi(k|x)$$

Thus the optimal decision rule is $\delta = \text{argmin}_{a \in \mathcal{A}} R(a|x)$. A measure of performance of the decision rule can be assessed through the Bayes risk

$$r = \mathop{\mathbb{E}}_{\mathcal{X}}[\min_{a \in \mathcal{A}} R(a|x)] = \sum_{k=1}^{K} \int_{\mathcal{X}} \min_{a \in \mathcal{A}} L(a, k) f(x|k)\pi_k \, dx$$

where $\mathcal{X}$ is the probability distribution of x. Note that the Bayes risk is a real number independent of x and θ and so can be ordered for each base-learner, permitting a direct comparison. Evaluation of the Bayes risk requires specification of the loss function and the posterior probability.

Taking the log ratio of equation 3.9 for two distinct events $\theta = k$ and $\theta = i$ gives

$$\log\left(\frac{\pi(k|x)}{\pi(i|x)}\right) = \log\left(\frac{f(x|k)}{f(x|i)}\right) + \log\left(\frac{\pi_k}{\pi_i}\right) \tag{3.10}$$

The above equation states that the DM posterior log-odds is equal to the sum of the log-likelihood ratio and its prior log-odds. The DM's prior log-odds is its initial odds about the events. So if the log-likelihood is positive then its posterior log-odds is greater than its prior log-odds and the minimum risk condition will make the DM favor the event $\theta = k$ over $\theta = i,\ i \neq k$. On the other hand if the log-likelihood is negative then the DM favors $\theta = i$ over $\theta = k$. Thus it can be seen that the log-likelihood ratio is a measure of how the DM values the base-learners opinion expressed in each x as indicative of one of the events $\theta = k$. For a positive log-likelihood ratio, the DM favors $\theta = k$ over all the others, this can be written as

$$-\log\Big(f(x|k)\pi_k\Big) + \max_{i\neq k}\Big\{\log\Big(f(x|i)\pi_i\Big)\Big\} < 0 \quad i,k = 1,\ldots k$$

Thus for each k, the sign of the discriminating function

$$S(x|k) = -\log\Big(f(x|k)\pi_k\Big) + \max_{i\neq k}\Big\{\log\Big(f(x|i)\pi_i\Big)\Big\}$$

indicates a decision selection criterion. If $S(x|k) < 0$ the DM favors the event $\theta = k$ for x and selects the decision $a = k$. If however $S(x|k) \geq 0$ when the true value is $\theta = k$, then an error will be committed and a loss incurred. A loss function for this decision making can

be defined as

$$L(i,k) = \begin{cases} \lambda_{ik} & \text{if } S(x|k) \geq 0 \text{ and } S(x|i) < 0 \ \forall \ i \neq k \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

where λ_{ik} specifies the error cost associated with choosing the event $a = i$ when the true event is $\theta = k$. This reduces to the "0/1" loss function when $\lambda_{ik} = 1, \ \forall \ i \neq k$. The sets $X_k = \{x : S(x|k) < 0\}$ are the decision regions which are mutually disjoint i.e $X_i \bigcap X_k = \emptyset, \forall \ i \neq k$ and $\mathcal{X} = \bigcup_k X_k$. Thus the Bayes risk and optimal decision rule can then be written respectively as

$$r = \sum_{k=1}^{K} \left(\int_{\bigcup_{i \neq k} X_i} \lambda_{ik} f(x|k) \pi_k \, dx \right) \tag{3.12}$$

$$\delta = \underset{i \in \mathcal{A}}{\operatorname{argmax}} \left(\sum_{k=1}^{K} \lambda_{ik} \pi(k|x) \right) \tag{3.13}$$

Computing the Bayes risk and optimal rule depends on the specification of the error cost λ_{ik} for every decision-event pair. Choosing the error cost is highly problem dependent and also on the computational complexity involved. For the simple 0/1 loss function, the optimal rule is simply to select the event with the highest posterior probability or *maximum a posteriori* (MAP) given the particular value x.

For a binary decision problem, $\mathcal{A} = \Theta = \{0, 1\}$ and $\pi_1 = 1 - \pi_0 = \pi$. The MAP condition implies that the DM should select the event $a = 1$ if $\lambda_{10}\pi(0|x) < \lambda_{01}\pi(1|x)$ and

$a = 0$ otherwise. That is

$$\delta = \begin{cases} 1 & \text{if} \quad \dfrac{\pi(1|x)}{\pi(0|x)} > \dfrac{\lambda_{10}}{\lambda_{01}} \\ \\ 0 & \text{otherwise} \end{cases}$$

Using the definition of the posterior in equation 3.9, the rule can be equivalently written as

$$\delta = \begin{cases} 1 & \text{if} \quad l(x) > \tau \\ \\ 0 & \text{otherwise} \end{cases}$$

where

$$l(x) = \log\left(\frac{f(x|1)}{f(x|0)}\right) \text{ and } \tau = \log\left(\frac{1-\pi}{\pi}\right) + \log\left(\frac{\lambda_{10}}{\lambda_{01}}\right).$$

This form makes it clear that the Bayes optimal decision rule tries to make a compromise between the evidence about the events supplied by the base-learners and prior knowledge about the events and the error cost. The combined prior log-odds and the log cost ratio yield a decision threshold τ. In medical diagnostic testing for example, this is simply the cut-off value of the test. Note that the definition of the threshold here assumes the prior and/or error cost are known. In problems where these parameters are not known or well defined, the approach may simply to decide on $a = 1$ if $l(x) > \tau$ where τ may be independent of the prior and/or cost. By varying the threshold different decisions can be made based on the problem.

This simple and intuitive Bayesian decision making framework has been extensively studied by statisticians and other researchers over the past few decades. It has been used in statistics, signal detection and weather forecasting to combine the subjective probabilities provided by multiple experts (Jacobs, 1995; Krzysztofowicz and Long, 1990).

3.6.2 Error Probabilities

Based on the optimal decision rule for the binary case, the decision regions can be written as $X_1 = \{x : \ l(x) > \tau\}$ and $X_0 = \{x : \ l(x) \leq \tau\}$. The Bayes risk (equation 3.12) can be written as

$$r = \lambda_{01}\pi \int_{X_0} f(x|1)\, dx \ + \ \lambda_{10}(1-\pi) \int_{X_1} f(x|0)\, dx$$

It can be seen that the integrals

$$I_\theta = \int_{X_\theta^c} f(x|\theta)\, dx, \quad \theta = 0, 1 \tag{3.14}$$

represents the probability of incorrect decision ("error") conditioned on each of the two possible events $\theta = 0, 1$ (X_θ^c is the complement of the set X_θ). The integral $I_0 = p(a = 1|\theta = 0)$ is the probability that the value x indicative of the event $\theta = 0$ will be taken to lie in the decision region X_1 associated with the event $\theta = 1$. If $\theta = 1$ represents the positive class and $\theta = 0$ the negative, then I_0 is simply the type 1 error or false positive. The integral $I_1 = p(a = 0|\theta = 1)$ on the other hand qualifies the probability of the other type of error or false negatives. The Bayes risk can then be written as

$$r = p(\text{“}error\text{”}|\theta = 1)\lambda_{01}\pi + p(\text{“}error\text{”}|\theta = 0)\lambda_{10}(1-\pi) = p(\text{“}error\text{”})$$

The probability of error is simply the sum of the off diagonal elements of a classification confusion matrix weighted by the prior and error cost. Thus the error or accuracy depends on prior and cost. Since these parameters are usually not known in most real applications, this dependence is often ignored and the probability of error is simply computed as

$$p(\text{“}error\text{”}) = 1 - p(\text{“}correct \;\; decision\text{”})$$

$$= 1 - p(a = 1|\theta = 1) - p(a = 0|\theta = 0).$$

Classifiers with small probability of error are preferred.

The erorr probability offers a useful means to evaluate the performance of a decision rule for a given classification problem. However, the probability of error can be very difficult to compute, and exact closed-form expressions are only attainable in some very simple situations. Approximate parametric and numerical methods are the usual approaches implemented to compute them.

3.6.3 Beta Likelihood Model for Prediction Probabilities

Despite the simplicity of the Bayesian decision theoretic framework presented in section 3.6.1, it might be impractical in some real applications. The tractability of the integrals involved in the Bayes risk depends upon the successful modelling of the likelihood function. Defining an appropriate likelihood function can be a difficult task and a poorly defined function may render computation of the Bayes risk impossible or very expensive. It is therefore common in decision theory to make certain simplifying distributional assumptions to ameliorate these problems.

For the binary decision problem, the marginal distribution of the base-learners' posterior probability is given by

$$m(x_j) = \pi f(x_j|1) + (1 - \pi) f(x_j|0), \quad j = 1, \ldots, L$$

A good base-learner provides reliable estimates of the true probability that a test sample is a member of the class of interest. One will therefore expect the probabilities to be clustered near $x_j = 0$ and $x_j = 1$ producing two distinct clusters in the distribution of $m(x_j)$. A poor

base-learner on the other hand produces unreliable estimates with possible high frequency of x_j values near 0.5 resulting in two or more clusters in $m(x_j)$. The heterogeneity of the distribution of x naturally motivates modeling it as a mixture of models. When data is characterized by population heterogeneity, that is, consisting of unlabeled observations each of which is thought to belong to one of distinct classes, finite mixture models have been recognized as a useful and powerful probabilistic tool for modelling the data (McLachlan and Peel, 2000). The probability of an observation falling into any of the components of the mixture or the component densities themselves can then be used to model the likelihood ratio.

Despite the favorable properties of finite mixtures, its is often computationally very expensive to implement and therefore not desirable for the problem addressed in this study. However, with the recent developments of Markov chain Monte Carlo techniques, these computational burdens can be lessened. Gelfand et al. (1995) for example applied a Gibbs sampling technique to estimate the parameters of a finite mixture of beta models used to model the likelihood function of multiple experts opinions expressed as partial probabilities. To keep all computations relatively simple, the more common approach of modeling each base-learners prediction probability by a member of a standard parametric family of models is pursued in this work.

The Gaussian family of densities is the most common choice for the likelihood function. Jacobs (1995) for example combined the opinions of multiple experts by modeling their likelihood functions as normal models. Due to the fact that the Beta distribution has support [0,1] and is frequently used for modeling probabilities and proportions, this work assumes that the likelihood functions are beta distributions. That is, the prediction probabilities x_j provided by base-learner j about the events $\theta = 1$ and $\theta = 0$ are modeled by the beta distribution given by

$$f(x_j|1) \sim \mathcal{B}(x_j; a_{j1}, b_{j1})$$

$$f(x_j|0) \sim \mathcal{B}(x_i; a_{j0}, b_{j0})$$

where $\mathcal{B}(x; a, b)$ is the probability density function of the beta distribution defined by

$$\begin{aligned}\mathcal{B}(x; a, b) &= \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\, x^{a-1}(1-x)^{b-1} \\ &= \frac{1}{\mathbf{B}(a, b)}\, x^{a-1}(1-x)^{b-1} \end{aligned} \tag{3.15}$$

with $0 \leq x \leq 1, \quad a, b > 0$, Γ is the gamma function and $\mathbf{B}$ the beta function. The mean and variance of the distribution are respectively

$$E(x) = \frac{a}{a+b}$$

$$Var(x) = \frac{ab}{(a+b)^2(a+b+1)}$$

The beta density can have quite different shapes depending on the values of the parameters a and b that index the distribution. It is a very flexible distribution that can produce a unimodal, uniform, or bimodal distribution of points that can either be symmetrical or skewed. The beta distribution is therefore more appropriate to model the base-learners prediction probabilities than the normal distribution. Furthermore, because the opinions of the base-learners are define by $x_j = p(\theta = 1|h_j)$ and $1 - x_j = p(\theta = 0|h_j)$ it is clear that a "symmetric beta likelihood" with only two parameters instead of four can be assumed. That is

$$f(x_j|1) \sim \mathcal{B}(x_j;\, a_j, b_j)$$

$$f(x_j|0) \sim \mathcal{B}(x_j;\, b_j, a_j)$$

This assumption is reasonable and can be argued for by the fact that, when $b_j > a_j$, values

close to 0 become more likely than those close to 1 and when $b_j < a_j$, values close to 1 are more likely than those close to 0. For $a_j = b_j$ the distribution is symmetric about $x_j = 0.5$, becoming more peaked as the common value of a_j and b_j increases. In other words, this is simply the reflective symmetry property of beta density: $\mathcal{B}(x_j;\, a_j, b_j) = \mathcal{B}(1 - x_j;\, b_j, a_j)$. These conditions fully describe the observed distribution of the prediction probabilities x_j. The parameters a_j and b_j can be estimated from the data using maximum likelihood methods.

3.6.4 ROC Analysis

The receiver operating characteristic (ROC) curve is a method for visualizing, assessment and comparison of classifiers based on their performance. For binary classification, the theoretical ROC curve of a classifier is a plot of its true positive rate (*sensitivity*) against its false positive rate (1 - *specificity*) for all possible values of the classification threshold. Plotting the ROC curve is a popular way of displaying the discriminatory accuracy of a classifier. The ROC curves compares the classifiers performance across an entire range of decision thresholds. If the ROC curve for one classifier h_1 is always above and to the left of another classifier h_2, then h_1 is said to dominate h_2. Most often this would mean that classifier h_1 has a lower probability of error than h_2. However, if h_1 has a smaller probability of error than h_2, this does not necessarily mean its ROC curve will always be above that of h_2. The two ROC curves may for example cross each other. Under such conditions, its difficult to establish dominance relationship. In some cases where no dominance relation exists between the ROC curves, the performances can be compared by considering the area under the curves.

The area under the ROC curve or simply AUC is a global measure similar to the overall accuracy for comparing the performance of learning algorithms. The AUC is the integrated

sensitivity as the specificity ranges over [0,1]. That is

$$\text{AUC} = \int_0^1 ROC(t)\, dt$$

where $ROC(t)$ is the *sensitivity* at *1 - specificity* $= t$. The larger the AUC, the better the overall performance of the classifier to correctly discriminate between the classes. It is clear that when there is a dominance relationship between the ROC curves the AUC will reflect this dominance, but as with the probability of error, the reverse is not true. AUC has a nice statistical property that, it represents the probability that a randomly chosen test sample $x \in X_0$ will have a small probability $p(a = 1|\theta = 0)$ of falling in X_1. Another nice property of AUC is that it is equivalent to the Wilcoxon test statistics (Hanley, 1982).

In Bradley (1997), several machine learning classification algorithms were compared using the AUC and the results showed that the AUC offers several desirable properties compared to the overall accuracy of the models. Theoretical arguments that the AUC is a better measure than overall accuracy can be found in Huang and Ling (2005).

Despite the appealing properties of the AUC, there are circumstances where two classifiers may perform quite differently, have different overall accuracy or crossing ROC curves but may have the same AUC. Under such conditions, it becomes difficult to compare the performance of the classifiers using ROC curves or AUCs. In such cases, and in situations where high *specificity* is a perquisite, the AUC can be computed for a confined range of *specificity values*. This area is called the partial area under the ROC curve (pAUC). Thus if $r \leq t \leq s$ is the required range of *1 - specificity*, then the pAUC (normalized) is given by

$$\text{pAUC} = \frac{1}{s - r} \int_r^s ROC(t)\, dt$$

It has been shown that the pAUC is preferred to the AUC for comparing learning algorithms especially in application problems where high *specificity* is demanded (Wang

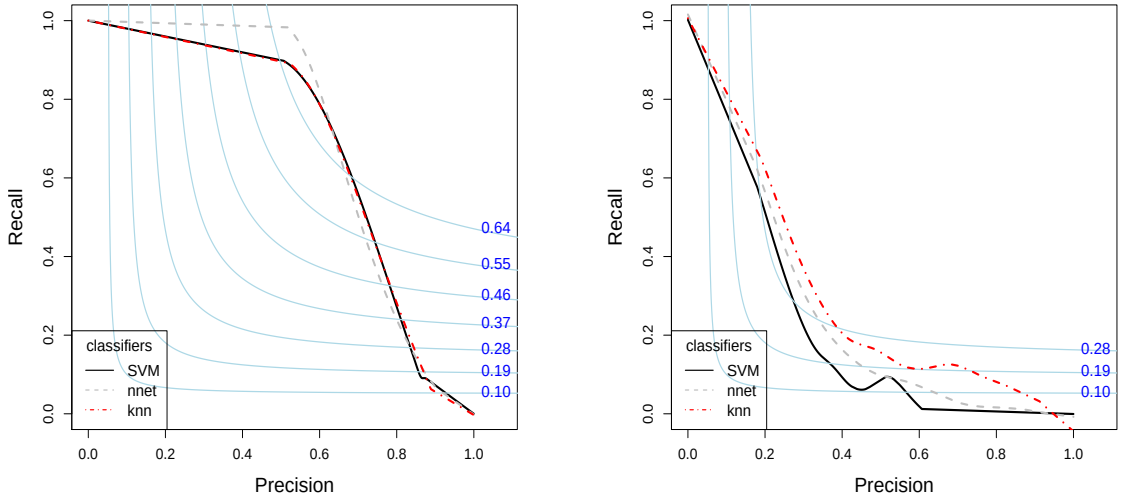
and Chang, 2011).

ROC curves can be easily computed from the Bayes decision framework and likelihood models presented in section 3.6.1. A plot of the probability of correct decisions or *sensitivities* $p(a = 1|\theta = 1)$ versus the false positives $p(a = 1|\theta = 0)$ for various decision thresholds τ gives the ROC curve. Various decision thresholds τ can be obtained by varying either the prior probability π over the interval (0,1) or the error cost ratio $\lambda = \lambda_{01}/\lambda_{10}$. The plotted probabilities do not depend on the prior or error cost. This means that, if all operating points or conditions are used, the ROC curve will remain invariant with respect to the operating condition (priors and/or error cost). This is an appealing property of the ROC curve and consequently the AUC: the performance of a set of classifiers can be graphed and compared without regard to class distribution or error cost. These conditions may change but the graph will remain the same.

3.6.5 Precision-Recall Analysis

The ROC analysis presented in the previous section assumed that the prior probabilities and/or cost are known, consequently, the analysis was found to be insensitive to changes in these operating points. However, the situation is different when the prior/cost are not known or are not well defined. There is therefore need to inspect the performance of the algorithms for different operating points using measures depending on the prior/cost. The insensitivity of ROC analysis to changes in prior/cost is advantageous in some situation but also disadvantageous in others. On datasets with highly skewed class distributions, a low false positive rate may still produce a large number of false positives. Thus a large change in the number of false positives can lead to a small change in the false positive rate used in ROC analysis (Davis and Goadrich, 2006).

The Precision-Recall (PR) analysis has recently received much attention in the machine learning literature as an alternative to ROC analysis for classification tasks on high skewed datasets or when knowledge of the skewness is not available (Davis and Goadrich, 2006;



(a) $\pi = 0.5$ (b) $\pi = 0.1$

Figure 3.2: PR curves for three algorithms: svm, nnet and kkn

Landgrebe et al., 2006). The PR analysis illustrates the tradeoff between the *sensitivity* ($p(a = 1|\theta = 1)$) or *recall* and the probability that the class is positive given that it was classified as positive ($p(\theta = 1|a = 1)$) or the *precision*. PR curves are obtained by plotting *recall* against *precision* for varying decision thresholds. Using properties of conditional probabilities and Bayes theorem, the *precision* can be written as

$$\begin{aligned} p(\theta = 1|a = 1) &= \frac{p(\theta = 1 \text{ and } a = 1)}{p(a = 1)} \\ &= \frac{p(a = 1|\theta = 1)p(\theta = 1)}{p(a = 1|\theta = 1)p(\theta = 1) + p(a = 1|\theta = 0)p(\theta = 0)} \\ &= \frac{p(a = 1|\theta = 1)\pi}{p(a = 1|\theta = 1)\pi + p(a = 1|\theta = 0)(1 - \pi)} \end{aligned} \tag{3.16}$$

Precision is thus seen to depend on the prior π. When the class distribution is highly imbalanced, PR analysis may be more appropriate than ROC analysis since it remains

sensitive to the performance of each class.

Taking the harmonic average of the *precision* and *recall* leads to another popular performance measure known as the F1-score:

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

F1 has high values only when both *precision* and *recall* have high values. The maximum F1 over a varying prior is also commonly used.

Figure 3.2 (a) and (b) demonstrates the dependence of the PR curve to two different prior settings ($\pi = 0.5$ and $\pi = 0.1$) for three classifiers trained on the Pima Indian diabetes dataset from UCI. The performances of the algorithms are comparable under a class balanced assumption ($\pi = 0.5$), however the situation is very different for an imbalance assumption ($\pi = 0.1$) where there is a clear first, second and third. Contours of F1 are also shown. Classifiers with PR curves that fall in higher F1 regimes are typically considered of higher quality.

Similar to the computation of AUC, the area under the PR curve (AUPR) is the integrated *recall* for all possible values of the *precision* at a given prior π:

$$AUPR_\pi = \int PR(t_\pi)\, dt_\pi$$

where $\mathrm{PR}(t_\pi)$ is the *recall* value at $precision = t_\pi$ for a given prior π. Under conditions where the prior is not known or may vary, it may be more informative to estimate the performance over various values of the priors. The easiest way to estimate the overall performance is to take the average of the $AUPR_\pi$ values. Another similar approach presented in Landgrebe et al. (2006) is to integrate $AUPR_\pi$ (IAUPR) over possible values of π.

3.6.6 Divergence of Beta Likelihood Models

The goal of this section is to use an information-theoretic method to measure the similarity between the likelihood of the prediction probabilities. The basic idea is that the distance between the likelihood functions $f(x_j|1)$ and $f(x_j|0),\ j,= 1,\dots L$ offers a measure of the discriminating power of the classifier.

Given the prediction probabilities x_i and x_j from two classifiers h_i and h_j, since the probabilities are subinterval of the unit interval, h_i is said to be of higher quality than h_j if x_i is "finer" than x_j, that is x_i is a refinement of x_j, meaning that x_i is closer to 0 and 1 than x_j. This refinement property can be seen in the light of the likelihood functions as being more concentrated at $x_i = 0$ and $x_i = 1$. Thus, comparing the distances between $f(x|1)$ and $f(x|0)$ for two classifiers, a measure of quality of the classifiers can be established.

The distance or divergence between probability distributions is central to the estimation of the dissimilarity between distributions in statistics, information theory, and many other fields. Many machine learning classification and clustering algorithms for example employ a variety of dissimilarity measures. The most popular and often used measures are the squared Euclidean distance and Kullback-Leibler divergence. The Rényi-α and other α-divergence measures are generalized alternative divergence measures that may provide more robust solutions with respect to outliers and improved accuracy (Cichocki and Amari, 2010). It is particularly very straight forward to compute the Rényi-α divergence for the beta distribution.

Definition 3.6.1. *Let f and g be continuous probability densities over $\mathbb{R}$ and $\alpha \in \mathbb{R}^+\backslash\{1\}$. The Rényi-$\alpha$ divergence of order α is defined by*

$$R_\alpha(f\|g) = \frac{1}{\alpha - 1} \log \int f^\alpha(x) g^{1-\alpha}\, dx$$

The Rényi-α divergence is asymmetric i.e $R_\alpha(f\|g) \neq R_\alpha(g\|f)$, however it can be symmetrized by taking the average of the two way measures: $R_{sym} = (R_\alpha(f\|g) + R_\alpha(g\|f))/2$.

Letting $D_\alpha(f\|g) = \int f^\alpha(x)g^{1-\alpha}\,dx$, and given two beta distributions $f(x) = \mathcal{B}(x;a,b)$ and $g(x) = \mathcal{B}(x;c,d)$ it can be shown that:

$$D_\alpha(f\|g) = \left(\frac{\mathbf{B}(c,d)}{\mathbf{B}(a,b)}\right)^{1-\alpha} \frac{\mathbf{B}(c_1,d_1)}{(\mathbf{B}(a,b))^2}$$

$$D_\alpha(g\|f) = \left(\frac{\mathbf{B}(a,b)}{\mathbf{B}(c,d)}\right)^{1-\alpha} \frac{\mathbf{B}(a_1,b_1)}{(\mathbf{B}(c,d))^2}$$

where $a_1 = a+(c-a)\alpha$, $b_1 = b+(d-b)\alpha$, $c_1 = c+(a-c)\alpha$ and $d_1 = d+(b-d)\alpha$. Classifiers with higher values of R_α are considered to be of higher quality.

3.7 Numerical Experiments

This section presents performance comparison of 8 classification algorithms on the Pima Indians diabetes and the Forest Covertype datasets from UCI. In the experiments, the active learning procedure presented in this chapter is mainly used as a tool to train the algorithms to ensure that all are trained on their "desired" training set. Training the algorithms on their desired training set is considered in this thesis as presenting a suitable condition for fair comparison of the classifiers. The active data selection procedure will find a very useful application in chapter 5.

Ten performance measures are used for comparing the classifiers: The Bayes Risk (B.Risk), ROC curves, area under ROC curves (AUC and pAUC), PR curves, area under PR curves ($\text{AUPR}_{0.5}$ and IAUPR), maximum F1-scores at $\pi = 0.5$ ($\text{Max.F1}_{0.5}$), mean maximum F1-scores for a range of values of π (Mean.Max.F1), and the (symmetric) Rényi-α divergence measure (R_α). These measures are all computed from the beta likelihood model for the prediction probabilities. All experiments are performed with the following settings: $\alpha = 0.5$ for Rényi divergence, $\pi = 0.5, \lambda_{01}/\lambda_{10} = 1$ for Bayes Risk, pAUC is reported for false positive lower limit set at $r = 0$ and upper limits $s = 0.01, 0.02, 0.05, 0.1, 0.2$. PR analysis is carried out for $\pi = 0.1, 0.2, \ldots, 0.9$.

3.7.1 Base Learning Algorithms

The 8 classification algorithms are respectively: Support Vector Machine (svm), Recursive Partitioning and Regression Trees (tree), logistic regression (log), Näive Bayes (nb), Linear discriminant analysis (lda), Quinlan's C5.0 rule-based classifier (c50), Neural network (nnet) and k-nearest neighbor (knn). All algorithms are trained with defualt parameter settings.

The Hausman specification test is applied to decide which of LDA or LR active learning schemes to adopt for training. This procedure is repeated 5 times on independent subsets of the datasets. Thus each classifier submits five distinct prediction probabilities to the DM. The performance measures are averaged over these five experiments.

3.7.2 Results

Tables 3.1 and 3.2 shows the performance of the classifiers on the two datasets considered. Except for the B.Risk, higher values indicates better performance. The five columns before the last of each table are values of pAUC for different false positive rates s. The interest here is to lookout for classifiers with high pAUC values for very small s. The higher the value, the more likely the classifier will detect the positive class. The last column represents overall ranking of the classifiers with the top 3 boldfaced. However, this does not indicate they are superior across all performance measures. The ranking is done in the following manner: First all the classifiers are ranked with respect to each performance metric. The scores are then added and ranked.

Figures 3.3 and 3.4 shows the ROC and PR curves for the two datasets. The ordering of the classifiers displayed in the legend is by decreasing AUC and IAUPR respectively. The following points can be made from observing the results in the tables and figures.

1. No Free Lunch Theorem: None of the algorithms are always the best, even for this two dataset experiment.

2. Learning algorithms with the smallest Bayes risk (probability of error) do not necessarily have their ROC or PR curves above those with higher risk. For example in Table 3.1 Neural network has the smallest risk, but Figure 3.3 shows that its curve is not the dominating curve. Interestingly, in Table 3.1 Quinlan's C5.0 rule-based classifier has the highest error rate but its ROC and PR curves are above Näive Bayes (nb) and decision trees (tree) classifiers with smaller error rates. This shows the usefulness of using several performance measures to evaluate classifiers.

3. Decision trees classifier performed well on the forest dataset but worst on the pima dataset. The reverse is true for k-nearest neighbor. Näive Bayes performance was consistently very poor.

4. Logistic regression, SVM, Neural network and Linear discriminant analysis have excellent performance on at least one dataset.

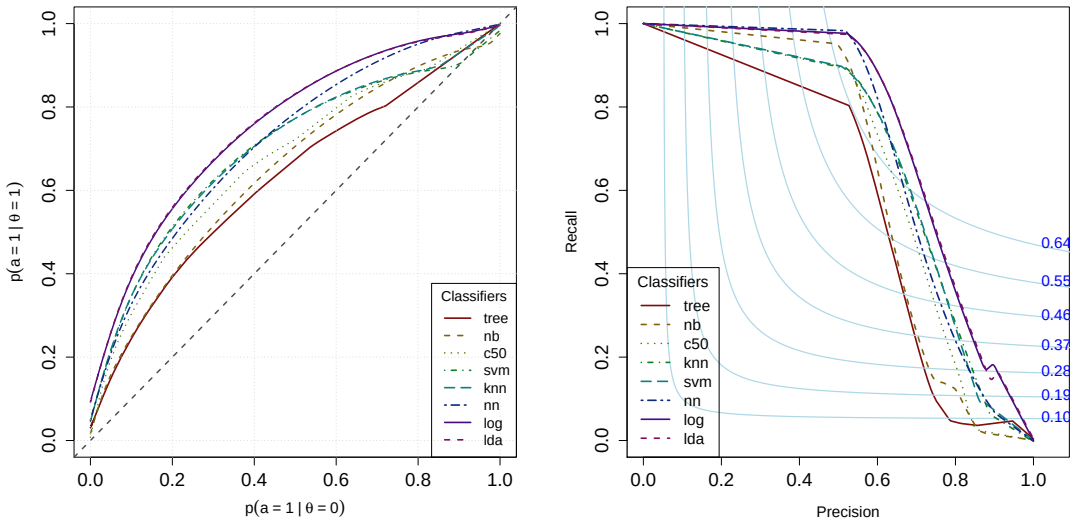
5. The values of the performance measures presented in tables are very close to some published results in the machine learning literature (Caruana and Niculescu-Mizil, 2006; Huang and Ling, 2005). This indicates that, modeling the prediction probabilities by the beta distribution is a viable approach to computing the performance measures.

Table 3.1: Performance Measures on Pima Indian Data

	Performance Metric							pAUC at s					
Classifier	B.Risk	AUC	$AUCPR_{.5}$	IAUCPR	$Max.F1_{.5}$	Mean.Max.F1	R_α	0.01	0.02	0.05	0.1	0.2	Rank
svm	0.21	0.70	0.69	0.65	0.68	0.65	0.31	0.04	0.07	0.13	0.21	0.32	5
tree	0.23	0.62	0.59	0.57	0.64	0.60	0.18	0.03	0.05	0.09	0.14	0.23	1
log	**0.13**	**0.75**	**0.75**	**0.71**	**0.72**	**0.69**	**0.28**	**0.05**	**0.10**	**0.17**	**0.25**	**0.37**	**7**
nb	0.13	0.64	0.64	0.61	0.67	0.63	0.13	0.03	0.05	0.10	0.15	0.24	2
lda	**0.13**	**0.75**	**0.75**	**0.71**	**0.72**	**0.69**	**0.29**	**0.05**	**0.10**	**0.17**	**0.26**	**0.37**	**8**
c50	0.30	0.67	0.66	0.62	0.67	0.64	0.25	0.03	0.05	0.11	0.18	0.28	3
nn	**0.11**	**0.71**	**0.71**	**0.67**	**0.70**	**0.67**	**0.19**	**0.05**	**0.08**	**0.13**	**0.20**	**0.31**	**6**
knn	0.20	0.70	0.69	0.65	0.68	0.65	0.32	0.04	0.07	0.14	0.21	0.32	5

Table 3.2: Performance Measures on Forest Data

Performance Metric								pAUC at s					
Classifier	B.Risk	AUC	$AUCPR_{.5}$	IAUCPR	$Max.F1_{.5}$	Mean.Max.F1	R_{α}	0.01	0.02	0.05	0.1	0.2	Rank
svm	**0.18**	**0.82**	**0.83**	**0.80**	**0.77**	**0.75**	**0.90**	**0.19**	**0.26**	**0.36**	**0.46**	**0.57**	**7**
tree	0.49	0.74	0.75	0.72	0.70	0.68	1.31	0.16	0.23	0.34	0.43	0.51	5
log	**0.07**	**0.95**	**0.94**	**0.92**	**0.87**	**0.85**	**1.35**	**0.42**	**0.49**	**0.60**	**0.69**	**0.78**	**8**
nb	0.08	0.70	0.70	0.66	0.69	0.66	0.15	0.05	0.07	0.13	0.19	0.28	2
lda	0.25	0.72	0.73	0.70	0.70	0.67	0.67	0.09	0.14	0.23	0.31	0.42	3
c50	0.24	0.76	0.76	0.73	0.72	0.70	0.79	0.13	0.18	0.27	0.36	0.47	4
nn	**0.23**	**0.79**	**0.80**	**0.77**	**0.75**	**0.73**	**1.01**	**0.19**	**0.25**	**0.35**	**0.44**	**0.55**	**6**
knn	0.39	0.63	0.59	0.57	0.61	0.58	0.69	0.04	0.07	0.13	0.20	0.30	1



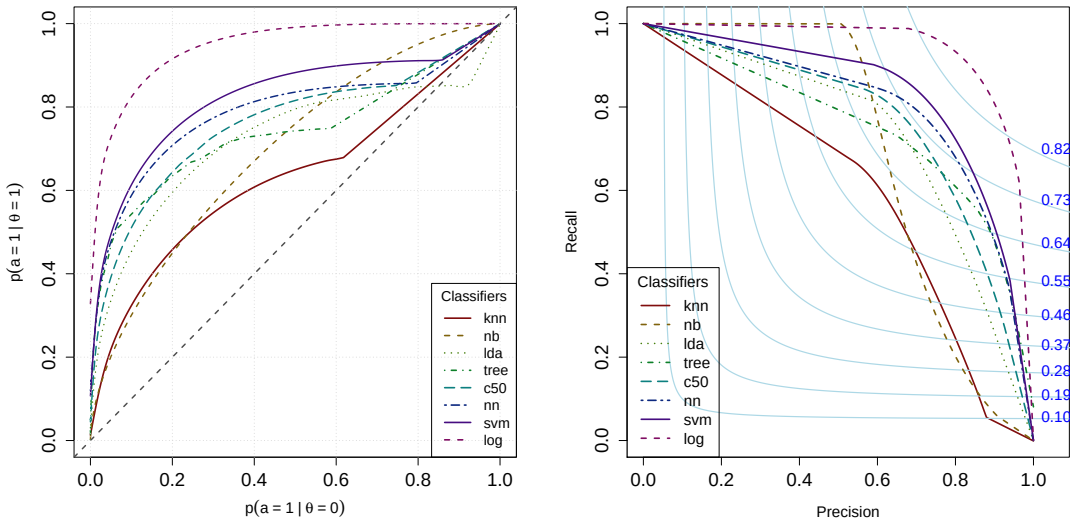
(a) ROC curves

(b) PR curves with contours of F1-score

Figure 3.3: ROC and PR curves for Pima Indian Data

3.8 Related Work

The STATLOG project (King et al., 1995) is one of the earliest evaluation study of learning algorithms. Since then some of the algorithms used in the project have been improved and new ones have been developed. A wide range of learning algorithms now exist today and its



(a) ROC curves (b) PR curves with contours of F1-score

Figure 3.4: ROC and PR curves for Forest Cover Data

not surprising various empirical comparison methods have appeared in the literature. Caruana and Niculescu-Mizil (2006) performed a similar large-scale empirical comparison of ten learning algorithms using eight performance measures such as the classification accuracy, F1-score, AUC average precision etc. Bradley (1997) uses ANOVA to test statistical difference in accuracy and AUC of different machine learning algorithms. The AUC is similarly used in Huang and Ling (2005) to evaluate learning algorithms. The recall, false positive rate and precision are used in Menzies et al. (2004) to compare detectors of software defects. Brazdil et al. (2008) uses meta-learning technique to compare and rank learning algorithms. Alpaydm (1999) presents a 5× 2 cv F test for comparing learning algorithms.

All these studies have a few things in common in their approach to comparison/ranking:

- If the learning algorithms are trained and tested on different subsets of the input space it may become difficult to apply these methods to compare their performance. Consequently, their application in distributed learning environment where data cannot

be easily moved to a central location is limited.

- Scalability of the methods is not assured.
- All algorithms are trained on the same training examples. Some training examples may not be suitable for learning by some algorithms. By making the algorithms take part in selecting training examples, they may better generalize on unseen test examples.

3.9 Summary

This chapter presented several principled methods to deal with the problem of data and classifier selection. Two simple, fast and effective active learning techniques are used to train a collection of learning algorithms using a small set of carefully selected training examples. The selection of examples is done through a joint action of the active learner and classifier, thus the acquired training set can be considered as the "desired" or most informative training set for the classifier. Through a Bayes decision framework, the prediction probabilities from each classifiers are modeled by the beta distribution from which the performances of the classifiers are evaluated and ranked.

Chapter 4: Optimal Integration of Classification Models

This chapter introduces three methods for integrating the outputs of a collection of classifiers optionally selected using the Bayes active data and algorithm selection method described in chapter 3. The uniqueness of the presented methods lies in their ability to incorporate additional information infered from the data to improve accuracy of the ensemble classifier. A generative Bayesian model and inference is proposed for learning the information and combining the classifiers. Specifically, a variational Bayesian factor common spatial pattern (VBFACSP) is proposed for inferring hidden informative feature vectors from the training set and three new combinations schemes are proposed to integrate the outputs of the classifiers. The first ensemble method uses the classical expectation-maximization (EM) optimization algorithm to estimate classifier combination weights while the last two approaches uses variational Bayesian inference with automatic relevance determination (ARD) hirarchical priors. A nice feature of the variational ensemble scheme is that the preprocessing step of algorithm selection presented in chapter 3 is optional since unreliable classifiers can be pruned automatically during the combination step.

4.1 Introduction

To the best of the knowledge of the author, the idea of incorporating (class specific) hidden information about each data point in an ensemble combination scheme has not been explored before. The obvious question one might ask is why bother searching for such hidden information. To answer this question, first one may observe that the learning process of most classification algorithms may be seen as learning by ignoring hidden variables, or by eliminating them through marginalization or "averaging out". Therefore, some vital information useful for classification may not be taken advantage of. A second observation in the

classification process can be summarized as follows:

1. Some classifiers are good in classifying certain data points and worse at others i.e they perform best on some subsets of the dataspace and worse on others. Relevant information from other sub areas may help in deficient areas.

2. Different classifiers may attach different costs to different misclassification errors. Different data points may carry different misclassification costs.

3. Some data points are inherently easy or difficult to classifier. A difficult or ambiguous data point may be classified incorrectly even by the most accurate classifier. The term "classification difficulty" used here refers to some hidden intrinsic property of the data point reflecting its ease to be classified by any given algorithm. How easy it is to classify a given data point depends on this intrinsic property.

4. Given sufficient information, it is possible to determine the true class of some misclassified samples by a classifier.

Based on these observations, it is assumed that there genuinely exists a unitvariate or multivariate hidden variable e that encodes the true nature of a data point with respect to its class membership. That is, the hidden variable represents some evidence that maps each data point x to its corresponding true class t. This hidden variable will be referred to as *classification evidence* or simple *evidence*. A generative Bayesian graphical model and inference is formulated to infer the classification evidence and parameters of the proposed ensemble models. While most of the presented results are valid for general multiclass problems, the focus is on binary classification tasks.

4.2 Bayesian Graphical Evidence and Ensemble Model

Graphical models provide a framework for representing dependencies among variables of a statistical modelling problem. The nodes in the graph correspond to variables while the

edges represent the dependencies among them. Thus a directed edge from a node **A** to a node **B** denotes that variable **B** depends on the value of variable **A**. Node **A** is said to be the parent of node **B**. Circles are used to represent random variables while square boxes denotes deterministic variables. Rounded boxes or plates enclosing one or more variables denotes repetitions of the variables. Each node represents a conditional probability density that defines the distribution of the variable given the values of its parents. The density at the node may be parametrized by a set of parameters θ. All models in this section are visualized as *directed acyclic graphs* also known as probabilistic directed acyclic graphical models or simply Bayesian probabilistic networks.

Bayesian probabilistic networks have gained a lot of popularity over the past few years. This is not surprising as these networks have been shown to be effective tools for decision making and reasoning under uncertainty. For example they perform well in complex decision-making domains such as medical diagnostics, image analysis, robot control etc. Further, the graphical nature of these networks makes the learning process typically comprehensible to humans.

The work presented in this section focuses on learning probabilistic networks where some of the variables are hidden. This situation is not unusual, for the real world is rarely fully observed. In medical diagnosis for example, the actual disease or cause of the disease may not be known even at the end of some treatment. More generally, in classification the observed input attributes may provide only partial evidence for classification. In the real world it is often not possible to observe all attributes and not all of them may be relevant. In this regard, one can intuitively envision the existence of some unknown mechanism that transforms the attributes into a single variable or a small group of variables that embeds most of the information about true classification. Equivalently, one can think of the existence of a ground truth model that transforms the input into a variable with complete information on classification. While this variable remain hidden, classical learning algorithms with certain learning biases or defects may not have complete access to it. Consider for example users online preferences or ratings of certain products such as movies,

books, DVDs. The hidden variable may represent a user profile, such as the users affinity for a certain item or group of items which is not directly observable. Access to this variable could help improve the performance of some systems such as a recommender system. The goal here is simply to draw inference from this variable through a probabilistic network for classification purposes, the meaning or interpretation of this variable is irrelevant to this work.

In learning probabilistic networks, it is very important to specify the correct network structure for the learning algorithm uses it as a strong source of prior information about the domain. The assumptions made here is that the network structure is known with the following causal relationships among the main variables of the model.

1. The observed data x depends on the classification evidence e. This relationship can be modeled as linear or non-linear. It is useful to think of this relationship as some mechanism described by the graphical model structure in which the evidence is responsible for generating the observed data.

2. The observed output $\hat{t}$ from each classifier depends on several factors including:

   (a) the classification evidence e

   (b) the true class t

   (c) conditionally independent given the true class

   (d) reliability and bias of the classifier

Given a training set $\mathcal{D}_l = \{(x_1, t_1), \ldots, (x_n, t_n)\} \in \mathcal{X} \times \mathcal{Y}$ on which a set of L classifiers $\mathcal{H} = \{h_1, \ldots, h_L\}$ have been trained and used to produce the predictions $\hat{Y} = \{\hat{t}_1, \ldots, \hat{t}_L\}$ for each data point in $\mathcal{D}_l$. The problem is to predict the class of a new data point $x^* \in \mathcal{X}$ based on the predictions of the base classifiers. Figure 4.1 shows the probabilistic graphical model with the assumed causal relationships. In the figure, observed variables are shaded grey while hidden variables are unshaded. Square boxes with numbers corresponds to repetitions of the respective random variables appearing in the box.

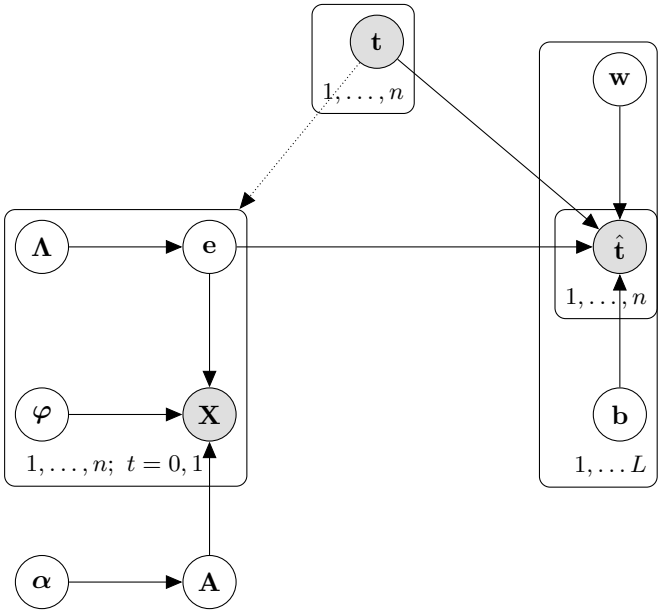


Figure 4.1: Graphical Model for Classification Evidence and Multiple Classifier Integration.

From the graphical model, the joint distribution of the variables can be written as

$$p(x, t, \hat{t}, e, \boldsymbol{\theta}) = \prod_{t=0}^{1} p(x^t|e^t, \boldsymbol{\varphi}^t, \mathbf{A}) p(e^t|\boldsymbol{\Lambda}^t) p(\hat{t}|e^t, \mathbf{w}, \mathbf{b}) p(\mathbf{A}|\boldsymbol{\alpha}) p(\boldsymbol{\varphi}^t) p(\boldsymbol{\Lambda}^t) p(\mathbf{w}) p(\mathbf{b}) \quad (4.1)$$

where $\boldsymbol{\theta} = (\boldsymbol{\varphi}, \boldsymbol{\Lambda}, \mathbf{A}, \boldsymbol{\alpha}, \mathbf{w}, \mathbf{b})$ and $p(a^t) \equiv p(a|t)$. The joint distribution can be split into two model classes: An evidence model and an ensemble model with specific parametric models specified as follows

1. **Evidence Model**

$$p(x, t, \hat{t}, e, \boldsymbol{\varphi}, \boldsymbol{\Lambda}, \mathbf{A}, \boldsymbol{\alpha}) = \prod_{t=0}^{1} p(x^t|e^t, \boldsymbol{\varphi}^t, \mathbf{A}) p(e^t|\boldsymbol{\Lambda}^t) p(\mathbf{A}|\boldsymbol{\alpha}) p(\boldsymbol{\varphi}^t) p(\boldsymbol{\Lambda}^t)$$

where x is assumed to be generated by

$$x^t = \mathbf{A}e^t + \boldsymbol{\varepsilon}^t, \quad t = 0, 1 \tag{4.2}$$

with $\mathbf{A}$ a $p \times K$ basis matrix. The K dimensional evidence e and p dimensional noise $\boldsymbol{\varepsilon}$ vectors are assumed to follow a zero-mean Gaussian distributions,

$$e^t \sim \mathcal{N}_K(\mathbf{O}, \left[\boldsymbol{\Lambda}^{-1}\right]^t), \tag{4.3}$$

$$\boldsymbol{\varepsilon}^t \sim \mathcal{N}_p(\mathbf{O}, \left[\boldsymbol{\varphi}^{-1}\right]^t) \tag{4.4}$$

where $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_K)$ and $\boldsymbol{\varphi} = \text{diag}(\phi_1, \ldots, \phi_p)$ are diagonal matrices. Note that a non-linear relationship between x and e can equally be specified. A linear relationship is assumed here for computational simplicity and readily available principled statistical inference methods.

A model structure (equations 4.2) having the evidence e depending on the class t as illustrated in figure 4.1 is equivalent to the probabilistic common spatial patterns (PCSP) model (Wu et al., 2009). Common spatial pattern (CSP) is a widely used feature extraction method applied in brain-computer interface systems for discriminating between positive and negative classes in electroencephalogram (EEG) data. It discriminates between the two classes by maximizing the variance of one class and minimizing the variance of the other.

If the dependency condition between the evidence and class is relaxed, then the model structure reduces to standard factor analysis with the exception that the distribution of the latent factor is zero mean Gaussian with non-identity covariance matrix $\boldsymbol{\Lambda}$. This modified factor model is useful in that, the parametrization represented by $\boldsymbol{\Lambda}$ serves as an ARD prior, which helps to prune irrelevant factors.

A variational Bayesian algorithm is presented for the estimation of the maximum

likelihood of the parameters of the two model structures. However, to keep the computations simple, subsequent analysis is restricted to the factor model. The class dependent model is reserved for feature work.

2. **Ensemble Model**

$$p(\hat{t}, t, e, \mathbf{w}, \mathbf{b}) = \prod_{t=0}^{1} p(\hat{t}|e^t, \mathbf{w}, \mathbf{b}) p(\mathbf{w}) p(\mathbf{b})$$

Because the output of the base classifiers can be binary $\hat{t} = 0, 1$ or continuous i.e class scores or class posterior probabilities $\hat{t} = p(t = 1|x)$ two modeling strategies are considered.

(a) Binary predictions

The class predicted by each classifier given the true class t and the classification evidence e is modeled by the logistic function

$$p(\hat{t} = t|e, \mathbf{w}, \mathbf{b}) = \frac{1}{1 + e^{-\mathbf{we}-\mathbf{b}}} \tag{4.5}$$

$\mathbf{w}$ is a $L \times K$ matrix whose rows weights the classifiers and columns the evidence while $\mathbf{b}$ is the vector of biases for each classifier.

(b) Class posterior probabilities

For class probabilities, a new vector is created by a concatenation of the classification evidence and the predicted probabilities, i.e $\Phi = (\hat{t}, e) \in \mathbf{R}^{L+K}$. The true label t is equally modeled by the logistic function

$$p(t = 1|\Phi, \mathbf{w}, \mathbf{b}) = \frac{1}{1 + e^{-\mathbf{w}^T\Phi-\mathbf{b}}} \tag{4.6}$$

In this case, $\mathbf{w}$ is a vector of weights for the classifiers and evidence.

Inference for the evidence and ensemble models can be performed jointly or decoupled into two successive steps: evidence then ensemble. Joint inference increases the computational complexity of the whole process and it might not be clear initially if any gain in performance will be archived. However, the full representational power of the graphical model is captured by a joint analysis. Any complex correlation patterns between the two models is explicitly modeled and thus one might suspect it could lead to more reliable predictions. The decoupled model inference on the other hand, is computationally simple and easy to implement. Further, it allows the same evidence model to be used for different ensemble learning schemes.

For computational reasons, the easier decoupled model structure is implemented in this thesis while joint model structure is reserved for future investigation. The next section presents a variational Bayesian method for estimating the classification evidence of the decoupled system.

4.3 Variational Bayesian Inference for Evidence Model

Factor analysis (FA) is a powerful multivariate analysis technique that is used to uncover or identify common hidden characteristics in a set of variables and to reduce the set to a smaller number of derived variables called factors. That is, it explains a multivariate data set $x \in \mathbb{R}^p$ in terms of $K < p$ factors that captures the joint correlation or dependencies between the p observed variables. The patterns uncovered by FA can help classify the original data. The application of FA range in various fields, from psychology, physics, economics, bioinformatics, machine learning, natural language processing and social sciences.

The maximum likelihood Expectation Maximization (EM) and Variational Bayesian (VB) algorithms are the two most popular methods for learning FA (Dempster et al., 1977; Ghahramani and Beal, 1999). EM methods work well in many substantive problems, but can perform poorly when applied to large data sets. More importantly, EM methods do not take into account model complexity. Thus very complex models are not penalized, which

may lead to overfitting. More over, EM methods rely on very costly procedures such as cross-validation to determine the best model size and structure (Ghahramani and Beal, 1999). Fully Bayesian approaches on the other hand overcome these problems by treating the parameters of the model as unknown random variables and giving them priors to express the model assumptions. By averaging out the unknown variables, very complex models are penalized. Thus Bayesian methods provide a natural data-driven method for ensuring that the model does not overfit the data.

The evidence model given by equation 4.2 is equivalent to the standard factor analysis model except that the latent factors have non-identity covariance matrix and are class dependent. The factor loadings on the other hand is shared between the classes. It can be seen as a mixture of factor models, where each component of the mixture is a class distribution. This model is commonly known as probabilistic common spatial patterns (PCSP). Therefore a standard VB treatment of FA can be applied with just a few modifications (Bishop, 1999; Lappalainen and Honkela, 2000).

4.3.1 Class Independent Evidence Model

This section presents a variational Bayesian algorithm for the evidence model assuming that the classification evidence e is independent of the true class t. The presented algorithm can be easily extended to the class dependent case. The graphical model for class independent evidence model is shown in Figure 4.2. Small square boxes represents the fixed parameters of the model.

From the assumptions stated for the evidence model (equation 4.2), the likelihood of the data is given by

$$p(\mathbf{X}|\mathbf{e}, \mathbf{A}, \boldsymbol{\varphi}) = \prod_{i=1}^{n} \mathcal{N}_p(x_i|\mathbf{A}e_i, \varphi^{-1}). \tag{4.7}$$

Here $\mathbf{X}$ denotes the $n \times p$ design matrix and $\mathbf{e}$ is a $n \times K$ matrix of factors. The prior

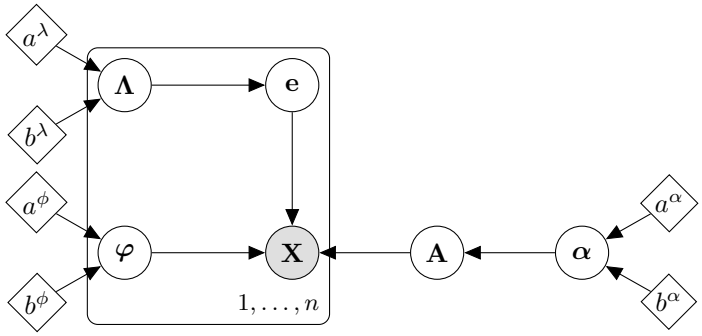


Figure 4.2: Class Independent Evidence Model.

models for the unknown variables are specified as

$$p(\mathbf{A}|\boldsymbol{\alpha}) = \prod_{k=1}^{K} \mathcal{N}_p(a_k|\mathbf{O}, \alpha_k^{-1}\mathbf{I}), \tag{4.8}$$

$$p(\boldsymbol{\alpha}) = \prod_{k=1}^{K} \mathcal{G}(\alpha_k|a_k^\alpha, b_k^\alpha), \tag{4.9}$$

$$p(\varphi) = \prod_{j=1}^{p} \mathcal{G}(\phi_j|a_j^\phi, b_j^\phi), \tag{4.10}$$

$$p(\Lambda) = \prod_{k=1}^{K} \mathcal{G}(\lambda_k|a_k^\lambda, b_k^\lambda), \tag{4.11}$$

where $\mathcal{G}(x|a,b)$ denotes a gamma distribution having mean $\langle x \rangle = a/b$ and $\langle \log x \rangle = \psi(x) - \log b$ with $\psi(x)$ the digamma function. a_k is a vector corresponding to the k'th column of the matrix $\mathbf{A}$. Since the columns of $\mathbf{A}$ are given zero mean Gaussian distribution with the parameter α_k controlling the precision, if the posterior distribution over $\boldsymbol{\alpha}$ concentrates on large values, then the variance of the k'th column goes to zero thus shutting off the column. This can be seen as an ARD prior which automatically shut off irrelevant factors and the number of factors K can be determined. The same reasoning applies to the Λ prior.

VB methods approximate the true posterior distribution $p(\boldsymbol{\theta})$ of the unknown variables $\boldsymbol{\theta} = (\mathbf{e}, \mathbf{A}, \boldsymbol{\varphi}, \boldsymbol{\alpha}, \boldsymbol{\Lambda})$ using a much simpler distribution $q(\boldsymbol{\theta})$ which is assume to factorize with respect to the unknown variables. Thus $q(\boldsymbol{\theta})$ factorizes as

$$q(\boldsymbol{\theta}) = q(\mathbf{e})q(\mathbf{A})q(\boldsymbol{\varphi})q(\boldsymbol{\alpha})q(\boldsymbol{\Lambda}).$$

This assumption is made purely to simplify tractability of the computations (Bishop and Nasrabadi, 2006). In the factorized form, all the factorized posteriors $q(\boldsymbol{\theta}_i)$ are usually of the same form as the priors $p(\boldsymbol{\theta}_i)$. This is one of the main objectives of variational Bayesian method as it greatly facilitates the derivation of the q distributions through conjugacy. The posterior $q(\boldsymbol{\theta})$ will generally differ from the true posterior $p(\boldsymbol{\theta})$ and is called *variational posterior*. It will be optimized by alternatively updating each component of the variational posterior to produce a best approximation to the true posterior. This is done by maximizing the lower bound of the marginal log-likelihood of the model obtained via Jensen's inequality:

$$\begin{aligned}
\log p(\mathbf{X}) &= \log \int p(\mathbf{X}, \boldsymbol{\theta})\, d\theta \\
&\geq \int q(\boldsymbol{\theta}) \log \left(\frac{p(\mathbf{X}, \boldsymbol{\theta})}{q(\boldsymbol{\theta})} \right) d\theta \\
&= \langle \log p(\mathbf{X}|\boldsymbol{\theta}) \rangle - \left\langle \log \left(\frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta})} \right) \right\rangle = \mathcal{F}(q),
\end{aligned} \tag{4.12}$$

where $\langle \cdot \rangle$ denotes the expectation over the distribution q. The first term of the variational lower bound $\mathcal{F}(q)$ corresponds to the likelihood term of the model while the second term is the Kullback-Leibler (KL) divergence between the true posterior $p(\boldsymbol{\theta})$ and the variational posterior $q(\boldsymbol{\theta})$. It can be seen from equation 4.12 that maximizing $\mathcal{F}(q)$ is equivalent to minimizing the KL divergence. Thus the KL term serves as a penalty term which penalizes complex models.

To obtain the variational posterior $q(\boldsymbol{\theta}_i)$ for any of the variables $\boldsymbol{\theta}_i$, the expectation of the complete log-likelihood $\log p(\mathbf{X}, \boldsymbol{\theta})$ is taken with respect to all other variational posteriors

$q(\boldsymbol{\theta}_j),\ j \neq i$. The complete log-likelihood of the model is given by

$$\begin{aligned}
\log p(\mathbf{X}, \boldsymbol{\theta}) &= \log p(\mathbf{X}|\mathbf{e}, \mathbf{A}, \boldsymbol{\varphi}) + \log p(\mathbf{e}|\boldsymbol{\Lambda}) + \log p(\mathbf{A}|\boldsymbol{\alpha}) + \log p(\boldsymbol{\varphi}) + \log p(\boldsymbol{\alpha}) + \log p(\boldsymbol{\Lambda}) \\
&= \frac{n}{2}\log|\boldsymbol{\varphi}| - \frac{1}{2}\sum_{i=1}^{n}(x_i - \mathbf{A}e_i)^T\boldsymbol{\varphi}(x_i - \mathbf{A}e_i) + \frac{n}{2}\log|\boldsymbol{\Lambda}| - \frac{1}{2}\sum_{i=1}^{n} e_i^T\boldsymbol{\Lambda}e_i \\
&\quad + \frac{p}{2}\sum_{k=1}^{K}\log\alpha_k - \frac{1}{2}\sum_{k=1}^{K}\alpha_k a_k^T a_k + \sum_{k=1}^{K}(a_k^{\alpha} - 1)\log\alpha_k - \sum_{k=1}^{K} b_k^{\alpha}\alpha_k \\
&\quad + \sum_{j=1}^{p}(a_j^{\phi} - 1)\log\phi_j - \sum_{j=1}^{p} b_j^{\phi}\phi_j + \sum_{k=1}^{K}(a_k^{\lambda} - 1)\log\lambda_k - \sum_{k=1}^{K} b_k^{\lambda}\lambda_k + constant,
\end{aligned} \tag{4.13}$$

where $|\mathbf{A}|$ is the determinant of the matrix $\mathbf{A}$. The distribution and update rules for the individual components of the variational posterior is presented in Algorithm 4.

Each update of the variational posteriors in Algorithm 4 is guaranteed to increase the variational lower bound $\mathcal{F}(q)$. Thus, the posteriors are updated iteratively until no significant change in the lower bound is observed. The lower bound can also be used for model selection. Models with higher $\mathcal{F}(q)$ and hence lower KL are preferred. With the variational posteriors determined, $\mathcal{F}(q)$ can be computed as

$$\mathcal{F}(q) = \langle \log p(\mathbf{X}|\boldsymbol{\theta}) \rangle - KL(q(\boldsymbol{\theta})\|p(\boldsymbol{\theta})) \tag{4.14}$$

with

$$\langle \log p(\mathbf{X}|\boldsymbol{\theta}) \rangle = -\frac{np}{2} + \frac{1}{2}\sum_{i=1}^{p}\langle \log\phi_j \rangle - \sum_{j=1}^{p}(\hat{a}_j^{\phi} - \hat{b}_j^{\phi}\langle\phi_j\rangle) \tag{4.15}$$

Algorithm 4: Variational Bayesian Inference for Class Independent Evidence Model

Input : Observed data $\mathbf{X}$, stopping criteria
Output: Variational posterior $q(\boldsymbol{\theta})$

Initialize moments of variational posterior $q(\boldsymbol{\theta})$;

repeat

1 Update $q(\mathbf{e})$;

$$q(\mathbf{e}) = \prod_{i=1}^{n} \mathcal{N}_K(e_i | m_e^i, \Sigma_e), \qquad \textbf{Moments} \quad \Sigma_e = \Big(\langle \boldsymbol{\Lambda} \rangle + \big\langle \boldsymbol{A}^T \boldsymbol{\varphi} \boldsymbol{A} \big\rangle \Big)^{-1}$$
$$m_e^i = \Sigma_e \langle \boldsymbol{A} \rangle^T \langle \boldsymbol{\varphi} \rangle x_i$$

2 Update $q(\mathbf{A})$

$$q(\mathbf{A}) = \prod_{j=1}^{p} \mathcal{N}_K(a_j | m_a^j, \Sigma_a^j), \qquad \textbf{Moments} \quad \Sigma_a^j = \Big(\langle \phi_j \rangle \sum_{i=1}^{n} \big\langle e_i e_i^T \big\rangle + \langle \boldsymbol{\alpha} \rangle \Big)^{-1}$$
$$m_a^j = \Sigma_a^j \langle \phi_j \rangle \sum_{i=1}^{n} x_{ij} \langle e_i \rangle$$

3 Update $q(\boldsymbol{\alpha})$

$$q(\boldsymbol{\alpha}) = \prod_{k=1}^{K} \mathcal{G}(\alpha_k | \hat{a}_k^\alpha, \hat{b}_k^\alpha), \qquad \textbf{Moments} \quad \hat{a}_k^\alpha = a_k^\alpha + \frac{p}{2}$$
$$\hat{b}_k^\alpha = b_k^\alpha + \frac{\langle a_k^T a_k \rangle}{2}$$
$$\langle \boldsymbol{\alpha} \rangle = \boldsymbol{diag}\left(\frac{\hat{a}^\alpha}{\hat{b}^\alpha}, k = 1, \ldots, K \right)$$

4 Update $q(\boldsymbol{\varphi})$

$$q(\boldsymbol{\varphi}) = \prod_{j=1}^{p} \mathcal{G}(\phi_j | \hat{a}_j^\phi, \hat{b}_j^\phi), \qquad \textbf{Moments} \quad \hat{a}_j^\phi = a_j^\phi + \frac{n}{2}$$
$$\hat{b}_j^\phi = b_j^\phi + \frac{1}{2} \sum_{i=1}^{n} \big\langle (x_{ij} - a_j^T e_i)^2 \big\rangle$$
$$\langle \boldsymbol{\varphi} \rangle = \boldsymbol{diag}\left(\frac{\hat{a}^\phi}{\hat{b}^\phi}, j = 1, \ldots, p \right)$$

5 Update $q(\boldsymbol{\Lambda})$

$$q(\boldsymbol{\Lambda}) = \prod_{j=1}^{K} \mathcal{G}(\lambda_k | \hat{a}_k^\lambda, \hat{b}_k^\lambda), \qquad \textbf{Moments} \quad \hat{a}_k^\lambda = a_k^\lambda + \frac{n}{2}$$
$$\hat{b}_k^\lambda = b_k^\lambda + \frac{1}{2} \Big(\sum_{i=1}^{n} \big\langle e_i e_i^T \big\rangle \Big)_{(k,k)}$$
$$\langle \boldsymbol{\Lambda} \rangle = \boldsymbol{diag}\left(\frac{\hat{a}^\lambda}{\hat{b}^\lambda}, k = 1, \ldots, K \right)$$

until *stopping criteria*;

and

$$KL\Big(q(\boldsymbol{\theta})\|p(\boldsymbol{\theta})\Big) = \Big\langle KL\Big(q(\mathbf{e})\|p(\mathbf{e}|\boldsymbol{\Lambda})\Big)\Big\rangle_{q(\boldsymbol{\Lambda})} + \Big\langle KL\Big(q(\mathbf{A})\|p(\mathbf{A}|\boldsymbol{\alpha})\Big)\Big\rangle_{q(\boldsymbol{\alpha})} + KL\Big(q(\boldsymbol{\alpha})\|p(\boldsymbol{\alpha})\Big) + KL\Big(q(\boldsymbol{\phi})\|p(\boldsymbol{\phi})\Big) + KL\Big(q(\boldsymbol{\Lambda})\|p(\boldsymbol{\Lambda})\Big). \tag{4.16}$$

The expression for KL is given in appendix B. During optimization, hyperparameters of the model i.e the fixed gamma parameters a and b can be set to small values such 10^{-6}. An alternative approach is to perform hyperparameter optimization, i.e find hyperparameter values that maximizes the expected complete log-likelihood. This can be done by standard optimization algorithms such the Newton-Raphson algorithm.

The EM and VB methods are known to be very slow to converge. The maximization of the lower bound in VB is carried out iteratively, one component update at a time. In addition, observation of Algorithm 4 shows that the variables in the factorized distribution are strongly coupled even though VB approximation assumed independence between the variables. Recent techniques such as the parameter expanded VB methods (Luttinen and Ilin, 2010) have been proposed to speed up the learning process. The general idea of the method is to introduce auxiliary parameters in the original model to reduce the effect of strong coupling between the variables.

4.3.2 Class Dependent Evidence Model

This section simply extends the VB estimation of the class independent evidence model to the class dependent model. Figure 4.3 shows the class dependent model which emphasis that the factor loading $\mathbf{A}$ is shared between the classes. The distribution and update rules for the individual components is presented in Algorithm 5. Observe that the covariance

matrix of the variational distribution of the factor loading $\mathbf{A}$ is the pool covariance matrix of the two class distribution and the mean is simply the sum.

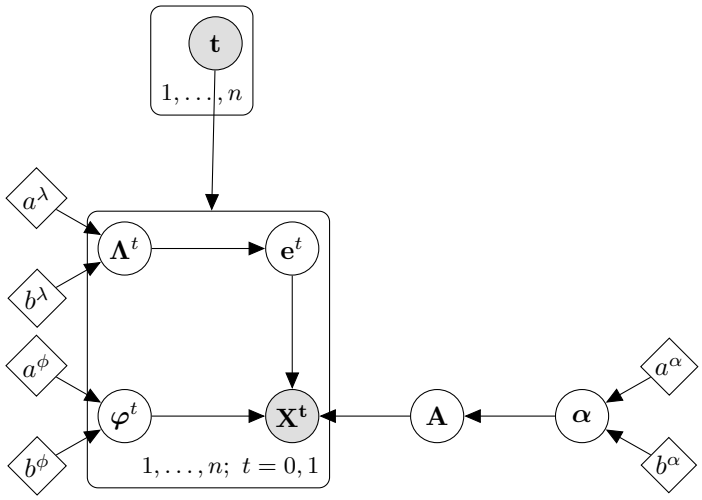


Figure 4.3: Class Dependent Evidence Model.

4.4 Optimal Bayesian Combination of Multiple Classifiers

The primary goal of combining multiple classifiers is to improve classification accuracy. To archive this goal the combination method must make the best use of the outputs of the classifiers in addition to any prior information that may be available. In a combination scheme made up of a list of diverse heterogeneous classifiers, the classifiers may differ significantly in reliability or strength. The list could be made up of multiple imperfect classifiers, such as in combining decision from human decision makers. In addition, the base classifiers are usually black boxes and one rarely have access to a reliable estimate of the confidence of their decisions. Combining multiple classifiers in the presence of such uncertainty can be optimally archived through a Bayesian inference. Bayesian learning provides a principled statistical framework for knowledge integration and for inferring the appropriate structure

Algorithm 5: Variational Bayesian Inference for Class Dependent Evidence Model

Input : Observed data $\mathbf{X}$, stopping criteria
Output: Variational posterior $q(\boldsymbol{\theta})$

Initialize moments of variational posterior $q(\boldsymbol{\theta})$;

repeat

Update $q(\mathbf{e}),$ for each $t = 0, 1$;

$$q(\mathbf{e}^t) = \prod_{i=1}^{n} \mathcal{N}_K(e_i^t | \left[m_e^i\right]^t, \Sigma_e^t), \qquad \textbf{Moments} \quad \begin{aligned} \Sigma_e &= \Big(\langle \boldsymbol{\Lambda} \rangle + \left\langle \boldsymbol{A}^T \boldsymbol{\varphi} \boldsymbol{A} \right\rangle \Big)^{-1} \\ m_e^i &= \Sigma_e \left\langle \boldsymbol{A} \right\rangle^T \langle \boldsymbol{\varphi} \rangle \, x_i \end{aligned}$$

Update $q(\mathbf{A})$;

$$q(\mathbf{A}) = \prod_{j=1}^{p} \mathcal{N}_K(a_j | m_a^j, \Sigma_a^j), \quad \textbf{Moments} \quad \begin{aligned} {\Sigma_a^j}^{-1} &= \sum_{t=0}^{1} \left\langle \phi_j^t \right\rangle \sum_{i=1}^{n} \left\langle e_i e_i^T \right\rangle^t + \langle \boldsymbol{\alpha} \rangle \\ m_a^j &= \sum_{t=0}^{1} \Sigma_a^j \left\langle \phi_j^t \right\rangle \sum_{i=1}^{n} x_{ij}^t \left\langle e_i^t \right\rangle \end{aligned}$$

Update $q(\boldsymbol{\alpha}^t),$ for each $t = 0, 1$;

$$q(\boldsymbol{\alpha}) = \prod_{k=1}^{K} \mathcal{G}\Big(\alpha_k^t \Big| \left[\hat{a}_k^\alpha\right]^t, \left[\hat{b}_k^\alpha\right]^t\Big), \quad \textbf{Moments} \quad \begin{aligned} \hat{a}_k^\alpha &= a_k^\alpha + \frac{p}{2} \\ \hat{b}_k^\alpha &= b_k^\alpha + \frac{\left\langle a_k^T a_k \right\rangle}{2} \\ \langle \boldsymbol{\alpha} \rangle &= \boldsymbol{diag}\left(\frac{\hat{a}_k^\alpha}{\hat{b}_k^\alpha}\right) \end{aligned}$$

Update $q(\boldsymbol{\varphi}),$ for each $t = 0, 1$;

$$q(\boldsymbol{\varphi}^t) = \prod_{j=1}^{p} \mathcal{G}\Big(\phi_j^t \Big| [\hat{a}_j^\phi]^t, [\hat{b}_j^\phi]^t\Big), \quad \textbf{Moments} \quad \begin{aligned} \hat{a}_j^\phi &= a_j^\phi + \frac{n}{2} \\ \hat{b}_j^\phi &= b_j^\phi + \frac{1}{2} \sum_{i=1}^{n} \left\langle (x_{ij} - a_j^T e_i)^2 \right\rangle \\ \langle \boldsymbol{\varphi} \rangle &= \boldsymbol{diag}\left(\frac{\hat{a}_j^\phi}{\hat{b}_j^\phi}\right) \end{aligned}$$

Update $q(\boldsymbol{\Lambda}),$ for each $t = 0, 1$;

$$q(\boldsymbol{\Lambda}^t) = \prod_{j=1}^{K} \mathcal{G}\Big(\lambda_k^t \Big| [\hat{a}_k^\lambda]^t, [\hat{b}_k^\lambda]^t\Big), \quad \textbf{Moments} \quad \begin{aligned} \hat{a}_k^\lambda &= a_k^\lambda + \frac{n}{2} \\ \hat{b}_k^\lambda &= b_k^\lambda + \frac{1}{2}\Big(\sum_{i=1}^{n} \left\langle e_i e_i^T \right\rangle \Big)_{(k,k)} \\ \langle \boldsymbol{\Lambda} \rangle &= \boldsymbol{diag}\left(\frac{\hat{a}_k^\lambda}{\hat{b}_k^\lambda}\right) \end{aligned}$$

until *stopping criteria*;

of a model by penalizing very complex model.

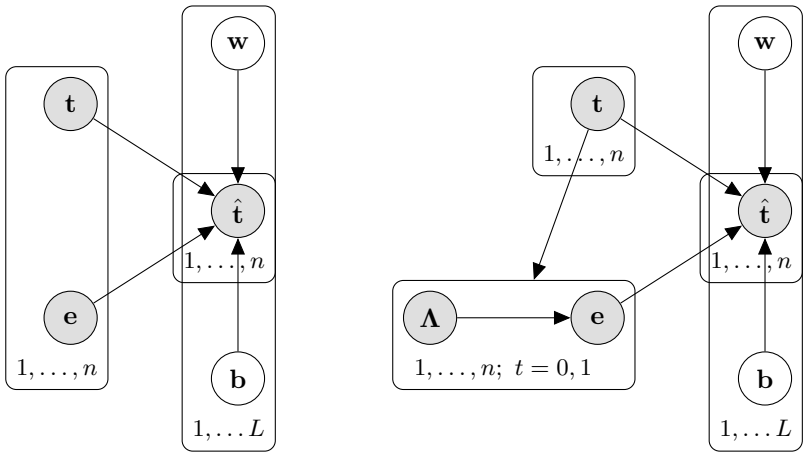
In the previous section, it was seen that in Bayesian inference, hierarchical priors can be constructed to automatically determine model structure or to determine the relevance of inputs to the model without need for testing all possible model structures. Inputs found to be irrelevant are automatically switch off. Such priors are called automatic relevance determination (ARD) (MacKay et al., 1994) hierarchical priors. This section presents a similar Bayesian framework for optimal integration of a set of classifiers of varying reliability.

Three combination schemes using the parametric model assumptions specified by equations 4.5 and 4.6 are presented. The first method obtains point estimates of the combination weights by the classical EM algorithm while the other two methods apply a full Variational Bayesian approach to obtain a posterior distribution over the weights. A ARD hierarchical prior is placed on the distribution of weights in the VB methods to switch off irrelevant classifiers in the ensemble. Note that, conceptually ARD prior is different from pruning. In pruning, if a member of the ensemble is completely irrelevant, then it is dropped from the ensemble and its contribution is zero. However, ARD technically allows all members to interact to some degree.

The ensemble models are derived for class independent and dependent classification evidence separately. However, only the general structure of the class dependent case is presented. The graphical models for the two cases is shown in Figure 4.4. Note that in the decoupled evidence and ensemble model inference procedures, the evidence is considered an observed random variable in the ensemble model.

4.4.1 An EM Method for Integrating Classification Models

The EM ensemble learning method presented in this section assumes that the output of the based classifiers are binary. The continuous case is treated under VB methods.



(a) **e** independent of class (b) **e** dependent on class

Figure 4.4: Graphical Model for Multiple Classifier Integration.

4.4.1.1 The EM algorithm

The EM algorithm (Dempster et al., 1977) attempts to find the maximum likelihood estimates of parameters of a model by marginalizing the likelihood of the observed data over hidden variables. Hidden variables are introduced into the model to make a complex and intractable likelihood easier to compute. Specifically, if $\boldsymbol{\theta}$ is a set of parameters of a model and the task is to find the value of $\boldsymbol{\theta}$ that maximizes the likelihood $p(x;\boldsymbol{\theta})$[1]. EM assumes access to some hidden variables z is available such that maximizing $p(x,z;\boldsymbol{\theta})$ is much easier. Once priors $p(z;\boldsymbol{\theta})$ for the hidden variables have been defined, one recovers the likelihood $p(x;\boldsymbol{\theta})$ by averaging out the hidden variables:

$$p(x;\boldsymbol{\theta}) = \int p(x,z;\boldsymbol{\theta})\,dz = \int p(x|z;\boldsymbol{\theta})p(z;\boldsymbol{\theta})\,dz.$$

[1]The notation $p(x;\boldsymbol{\theta})$ is used to indicate $\boldsymbol{\theta}$ is a parameter, if $\boldsymbol{\theta}$ is a random variable the notation $p(x|\boldsymbol{\theta})$ is used

In most problem, the interest is on the posterior of the hidden variables $p(z|x;\boldsymbol{\theta})$, as they can be ascribed domain specific semantics to them. By application of Bayes theorem, the posterior of the hidden variable can be obtained as

$$p(z|x;\boldsymbol{\theta}) = \frac{p(x|z;\boldsymbol{\theta})p(z;\boldsymbol{\theta})}{p(x;\boldsymbol{\theta})}.$$

However in most cases, the integral above is either impossible or very difficult to compute in close form. Assuming knowledge of the posterior $p(z|x;\theta)$ is the cornerstone of the EM algorithm. Introducing any arbitrary distribution $q(z)$ over the hidden variables, EM maximizes the log-likelihood function through an auxiliary lower bound $\mathcal{F}(q,\boldsymbol{\theta})$ given by

$$\begin{aligned}\log p(x;\boldsymbol{\theta}) &= \log \int q(z)\frac{p(x,z;\boldsymbol{\theta})}{q(z)}\,dz \\ &\geq \int q(z)\log\left(\frac{p(x,z;\boldsymbol{\theta})}{q(z)}\right)\,dz = \mathcal{F}(q,\boldsymbol{\theta}).\end{aligned} \tag{4.17}$$

The inequality above becomes an equality when $q(z)$ equals the true posterior $p(z|x;\boldsymbol{\theta})$.

The EM is a two step iterative algorithm that maximizes the lower bound $\mathcal{F}(q,\boldsymbol{\theta})$ and hence the log-likelihood. The standard EM iteration performs coordinate ascent on $\mathcal{F}(q,\boldsymbol{\theta})$ as follows:

E-Step

$$q^{new}(z) = \underset{q(z)}{\operatorname{argmax}}\,\mathcal{F}(q,\boldsymbol{\theta}^{old}). \tag{4.18}$$

The maximum occurs when $KL(q(z)\|p(z|x;\boldsymbol{\theta}^{old})) = 0$ i.e when $q(z) = p(z|x;\boldsymbol{\theta}^{old})$, thus E-Step basically computes $p(z|x;\boldsymbol{\theta}^{old})$

M-Step

$$\boldsymbol{\theta}^{new} = \underset{\theta}{\operatorname{argmax}}\,\mathcal{F}(q^{new},\boldsymbol{\theta}). \tag{4.19}$$

A different expression of the lower bound $\mathcal{F}(q, \boldsymbol{\theta})$ can be obtain by substituting $q(z) = p(z|x; \boldsymbol{\theta}^{old})$ into the lower bound:

$$\mathcal{F}(q, \boldsymbol{\theta}) = \int p(z|x; \boldsymbol{\theta}^{old}) \log p(x, z; \boldsymbol{\theta})\, dz - \int p(z|x; \boldsymbol{\theta}^{old}) \log p(x, z; \boldsymbol{\theta}^{old})\, dz \tag{4.20}$$

$$= Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) + \text{Entropy}. \tag{4.21}$$

Because the Entropy term is constant with respect to $\boldsymbol{\theta}$, the function $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})$ which is the expectation of the *complete* log-likelihood (log-likelihood of data and hidden variables) is the function to maximized in the M-Step. Thus the EM computes $p(z|x; \boldsymbol{\theta}^{old})$ in the E-Step and maximizes $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})$ with respect to $\boldsymbol{\theta}$ in the M-Step.

4.4.1.2 Ensemble Method with EM

Let $\hat{t}_{ij} \in \{0, 1\}$ be the class assigned by classifier $h_j \in \{h_1, \ldots, h_L\}$ for the data point x_i with classification evidence e_i. Recall that by the assumptions of the model, the predictions depends on the true class t_i, the classification evidence e_i, the reliability and bias of the classifier and are modeled by the logistic function:

$$p(\hat{t}_{ij}|t_i, e_i, \mathbf{w}_j, b_j) = p(\hat{t}_{ij} = t_i|e_i; \mathbf{w}_j, b_j) = \sigma(\mathbf{w}_j^T e_i + b_j),$$

where $\sigma(x) = 1/(1 + e^{-x})$ and $\mathbf{w}_j$ is a K-dimensional weight or reliability vector for classifier j. Using Bayes theorem and the independence condition of the graphical model, the posterior probability of the true class given the the predictions can be written as

$$p(t_i|\hat{t}_{ij}) = \frac{p(\hat{t}_{ij}|t_i, e_i; \mathbf{w}_j, b_j)p(t_i)}{\sum_{t_i=0}^{1} p(\hat{t}_{ij}|t_i, e_i; \mathbf{w}_j, b_j)p(t_i)}. \tag{4.22}$$

The predictions can be grouped into an $n \times L$ matrix $\mathbf{Y} = \{\hat{t}_{ij},\ i = 1, \ldots n,\ j = 1, \ldots L\}$. So if $\hat{\mathbf{t}}_i$ is the i'th row of the matrix $\mathbf{Y}$, then using the conditional independence assumption gives

$$p(t_i|\hat{\mathbf{t}}_i) \propto p(t_i) \prod_{j=1}^{L} p(\hat{t}_{ij}|t_i, e_i; \mathbf{w}_j, b_j) \tag{4.23}$$

The EM algorithm will be applied to obtain the MLE of the parameters of the model $\boldsymbol{\theta} = (\mathbf{w}, \mathbf{b})$ where $\mathbf{w} \in \mathbb{R}^{L \times K}$ is a matrix containing the combination weights for each classifier and $\mathbf{b} \in \mathbb{R}^{L}$ is the vector of their biases. To derive the EM method for combining the outputs of the base classifiers, it is assumed that the true class labels t_i are the hidden variables in the EM algorithm. Also it should be noted that in the presented EM ensemble method, the parameters $\boldsymbol{\theta}$ are deterministic and not random and should be treated as such in the graphical model in Figure 4.4.

Ignoring the normalization constant, the EM algorithm proceeds as follows

E-Step: compute $p(t_i|\hat{\mathbf{t}}_i) = p(t_i) \prod_{j=1}^{L} p(\hat{t}_{ij}|t_i, e_i; \mathbf{w}_j, b_j)$

M-Step: maximize the auxiliary function $Q(\mathbf{w}, \mathbf{b})$, which is the expectation of the complete log-likelihood with respect to the class posterior $p(t_i|\hat{\mathbf{t}}_i)$ computed in the last E-Step. Thus

$$\begin{aligned}
Q(\mathbf{w}, \mathbf{b}) &= \langle \log p(\mathbf{Y}, \mathbf{t}, \mathbf{e}; \mathbf{w}, \mathbf{b}) \rangle \\
&= \langle \log p(\mathbf{t}) p(\mathbf{Y}|\mathbf{t}, \mathbf{e}; \mathbf{w}, \mathbf{b}) \rangle \\
&= \left\langle \log \prod_{i=1}^{n} p(t_i) \prod_{j=1}^{L} p(\hat{t}_{ij}|t_i, e_i; \mathbf{w}_j, b_j) \right\rangle \\
&= \left\langle \sum_{i=1}^{n} \log p(t_i) + \sum_{i=1}^{n} \sum_{j=1}^{L} \log p(\hat{t}_{ij}|t_i, e_i; \mathbf{w}_j, b_j \right\rangle \\
&= \sum_{i=1}^{n} \langle \log p(t_i) \rangle + \sum_{i=1}^{n} \sum_{j=1}^{L} \left\langle \log p(\hat{t}_{ij}|t_i, e_i; \mathbf{w}_j, b_j) \right\rangle .
\end{aligned} \tag{4.24}$$

Let $p_i^k = p(t_i = k|\hat{\mathbf{t}}_i),\ k = 0, 1$, then

$$Q(\mathbf{w}, \mathbf{b}) = \sum_{i=1}^{n}\sum_{k=0}^{1} p_i^k \log p(t_i) + \sum_{i=1}^{n}\sum_{j=1}^{L}\sum_{k=0}^{1} p_i^k \log p(\hat{t}_{ij}|t_i, e_i; \mathbf{w}_j, b_j)$$

$p(\hat{t}_{ij}|t_i, e_i; \mathbf{w}_j, b_j) = \sigma(\mathbf{w}_j^T e_i + b_j)$ can be expressed in terms of the two classes as

$$p(\hat{t}_{ij}|t_i = 1, e_i; \mathbf{w}_j, b_j) = \sigma(\mathbf{w}_j^T e_i + b_j)^{\hat{t}_{ij}} \Big(1 - \sigma(\mathbf{w}_j^T e_i + b_j)\Big)^{1-\hat{t}_{ij}} \tag{4.25}$$

$$p(\hat{t}_{ij}|t_i = 0, e_i; \mathbf{w}_j, b_j) = \sigma(\mathbf{w}_j^T e_i + b_j)^{1-\hat{t}_{ij}} \Big(1 - \sigma(\mathbf{w}_j^T e_i + b_j)\Big)^{\hat{t}_{ij}}. \tag{4.26}$$

Letting $\sigma_{ij} = \sigma(\mathbf{w}_j^T e_i + b_j)$, $Q(\mathbf{w}, \mathbf{b})$ can be written as

$$\begin{aligned} Q(\mathbf{w}, \mathbf{b}) &= \sum_{i=1}^{n}\sum_{k=0}^{1} p_i^k \log p(t_i) + \sum_{i=1}^{n}\sum_{j=1}^{L} \Big[p_i^0 \log p(\hat{t}_{ij}|t_i = 0, e_i; \mathbf{w}_j, b_j) \\ &\quad + p_i^1 \log p(\hat{t}_{ij}|t_i = 1, e_i; \mathbf{w}_j, b_j)\Big] \\ &= \sum_{i=1}^{n}\sum_{k=0}^{1} p_i^k \log p(t_i) + \sum_{i=1}^{n}\sum_{j=1}^{L} \Big[p_i^0\big((1 - \hat{t}_{ij}) \log \sigma_{ij} + \hat{t}_{ij} \log(1 - \sigma_{ij})\big)\Big] \\ &\quad + \sum_{i=1}^{n}\sum_{j=1}^{L} \Big[p_i^1\big(t_{ij} \log \sigma_{ij} + (1 - \hat{t}_{ij}) \log(1 - \sigma_{ij})\big)\Big]. \end{aligned} \tag{4.27}$$

To maximize $Q(\mathbf{w}, \mathbf{b})$ with respect to $\boldsymbol{\theta} = (\mathbf{w}, \mathbf{b})$ in the M-Step, its derivative with respect to $\boldsymbol{\theta}$ is taken and set to zero. By including a constant term of 1 in e_i i.e $e_i := (e_i, 1)$ and

also augmenting $\mathbf{w}_j$ with b_j i.e $\mathbf{w}_j := (\mathbf{w}_j, b_j)$ the derivative can be written as

$$\begin{aligned}\frac{\partial Q}{\partial \mathbf{w}_j} &= \sum_{i=1}^{n} \Big[p_i^0\big((1-\hat{t}_{ij})(1-\sigma_{ij})e_i - \hat{t}_{ij}\sigma_{ij}e_i\big) + p_i^1\big((\hat{t}_{ij})(1-\sigma_{ij})e_i - (1-\hat{t}_{ij})\sigma_{ij}e_i\big)\Big] \\ &= \sum_{i=1}^{n} \Big[p_i^0(1-\hat{t}_{ij}) + p_0^1\hat{t}_{ij} - \sigma_{ij}\Big]e_i. \end{aligned} \tag{4.28}$$

The derivative is non-linear, therefore standard iterative methods such as the Newton-Raphson or gradient ascent can be used to find a local maxima.

Letting $\boldsymbol{\sigma}_j$ be the j'th column of the $n \times L$ matrix $\{\sigma_{ij},\ i = 1, \dots n, j = 1, \dots, L\}$ and $\mathbf{V}_j = 1 - \mathbf{Y}_j - \mathbf{p}^1 + 2\mathbf{p}^1 \circ \mathbf{Y}_j$ where $\mathbf{Y}_j$ is the j'th column of $\mathbf{Y}$, $\mathbf{p}^1 = (p_1^1, \dots, p_n^1)$ and $\circ$ represents element wise matrix multiplication, the derivative can be written in matrix form as

$$\frac{\partial Q}{\partial \mathbf{w}_j} = \mathbf{e}^T(\mathbf{V}_j - \boldsymbol{\sigma}_j),\ j = 1, \dots L$$

The second derivative or Hessian matrix required by the Newton-Raphson algorithm is given by

$$\frac{\partial^2 Q}{\partial \mathbf{w}_j \partial \mathbf{w}_j^T} = -\mathbf{e}^T\mathbf{W}_j\mathbf{e},\ j = 1, \dots L, \tag{4.29}$$

where $\mathbf{W}_j$ is a diagonal matrix with i'th element $\sigma_{ij}(1-\sigma_{ij})$.

The two steps of the EM are iterated until convergence. With the parameter estimates, the posterior probabilities of the true class given the class predictions can be estimated using equation 4.23. The class prior $p(t_i)$ can be estimated by $p(t_i = k) = \frac{n_k}{n}$, where n_k is the number of observations in class k in the training set.

4.4.2 Variational Bayesian Ensemble Methods

A major draw back of the EM combination scheme presented in the previous section is that it assumes the true class t is the hidden variable, and therefore does not benefit from the information provided by this variable during learning. Another major problem, which is typical with most maximum likelihood methods is that they often do not take account of model complexity. Overfitting is very common with such methods as more complex models are not penalized. By treating the parameters of the model as random variables and averaging them out over an ensemble of models, full Bayesian approaches are able to overcome these problems. All information inferred from the data about the parameters (based on the assumed model) is captured by the posterior distribution of the parameters rather than in the point estimates of maximum likelihood approaches. The Bayesian approach is optimal and naturally data-driven.

In this section the principles of the variational Bayesian (VB) method are applied to optimally combine the outputs of multiple classifiers. The reliability of each classifier in the ensemble is determined by the use of a hierarchical ARD prior placed on the distribution of the combination weights. In this way, the effect of weak classifiers in the ensemble can be mitigated while allowing the strength of good and complementary classifiers to be reinforced.

The outputs of the classifiers can be discrete (binary in this case) or continuous. For discrete outputs, the presented VB method modify the standard VB logistic regression (Jaakkola and Jordan, 1997; Bishop and Nasrabadi, 2006) allowing it to model the discrete outputs, the true class and the classification evidence in a unique way. When the outputs are continuous, the classification evidence is simply combined with the predictions and a straightforward learning by the VB logistic regression or any other classification algorithm can be used. Figure 4.5 shows the graphical model for VB ensemble learning.

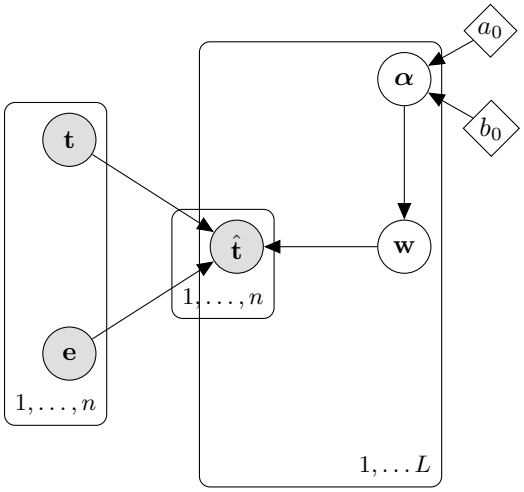


Figure 4.5: Variation Bayesian Ensemble Learning for Class Independent Evidence

4.4.2.1 VB Ensemble for Classifiers with Discrete Outputs (VBD)

Based on the model assumptions given by equation 4.5, the predicted probabilities satisfy

$$p(\hat{t}_{ij} = 1 | t_i, e_i, \mathbf{w}_j) = \sigma(\mathbf{w}_j^T e_i),$$

$$p(\hat{t}_{ij} = 0 | t_i, e_i, \mathbf{w}_j) = 1 - \sigma(\mathbf{w}_j^T e_i).$$

Let $s_{ij} = 2\hat{t}_{ij} - 1$ ($s_{ij}^2 = 1$), then the probabilities for the two classes can be computed as

$$p(\hat{t}_{ij} | t_i = 1, e_i, \mathbf{w}_j) = \sigma(s_{ij} \mathbf{w}_j^T e_i),$$

$$p(\hat{t}_{ij} | t_i = 0, e_i, \mathbf{w}_j) = 1 - \sigma(s_{ij} \mathbf{w}_j^T e_i).$$

These two equation can be combined as

$$p(\hat{t}_{ij} | t_i, e_i, \mathbf{w}_j) = \sigma(s_{ij} \mathbf{w}_j^T e_i)^{t_i} \Big(1 - \sigma(s_{ij} \mathbf{w}_j^T e_i)\Big)^{1 - t_i}. \tag{4.30}$$

Letting $\mathbf{Y}$ be the $n \times L$ matrix containing the predictions $\hat{t}_{ij}$ and $\mathbf{w}$ the $L \times K$ matrix of weights, the conditional independence assumption implies

$$p(\mathbf{Y}|\mathbf{t}, \mathbf{e}, \mathbf{w}) = \prod_{i=1}^{n} \prod_{j=1}^{L} \sigma(s_{ij} \mathbf{w}_j^T e_i)^{t_i} \Big(1 - \sigma(s_{ij} \mathbf{w}_j^T e_i)\Big)^{1-t_i}$$

$$\log p(\mathbf{Y}|\mathbf{t}, \mathbf{e}, \mathbf{w}) = \sum_{i=1}^{n} \sum_{j=1}^{L} \Big[t_i \log \sigma(s_{ij} \mathbf{w}_j^T e_i) + (1 - t_i) \log \big(1 - \sigma(s_{ij} \mathbf{w}_j^T e_i)\big) \Big]. \tag{4.31}$$

In VB logistic regression, the parameters $\mathbf{w}$ are treated as random variables and are given zero mean Gaussian priors. These priors can also be given hierarchical ARD Gamma priors. The specification of these priors presented here is some what different. In standard variational logistic regression the parameters $\mathbf{w}$ is a vector where each element weights the attributes of the training set. In this case however, $\mathbf{w}$ is a matrix that simultaneously weights the classifiers and evidence. The rows weights the classifiers condition on the true class while columns weights the evidence. To model a prior for $\mathbf{w}$, each row is modeled as a zero mean Gaussian distribution with a gamma prior controlling the precision. Thus the following priors are specified for the parameters

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{j=1}^{L} \mathcal{N}_K(\mathbf{w}_j|\mathbf{O}, \alpha_j^{-1}\mathbf{I}) \tag{4.32}$$

$$p(\boldsymbol{\alpha}) = \prod_{j=1}^{L} \mathcal{G}(\alpha_j|a_j, b_j). \tag{4.33}$$

VB Inference

VB inference is based on maximizing a lower bound of the log-likelihood of the data. The data in this case are the outputs of the classifiers $\mathbf{Y}$ given the true class t_i and the classification evidence e_i. The log-likelihood can be written as

$$\log p(\hat{t}_{ij}|t_i, e_i) = \log \int p(\hat{t}_{ij}, \boldsymbol{\theta}|t_i, e_i)\, d\boldsymbol{\theta}$$

$$= \log \int p(\hat{t}_{ij}|t_i, e_i, \boldsymbol{\theta}) p(\boldsymbol{\theta})\, d\boldsymbol{\theta}. \tag{4.34}$$

Using Bayes theorem, inference on predicted class given the true class and evidence can be transformed into inference on true class given the predicted class and evidence as

$$\log p(t_i|\hat{t}_{ij}, e_i) = \log p(\hat{t}_{ij}|t_i, e_i) p(t_i|e_i) + constant.$$

Thus

$$\log p(t_i|\hat{t}_{ij}, e_i) = \log \int p(\hat{t}_{ij}|t_i, e_i, \boldsymbol{\theta}) p(t_i|e_i) p(\boldsymbol{\theta})\, d\boldsymbol{\theta} + constant$$

$$\geq \int q(\boldsymbol{\theta}) \log \left(\frac{p(\hat{t}_{ij}|t_i, e_i, \boldsymbol{\theta}) p(t_i|e_i) p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} \right) d\boldsymbol{\theta} = \mathcal{F}(q), \tag{4.35}$$

where the variational posterior $q(\boldsymbol{\theta}) = q(\mathbf{w}, \boldsymbol{\alpha})$ approximates the true posterior $p(\boldsymbol{\theta}) = p(\mathbf{w}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})$ of the parameters and is assumed to factorize as $q(\mathbf{w}, \boldsymbol{\alpha}) = q(\mathbf{w})q(\boldsymbol{\alpha})$.

In the discussion on VB for factor analysis in section 4.3.1, the variational posteriors $q(\boldsymbol{\theta}_i)$ for any of the parameters were obtained by taking the expectation of the complete log-likelihood with respect to the other variational posteriors $q(\boldsymbol{\theta}_j),\ j \neq i$. This led to simple analytic expressions for $q(\boldsymbol{\theta})$ because the the model structure was Gaussian i.e it has a conjugate-prior in the exponential family. In this case, similar analytic expressions cannot be obtained because the logistic model does not have a conjugate-prior in the exponential family. It is nevertheless possible to find an accurate variational transformation of the model such that the desired variational distributions can be computed. In Jaakkola and Jordan (1997), a lower bound on the logistic function is introduced which allow the log-likelihood for the logistic regression to be approximated by the exponential of a quadratic

form. Specifically, the logistic function is lower bounded by

$$\sigma(x) \geq \sigma(\xi) \exp\left\{\frac{1}{2}(x-\xi) - \lambda(\xi)(x^2-\xi^2)\right\}, \tag{4.36}$$

where $\lambda(\xi) = \frac{1}{2\xi}\left\{\sigma(\xi) - \frac{1}{2}\right\}$ and ξ is a variational parameter corresponding to each data point. Applying this lower bound to equation 4.31 with $x = s_{ij}\mathbf{w}_j^T e_i$, the log-likelihood of the data is lower bounded by

$$\begin{aligned}
\log p(\mathbf{Y}|\mathbf{t},\mathbf{e},\mathbf{w}) &\geq \sum_{i=1}^{n}\sum_{j=1}^{L}\left[t_i\left(\log\sigma(\xi_i) + \frac{1}{2}s_{ij}\mathbf{w}_j^T e_i - \lambda(\xi_i)\mathbf{w}_j^T e_i e_i^T \mathbf{w}_j - \frac{\xi_i}{2} + \lambda(\xi)\xi_i^2\right)\right. \\
&\quad \left. + (1-t_i)\left(\log\sigma(\xi_i) - \frac{1}{2}s_{ij}\mathbf{w}_j^T e_i - \lambda(\xi_i)\mathbf{w}_j^T e_i e_i^T \mathbf{w}_j - \frac{\xi_i}{2} + \lambda(\xi)\xi_i^2\right)\right] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{L}\left[\log\sigma(\xi_i) + \left(t_i - \frac{1}{2}\right)s_{ij}\mathbf{w}_j^T e_i - \lambda(\xi_i)\mathbf{w}_j^T e_i e_i^T \mathbf{w}_j - \frac{\xi_i}{2} + \lambda(\xi)\xi_i^2\right] \\
&= \log H(\mathbf{w},\boldsymbol{\xi}).
\end{aligned} \tag{4.37}$$

Substituting this expression into equation 4.35 result in a new variational lower bound given by

$$\begin{aligned}
\log p(\mathbf{t}|\mathbf{Y},\mathbf{e}) &\geq \int q(\boldsymbol{\theta}) \log\left(\frac{p(\mathbf{Y}|\mathbf{t},\mathbf{e},\boldsymbol{\theta})p(\mathbf{t}|\mathbf{e})p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})}\right) d\boldsymbol{\theta} = \mathcal{F}(q) \\
&\geq \int q(\boldsymbol{\theta}) \log\left(\frac{H(\mathbf{w},\boldsymbol{\xi})p(\mathbf{t}|\mathbf{e})p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})}\right) d\boldsymbol{\theta} = \widetilde{\mathcal{F}}(q),
\end{aligned} \tag{4.38}$$

where $p(\mathbf{t}|\mathbf{e}) = p(\mathbf{t})$ for the class independent evidence model. With this new approximation,

the complete log-likelihood is approximated by

$$\begin{aligned}\log p(\mathbf{t},\mathbf{Y},\mathbf{e},\mathbf{w}) &= \log p(\mathbf{t}) + \log p(\mathbf{Y}|\mathbf{t},\mathbf{e},\mathbf{w}) + \log p(\mathbf{w}|\boldsymbol{\alpha}) + \log p(\boldsymbol{\alpha}) \\ &\approx \sum_{i=1}^{n}\sum_{k=0}^{1} p(t_i = k) + \sum_{j=1}^{L}\left[\mathbf{w}_j^T\left(\sum_{i=1}^{n}\left(t_i - \frac{1}{2}\right)s_{ij}e_i\right)\right. \\ &\left. + \mathbf{w}_j^T\left(\sum_{i=1}^{n}\lambda(\xi_i)e_i e_i^T\right)\mathbf{w}_j\right] + \sum_{i=1}^{n} L\left(\log\sigma(\xi_i) - \frac{\xi_i}{2} + \lambda(\xi_i)\xi_i^2\right) \\ &+ \frac{K}{2}\sum_{j=1}^{L}\log\alpha_j - \frac{1}{2}\sum_{j=1}^{L}\alpha_j\mathbf{w}_j^T\mathbf{w}_j + \sum_{j=1}^{L}(a_j - 1)\log\alpha_j - \sum_{k=1}^{L} b_j\alpha_j + \textit{constant}.\end{aligned} \tag{4.39}$$

The distribution of variational posterior $q(\mathbf{w})$ can now be computed by taking the expectation of the complete log-likelihood with respect to $q(\boldsymbol{\alpha})$ and keeping only terms in $\mathbf{w}$. The remaining expression can be identified as the joint distribution of L multivariate normal distributions. Similarly the posterior $q(\boldsymbol{\alpha})$ is obtained by taking the expectation with respect to $q(\mathbf{w})$ and keep only terms in $\boldsymbol{\alpha}$. The remaining expression can be identified as the product of L gamma distributions. These distributions are alternately updated just as in VB factor analysis until some convergence criterion, which is usually when a plateau in the lower bound is observed. The updates of the distribution and their moments is presented in Algorithm 6.

The expression for the new approximate variational lower bound is given by

$$\widetilde{\mathcal{F}}(q) = \Big\langle H(\mathbf{w},\boldsymbol{\alpha})\Big\rangle_{q(\mathbf{w})q(\boldsymbol{\alpha})} + \Big\langle \log p(\mathbf{t})\Big\rangle_{q(\mathbf{w})q(\boldsymbol{\alpha})} - KL, \tag{4.40}$$

Algorithm 6: VB Ensemble for Classifiers with Discrete Outputs

Input : Observed data **t, Y, e**, stopping criteria.
Output: Variational posteriors $q(\mathbf{w})$ and $q(\boldsymbol{\alpha})$.

Initialize moments of variational posteriors ;
repeat

1 Update $q(\mathbf{w})$;

$$q(\mathbf{w}) = \prod_{j=1}^{L} \mathcal{N}_K(\mathbf{w}_j | m^j, \Sigma^j), \quad \textbf{Moments} \quad \begin{aligned} \boldsymbol{\Sigma}^j &= \left(\langle \alpha_j \rangle \boldsymbol{I} + 2 \sum_{i=1}^{n} \lambda(\xi_i) e_i e_i^T \right)^{-1} \\ \boldsymbol{m}^j &= \boldsymbol{\Sigma}^j \sum_{i=1}^{n} \left(t_i - \frac{1}{2} \right) s_{ij} e_i \end{aligned}$$

2 Update $q(\boldsymbol{\alpha})$;

$$q(\boldsymbol{\alpha}) = \prod_{j=1}^{L} \mathcal{G}(\alpha_j | \hat{a}_j, \hat{b}_j), \qquad \textbf{Moments} \qquad \begin{aligned} \hat{a}_j &= a_j + \frac{K}{2} \\ \hat{b}_j &= b_j + \frac{\left\langle \boldsymbol{w}_j^T \boldsymbol{w}_j \right\rangle}{2} \\ \langle \alpha_j \rangle &= \frac{\hat{a}_j}{\hat{b}_j} \end{aligned}$$

3 Update ξ_i ;

$$\xi_i^2 \longleftarrow \frac{1}{L} e_i^T \left(\sum_{j=1}^{L} \left\langle \mathbf{w}_j \mathbf{w}_j^T \right\rangle \right) e_i$$

until *stopping criteria*;

where

$$\begin{aligned} \left\langle H(\mathbf{w}, \boldsymbol{\alpha}) \right\rangle_{q(\mathbf{w})q(\boldsymbol{\alpha})} = & \sum_{j=1}^{L} \langle w_j \rangle^T \left(\sum_{i=1}^{n} \left(t_i - \frac{1}{2} \right) s_{ij} e_i \right) \\ & - \operatorname{tr} \left[\left(\sum_{i=1}^{n} \lambda(\xi_i) e_i e_i^T \right) \left(\sum_{j=1}^{L} \left\langle \mathbf{w}_j \mathbf{w}_j^T \right\rangle \right) \right] \\ & + \sum_{i=1}^{n} L \left(\log \sigma(\xi_i) - \frac{\xi_i}{2} + \lambda(\xi_i) \xi_i^2 \right) \end{aligned} \tag{4.41}$$

and

$$KL = \left\langle KL(q(\mathbf{w}) \| p(\mathbf{w}|\boldsymbol{\alpha}) \right\rangle_{q(\boldsymbol{\alpha})} + KL(q(\boldsymbol{\alpha}) \| p(\boldsymbol{\alpha}))$$

is the Kullback-Leibler divergence between the variational posterior and the true posterior and its computation is similar to that for VB factor analysis. The function $\mathrm{tr}(\cdot)$ computes the trace of a matrix.

Notice that, in Algorithm 6 the covariance matrix for each $\mathbf{w}_j$ depends on the local variational parameters ξ_i through $\lambda(\xi_i)$ and so its value needs to be specified. The values of ξ_i can be obtain by optimizing the approximate lower bound $\widetilde{\mathcal{F}}(q)$ as a function of ξ_i. In $\widetilde{\mathcal{F}}(q)$, only $\langle H(\mathbf{w}, \boldsymbol{\alpha})\rangle_{q(\mathbf{w})q(\boldsymbol{\alpha})}$ is a function of ξ_i, thus taking its derivative with respect to ξ_i and setting it to zero leads to the update rule

$$(\xi_i^{new})^2 = \frac{1}{L} e_i^T \left(\sum_{j=1}^{L} \langle \mathbf{w}_j \mathbf{w}_j^T \rangle \right) e_i. \tag{4.42}$$

The variational lower bound is maximized by iterating over each update in Algorithm 6 and the local variational parameter ξ_i until no significant change is observed in $\widetilde{\mathcal{F}}(q)$. Optionally, the hyperparameters a and b of the gamma distribution can also be optimized after each update. This is done by taking the derivative of the complete log-likelihood with respect to these parameters and setting them to zero. The resulting equation does not have a close form solution and can be solve using iterative methods like the Newton-Raphson method.

Algorithm 6 appears to be a little complex than standard VB logistic regression. However, because the outputs of the classifiers are binary and the dimension of e is usually very small, the algorithm converges very fast taking far less number of iterations than standard VB logistic regression.

Making Predictions

Once the variational posterior distributions and moments have been computed, predictions for new inputs, i.e, new outputs $\mathbf{Y}^*$ from the base classifiers and classification evidence $\mathbf{e}^*$

from the test set can be obtained by substituting the most probable value or mean of the posterior distribution of the weights $\mathbf{w}_{MP} = \langle \mathbf{w} \rangle$ into equation 4.22 to give the predictive distribution in the form $p(\mathbf{t}^*|\mathbf{Y}^*, \mathbf{e}^*, \mathbf{w}_{MP})$.

An alternative and perhaps more accurate approach is to marginalize $p(\mathbf{t}^*|\mathbf{Y}^*, \mathbf{e}^*, \mathbf{w})$ over the posterior distribution of the weights so as to account for any uncertainty in the weights. That is, given new data $(\mathbf{Y}^*, \mathbf{e}^*)$ and the training data $\mathcal{D}_l$, compute the predictive distribution

$$\begin{aligned} p(\mathbf{t}^* = 1|\mathbf{Y}^*, \mathbf{e}^*, \mathcal{D}_l) &= \int p(\mathbf{t}^* = 1|\mathbf{Y}^*, \mathbf{e}^*, \mathbf{w}) p(\mathbf{w}|\mathcal{D}_l)\, d\mathbf{w} \\ &\approx \int p(\mathbf{t}^* = 1|\mathbf{Y}^*, \mathbf{e}^*, \mathbf{w}) q(\mathbf{w})\, d\mathbf{w} \\ &= \int \frac{p(\mathbf{Y}^*|\mathbf{t}^* = 1, \mathbf{e}_i^*, \mathbf{w}) p(\mathbf{t} = 1)}{Z(\mathbf{w})} q(\mathbf{w})\, d\mathbf{w}. \end{aligned} \tag{4.43}$$

This gives for each data point

$$p(\mathbf{t}_i^* = 1|\hat{t}_i^*, e_i^*, \mathcal{D}) = p(\mathbf{t}_i = 1) \prod_{j=1}^{L} \int \frac{\sigma(s_{ij}^* \mathbf{w}_j^T e_i^*)}{Z(\mathbf{w}_j)} \mathcal{N}_k(\mathbf{w}|m^i, \Sigma^j)\, d\mathbf{w}_j \tag{4.44}$$

where $Z(\mathbf{w}_j) = \sum_{t_i} p(\hat{t}_{ij}^*|t_i, e_i^*, \mathbf{w}_j) p(t_i)$ is the normalization constant in equation 4.22. Unfortunately, the integral above involves a product between a logistic sigmoid function and a Gaussian which is intractable and requires numerical methods. An approximation to such integrals is given in MacKay (1992). However, the normalization constant in the denominator of the expression above depends on $\mathbf{w}_j$ making it difficult to apply MacKay's approximation directly. Since $Z(\mathbf{w}_j)$ is a normalization constant, it can be assumed to be constant over all components of $\mathbf{w}_j$ and thus can be approximated by $Z(\mathbf{w}_{MP})$. This approximation was however found to yield poor quality estimates of the class posterior

probability and was dropped in favor of the first approach.

4.4.2.2 VB Ensemble for Classifiers with Continuous Outputs (VBC)

When the outputs of the classifiers are continuous, a simple ensemble scheme that make use of the classification evidence is to combine the evidence with the predictions into a new variable and learn using any standard classification algorithm. The VB logistic regression is preferred in this work as it provides a principled approach to determine the reliability of the classifiers and for easy comparison with the ensemble method using discrete outputs.

Let $\hat{\mathbf{t}}_i = \left\{\hat{t}_{i1}, \ldots, \hat{t}_{iL}\right\} \in \mathbb{R}^L$ be the continuous class predictions from L base classifiers for the data point x_i with classification evidence e_i and true class $t_i \in \{0, 1\}$. Form the new vector $\Phi_i = (\hat{\mathbf{t}_i}, e_i) \in \mathbb{R}^{L+K}$ so that the training data becomes $\mathcal{D}_l = \{(t_i, \Phi_i), i = 1, \ldots n\}$. From equation 4.6, the distribution of the class posterior probability given the data is given by

$$p(t_i|\Phi_i, \mathbf{w}) = \sigma(\mathbf{w}^T\Phi_i)^{t_i}\left(1 - \sigma(\mathbf{w}^T\Phi_i)\right)^{1-t_i} = \sigma\left((2t_i - 1)\mathbf{w}^T\Phi_i\right) \tag{4.45}$$

where $\mathbf{w} \in \mathbb{R}^{L+K}$ is the vector of weights for the classifiers and the classification evidence.

In VB logistic regression, the weights are given zero mean Gaussian priors whose precision are given gamma priors. Thus

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{j=1}^{L+K} \mathcal{N}(\mathbf{w}_j|0, \alpha_j^{-1}), \tag{4.46}$$

$$p(\boldsymbol{\alpha}) = \prod_{j=1}^{L+K} \mathcal{G}(\alpha_j|a_j, b_j). \tag{4.47}$$

The likelihood of the data is

$$p(\mathbf{t}|\boldsymbol{\Phi}, \mathbf{w}) = \prod_{i=1}^{n} \sigma\left((2t_i - 1)\mathbf{w}^T\Phi_i\right). \tag{4.48}$$

The likelihood does not have a conjugate prior in the exponential family, but by introducing a lower bound on the logistic sigmoid function as done in the previous section, the log-likelihood can be approximated by the exponential of a quadratic form. Using similar variational inference as done in the previous section, the updates for variational posterior can be derived as shown in Algorithm 7.

Algorithm 7: VB Ensemble for Classifiers with Continuous Outputs

Input : Observed data $\mathcal{D}_l = (\mathbf{t}, \mathbf{\Phi})$ stopping criteria.
Output: Variational posteriors $q(\mathbf{w})$ and $q(\boldsymbol{\alpha})$.
Initialize moments of variational posteriors ;
repeat

1 Update $q(\mathbf{w})$;

$$q(\mathbf{w}) = \mathcal{N}_{L+K}(\mathbf{w}|\mathbf{m}, \mathbf{\Sigma}), \quad \textbf{Moments} \quad \begin{aligned} \mathbf{\Sigma} &= \left(\langle \boldsymbol{\alpha} \rangle + 2 \sum_{i=1}^{n} \lambda(\xi_i) \Phi_i \Phi_i^T \right)^{-1} \\ \boldsymbol{m} &= \frac{1}{2} \mathbf{\Sigma} \sum_{i=1}^{n} (2t_i - 1) \Phi_i \end{aligned}$$

2 Update $q(\boldsymbol{\alpha})$;

$$q(\boldsymbol{\alpha}) = \prod_{j=1}^{L+K} \mathcal{G}(\alpha_j|\hat{a}_j, \hat{b}_j), \quad \textbf{Moments} \quad \begin{aligned} \hat{a}_j &= a_j + \frac{K}{2} \\ \hat{b}_j &= b_j + \frac{\left\langle \boldsymbol{w}_j^2 \right\rangle}{2} \\ \langle \boldsymbol{\alpha} \rangle &= \boldsymbol{diag} \left(\frac{\hat{a}_j}{\hat{b}_j} \right) \end{aligned}$$

3 Update ξ_i ;

$\xi_i^2 \longleftarrow \Phi_i^T \left(\left\langle \mathbf{w}\mathbf{w}^T \right\rangle \right) \Phi_i$

until *stopping criteria*;

Making Predictions

Predictions can be made for new data $\mathbf{\Phi}^* = (\mathbf{Y}^*, \mathbf{e}^*)$ by substituting the most probable value of the weights such as the posterior mean weights $\mathbf{w}_{MP} = \langle \mathbf{w} \rangle$ into equation 4.6 to

give the class posterior distribution

$$p(\mathbf{t}^*|\mathbf{\Phi}^*, \mathbf{w}_{MP}) = \sigma\left(\mathbf{w}_{MP}^T \mathbf{\Phi}^*\right). \tag{4.49}$$

The above predictive distribution is computed using $\mathbf{w}_{MP}$ as the sole representative of the true posterior distribution of the weights. This approach may not be very accurate since there may be regions in the input space where the ensemble is very uncertain about class memberships. On such regions, one will require the ensemble to produce probabilities close to 0.5. Class posterior probabilities computed with the most probable value of the weights may produce extreme, unrepresentative and overconfident results (MacKay, 1992) especially when unreliable classifiers are present.

MacKay Approximation

A more accurate result that takes into account uncertainties in the weights can be obtained by marginalizing the posterior distribution $p(\mathbf{t}^*|\mathbf{\Phi}^*, \mathbf{w})$ over the posterior distribution of the weights. That is, given a new data point $\mathbf{\Phi}^*$ and the training set $\mathcal{D}_l$, compute the predictive distribution

$$\begin{aligned} p(\mathbf{t}^* = 1|\mathbf{\Phi}^*, \mathcal{D}_l) &= \int p(\mathbf{t}^* = 1|\mathbf{\Phi}^*, \mathbf{w}) p(\mathbf{w}|\mathcal{D}_l)\, d\mathbf{w} \\ &\approx \int p(\mathbf{t}^* = 1|\mathbf{\Phi}^*, \mathbf{w}) q(\mathbf{w})\, d\mathbf{w} \\ &= \int \sigma\left(\mathbf{w}^T \mathbf{\Phi}^*\right) \mathcal{N}_{L+k}(\mathbf{w}|\mathbf{m}, \mathbf{\Sigma})\, d\mathbf{w}. \end{aligned} \tag{4.50}$$

The convolution of a logistic sigmoid function and a Gaussian cannot be computed analytically. However, an approximate solution can be obtained by approximating the sigmoid function $\sigma(\xi)$ by a probit function $\phi(\lambda\xi)$ for a suitable parameter λ such that the two functions have the same slope at the origin. This yields $\lambda = (\pi/8)^2$. Since the

convolution of a probit and a Gaussian is a probit, one has

$$\begin{aligned}\int \sigma(\xi)\mathcal{N}(\xi|\mu, s^2)\, d\xi &\approx \int \phi(\lambda\xi)\mathcal{N}(\xi|\mu, s^2)\, d\xi \\ &= \phi\left(\frac{\mu}{\sqrt{\lambda^{-2} + s^2}}\right) \\ &\approx \sigma\left(\frac{\mu}{\sqrt{1 + \pi s^2/8}}\right). \end{aligned} \tag{4.51}$$

This result can then be applied to equation 4.50 with

$$\mu = \langle \mathbf{w} \rangle^T \Phi^* \tag{4.52}$$

$$s^2 = \mathbf{\Phi}^{*T} \mathbf{\Sigma} \mathbf{\Phi}. \tag{4.53}$$

See MacKay (1992) for details.

Note that even though the VBC method is presented for classifiers with continuous outputs, it can also combine classifiers with discrete outputs. Its restriction to continuous output is based on the observations by Džeroski and Ženko (2004a) that using discrete outputs in stack generalization cannot capture the confidence and uncertainties of the base learners as class probabilities would.

4.5 Ensemble Method For Class Dependent Evidence Model

This section briefly presents the structure of the ensemble method with a class dependent evidence. The EM and VB inference methods previously described for class independent evidence model can be applied here with only minimal modifications.

From the graphical model of ensemble learning with the evidence depending on the true class membership shown in Figure 4.4(b), the joint distribution of the random variables can be written as

$$p(\hat{\mathbf{t}}, \mathbf{t}, \mathbf{e}, \mathbf{\Lambda}, \mathbf{w}) = p(\hat{\mathbf{t}}|\mathbf{t}, \mathbf{e}, \mathbf{\Lambda}, \mathbf{w})p(\mathbf{t}|\mathbf{e}, \mathbf{\Lambda}, \mathbf{w})$$

$$= p(\mathbf{t}|\hat{\mathbf{t}}, \mathbf{e}, \mathbf{\Lambda}, \mathbf{w})p(\hat{\mathbf{t}}|\mathbf{e}, \mathbf{\Lambda}, \mathbf{w}).$$

Using the independence assumptions of the graphical model leads to the relation

$$p(\mathbf{t}|\hat{\mathbf{t}}, \mathbf{e}, \mathbf{\Lambda}) \propto p(\hat{\mathbf{t}}|\mathbf{t}, \mathbf{e}, \mathbf{w})p(\mathbf{t}|\mathbf{e}, \mathbf{\Lambda}).$$

The unknown distribution in this expression is $p(\mathbf{t}|\mathbf{e}, \mathbf{\Lambda})$, however, it can be easily derived through Bayes theorem. Successive application of Bayes theorem gives

$$p(\mathbf{t}|\hat{\mathbf{t}}, \mathbf{e}, \mathbf{\Lambda}) \propto p(\hat{\mathbf{t}}|\mathbf{t}, \mathbf{e}, \mathbf{w})p(\mathbf{e}|\mathbf{t}, \mathbf{\Lambda})p(\mathbf{\Lambda}|\mathbf{t})p(\mathbf{t}) \tag{4.54}$$

Thus if $\hat{\mathbf{t}}_i$ is the i'th row of the matrix $\mathbf{Y}$, then the class posterior probability for each data point is given by

$$p(t_i|\hat{\mathbf{t}}_i, e_i, \mathbf{\Lambda}) \propto p(e_i|t_i, \mathbf{\Lambda})p(\mathbf{\Lambda}|\mathbf{t}_i)p(t_i) \prod_{j=1}^{L} p(\hat{t}_{ij}|t_i, e_i, \mathbf{w}_j). \tag{4.55}$$

Since the distribution of the evidence $\mathbf{e}$ and the precision $\mathbf{\Lambda}$ are known (see equations 4.4 and 4.11), inference can be done with the same EM or VB ensemble learning methods presented in this chapter. Comparing with equation 4.23 shows that the class dependent evidence model supply more information for the estimation of class posterior probabilities taking into account uncertainties in the evidence. This property may likely improve accuracy of the ensemble models.

4.6 Experiments

Numerical experiments are presented in this section to demonstrate the performance of the three ensemble methods introduced in this chapter compared to other ensemble methods. Specifically, the EM, VBD and VBC ensemble schemes are compared with three popular ensemble methods: Stack generalization with a logistic regression meta-learner (Stack), Mean Rule (M.Rule) and Majority Votes (M.Votes). The performance of the base classifiers are also reported. The performance of the classifiers are evaluated using five measures: Percent correctly classified (PCC), area under receiver operating characteristic curve (AUC), sensitivity, specificity and G-Mean. Except for the AUC, the other measures can be computed directly from the classification confusion matrix:

Table 4.1: Confusion Matrix

		Predicted Class		
		Positive	Negative	Total
True Class	Positive	a	b	$a+b$
	Negative	c	d	$c+d$
	Total	$a+c$	$b+d$	n

$$\text{sensitivity} = \frac{a}{a+b}$$

$$\text{specificity} = \frac{d}{c+d}$$

$$\text{G-Mean} = \sqrt{\text{sensitivity} * \text{specificity}}$$

$$\text{PCC} = \frac{a+d}{a+b+c+d}$$

AUC requires predicted probabilities and thus it is only reported for ensemble methods that generate probabilities. AUC is also not reported for the base classifiers.

4.6.1 Base Algorithms and Training

The Magic Gamma Telescope and Car evaluation datasets from the UCI Machine learning repository are used for the experiments. Magic dataset consist of 19020 observations with 10 features on the real continuous scale. The classification task represented by the dataset is to discriminate hadrons (background) from gamma rays (signal). This is an example of a classification problem where high specificity is required, since classifying a background event as signal is worse than classifying a signal event as background. The authors of the data set recommends ROC curves for comparing different classifiers with the following relevant false positive upper limits: $s = 0.01, 0.02, 0.05, 0.1, 0.2$. The Car dataset consist of 1728 observations with all categorical features (see section 5.8.1 for full description of the data). The Bayes active data and algorithm selection scheme with the Hausman specification test presented in chapter 3 is used to compare and rank the base learners based on a series of performance measures. The top k classifiers can then be selected for ensemble learning.

Eleven classification algorithms were chosen for ensemble learning: iterative re-weighted least squares extreme logistic regression (irls-elr) given by equation 2.10, linear discriminant analysis (lda), support vector machines (svm), Näive Bayes (nb), logistic regression (log), neural network (nn), decision trees (tree), extreme learning machine (elm) (see Huang et al. (2012)), least-squares extreme logistic regression (ls-elr) given by equation 2.9, Quinlan's C5.0 rule-based classifier (c50) and k-nearest neighbor (knn).

The datasets are randomly split into three parts: training, validation and testing. The training set is used to actively trained the base algorithms, ensemble combination weights are generated on the validation set and final evaluation is done on the test set. For active learning, the training set is randomly split into a small labeled set $\mathcal{D}_l$ and a large unlabeled pool $\mathcal{D}_u$. At each round of the active learning as illustrated in Algorithms 2 and 3,

informative points are queried from $\mathcal{D}_u$ and add to $\mathcal{D}_l$. The predictions probabilities from all classifiers on $\mathcal{D}_l$ is then analyzed by the Bayes decision marker (DM) as described in chapter 3. After active training the top k classifiers are selected for ensemble learning.

However, for the numerical experiments reported in this section, all base classifiers were used for ensemble learning. The experiment was repeated 50 times and the average of the five performance measures reported. The standard deviation of AUC is also reported. Because the k-nearest neighbor classifier requires the training set to be available at test time, the computational cost increases for the 50 runs and so the classifier was dropped.

4.6.2 Results

Results for the performance of the algorithm selection and ensemble methods on the two datasets are presented in this section.

4.6.2.1 Active Training and Ranking

Table 4.2 and 4.3 shows the performance of the base algorithms across several performance measures as computed by the DM during active data and algorithm selection step. The last column is the overall ranking of the classifiers. The results clearly show that a good number of the algorithms may perform very well on these datasets. A favorable advantage can be seen for irls-elr, elm, elr and tree for the magic dataset while irls-elr, elm, elr, log and c50 for the car dataset.

Surprisingly, decision trees performed very well on the magic dataset while its performance deteriorates on the car dataset. A similar performance trend was observed in the experiments reported in chapter 3. Neural network seems to not perform poor on these datasets. This is quite opposite to its performances in chapter 3.

Table 4.2: Active Training Performance on Magic Data

Performance Metric								pAUC at s					
Classifier	B.Risk	AUC	$AUCPR_{.5}$	IAUCPR	$Max.F1_{.5}$	Mean.Max.F1	R_α	0.01	0.02	0.05	0.1	0.2	Rank
irls-elr	0.21	0.72	0.76	0.72	0.72	0.70	0.56	0.01	0.03	0.07	0.15	0.29	**11**
lda	0.20	0.64	0.65	0.62	0.66	0.64	0.24	0.01	0.02	0.05	0.09	0.19	**7**
svm	0.16	0.57	0.57	0.55	0.64	0.61	0.09	0.01	0.01	0.04	0.07	0.14	**5**
nb	0.25	0.55	0.54	0.52	0.61	0.58	0.15	0.01	0.01	0.04	0.07	0.14	**4**
log	0.17	0.58	0.58	0.56	0.64	0.61	0.11	0.01	0.01	0.04	0.07	0.15	**5**
nn	0.03	0.57	0.58	0.56	0.67	0.63	0.03	0.01	0.01	0.03	0.06	0.13	2
tree	0.31	0.57	0.55	0.53	0.60	0.57	0.17	0.01	0.02	0.05	0.10	0.18	**8**
elm	0.21	0.68	0.70	0.66	0.69	0.66	0.32	0.01	0.02	0.06	0.11	0.22	**10**
ls-elr	0.20	0.66	0.68	0.65	0.68	0.65	0.29	0.01	0.02	0.05	0.10	0.21	**9**
c50	0.19	0.58	0.56	0.55	0.63	0.60	0.12	0.01	0.02	0.04	0.08	0.15	**4**
knn	0.62	0.51	0.37	0.37	0.43	0.40	0.27	0.01	0.01	0.03	0.06	0.12	1

Table 4.3: Active Training Performance on Car Data

Performance Metric								pAUC at s					
Classifier	B.Risk	AUC	$AUCPR_{.5}$	IAUCPR	$Max.F1_{.5}$	Mean.Max.F1	R_α	0.01	0.02	0.05	0.1	0.2	Rank
irls-elr	0.07	0.93	0.95	0.94	0.90	0.89	1.82	0.05	0.10	0.25	0.49	0.71	**11**
lda	0.13	0.80	0.84	0.80	0.77	0.75	0.61	0.02	0.03	0.08	0.16	0.33	**6**
svm	0.13	0.78	0.82	0.78	0.75	0.73	0.55	0.02	0.03	0.08	0.15	0.30	**6**
nb	0.16	0.59	0.60	0.58	0.66	0.63	0.05	0.01	0.01	0.03	0.07	0.13	2
log	0.15	0.81	0.84	0.81	0.78	0.76	0.87	0.02	0.04	0.09	0.19	0.38	**8**
nn	0.18	0.66	0.68	0.64	0.68	0.65	0.30	0.01	0.02	0.05	0.10	0.20	4
tree	0.91	0.46	0.23	0.25	0.22	0.19	1.10	0.01	0.01	0.03	0.05	0.08	1
elm	0.11	0.85	0.89	0.85	0.82	0.80	0.83	0.02	0.04	0.11	0.22	0.44	**10**
ls-elr	0.12	0.82	0.86	0.82	0.79	0.76	0.67	0.02	0.04	0.09	0.18	0.36	**9**
c50	0.34	0.68	0.67	0.64	0.66	0.63	0.81	0.03	0.06	0.14	0.24	0.34	**10**
knn	0.47	0.56	0.51	0.49	0.55	0.53	0.54	0.01	0.02	0.05	0.09	0.19	3

4.6.2.2 Ensemble Performance

Table 4.4 , 4.5, 4.7, and 4.8 shows the average performances over 50 different experiments for the six ensemble methods and the base classifiers actively trained on the magic and car datasets. The following observations can be easily inferred from the results:

- All algorithms except perhaps EM and Näive Bayes performed excellent on the validation and test set respectively.
- VBC outperformed all other algorithms on the Magic dataset. This is followed by Stack, VBD and M.Rule. Table 4.6 further compares these four ensemble methods in terms of the two sample student-t test that one sample mean AUC is greater than the

other. Clearly, VBC is statistically better than Stack (p.value = 0.01) and M.Rule (p-value $\ll 10^{-16}$). VBD and S.Scores are not statistically different on this dataset.

- Stack outperformed all other algorithms on the Car dataset. Table 4.9 shows that Stack is statistically better than VBD and VBC. On the other hand both VBD and VBC are statistically better than M.Rule. It should be recalled that the car dataset has all categorical attributes which violates the normal assumption of the evidence model. Hence the performance of VBD and VBC may not be optimal.

- The performance of EM ensemble method was very poor. Possible reasons for this poor results can be given as: first EM does not make use of the information provided by the true class label during training, second the algorithm does not penalize complex models and so it is very prone to overfitting. And finally, as with VBD, its performance depends on the quality of the predicted classes and on the classification evidence. However, unlike VBD, poor or unreliable classifiers and evidence scores are not "pruned" through ARD priors from the final decision. The results for EM indicates that, even with supplementary hidden information, a good combination method is very important for accurate results.

Table 4.4: Performance Measures on Magic Validation Set

Model	PCC	AUC	AUC.sd	sensitivity	specificity	G-mean
VBD	0.81	0.88	0.01	0.72	0.86	0.79
VBC	0.84	0.91	0.00	0.81	0.85	0.83
EM	0.73	0.74	0.01	0.37	0.92	0.58
Stack	0.83	0.90	0.00	0.81	0.84	0.83
M.Rule	0.81	0.88	0.01	0.79	0.82	0.80
M.Votes	0.80			0.61	0.90	0.74
irls-elr	0.77			0.50	0.91	0.68
lda	0.79			0.58	0.90	0.72
svm	0.80			0.60	0.92	0.74
nb	0.64			0.62	0.65	0.61
log	0.79			0.62	0.88	0.74
nn	0.77			0.77	0.76	0.76
tree	0.80			0.67	0.87	0.76
elm	0.78			0.56	0.91	0.71
ls-elr	0.79			0.57	0.90	0.72
c50	0.81			0.70	0.87	0.78

Table 4.5: Performance Measures on Magic Test Set

Model	PCC	AUC	AUC.sd	sensitivity	specificity	G-mean
VBD	0.80	0.89	0.01	0.72	0.85	0.78
VBC	0.82	0.90	0.00	0.80	0.84	0.82
EM	0.68	0.72	0.01	0.11	0.99	0.33
Stack	0.82	0.90	0.00	0.80	0.83	0.81
M.Rule	0.80	0.88	0.01	0.78	0.82	0.80
M.Votes	0.80			0.61	0.90	0.74
irls-elr	0.77			0.51	0.91	0.68
lda	0.79			0.58	0.90	0.72
svm	0.80			0.60	0.92	0.74
nb	0.66			0.58	0.70	0.62
log	0.79			0.62	0.88	0.74
nn	0.78			0.76	0.79	0.77
tree	0.81			0.67	0.88	0.77
elm	0.78			0.56	0.91	0.71
ls-elr	0.79			0.57	0.90	0.72
c50	0.80			0.69	0.86	0.77

Table 4.6: p-values: Pr(VBD or VBC > Stack or M.Rule) on Magic dataset

	VBC	VBD
Stack	0.01	.88
M.Rule	0.00	0.44

Table 4.7: Performance Measures on Car Validation Set

Model	PCC	AUC	AUC.sd	sensitivity	specificity	G-mean
VBD	0.95	0.99	0.00	0.97	0.94	0.95
VBC	0.98	0.99	0.00	0.98	0.97	0.98
EM	0.76	0.90	0.02	0.69	0.78	0.73
Stack	0.98	1.00	0.00	0.99	0.97	0.98
M.Rule	0.94	0.98	0.01	0.98	0.93	0.96
M.Vots	0.92			0.96	0.91	0.93
irls-elr	0.81			0.44	0.94	0.64
lda	0.90			0.90	0.89	0.90
svm	0.93			0.97	0.92	0.94
nb	0.68			1.00	0.56	0.75
log	0.90			0.88	0.91	0.90
nn	0.97			0.96	0.97	0.97
tree	0.90			0.88	0.90	0.89
elm	0.88			0.78	0.92	0.84
ls-elr	0.89			0.87	0.91	0.89
c50	0.93			0.91	0.94	0.92

Table 4.8: Performance Measures on Car Test Set

Model	PCC	AUC	AUC.sd	sensitivity	specificity	G-mean
VBD	0.93	0.98	0.01	0.94	0.93	0.94
VBC	0.97	0.99	0.00	0.96	0.97	0.97
EM	0.73	0.90	0.02	0.00	1.00	0.00
Stack	0.97	0.99	0.00	0.97	0.97	0.97
M.Rule	0.94	0.97	0.01	0.97	0.93	0.95
M.Votes	0.92			0.95	0.91	0.93
irls-elr	0.80			0.42	0.94	0.62
lda	0.89			0.89	0.89	0.89
svm	0.93			0.97	0.92	0.94
nb	0.68			1.00	0.57	0.75
log	0.90			0.87	0.91	0.89
nn	0.97			0.96	0.97	0.97
tree	0.89			0.87	0.90	0.89
elm	0.87			0.75	0.92	0.83
ls-elr	0.89			0.85	0.90	0.88
c50	0.93			0.90	0.94	0.92

Table 4.9: Car dataset p-values: Pr(VBD or VBC > Stack or M.Rule) on Car dataset

	VDC	VBD
Stack	0.80	1.00
M.Rule	0.00	0.00

4.7 Summary and Future Work

This chapter introduced an optimal framework for integrating a set of machine learning classification models. A VBFACSP method was proposed for inference of the classification evidence to be used in the integration step. Three methods for integrating the classification models are then presented. The first method uses the EM algorithm to combine classifiers with discrete outputs assuming the true classes are hidden variables. The second method re-implements the VB logistic regression to combine classifiers with discrete outputs. The third method applies standard VB logistic regression to combine classifiers with continuous outputs.

Some classifiers can only generate discrete or class labels. While it may be possible to transform the discrete outputs to probabilities, the quality of the result may be very poor. The EM and VBD ensemble methods presented in this chapter can be advantageously applied to combine such classifiers to produce quality class probabilities. A drawback with these two ensemble method is that they rely on good estimates of the evidence e. Poor values of e may result in a poor ensemble. The VBC ensemble on the otherhand can operate on both continuous and discrete outputs. Because of the ADR priors, the dependence of this method on the evidence is not critical. If some dimensions of e are irrelevant, the ARD simply turn them off.

Future directions of the work presented in this chapter include a full implementation of the VBFACSP method and the corresponding ensemble methods for class dependent evidence model. It will be interesting to compare the performance of ensemble methods with class dependent and independent evidence.

Chapter 5: Collective Machine Learning

This chapter introduces a framework for learning large-scale distributed datasets. The proposed framework builds efficient, scalable parallel and distributed machine learning models from homogeneous or heterogeneous distributed data sites. Each data site (modeled as an agent) is tightly integrated with a Bayes active data and algorithm selection scheme, a classification evidence model and an optimal Bayesian multiclassifier integration scheme as described in chapters 3 and 4 respectively. During learning, performance of the agents can be improved by sharing

- classification evidence;
- local classification models and/or classification evidence models;
- local ensemble models.

With these learning properties, the system can be described as collective or cooperative machine learning teams. When a global solution is required, a designated agent can combine the local ensemble decisions of the agents through the same ensemble learning scheme embedded within the agents. Two parallel programming frameworks are explored for collective learning: MapReduce and traditional High Performance Computing (HPC) with Message Passing Interface (MPI). Because the Hadoop-MapReduce technology is relatively new, most of the implementation details are focused on it.

5.1 Introduction

In the past, one of the bottlenecks preventing the development of more intelligent systems was information scarcity. Today, information has gone from scarce to superabundant. The

world today contains an unimaginably vast amount of information which is getting ever vaster more and more rapidly. Merely keeping up with the flood is hard and the ability to collect and store the bits that might be useful is proving to surpass traditional database management tools. Further, analyzing the data to identify and extract relevant information is even more challenging. Even so, the "data deluge" is already transforming the way research is conducted, business, government and every day life. It has spawn a large industry in "Big Data" Analytics.

The MapReduce framework (Dean and Ghemawat, 2008) and its variants are recent tools that have been developed to help solve the Big Data problem. MapReduce allows application developers to effectively run parallel applications on a cluster of machines using high-level interfaces which hides low-level details such as data locality, load balancing, fault tolerant etc. The open source implementation of MapReduce mainly Hadoop (http://hadoop.apache.org/) is one of such interfaces and has become the de facto standard for performing large-scale processing tasks that require single pass over the dataset.

Big Data not only changed the tools traditionally used for collection and storage of data, it also changed the algorithms traditionally used for predictive analytics. Further, it entirely changed the way of thinking about knowledge extraction and interpretation. The Manhout, Graphlab and many others (Low et al., 2010; Kraska et al., 2013; Chu et al., 2007; Ngufor and Wojtusiak, 2013, 2014b) are some recently proposed scalable machine learning algorithms for efficient processing of large and distributed datasets.

These parallel and distributed machine learning tasks have help derived accurate predictions of various kinds; collaborative filtering, clustering and classification. Some application examples include large scale signal detection of adverse drug events from millions of spontaneous reports (Fan et al., 2010), discovery of unknown drug interactions in millions of electronic health records, learning online users preferences for product recommendations, credit card fraud detection, market segmentation in other to learn customers habits, providing incentives and preventing churn. These examples illustrate that, the field of parallel and distributed machine learning is of considerable importance, but also it is relatively

young and growing. The work presented in this chapter can be considered as an attempt to contribute to that growth.

5.2 Big Data Analytics

The term Big Data refers to voluminous datasets that cannot be stored and analyzed using conventional database management systems. The data most often need to be accessed and processed in real time to extract information. How useful such information is to scientific or business decisions depends on the nature of the data. The nature of Big Data is conventionally described in terms of its volume, velocity and variety. Analytics with Big Data or simply Big Data Analytics is commonly carried out with tools from statistics, machine learning, natural language processing, visualization, and data mining. The combination of all these approaches have given birth to a new inter-disciplinary field called "data science".

The recent interest in research in Big Data has been due to the increasing ability to collect and store large datasets, the ability to tap its vast potentials by the research community, business, government and the society. A few examples illustrating how the use of Big Data can create value include;

- Big Data can help unlock significant value by making information transparent and usable at a much faster and higher frequency.
- As organizations and industries amass more and more data, they can collect more accurate and detailed performance information on various items such as product inventories, drug interactions, disease patterns etc.
- Big Data is increasingly use in research and industry to perform controlled experiments so as to develop and test optimal solutions to problems before implementation.
- Big Data allows ever-narrower segmentation of customers making companies to precisely tailor products and services.

Many other examples of the application of Big Data can be found in Manyika et al. (2011). These examples reveal the need for Big Data analytics which in turn begs for the development and use of more efficient machine learning and statistical methods that integrates parallel processing of data. Conventional machine learning applications will need to be redesigned or scale-up to meet the parallel and distributed nature of current and future datasets. The MapReduce and its variants are promising tools in the industry.

5.3 Hadoop Distributed File System, MapReduce and MPI

The fast emerging field of data science in which massive data need to be stored processed, analyzed, visualized and used brings new computing challenges. Consequently, high performance computing (HPC) is becoming increasingly important in many data intensive areas such as Biotechnology, geology, nuclear physics, web search, network video and multimedia streaming, medical imaging and diagnosis, pharmaceutical design, mathematics, defense, financial and economic modeling just to name a few. HPC typically involves distribution of work across a cluster of machines which may or may not have access to a shared file system hosted on a storage area network. Traditionally, parallelization in HPC has been implemented with the Message Passing Interface (MPI). Recently, the implementation is fast shifting towards the popular Hadoop's MapReduce which is currently the large-scale data analysis tool of choice.

Hadoop is an Appache project which develops open source software for reliable and scalable computing. Inspired by technologies created inside Google, Appache Hadoop was created in 2005 with two primary components: the Hadoop Distributed File System (HDFS) for distributed storage of large-scale datasets and MapReduce parallel processing engine to support distributed processing of data workloads on HDFS. In 2012, Hadoop underwent a complete overhaul resulting in Hadoop YARN or Yet Another Resource Negotiator for Hadoop, which is a more general usable framework that supports not only MapReduce, but other distributed processing models. The Hadoop implementation in this thesis is however

based on the pre-YARN Hadoop architecture.

5.3.1 HDFS

The HDFS is a fault tolerant and self healing distributed file system designed to turn low cost clusters of commodity hardware into a massively scalable pool of storage. HDFS was developed specifically for large-scale data processing applications where scalability, flexibility and high throughput are critical. It accepts data in any format regardless of the schema and optimizes for high bandwidth streaming. Thus HDFS is designed more for batch processing rather than interactive or iterative processing.

An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are several DataNodes, one per node or machine in the cluster, which manage storage attached to the node that they run. For failure resilient purposes, it has a secondary NameNode which replicates the data of NameNode at regular intervals. Internally, HDFS is a block structured file system, that is individual files are split into one or more blocks of fixed sizes (default is 64 MB) and these blocks are stored in a set of DataNodes. The NameNode is responsible for operations such as opening, closing and renaming files. It also determines the mapping of blocks to DataNodes. The DataNodes on the other hand are responsible for read and writing data to the HDFS. DataNodes also performs block creation, deletion and replication upon instruction from the NameNode. Because all blocks of a single file may not be stored on the same DataNode, a fault in any one of the DataNode may render a file unavailable. HDFS overcome this by replicating each block across a number of DataNodes (three by default). The number of copies of a file is called its replication factor. The metadata containing information about file partitioning to different DataNodes is stored in the NameNode.

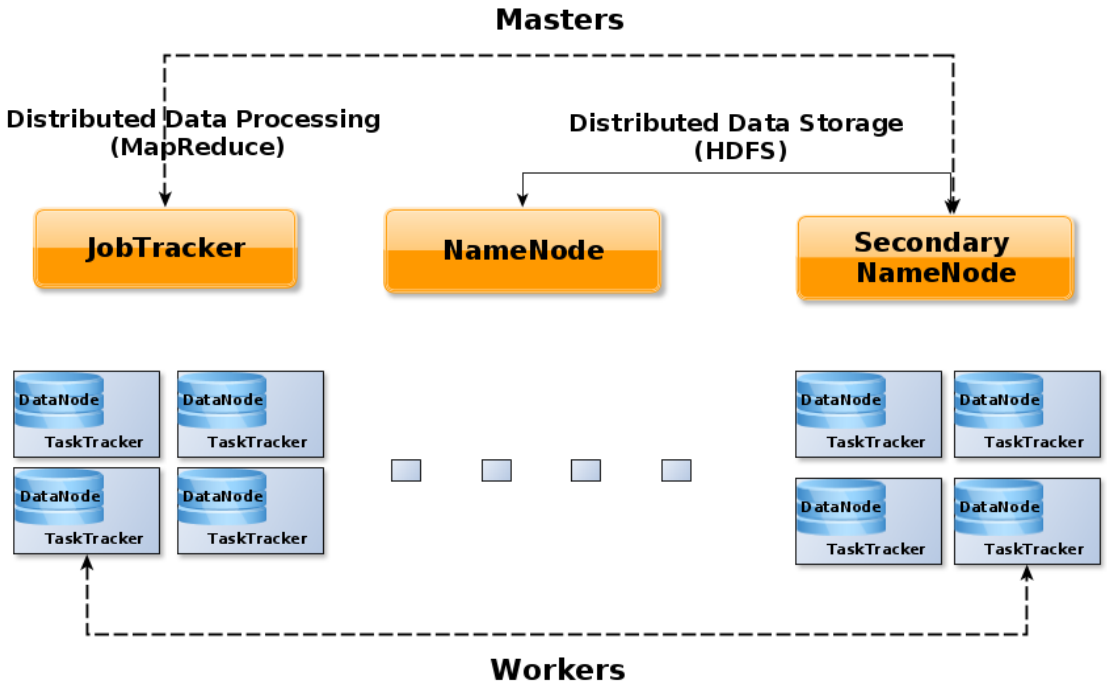


Figure 5.1: Hadoop-MapReduce Architecture

5.3.2 MapReduce

MapReduce is a massively scalable, parallel processing framework that works in tandem with HDFS. With Hadoop-MapReduce, computation is executed at the location of the data rather, than moving data to the computation, i.e, data storage and computation coexist on the same physical node.

The MapReduce framework consist of a single master JobTracker and one or more TaskTrackers per node. The JobTracker is responsible for managing the TaskTrackers on the worker nodes, tracking resource consumption and availability, scheduling individual jobs task, tracking progress and providing fault tolerance through heartbeats (periodic messages). The TaskTrackers communicates through heartbeats to the JobTracker. If the JobTracker does not receive a heartbeat form a TaskTracker, it assumes it has failed and

takes appropriate action (such as restarting the map task on another node). Figure 5.3 illustrates the HDFS and MapReduce architecture.

In distributed data processing with MapReduce, a MapReduce job splits the input dataset into independent chunks which are process by a Map and Reduce task in a completely parallel manner. The splits can be made up of one or more HDFS data blocks. MapReduce operates entirely on a $\langle key\ ,\ value \rangle$ pairs, that is, the input to the job is a set of $\langle key\ ,\ value \rangle$ pairs and produces a set of $\langle key\ ,\ value \rangle$ pairs as the output. The map function takes an input $\langle key_1\ ,\ value_1 \rangle$ pairs and produces a set of intermediate $\langle key_2\ ,\ value_2 \rangle$ pairs. The Map output can have multiple values for the same key. The reduce function then operates on the intermediate *key* and set of *values* associated with that key to produce a smaller set of output values.

1. map: $\langle key_1\ ,\ value_1 \rangle \longrightarrow \text{list}\langle key_2\ ,\ value_2 \rangle$

2. reduce: $\langle key_2\ ,\ \text{list}\{value_2\} \rangle \longrightarrow \text{list}\{value_3\}$

The MapReduce framework hides all the complexity of parallelization, data distribution, load balancing, task scheduling, monitoring and fault-tolerance from the client. All that is required is for the client to specify the map and reduce functions. Also, although the Hadoop framework is implemented in Java, MapReduce applications need not be written in Java.

5.3.3 SPMD with Message Passing Parallel Programming Model

MPI is a message passing library standard used by industry and academia for parallel computing on shared, distributed or hybrid (distributed-shared) memory architectures. The Single Program Multiple Data (SPMD) with message passing is probably the most commonly used parallel programming model for multi-node clusters. In this model, all tasks execute a copy of the same program simultaneously and exchange data through communications by sending and receiving messages. Data transfer between processes usually requires

cooperative operation by each process in the "communicator". For example, a send operation must have a matching receive operation. MPI libraries provide point to point and collective communication routines. Common routines are MPI_Send, MPI_Recv, MPI_Isend, MPI_Irecv and MPI_Sendrecv, MPI_Bcast, MPI_Scatter,MPI_Allreduce, MPI_Allgather and MPI_Alltoall.

MPI programs are traditionally written in lower level languages like C, C++, or Fortran. Recently, wrapper programs have been written to enable simplified interfaces to MPI for high level languages like R, Python and Java. For example, the "Programming with Big Data in R" project (pbdR) (Ostrouchov et al., 2012) used in this thesis is a SPMD model that enables high-level distributed data parallelism in R. With pbdR packages, R users can easily utilize large HPC platforms with thousands of cores while maintaining R original syntax convenience.

5.4 Parallel and Distributed Machine Learning

The data deluge has open new ways for application of machine learning and statistical methods. Automatic methods for extracting value from such Big Data is a growing need and machine learning provides the necessary techniques to enable users to extract underlying structure and make prediction from large datasets. Big Data also raises new problems and challenges regarding the scalability, efficiency, accuracy, and computational time of learning algorithms. Most existing algorithms operates on the assumption that all training data can fit on a centralized memory. Consequently, they do not scale when confronted with very large and possibly distributed datasets. This has led to a significant amount of research in scaling methods, that is, designing algorithms to efficiently learn from very large and distributed datasets. Two general approaches can be identified in this endeavour: *scaling down* and *scaling up* (Bekkerman et al., 2011; Kargupta and Chan, 2000).

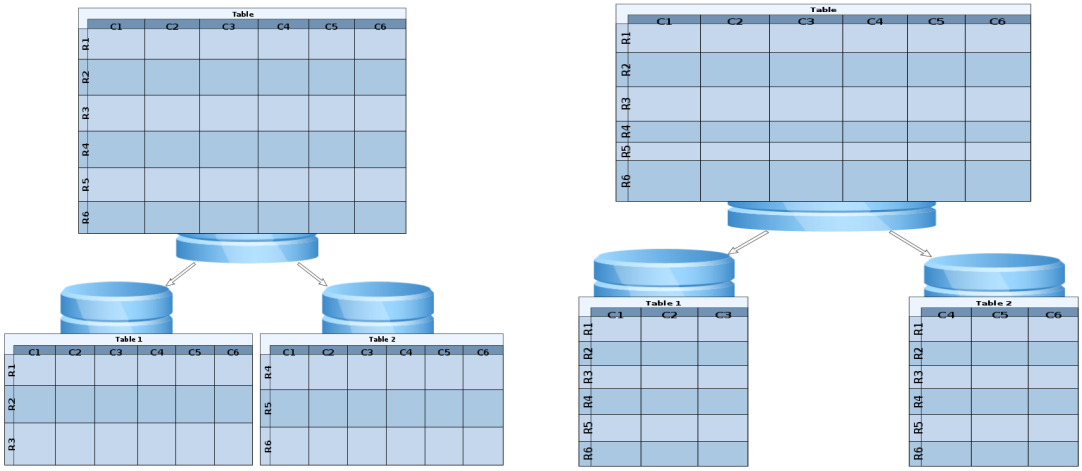
Following the latter approach, existing machine learning algorithms are modified or new ones develop with the property of being able to learn from very large datasets. There has

been a rapid rise in research methods for scaling up machine learning algorithms. This research has been aided in part by the fact that some machine learning algorithms can be readily deployed in parallel. For example Chu et al. (2007) showed that ten commonly used machine learning algorithms (logistic regression, näive Bayes, k-means clustering, support vector machines etc) can be easily written as MapReduce programs on multi-core machines.

The other part can be attributed to the rapid evolution of hardware and programming architectures (Bekkerman et al., 2011). These new technologies are highly optimized for distributed computing in the sense that they are parallel efficient, reliable, fault tolerant and scalable. The Hadoop-MapReduce framework for example has been successfully applied to a broad range of real world machine learning applications. The Apache Mahout project (Owen et al., 2011) for example contains implementation of some standard machine learning algorithms on the Hadoop-MapReduce framework. RHadoop (`https://github.com/RevolutionAnalytics/RHadoop/wiki`) is a collection of packages that allows R programmers to manage and perform statistical analysis through MapReduce on a Hadoop cluster. Other successful scalable implementations of machine learning algorithms include Google's cloud-based machine learning API (`https://developers.google.com/prediction/`), GrapLab (Low et al., 2010), MLbase (Kraska et al., 2013), Vowpal Wabbit (Langford et al., 2007) and many others. Lastly, in the course of writing this thesis, some major contributions were also made in the line of developing accurate, fast and parallel efficient machine learning algorithms (Ngufor and Wojtusiak, 2013, 2014b) from which some of the ideas form an integral part of this thesis.

5.5 Distributed Machine Learning Systems: An Agent-based Approach

Parallel and distributed learning has been recognized as an optimal approach for effective learning of large-scale datasets. As the examples in the previous section indicated, a great



(a) Horizontal Partitioning (b) Vertical Partitioning

Figure 5.2: Homogeneous and Heterogeneous Data Sites

number of learning strategies have been proposed. Most of these distributed algorithms are based on the basic principle of ensemble learning. Although the various methods may vary, however the basic idea is to build local models on each data site and to combine the outputs of the local models in some way. The different data sites may be homogeneous or heterogeneous. Homogeneous data sites contains data for exactly the same set of features but the observations may be different. This corresponds to a horizontal partitioning of the distributed data. Heterogeneous data sites on the other hand, contains only a subset of the features for the same observations. This corresponds to a vertical partitioning of the distributed data (see Figure 5.2).

Knowledge extraction for large distributed databases has been traditionally done by the use collaborative systems. These systems are often designed following the agent based modeling approach popular in the field of artificial intelligence. The agents are basically data mining tools which can interacts with each other and with the environment to solve a particular task. Among the most successful implementation of cooperative learning systems include PADMA, Papyrus and JAM.

PADMA (PArallel Data Mining Agents) (Kargupta et al., 1997), is a parallel/distributed data mining system that employs software agents for local data accessing, analysis and visualization. The framework is mainly based on homogeneous distributed databases. Local models (clusters) are generated at the local sites and a centralized site is designed to integrate the local models. Papyrus (Bailey et al., 1999) is a Java-based system wide area distributed data mining framework over clusters of heterogeneous data sites. It employs mobile data mining agents to move data, intermediate results and models between clusters to either perform local computation or from local to a central cluster for final computation. Papyrus supports various methods for combining and exchanging local models and metadata required to describe them using a special makeup language. JAM (Java Agents for Meta-learning) (Prodromidis et al., 2000) is equally a Java-based meta-learning system for large-scale distributed data mining. JAM consist of two types of agents: a learning agent and a meta-learning agent. Each learning agent trains a set of machine learning classification models and sends the output to a meta-learning agent for combination. The collective machine learning system presented in this section draws some motivations from JAM. Other agent systems with data mining/machine learning capabilities can be found in the following references (Bui and Lee, 1999; Tozicka et al., 2007; Kargupta et al., 1999; Wojtusiak et al., 2012; Zhang et al., 2005; Weiß, 1998).

A major drawback with most of these approaches is that they do not guarantee properties like scalability, fault-tolerance/failure-resilient, load balancing, task monitoring and dynamic addition of agents as offered by the Hadoop framework. For example, in a multi-agent system involving hundreds of millions of agents such as modeling human decisions, sensors, city traffic, fish schools, etc. the running time required to obtain acceptable solution can take several hours or days. If any of the agents fails during run time, the whole process need to be restarted.

Another drawback with most traditional agent-based distributed data mining and machine learning systems which sometimes limits their applicability is the level of programming involved. Most of them typically offer very complex programming abstractions which are

difficult to program by domain scientists. This has led many organizations to often avoid to build multi-agent based solutions even when there is clear need for one because of investment required and the perceived complexity of agent programming. On the other hand, Hadoop-MapReduce framework hides all these complex programming constructs from the user, in addition it offers the Hadoop Streaming and Hadoop Pipes API allowing non Java applications to use the framework. This greatly opens the possibility to employ many mathematical and statistical libraries which are limited in Java but very extensive in other programming languages like R, Python, C and C++.

5.6 A Large-Scale Cooperative Machine Learning Framework

This section presents two approaches for integrating multiple machine learning models generated from distributed data sites modeled as agents on the Hadoop or HPC platforms. The data sites may be homogeneous i.e each site contains data for exactly the same feature set or heterogeneous i.e each site may contain dataset with different feature vectors. Each agent in the system is endowed with a set of machine learning algorithms, a classification evidence algorithm and can cooperate with each other through MapReduce or MPI.

To simplify the presentation, a brief overview of how agents are represented and trained in MapReduce is first presented. The MapReduce and MPI implementations for homogeneous and heterogeneous data sites with their respective assumptions are then presented separately.

5.6.1 Agent Representation and Training with MapReduce

A data site may consist of one or more databases stored on the same or on different machines. A machine corresponds to a node in Hadoop. Hadoop splits the input file to a MapReduce job into fixed-sizes (default equal to block size) called *splits* and assigned a map task for each split. The splits are just references to the location (host names of the machines) of the actual data. MapReduce uses this information to try to execute the map task on the

node where the actual data resides in HDFS. This configuration can be changed to allow a different input size splits. Each data site or agent is modeled as a single input file. Since they can be multiple input files in a given node, therefore each of these files represent a separate agent. This specification accommodates for homogeneous or heterogeneous agents i.e all mappers operating on the same file contains the same type of features.

Each agent has two unique properties: a unique $identifier$ which is assigned by Hadoop and stored in the NameNode and the *type* of agent provided by the user. The type of an agent is a file that is broadcasted to all agents and contains information on the number and type of features the agent has. During communication, models or variables can be shared between agents. Thus, an agent need to know the type of model or variables it is receiving so as to be compatible with its learning theory. As will be seen below, the agent type is only required for heterogeneous agents.

To avoid overfitting, each agent's input data is split into a training and validation set. The training set is used to select and train the base classifiers while the validation set is use to predict class labels for ensemble learning. Selection of the base classifiers can be (optionally) done using the the Bayes active data and algorithm selection method described in section 3. Recall that in the selection process, the base algorithms are also trained on the training set so no further training is required. The classification evidence required for integration can be computed in two ways. The easiest approach is to generate the evidence model on the validation set and predict on the test set. The second approach which may be more accurate and robust is to use information from the active learning step to build the evidence model. Recall that during active training of the base classifiers, each classifier selects the data points it considers most informative for learning. The informative data points from all selected classifiers can then be aggregated to build the evidence model. The model is then used to predict the classification evidence on the validation and test set respectively.

5.6.2 Homogeneous Agents

Homogeneous agents contains the same number and type of features i.e it is equivalent to a horizontal partitioning of the dataset. This is the most common type of distributed databases and many mult-agent based distributed machine learning applications in the literature are based on horizontal partitioned datasets (Prodromidis et al., 2000; Kargupta et al., 1997; Bailey et al., 1999). Learning and programming design for homogeneous agents is very straight forward involving very little or no communication between agents. Unless some agents absolutely need to communicate, the design can be written as a single-pass MapReduce job.

A situation where communication may be warranted is when agents select different algorithms during the active learning step. For example, consider a system with two agents **A** and **B**. **A** selects and trains classifiers $\{h_1, h_2, h_3\}$ and generates the ensemble $h_A = h_1 + h_2 + h_3$. **B** selects $\{h_1, h_4, h_5\}$ and generates the ensemble $h_B = h_1 + h_4 + h_5$. The global ensemble model is then constructed as $h = h_A + h_B$. At application time, the global model need to be applied to any test data available at sites **A** or **B**. However, to successfully carry out this operation either **A** needs to communicate classifiers h_2 and h_3 to **B**, and likewise **B** communicates h_4 and h_5 to **A**. This is however not a major problem as it can be easily resolved by assigning zero weights to the non-selected classifiers in each agent. This is actually what the Variational Bayesian (VB) integration scheme described in chapter 4 section 4.4.2 does. Recall that this scheme does not require algorithm selection since it performs automatic relevance determination (ARD). ARD practically sends the weights of irrelevant classifiers to zero. Based on this fact, the algorithm for homogeneous agents in this chapter assumes no communication.

5.6.2.1 Training

With no other form of communication between agents, the algorithm for training homogeneous agents is implemented as a single-pass MapReduce job. Each map function operating

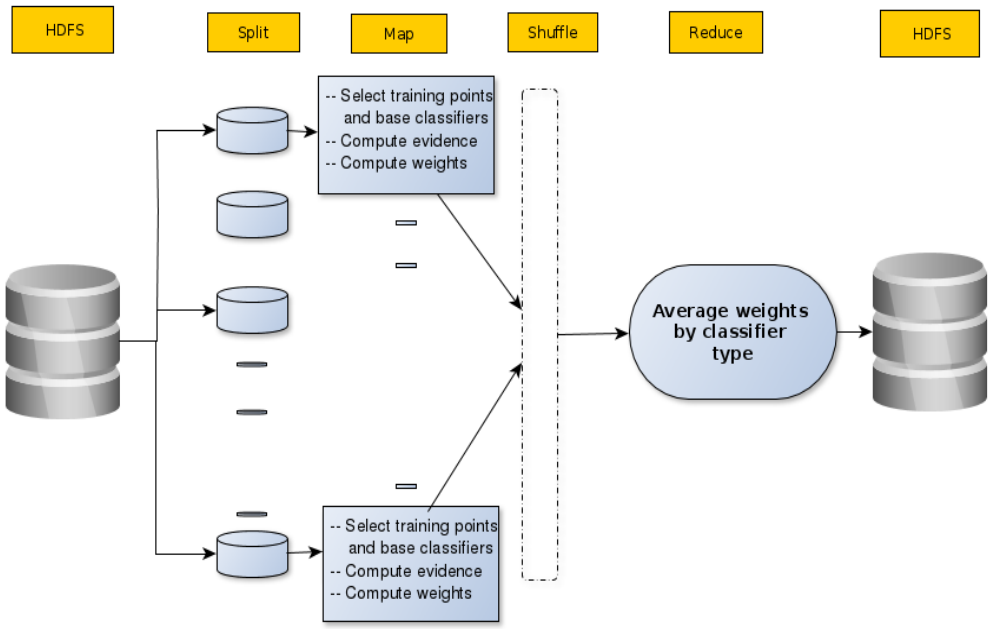


Figure 5.3: Training with MapReduce

on a block of the input file selects and trains a set of base algorithms. It also uses the training (or validation) set to generate parameters of the evidence model. The classification evidence and predictions of the classifiers on the validation set is used to generate combination weights for each classifier. This constitutes the local ensemble model for each agent.

For each base classifier type selected, the reducer simply computes the average of the combination weights. Since prediction using the Mackay Approximation (MacKay, 1992) requires the covariance matrix of the combination weights (see equation 4.53), the single-pass covariance matrix formula described in chapter 2 section 2.3.2 can be used to combine the covariance matrix of the weights.

The computed parameters constitutes the global ensemble model to be used for prediction on new cases and can be stored locally or on HDFS. The overall training process is illustrated in Figure 5.3.

5.6.2.2 Prediction

Prediction is equally carried out as a single-pass MapReduce job. When new test data is available at any of the sites, each agent uses the global ensemble model to predict the output of the test set. Note that if the agent is more confident in its local ensemble model, it may decide to use it instead of the global model.

5.6.3 Heterogeneous Agents

Heterogeneous agents represents the most general case of distributed machine learning task that has not been adequately studied in the literature. A few authors have attempted to address this very important topic with varying success (Kargupta et al., 1999; Provost and Buchanan, 1995). As demonstrated in Kargupta et al. (1999), naive approaches to distributed data analysis in heterogeneous environments may face ambiguous situations and may lead to incorrect global solutions. The challenge with heterogeneous data sites is that each dataset only has a subset of the feature space and models generated with with such "incomplete" data may not be representative of the global model i.e the models are not exact. However, some algorithms like decision trees and rules based learning are capable of generating exact models on such datasets. Thus it is not surprising that most existing studies on distributed machine learning with heterogeneous datasets are based on decision trees (Kargupta et al., 1999; Ye et al., 2009). The authors in Kargupta et al. (1999) proposed to used a set of orthogonal basis functions to generate local models which guarantees better global models with minimal communication.

This section offers an alternative solution to the problem of learning from heterogeneous database that makes use of the classification evidence to pull bits of information from the various data sites together to generate more accurate local models and hence more accurate global model. With this technique, the models generated can be considered approximately exact. Since the dimension of the classification evidence can be far less than the dimension of the dataset, communication cost can be maintained at a low and acceptable level.

Consider a collection of M heterogeneous datasets $\mathcal{D}_1 \in \mathbb{R}^{n \times p_1}, \ldots, \mathcal{D}_M \in \mathbb{R}^{n \times p_M}$, the evidence model represented by equation 4.2 can be seen in the context of factor analysis as the problem of finding $K \ll \sum_{m=1}^{M} p_m = p$ factors that describes the collection and in particular the dependencies between the datasets or groups instead of the individual variables. The solution to this problem includes the standard factor analysis as a special case. This problem was recently introduced in Virtanen et al. (2011) and coined Group Factor Analysis (GFA). An appealing property of GFA is that each factor provide weights over the datasets in a sparse manner. Thus, one can analyze the factors just as in traditional factor analysis. For example, with ARD priors, factors (groups) in GFA become sparse in the sense that elements corresponding to some subsets of the collection becomes zero indicating such datasets are irrelevant. A brief discussion of GFA is presented next and some simplifying assumptions are made to enable the use of equation 4.2 to model GFA for distributed datasets.

5.6.3.1 Group Factor Analysis

Given a collection $\mathcal{D}_1 \in \mathbb{R}^{n \times p_1}, \ldots, \mathcal{D}_M \in \mathbb{R}^{n \times p_M}$ of datasets with n co-occurring observations, the problem of GFA is to find a set of $K \ll p = \sum_{m=1}^{M} p_m$ factors that describes the joint data $\mathcal{Y} = [\mathcal{D}_1, \ldots, \mathcal{D}_M]$, where $\mathcal{Y}$ is a concatenation of the datasets $\mathcal{D}_m$ (Virtanen et al., 2011). The model can be written as

$$\mathcal{Y} = \mathcal{Z}\mathcal{W}^T + \mathcal{E} \tag{5.1}$$

where $\mathcal{W} \in \mathbb{R}^{p \times K}$ is the weight or loading matrix, $\mathcal{Z}$ is the factors and $\mathcal{E}$ is a zero mean Gaussian noise with diagonal covariance. GFA model can be solved by a simple extension of the VB evidence model presented in Algorithm 4.

One main advantage of GFA model is that, it is essentially a Bayesian factor analysis model with group-wise sparsity, where the groups corresponds to the datasets $\mathcal{D}_m$ instead

of variables. Another advantage is that it enables the analysis of a completely new kind of problem of which one of them is the problem studied in this section. That is, by introducing factors in GFA as classification evidence into the problem of integrating heterogeneous agents, the problems associated with learning from heterogeneous datasets with incomplete feature space can be minimize or eliminated since the evidence offers a higher level of information about the whole distributed datasets. A motivating example of the application of GFA taken from Virtanen et al. (2011) is the case where the authors studied drug responses by modeling four different datasets. Three datasets contained gene expression measurements of responses of different cell lines and one dataset contained chemical descriptors of the drugs. The authors did a joint analysis of the four datasets using GFA and obtained indicators on which drug descriptors were predictive of responses in a specific disease.

Unfortunately, with distributed data, GFA cannot be directly implemented without incurring some significant cost in communication. Consequently, in this work, it is assumed that the factors in GFA can be computed for each data site and distributed. That is, $\mathcal{Z}$ is a collection of M distributed datasets $\mathcal{Z}_1 \in \mathbb{R}^{n \times K_1}, \ldots \mathcal{Z}_M \in \mathbb{R}^{n \times K_M}$ with $K = \sum_{m=1}^{M} K_m$. With this assumption, computing $\mathcal{Z}$ corresponds to applying the evidence model to each data site (with local or global parameter estimates). The price paid is that the group-wise sparsity property may be lost. Full implementation of GFA for distributed datasets is reserve for future work as discussed in chapter 7.

During training, the evidence $\mathcal{Z}_m$ computed by each agent is broadcasted to all agents. Each agent therefore receives a copy of $\mathcal{Z}$ which describes the whole distributed datasets. Because the original information available to each agent may be incomplete for training the base classifiers, the supplementary information it receives from other agent may be included in the training set for training the base classifiers. However, to demonstrate the fact that the use of supplementary information by a well designed ensemble method can help correct for some misclassification by the base classifiers, the shared evidence is used only for constructing the ensemble models.

The challenge remaining now is how to communicate $\mathcal{Z}$ to all agents (or groups of

agents). Two communication strategies are described to solve this problem, the first is a pure MapReduce implementation while the second is based on MPI.

5.6.3.2 Training Heterogeneous Agents with MapReduce

Training homogeneous agents on MapReduce is very straightforward and proceeds as a simple single-pass MapReduce job. Heterogeneous agents on the other hand presents a rather challenging situation. The classification evidence from each agent need to be communicated to all agents. In addition, if the base classifiers are to be shared, then the type of agent need to be communicated as well. However, by design, the MapReduce framework does not support communication between map tasks. The common approach to have mappers communicate is to implement some ad hoc workarounds. For example communication can be achieved by having a map task write some data to HDFS and another to retrieve it.

This strategy can be conveniently implemented through a series of MapReduce steps, that is, the output of one MapReduce job becomes the input of the next:

$$Map1 \quad \mapsto \quad Reduce1 \quad \mapsto \quad Map2 \quad \mapsto \quad Reduce2 \quad \mapsto \quad Map3 \quad ...$$

The first job in the chain writes its output to a path which is then used as input path for the second job. This process can be repeated for as many jobs are necessary to arrive at a complete solution to the problem.

To use chained MapReduce for training heterogeneous agents, two MapReduce jobs are required;

- The first map task $Map1$ reads the input data and splits into training and validation. The training set is used for training the classifiers and generation of parameters of the evidence model. Class labels or probabilities and classification evidence are predicted on the validation set. The classification evidence is stored on HDFS while the class labels or probabilities, base classifiers and evidence model can be stored locally or on HDFS. Local storage is preferred due to the small sizes of the data and for fast local

access.

- The first reducer task $Reduce1$ does nothing, i.e this is no-operation reduce class.
- The next map task $Map2$ reads in the classification evidence for all agents and class probabilities (stored locally) and generates the ensemble model parameters.
- The ensemble parameters are passed to the last reducer $Reduce2$ who aggregates them. The parameters can be saved on HDFS or locally. Once again local storage is preferred.

5.6.3.3 Prediction

Prediction equally requires two chained MapReduce jobs. The first mapper reads the local classifiers, evidence models and predict on the test set. The second mapper reads the predicted evidence for all agents from HDFS, the predicted class labels or probabilities and global ensemble parameters and make final predictions.

This pure MapReduce training is relatively easy to implement and requires only two chained jobs with a missing reduce step. However, because communication is achieved by writing multiple files to HDFS, the process can suffer from serious system overhead especially when the files written are huge. Numerical experiments are not reported for heterogeneous agents on Hadoop in this thesis. This is reserve for future work. The next section presents an MPI training approach for efficient communications.

5.6.3.4 Training Heterogeneous Agents with MPI

One of the keys to Hadoop-MapReduce popularity and success is its lack of data motion. The framework is design to take computation to data rather than the other way around. Thus Hadoop can achieve very high performance on applications involving very little or no communication between processes. The Hadoop-MapReduce framework has been reported by many authors (Bu et al., 2010; Ekanayake et al., 2010; Ngufor and Wojtusiak, 2013) to

performs very poorly on problems requiring some sort of communication such as iterative processing of data. Since most machine learning algorithms are inherently iterative, their implementation on Hadoop is complex, inefficient and very slow. Even though projects like Apache Mahout, GrapLab, MLbase have constructed machine learning on MapReduce the implemented algorithms are rather too costly and slow. This major drawback of Hadoop-Mapreduce makes the implementation of some very popular machine learning algorithms such as Expectation Maximization, Gradient based optimization methods, SVM and many others on MapReduce to be inefficient. The MPI-based parallel distributed framework on very large HPC clusters is a more favorable alternative for the implementation of these algorithms. Though some of the nice properties of Hadoop such as fault-tolerance and task monitoring may be limited, the trade-off in terms of computational cost heavily favors MPI (`http://www.admin-magazine.com/HPC/Articles/The-New-Hadoop`).

Under the MPI framework agents are modeled as individual processes. Thus on a multicore machine, each of the CPU represents an agent. The training process is a straight forward HPC computing tasks and follows the same procedure as the previously described chain MapReduce jobs except that instead of writing files to HDFS, explicit point-to-point or collective communication between agents is performed.

5.7 Experiments

This sections evaluates and compares the performances of the classifier integration schemes presented in this thesis on 14 benchmark datasets and on a large public available flight dataset. Eleven of the benchmark datasets are from the the UCI machine learning repository (Frank and Asuncion, 2010), two from the website of the book The "Elements of Statistical Learning" (Hastie et al. (2001), `http://statweb.stanford.edu/~tibs/ElemStatLearn/`) and one from the KEEL repository (Alcalá-Fdez et al., 2009). Though the benchmark datasets are relatively small with no need for distributed learning, the main purpose of this

experiment is to demonstrate on a series of datasets the optimality of the presented framework when learning from heterogeneous data sites and to show that the models generated are approximately exact. Thus two main experiments are performed: The first experiment models heterogeneous data sites using the benchmark datasets while the second experiment models homogeneous data sites using the flight dataset.

To model heterogeneous data sites, a random subsets of the feature space is distributed across several computing nodes and trained using the MPI framework described in section 5.6.3.4. The MPI programming framework is implemented using the recent R package **pdbR** or "Programming with Big Data in R" (Ostrouchov et al., 2012). **pdbR** is a SPMD model that enables high-level distributed data parallelism in R on large scale computing clusters with focus on analyzing big data. With pbdR packages, R users can readily access compiled MPI codes through R traditional classes and methods while maintaining the language original syntax convenience. However, the underlying code is readily available if the user wishes to change or implement new constructs. The package also supports high level functions for distributed dense matrix operations and scalable linear algebra, however in the experiments only the basic MPI routines such as MPI_Bcast, MPI_Scatter, MPI_Allgather and MPI_Allreduce are used.

The MapReduce framework with active data and algorithm selection as described in section 5.6.2 is implemented for learning the flight data. However, the active algorithm was mainly used for sample size reduction while all base classifiers were selected a priori. The experiments were performed using Hadoop version 1.2.1 on a small cluster of three machines: A 6 core 8 GB RAM and two 4 core 4 GB RAM each.

For each dataset, a total of six ensemble models were constructed:

1. The three integration schemes presented in chapter 4:

    - Ensemble method with EM (EM) (see section 4.4.1.2).
    - Variational Bayes Ensemble for Discrete outputs (VBD) (see section 4.4.2.1).
    - Variational Bayes Ensemble for Continuous outputs (VBC) (see section 4.4.2.2).

2. Stack generalization (Stack) with logistic regression as the combiner (see section 2.2.2).

3. Mean Rule (M.Rule) (see section 2.2.3).

4. Majority Votes (M.Votes).

The accuracy or percent correctly classified (PCC) and AUC are used to evaluate all ensemble models. Since Majority Votes is determined using only the hard outputs $\{0, 1\}$, the AUC is not reported for this method. The AUC of the base classifiers are also not reported, but the PCC of the best base classifier (B.Base) is reported.

The same collection of algorithms used for the experiments in chapter 4 are equally used as base algorithms. The algorithms are respectively: Support Vector Machine (svm), Linear Discriminant Analysis (lda), Naive Bayes (nb), Logistic Regression (log), Neural Networks (nn), Recursive Partitioning and Regression Trees (tree), Least-Squares Extreme Logistic Regression (ls-elr), Iterative Least-Squares Extreme Logistic Regression (irls-elr), Extreme learning machine (elm) and Quinlan's C5.0 rule-based classifier (c50). Three random algorithms were dropped from the hadoop experiments to lessen the computational cost. All algorithms are trained with default parameter settings.

5.7.1 Benchmark Datasets

The 14 benchmark datasets consists of Abalone (abalone), Breast Cancer (breast), Car Evaluation (car), Australian Credit Approval (credit), Magic Gamma Telescope (magic), Mammographic Mass (mam), Ozone Level Detection (ozone), Page Blocks (pageblock), Phoneme (phoneme) (data from `http://statweb.stanford.edu/~tibs/ElemStatLearn/`) Pima Indians diabetes (pima), South African Hearth Disease (saheart) (data from `http://statweb.stanford.edu/~tibs/ElemStatLearn/`), Spambase (spam), Twonorm (twonorm) (data from Alcalá-Fdez et al. (2009)) and Wine Quality (wine). Each dataset is randomly split into three equal parts: One part for training, one part for validation and the last part for testing.

The number of attributes and sample sizes of the datasets is presented in Table 5.1. Most of the datasets represented binary classification problems. Multiclass datasets such as the car, abalone and and wine were converted to binary. The car dataset has four classes representing car evaluations. These were aggregated into two classes: class 1 = ("unacceptable", "acceptable") and class 0 = ("good" , "very good"). The abalone dataset represents the problem of predicting the age of abalone from physical measurements. The age is determined by the number of rings in the Abalone's cone. The number of rings varied from 1 to 29. The negative class was taken to represent rings in the range 1 to 10 while the rest made up the positive class. Finally, the outcome variable in the wine dataset is wine quality ranging from 0 (very bad) to 10 (very excellent). The binary classification problem constructed out of this dataset was to distinguish poor or normal wine (0-6) from good or excellent (7-10) wine.

Table 5.1: Classification datasets

Data	Attributes	Sample Size
abalone	8	4177
breast	9	277
car	21	1728
credit	14	690
magic	10	19020
mam	15	830
ozone	72	1847
pageblock	10	5473
phoneme	5	5404
pima	8	768
saheart	9	462
spam	57	4597
twonorm	20	7400
wine	11	4898

A three-fold cross-validation learning procedure is carried out to evaluate the generated models. That is, each of the splits of the data is used alternatively as training, validation or

testing. To model heterogeneous data sites, the base classifiers on each site are trained only on a random sample of 50% of features. A total of 5 processors on an 8-core 16-GB memory machine are used to model the data sites. Thus each processor reads the full dataset and randomly selects 50% of the features for learning. The dimension of the evidence extracted by each processor is set to $K = 1$, thus each data site broadcast a single evidence feature vector to the rest of the data sites.

The car and magic datasets were also used for the experiments reported in chapter 4. However, in chapter 4 the base classifiers were trained using all attributes of the datasets. Thus the experiments on the car and magic datasets provides an empirical way to compare the relative performance of the ensemble models when learning from distributed data sites with different observed attributes against a centralized learning with all attributes. This experiments will show that the ensemble methods proposed in this thesis generates approximately exact models.

5.7.2 Flight Dataset

The Bureau of Transportation Statistics (BTS) provides arrival and departure information for every commercial flight within the U.S from 1987 to date. Monthly data with the option of filtering relevant features can be downloaded from the BTS website in CSV format. Approximately 120 million records are contained in the dataset between 1987 and 2008. This subset has been used for the Visualization Poster Competition of the 2009 Joint Statistical Meetings (results of the competition can be found here `http://stat-computing.org/dataexpo/2009/posters/`). This subset is also currently used as a benchmark dataset to illustrate big data concepts in some implementations of the Hadoop-MapReduce framework such as RHadoop, RHIPE (`http://www.datadr.org/`), Oracle R Advanced Analytics for Hadoop[1].

For the experiments performed in this chapter, only flight details for the most recent years are considered. Specifically, flight details of the last 5 months of 2012 is used as

[1] `http://www.oracle.com/us/products/database/big-data-connectors/overview/index.html`

training set while details for the last 5 months of 2013 used for testing. The classification problem formulated here is to determine whether flight details of previous years can be used to predict flight (arrival) delays with high accuracy in the feature.

This problem is of significant importance as it has severe impact on the U.S economy. Using data from the U.S department of transportation, the Joint Economic Committee of the U.S Congress estimated that the total cost of domestic air traffic delays to the U.S economy was as much $41 billion in 2007 alone and cause an extra consumption of 740 million gallons of jet fuel (Schumer and Maloney, 2008). Further, almost 20% of total domestic flight time in the same year was wasted in delays costing travelers time worth of $12 billion. When air travel takes extra time than scheduled, delayed travelers, their employers, and others lost productivity, business opportunities, and leisure activities. The flight delay predictive analysis considered here can thus provide a useful tool for a traveler to predict several months ahead if her flight will be delayed and therefore make inform decision and best plan her travel.

5.7.2.1 Flight Data Preprocessing

As the primary objective is to predict flight arrival delays with high accuracy, a secondary data source was used to enrich the flight details from BTS. Specifically, for each aircraft in BTS identified by its Tail number, information such as year manufactured, type of aircraft, type of engine were pulled from the Federal Aviation Administration (FAA) Aircraft Registry by matching with the N-Number. After merging the two databases, the following variables were selected for further processing and learning:

1. Flight details: Month, Day of Month, Day of Week, Unique Carrier Code, Flight Date, Origin, Destination, CRS Departure Time, Departure Delay and Distance in miles

2. Aircraft Details: Age (year of flight - year manufactured), Type of Aircraft, Type of Engine, Number of Engines, Number of Seats, Aircraft Weight and Average Cruising Speed.

The average cruising speed was not available for all flights so a simple imputation by the mean was performed.

All categorical variables were converted to binary. To predict flight Arrival Delays, a continuous variable which represented the difference in minutes between Scheduled and Actual Arrival Time was converted into binary. Values greater than 15min were coded as 1 and 0 otherwise. This threshold can be turned to obtained different delay catergorization as desired.

BTS also provide information if a flight was canceled or not and a reason if there was a delay (departure or arrival). All flights that were canceled were removed and because weather details of the flights were not modeled in the experiments, all flights that were delayed because of weather conditions were also dropped.

Before loading the training data onto Hadoop, simple linux command line tools such as *cat, shuff* and *split* were used to shuffle the data. This ensures that approximately a balanced set of attributes is distributed to all nodes. Each of the shuffled dataset corresponding to a month's data is then compressed and loaded into hadoop. The compression step is however not required, but because each map task selects only informative data points for learning, the compression prevents further splitting of the data by Hadoop and ensures that a single mapper is assigned to a single file. The MapReduce job is then run with 5 mappers and a single reducer.

5.7.2.2 Training and Testing

Each month of the flight data in the training set is modeled as a homogeneous data site. Thus all data sites contained exactly the same number and type of features. Training proceeds as a single-pass MapReduce job. The active data and algorithm selection is first applied to the input data to select only informative points for training. The scaled down dataset is then split into traning and validation. Table 5.2 shows the sample sizes of the five input and the scaled down training sets respectively. It can be seen that each mapper learns on approximately 1% of the input data.

Table 5.2: MapReduce input data sizes and selected training sizes

Input	Training	Validation
509519	5348	2674
469746	5082	2541
552312	5091	2545
536393	5092	2546
548642	5091	2545

The classification evidence and base classifiers are then trained on the training set and predictions made on the validation set. Parameters of the ensemble models are computed using the predictions and passed to the reducer who simply aggregated them by classifier type. The trained models can either be saved locally or on Hadoop. Because access time for files on Hadoop can be longer, it is easier to save the models locally.

Testing equally proceed as a single-pass MapReduce job. The trained ensemble model parameters can be passed to the mappers as distributed cache or read locally depending on how it was saved during training. Each mapper predicts class labels or probabilities on the test set using the base classifiers and combine the predictions using the ensemble model.

5.8 Results

The results of the two experiments are presented and discussed in this section.

5.8.1 Performance on Benchmark Datasets

Table 5.4 and 5.5 shows the average validation and test performance of the six ensemble methods and the best base classifier on the 14 benchmark datasets. The following observations can be drawn from the results

1. The overall performance of VBC on the validation set was superior to the other methods. Given that the ensemble parameters were estimated on the validation set, the

performance of VBC and VBD indicates that the methods are not overly optimistic.

2. A remarkable performance can be seen for both VBC and VBD on the test set. Overall, these two ensemble methods outperformed the other methods. No noticeable change in performance can be observed for Stack, M.Rule, M.Votes and B.Base between the validation and test experiments.

3. The EM ensemble method appears to be the weak link among the proposed methods. Its performance is almost comparable to majority votes. The possible reasons for the poor performance as discussed in chapter 4 also applies here.

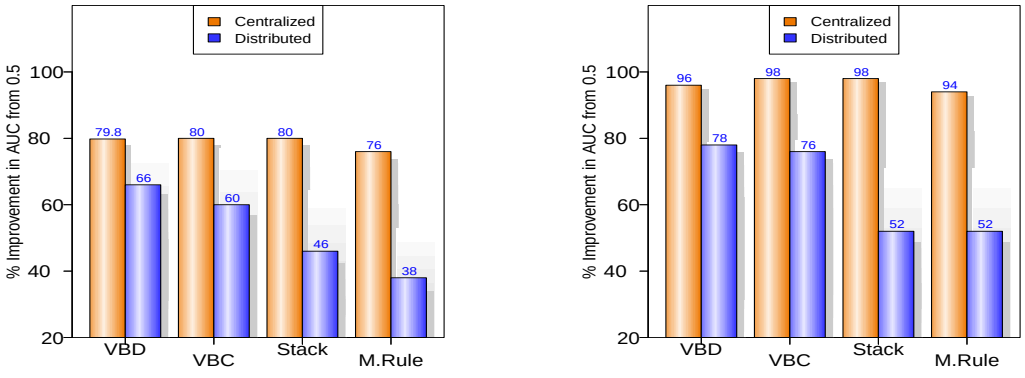
4. Consider the performance of the methods on the magic and car datasets for example; In chapter 4, the base classifiers were trained in a centralized manner i.e. using all attributes of the datasets. All classifiers performed very well and further inspection showed that except for the EM method, the results for VBD, VBC, Stack, and M.Rule were very close in magnitude. A reduction in AUC values from centralized to distributed training is obtained for each ensemble method as follows: VBD 6.7%, VBC 11%, EM 18%, Stack 19% and M.Rule 22% on magic dataset and VBD 9%, VBC 11%, EM 8%, Stack 23% and M.Rule 22%. Dropping the EM method because of its unstable performance, these results can be illustrated more clearly by showing the centralized and distributed results together as in Table 5.3. Figure 5.4 shows an alternative way to illustrate these results. It shows the percentage improvement in AUC from a baseline random classifier whose AUC is 0.5 in both centralized and distributed learning.

   Clearly VBD and VBC are approximately consistent in performance in a distributed environment. This can be attributed to the extra information provided by the classification evidence that is shared among the data sites and used to combine the decisions of the base classifiers.

5. In a distributed learning environment, some data sites may be irrelevant or contain noisy features. The factors in group factor analysis describes the data sites as a whole

Table 5.3: Approximate Exact Models

Model	Data	Centralized	Distributed	% Reduction
VBD	magic	0.89	0.83	6.7%
	car	0.98	0.89	9%
VBC	magic	0.90	0.80	11%
	car	0.99	0.88	11%
Stack	magic	0.90	0.73	19%
	car	0.99	0.76	23%
M.Rule	magic	0.88	0.69	22%
	car	0.97	0.76	22%



(a) magic

(b) car

Figure 5.4: magic and car experiment

and not just the features. With ARD priors, irrelevant data sites are pruned from the ensemble decision and thus contribute to improving the accuracy of the methods.

5.8.2 Performance on Flight Dataset

Table 5.6 shows the performance of the Hadoop experiment on the flight data. Tables 5.6 (a) and (b) are performances for the first two data sites while (c) is the average performance over the 5 months data. Judging from the AUC values, the proposed ensemble methods once again proved to be the better algorithm for learning the large-scale flight dataset on MapReduce.

An important factor which contributed to improve the accuracy of the methods is that the original flight data from BTS was enriched by several informative features about the individual aircraft obtained from the aircraft registry. Some of the base classifiers may not be able to make use of this information, but by supplementing their decision with hidden information about them, VBC, VBD and EM ensemble methods were able to improve prediction accuracy of flight delays by many percentage points.

While the performance of the EM method was poor on the benchmark datasets, it however outperformed all methods on the flight data. The limitations of the method as given for the benchmark experiments does not seems to apply here. This clearly indicates that ensemble learning is not a trick that will always work in all cases.

Table 5.4: Validation Performance on Benchmark Datasets

Data	Measure	VBC	VBD	EM	Stack	M.Rule	M.Votes	Best.Base
abalone	PCC	**0.78**	0.74	0.72	0.75	0.72	0.73	0.74(nn)
	AUC	**0.85**	0.78	0.64	0.81	0.79		
breast	PCC	0.69	0.71	0.69	**0.73**	0.69	0.71	0.71(nb)
	AUC	0.62	0.54	0.54	**0.69**	0.6		
car	PCC	**0.82**	0.78	0.77	0.76	0.77	0.77	0.77(svm)
	AUC	**0.89**	0.86	0.55	0.79	0.78		
credit	PCC	**0.85**	0.78	0.48	0.8	0.79	0.79	0.78(elr)
	AUC	**0.92**	0.83	0.7	0.88	0.87		
magic	PCC	**0.75**	0.73	0.72	**0.75**	0.73	0.73	0.74(nn)
	AUC	**0.76**	0.65	0.6	0.75	0.71		
mam	PCC	**0.82**	0.78	0.49	0.73	0.72	0.73	0.73(tree)
	AUC	**0.87**	0.79	0.5	0.77	0.76		
ozone	PCC	**0.94**	0.93	0.93	**0.94**	0.93	0.93	0.93(svm)
	AUC	**0.89**	0.86	0.7	0.88	0.87		
pageblock	PCC	**0.96**	0.94	0.92	**0.96**	0.93	0.94	0.96(tree)
	AUC	**0.98**	0.9	0.79	0.97	0.94		
phoneme	PCC	**0.79**	0.75	0.71	**0.79**	0.77	0.76	0.79(nn)
	AUC	**0.85**	0.78	0.6	**0.85**	0.82		
pima	PCC	**0.65**	0.64	0.63	0.63	0.64	0.63	0.64(nb)
	AUC	**0.72**	0.56	0.54	0.71	0.7		
saheart	PCC	**0.68**	0.65	0.62	**0.68**	0.63	0.64	0.66(lda)
	AUC	**0.71**	0.6	0.58	0.7	0.62		
sona	PCC	0.85	0.83	0.54	0.88	0.82	0.81	0.83(log)
	AUC	0.92	0.9	0.63	0.92	0.91		
spam	PCC	**0.87**	0.85	0.64	0.86	0.85	0.84	0.85(nn)
	AUC	**0.94**	0.89	0.61	0.92	0.91		
twonorm	PCC	**0.92**	0.5	0.5	0.87	0.87	0.87	0.87(nb)
	AUC	**0.98**	0.52	0.51	0.94	0.94		
wine	PCC	**0.81**	0.8	**0.81**	**0.81**	**0.81**	**0.81**	0.8(nn)
	AUC	**0.82**	0.8	0.76	0.81	0.8		

Table 5.5: Test Performance on Benchmark Datasets

Data	Measure	VBC	VBD	EM	Stack	M.Rule	M.Votes	B.Base
abalone	PCC	**0.78**	0.76	0.65	0.75	0.74	0.74	0.75(nn)
	AUC	**0.84**	0.82	0.76	0.81	0.81		
breast	PCC	0.71	**0.74**	0.71	0.72	0.71	0.72	**0.74(log)**
	AUC	0.6	0.54	**0.65**	0.63	0.63		
car	PCC	0.82	**0.85**	0.80	0.73	0.73	0.72	0.73(elr)
	AUC	0.88	**0.89**	0.83	0.76	0.76		
credit	PCC	0.81	**0.82**	0.77	0.79	0.8	0.79	0.8(log)
	AUC	0.85	**0.88**	0.84	**0.88**	**0.88**		
magic	PCC	0.78	**0.80**	0.72	0.73	0.71	0.71	0.73(nn)
	AUC	0.80	**0.83**	0.59	0.73	0.69		
mam	PCC	0.77	**0.78**	0.57	0.71	0.71	0.72	0.72(log)
	AUC	**0.84**	0.83	0.5	0.72	0.72		
ozone	PCC	**0.93**	0.92	0.92	**0.93**	**0.93**	**0.93**	**0.93(log)**
	AUC	0.83	**0.88**	0.82	0.87	**0.88**		
pageblock	PCC	**0.96**	**0.96**	0.92	**0.96**	0.94	0.94	**0.96(tree)**
	AUC	**0.97**	0.96	0.82	0.96	0.94		
phoneme	PCC	0.78	**0.79**	0.73	0.78	0.75	0.75	0.77(nn)
	AUC	**0.85**	**0.85**	0.74	**0.85**	0.82		
pima	PCC	**0.74**	**0.74**	0.69	0.73	0.74	0.72	0.73(elr)
	AUC	**0.77**	0.75	0.72	0.76	0.76		
saheart	PCC	0.66	0.69	**0.7**	0.66	0.7	0.68	0.7(svm)
	AUC	0.66	0.65	**0.72**	0.62	0.62		
spam	PCC	0.87	**0.88**	0.67	0.86	0.84	0.83	0.85(nn)
	AUC	**0.93**	**0.93**	0.71	0.92	0.91		
twonorm	PCC	0.91	**0.92**	0.91	0.86	0.86	0.86	0.86(nb)
	AUC	**0.98**	0.97	0.97	0.94	0.94		
wine	PCC	0.8	0.80	0.78	**0.81**	0.8	0.8	**0.81(tree)**
	AUC	0.81	0.81	0.71	**0.82**	0.81		

	Validation		Testing	
Model	PCC	AUC	PCC	AUC
VBC	0.74	0.81	0.80	0.83
VBD	0.74	0.79	0.80	0.82
EM	0.73	0.79	0.80	**0.89**
Stack	0.74	0.81	0.80	0.79
M.Rule	0.74	0.80	0.80	0.64
M.Votes	0.74		0.80	
svm	0.73		0.80	
nb	0.62		0.66	
log	0.74		0.80	
nn	0.69		0.80	
tree	0.74		0.80	
elr	0.74		0.80	
lda	0.74		0.80	

(a) First Data Site

	Validation		Testing	
Model	PCC	AUC	PCC	AUC
VBC	0.72	0.79	0.80	0.81
VBD	0.73	0.77	0.80	0.82
EM	0.72	0.79	0.80	**0.89**
Stack	0.72	0.79	0.80	0.77
M.Rule	0.72	0.78	0.80	0.62
M.Votes	0.72		0.80	
svm	0.72		0.80	
nb	0.63		0.66	
log	0.72		0.80	
nn	0.65		0.80	
tree	0.72		0.80	
elr	0.72		0.80	
lda	0.72		0.80	

(b) Second Data Site

	Validation		Testing	
Model	PCC	AUC	PCC	AUC
VBC	0.73	0.79	0.80	**0.83**
VBD	0.73	0.77	0.80	**0.81**
EM	0.70	0.77	0.80	**0.90**
Stack	0.73	0.79	0.80	0.78
M.Rule	0.72	0.78	0.80	0.65
M.Votes	0.71		0.80	
svm	0.72		0.80	
nb	0.62		0.65	
log	0.72		0.80	
nn	0.66		0.80	
tree	0.71		0.80	
elr	0.72		0.80	
lda	0.72		0.80	

(c) Global Performance

Table 5.6: MapReduce Validation and Test Performance on Flight Data.

5.9 Summary

This chapter presented a large-scale parallel and distributed machine learning framework or collective machine learning. By integrating the Bayes active data and algorithm selection scheme presented in chapter 3, the classification evidence model and optimal Bayesian multiclassifier integration schemes presented in chapter 4 within each data site, a parallel efficient collective and cooperative machine learning system is developed for learning homogeneous and heterogeneous distributed data sites. Agents can collective communicate by sharing informative features vectors not observed in the distributed datasets to improve accuracy by correcting misclassifications by the base classifiers. By pulling these bits of information from other datasets to enrich a given dataset, it is shown that the global models generated are approximately exact. Two parallel programming frameworks are proposed for collective learning: the MapReduce and MPI models. A series of numerical experiments on benchmark datasets and on a large-scale flight dataset demonstrates the superior performance of the proposed framework.

Chapter 6: Systematic Prediction of Adverse Drug Reactions from FAERS and MedEffect Canada

Adverse drug reactions (ADRs) are a major global health concern accounting for more than two million injuries, hospitalization and death each year in the US alone (Lazarou et al., 1998). With these numbers, it is not surprising that ADRs are equally a leading cost of healthcare (Classen et al., 1997). Each year billions of dollars are spent by the US public to treat disease cause by side-effects from prescription drugs which by themselves could lead to more side-effects. Therefore the timely and accurate detection and prediction of ADRs at the post-approval period is very important in healthcare. A reduction in both the harm to patients and cost can be archived if accurate models are available to predict at prescription time for each drug the likelihood of a patient developing a known or potentially new ADR based on known properties of the drug and characteristics of the patient.

This section presents a novel systematic and structured approach to a solve this problem based on the optimal Bayesian integration framework presented in this thesis.

6.1 Introduction/Background

ADRs have become a major healthcare care concern: the US public for example spends billions of dollars every year on prescription drugs of which an alarming number of them result in serious side-effects. Over the past decade both reported ADRs and related deaths have increased significantly, leading to the withdrawal of a number of drugs from the market (`http://www.fda.gov/safety/recalls/`). Hence accurate predictions of ADRs is extremely important at both the pre-approval and post-approval cycle of a drug.

Various approaches have been proposed to study the drug-event relationship. The traditional method for predicting or assessing potential ADRs has been done using two main approaches: during clinical trials and post-marketing surveillance. At the early stage of drug development, drug safety profiles are carried out by testing compounds with biochemical and cellular assays. One crucial advantage of clinical trails is that potential drug-event identified most often represent causal relationships. Despite the methodological rigor of clinical trials, it is generally not possible to identify all safety issues associated with drugs primarily due to cost and efficiency. The size and characteristics of patient population, drug doses and duration of use, and other realistic variables frequently observed at the post-marketing phase can be impossible to model at the clinical trial phase.

The second approach based on post-marketing surveillance aims at applying data mining techniques to identify significant patterns of association or "signals" between drugs and ADRs using spontaneous reporting systems (SRS). The most commonly used data mining methods used in this phase include Proportional Reporting Ratio (PPR)(Evans et al., 2001), Reporting Odds Ratio (van Puijenbroek et al., 2002), Information Component (IC) and Empirical Bayesian Geometric Mean (EBGM). The US Food and Drug Administration (FDA) has adopted the EBGM implemented as the Gamma Poisson Shrinker (GPS) (Szarfman et al., 2002) while the World Health Organization (WHO) implements the IC using a Bayesian Confidence Propagation Neural Networks (BCPNN) (Bate et al., 1998). The Bayesian methods accounts for variability associated with small report counts. Most of these methods are based on the use of a disproportionality statistics to quantify the degree of "unexpectedness" of a drug-event associations (Bate and Evans, 2009). However, analyzing SRS using these traditional data mining techniques is a very challenging task primarily because these voluntary reports are subject to various limitations such as under/over-reporting, reporting biases, unverified data, misattributed drug-event combination, missing and incomplete data, duplicated reporting and unspecified causal relationships. Another concern with these methods is that, potential drug-events relationships identified does not necessarily demonstrate causality. Reports of ADRs associated with a drug are

not necessarily true ADRs, that is, they may be temporally associated with a drug but not caused by the drug. Hypothesis generation of new possible side effects from such data is referred to as signal detection (Bate and Evans, 2009) and thus cannot be used to prove or refute causal relationship between a drug and an ADR.

Recently, a third approach has emerged that uses machine learning techniques and available large public drug databases to predict ADR at both the clinical and post-marketing phases. Most of these methods typically use chemical, biological and phenotypic properties of drugs to build predictive models. For example Liu et al. (2012) applied five machine learning algorithms, namely: logistic regression, naïve Bayes, k-nearest neighbor, random forest, and support vector machine to investigate the use of phenotypic information on drugs together with chemical and biological properties to predict ADRs. The underlying principle behind most of these approaches is that drugs with similar chemical and or biological properties most often exhibit similar ADRs (Fliri et al., 2005). To the best knowledge of the author, none of these studies have considered incorporating reports of ADRs from SRS databases to predict ADRs using a distributed machine learning approach such as the collective machine learning system presented in this thesis.

The public availability of drug databases such as DrugBank (Knox et al., 2011) and SIDER (Kuhn et al., 2010) containing chemical, biological, and phenotypic properties of drugs provide a unique opportunity to bridge the gap between the clinical and post-marketing domains in studying ADRs. A systematic leveraging of this type of information i.e. combining information on drug protein binding sites; biological pathways of drug action and metabolism; drug chemical similarities; demographic characteristics of patients such as age, gender, weight, height; route of administration, geographic location; and other relevant information of patients who consumed the drug, the drug-event association can be better understood. Further, and more importantly, individualized patient specific predictive models can be generated. Before prescription, based on patient information and known drug information, the likelihood of developing a known or potentially new ADR can be determined and the appropriate decisions made.

The objective of this chapter is to present a new approach to learning ADRs from chemical and biological properties of drugs publicly available in drug databases such as SIDER and DrugBank combined with information from SRS databases in a distributed high performance computing framework. Chemical and biological properties of FDA approved drugs in DrugBank are mapped to drugs in SIDER to retrieve known drug side effects. These drugs are then mapped to drugs reported in FDA Adverse Events Reporting System (FAERS) and MedEffect Canada and predictive models are generated using the optimal integration framework presented in this thesis. To increase the sensitivity of the approach, BCPNN is initially applied to the drug-events reported in both SRS databases to identify potential signals. Based on these signals, the data is then clustered by the chemical and biological properties of the drugs and patient demographic variables. Learning of ADRs is done within each cluster. The intuition behind clustering is based on the observation that similar drugs most often exhibit similar side-effects, thus the accuracy of ADRs predictive models constructed within each cluster can significantly improve. In addition, by examining the association between known ADRs and reported ADRs within each cluster novel ADRs can be identified while spurious reported ADRs can be eliminated. Including patient demographic information in the clustering can further strengthen the association.

6.2 Drug-ADRs Signal Detection and Clustering

Most Bayesian methods for signal generation between a single drug and a single event in SRS use the information component (IC) measure that attempts to account for uncertainty in the disproportionality measure. When the IC is positive for a drug-ADR association, this implies that the association is more strong than expected. Values of IC close to zero represent independence between the drug and event. The idea then is to search the SRS database for all positive values of IC.

This one-to-one signal generation however ignores the fact that certain groups of drugs

may share a common set of ADRs. Identifying such groups could provide valuable information about the drugs and ADRs and possible facilitate the identification of new ADRs (Harpaz et al., 2011).

Clustering of drug-ADRs is another data mining approach that has recently been applied to detect signals in SRS. Harpaz et al. (2011) applied a biclustering technique to simultaneously cluster drugs and ADRs in FAERS. The authors pointed out that examining similar drugs and ADRs in a bicluster, valuable information can be gained about the drugs and ADRs, and the classes of drugs identified can be used to predict potential ADRs for new drugs. To strengthen the clusters generated, clusters can be generated for drug-ADRs for which the IC is positive.

The BCPNN is chosen for generation of signals in this study because of its efficient computational power in handling all possible drug-ADR combinations. Another appealing property of BCPNN is that it is very suitable for implementation on parallel computers.

6.3 Experiments

To systematically predict adverse drug reactions, information on FDA approved drugs, known side-effects, spontaneous reported side-effects and characteristics of patients who consumed the drugs are pulled together. This information is then used in a structured way to predict side-effects of drugs: Chemical and biological properties of drugs are combined with demographic information of patients who consumed the drugs to build clusters of similar drugs and patients from which ensemble models are generated.

The idea behind the approach is based on the observation that similar drugs have similar side-effects. Thus if patients were to be grouped into homogeneous groups based on some demographic characteristics such as age, gender, height, weight, geographic location were side-effect occurred, duration of use, etc., then it is reasonable to assume that similar patients should have similar outcomes when administered the same treatment.

This assumption motivated the systematic and structured approach of combining drugs

and their known side-effects with reported side-effects about the drugs and characteristics of the patients to create a dataset whose observations are characterized by a drug and a patient. Further, since the content and structure of most SRS databses are different but contains information on the same type of drugs, sharing some information among SRS data sites can substantially improve the performance of the predictive models. To this end, the ensemble models are constructed from two SRS data sites: FAERS and MedEffect Canada. WHO Global ICSR Database System (VigiBase) was also considered for a third SRS data site, but unfortunately the reports are not freely available. Classification evidence for each drug stratified by patient characteristics is extracted from each data site and shared.

The overall learning process follows a series of steps as illustrated in Figure 6.1.

1. Preprocessing

    - Map drugs from SIDER to drugs in DrugBank to pull drug side-effects, drug chemical structures, information on proteins, targets and enzymes. This creates a known ADRs (KADR) dataset.
    - Find BCPNN signals from drugs-events reported in FAERS and MedEffect Canada. Select drug-event pairs with information content IC > 0.
    - Map drugs from KADR to signaled drugs in FAERS and MedEffect Canada to creat two data sites: FAERS-KADR and MedEffect-KADR.

2. Compute classification evidence from each data site using demographic variables and other reporter specific variables. Broadcast evidence to all sites.

3. Perform k-means clustering of each data site using biological, chemical, and demographic variables. The optimal number of clusters is determined by the cluster-wise assessment of cluster stability through a bootstrap procedure described in Hennig (2007).

4. Within each cluster, match known ADRs (from KADR) with reported ADRs. The modified Jaro-Winkler string comparator method with spelling correction is used for

this exercise. The comparator score is set to a very high value such as 0.95. Matches with scores less than 0.95 can be set aside for further investigation or can form a new cluster.

5. For each side-effect within a cluster, build a set of binary classification models and combine their predictions using the ensemble methods described in chapter 4.

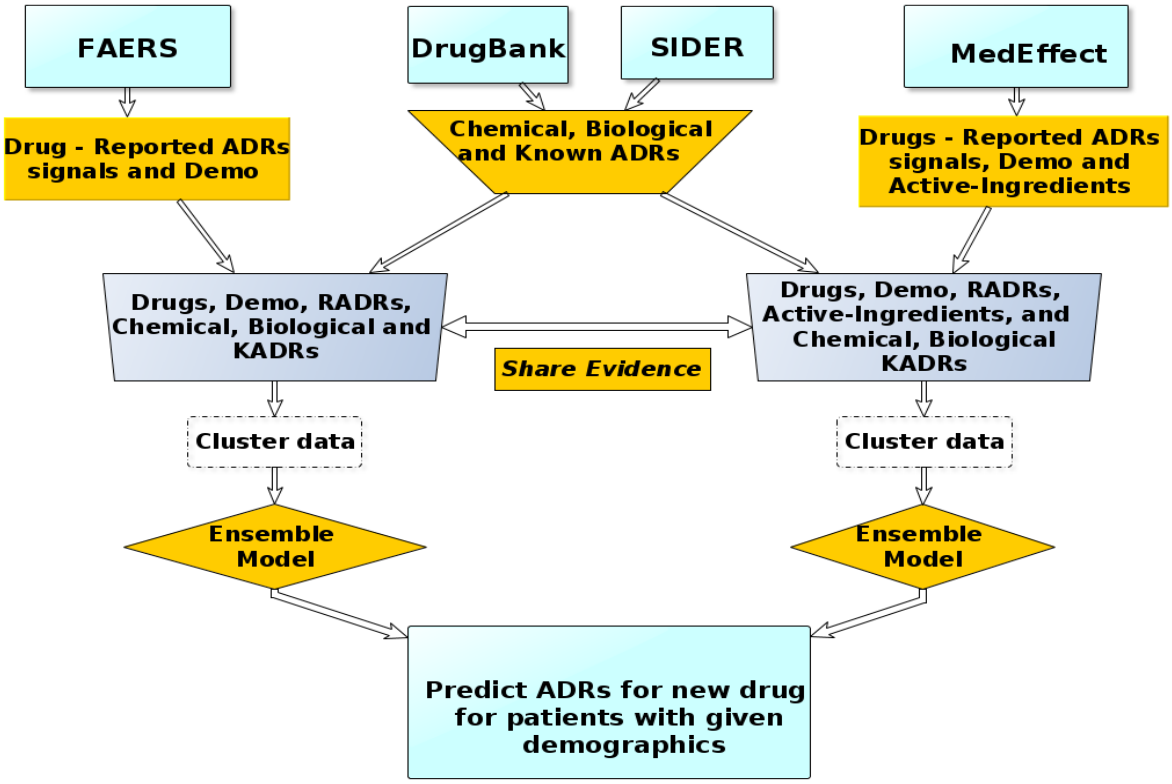


Figure 6.1: Systematic Prediction of ADRs

6.3.1 SIDER and DrugBank Data

Known ADRs (KADRs) of drugs are obtained from SIDER. The SIDER database contains drug side-effects relationship information on 888 drugs and 1450 side-effects. 70% of the drugs have between 10 and 100 different side effects.

Chemical and biological properties of drugs are obtained from DrugBank. The chemical structure of each drug is extracted from a structure-data file (SDF) which holds information about the atoms, bonds, connectivity, molecular weights, coordinates of a molecule etc. Some of the chemical attributes extracted include: solubility, number of rotatable bonds, polar surface area, exact mass, molecular weight, acid dissociation constant etc. All the chemical attributes are continuous. Biological attributes of drugs include: protein targets, transporters and enzymes. The biological attributes are all categorical and were converted to binary.

Drugs from SIDER were then mapped to drugs in DrugBank to create the dataset KADR. The mapping was done using drug names with the Jaro-Winkler string comparator algorithm and only exact matches were retained.

6.3.2 FAERS Database

FDA collects individual reports on ADRs (RADRs) voluntary reported by health care professionals and consumers. Data for the 4th quarter of 2012 with a total of 3,553,844 observations was used in this study. The database consists of 7 tables, but only data from the patient demographic and administration, adverse events, and drug tables were used. All data was loaded into a PosgreSQL database and merged. Selected demographic variables include: age, gender, weight, reporter country, occur country, dose amount, and route of administration.

Drug signals are generated from the processed FAERS data using BCPNN. Drugs with IC > 0 are then mapped to drugs in KADR to create the FAERS-KADR dataset. The k-means clustering algorithm is used to cluster the data and within each cluster known side-effects (from KADRs) is matched with reported side-effects to generate the dependent variable for classification.

6.3.3 MedEffect Database

The Canada Vigilance Adverse Reaction Database (`http://www.hc-sc.gc.ca/dhp-mps/medeff/index-eng.php`) contains information about suspected adverse reactions reported voluntary by consumers and health professional. The database consists of 11 tables but only data from 4 tables were used for this study: Reports (demographics), Reaction (RADRs), Report_Drug and Drug_Product_Ingredients (contains active ingredients associated with all drugs). Reports from 2008 to 2013 with a total of 2,714,445 observations was used for the analysis. Attribute selected include age, gender, weight, height, seriousness (yes/no), system organ class, route of administration, dose amount and active ingredients. As can be seen, MedEffect contains a richer set of attributes than FAERS for the same type of drugs.

Similarly, signals are generated with BCPNN and mapped to drugs in KADR to create the MedEffect-KADR dataset. Clustering is similarly done as for the FAERS-KADR data.

6.3.4 Implementation

Detecting and predicting ADRs is a very challenging task in terms of the computational complexity involved. For instance, calculating measure of disproportionality for detecting drug-event signals involves computing two-by-two contingency tables for all pairs of drug-events found in the SRS database. Due to the large volume and rapid accumulation of reports in SRS sites, the computational cost can be very expensive. Further, the clustering method can generate very large number of clusters and with ensemble learning within each cluster it is not hard to see that the computational task quickly becomes very difficult if not impossible for a single computing machine. Therefore a distributed high performance computing model is required for this task. A suitable computing framework for this task is cloud computing. Cloud computing provides a pay per service computing where high intensive applications in terms of CPU and storage requirements can be launch on a large number of computing nodes at relatively low cost. To this end, the systematic prediction of ADRs is performed on a cloud computing framework. Specifically, the recent publicly

available Google Compute Engine (GCE) (`https://developers.google.com/compute/docs/instances`) which claims to support high-performance, computationally intensive and sophisticated scientific problems is chosen for this task. A total of 24 nodes each with 4 cores and 26GB of memory is setup for the experiments. Thus a total of 96 processes can be lunched in parallel. Load balancing can be enforced such that more processes are allocated to FAERS-KADR datasets.

The programming model set up is similar to the experiments on benchmark datasets described in chapter 5 section 5.8.1. In this case there are only two data sites FAERS and MedEffect with different subsets of observed attributes. Parellization is archived by having each process read a subset of the observations and perform the predictive task as previously described. Since the classification evidence is stratified by patient characteristics, sharing is done by drug type.

6.4 Results

Due to the limitation of the allowed budget for this thesis project, only 11 random side effects were selected within each cluster for learning. For the same reason, only 3 ensemble learning methods are implemented: VBC, VBD (see chapter 5) and Majority Votes. The linear discriminant analysis base classifier was also dropped due to the large number of binary attributes in the datasets. Note that unlike the experiment on benchmark datasets, the active data selection is implemented in these experiments. That is, during ensemble learning, each node or process selects most informative instances for learning within each cluster, thus computational cost can be reduced significantly.

Table 6.1 shows the average PCC over all computing nodes for the three ensemble learning methods. Clearly the two proposed ensemble methods out performed majority voting. Overall, VBC outperformed the other methods. Highlighted in the table are the few ADRs for which majority vote performed better than VBD. VBC outperformed majority votes on almost all ADRs and by very large margin on a few.

Table 6.2 shows the PCC, sensitivity, specificity, and AUC for VBC from a random compute node. In terms of sensitivity, the method is very accurate in distinguishing the presence of ADRs for a drug. Similar results were also observed for VBD.

Finally Table 6.3 presents the overall accuracy for all models. This is the average accuracy for all ADRs presented in table 6.1 with values of the base classifiers included. VBC archived a 91% while VBD 88% prediction accuracy. These are very encouraging results and demonstrates the efficacy of the systematic approach to predict ADR.

Table 6.1: Average performance of VBC, VBD and Majority Votes

ADRs	VBC	VBD	M.Votes
Abdominal pain	0.81	0.81	0.79
Agranulocytosis	0.89	0.87	0.74
Akathisia	1.00	1.00	1.00
Alopecia	0.82	0.80	0.79
Amnesia	0.93	0.92	0.89
Anal Hemorrhage	1.00	1.00	1.00
Anemia	0.85	0.85	0.77
Anxiety	0.79	0.78	0.74
Anxiety Disorder	1.00	0.98	0.96
Arthralgia	0.91	0.85	0.83
arthritis	**0.97**	**0.95**	**0.96**
Atrial fibrillation	1.00	1.00	1.00
Constipation	0.95	0.92	0.85
Convulsions	**0.88**	**0.60**	**0.67**
Corneal deposits	1.00	1.00	1.00
Corneal.opacity	0.95	0.94	0.94
Diarrhea	0.97	0.83	0.69
Disorientation	**0.94**	**0.79**	**0.88**
Dyskinesia	**0.78**	**0.78**	**0.83**
Ecchymosis	0.99	0.96	0.97
Erythema	0.68	0.68	0.68
Headache	**0.77**	**0.59**	**0.60**
Hypertonia	1.00	1.00	1.00
Somnolence	1.00	1.00	1.00
Tremor	0.73	0.65	0.63

Table 6.2: VBC performance for 11 ADRs on a single node

ADR	PCC	sensitivity	specificity	AUC
Abdominal pain	0.80	0.64	0.90	0.89
Alopecia	0.79	0.82	0.68	0.92
Amnesia	0.93	0.85	0.96	0.98
Anemia	0.82	0.80	0.83	0.93
Anxiety	0.79	0.72	0.85	0.89
Anxiety.Disorder	1.00	0.90	1.00	1.00
Arthralgia	0.91	0.89	0.94	0.98
Arthritis	0.97	0.89	0.99	0.99
Atrial fibrillation	1.00	1.00	1.00	1.00
Constipation	1.00	1.00	1.00	1.00
Disorientation	0.94	0.89	0.95	0.99

Table 6.3: Comparing Ensemble and Base Classifiers Averaged over all ADRs

Classifier	PCC
VBC	0.91
VBD	0.88
M.Votes	0.85
elr	0.82
log	0.84
nb	0.55
nn	0.86
svm	0.83
tree	0.86

6.5 Summary

This section presented a novel systematic and structured approach to predicting ADRs pulling information from multiple sources. The excellent performance of the proposed evidence and ensemble methods is demonstrated once again on a real world large-scale learning task. In particular, the proposed evidence model is advantageously applied to enrich distributed databases with very different observed attributes and observation. This results in very efficient and accurate predictive models for ADRs.

Chapter 7: Conclusion and Future Work

This thesis has described a framework for integrating information from multiple sources in the presence of uncertainties. A generative Bayesian graphical model and inference was presented for the integration of a collection of optimally selected machine learning classification models making use of supplementary information from the training data. A collective learning mechanism was then presented that integrates the Bayesian integration scheme within each data site for learning large-scale distributed datasets with possibly different observed observations and features. A number of interesting features of the proposed framework were described and the methods shown to be effective in improving accuracy of a general classification problem.

This chapter reviews these results in the light of the original research questions and goals of the thesis. It re-examines the open issues and challenges of ensemble learning and shows how the presented work meets the stated goals of the thesis. The chapter concludes with the main case study of the work, other possible applications, and future directions of the research.

7.1 Challenges and Solutions

A general discussion of the open issues and challenges in ensemble learning was presented in chapter 1. Three main challenges were of interest to the thesis: first, development of techniques that scale up to large and possibly physically distributed databases. Second, construction of exact or approximately exact global models from distributed heterogeneous datasets with minimal data communication while preserving privacy of the data. Third, how to efficiently learn from modern large-scale datasets which are often characterized by noisy data points, unlabeled or poorly labeled, sample bias, missing values, etc.

The thesis addressed these challenges by the introduction of generative Bayesian graphical model and inference to optimally integrate a collection of carefully selected classification models with the aid of supplementary hidden information for learning homogeneous and heterogeneous datasets. A Bayesian graphical model was chosen because it offers a principled and powerful representative model that accounts for the structural dependency/independence between the classifiers and other variables, and dealing with uncertainties and complexities of the problem. In addition, the model offers a flexible way to represent and infer hidden information from the problem domain which is of primary interest to the thesis.

Integrating the decisions of the classifiers to meet the goals of the thesis raised a number of interesting questions:

1. How to select the base classifiers?

2. How to combine the outputs of these base classifiers to maximize classification accuracy?

3. How to make the method scalable?

Based on these simple questions and the challenges of parallel and distributed machine learning discussed in section 1.2, the thesis formulates three related research questions:

1. How to effectively evaluate, compare, and rank learning algorithms on modern datasets characterized by noise, poor labeling, class imbalance, sample bias, missing values, and other complex phenomenons.

2. How can a machine learning system identify and correct the mistakes of learning algorithms for distributed homogeneous/heterogeneous datasets. Producing exact models while not compromising the sensitivity of the data.

3. How to solve the two problems above and still remain efficient in both sample complexity and computational complexity.

These questions were addressed systematically in chapters 2, 3, 4, and 5. The next section describes how the work presented in these chapters meets the goals of the thesis.

7.2 Meeting the Goals: Contributions of the Thesis

A review of some state of the art machine learning classification and ensemble methods was given in chapter 2. Ensemble methods can be broadly classified into two groups:

- Methods based on using the same learning algorithms trained by manipulating the the training set or tuning parameters or some other way.
- Methods based on using different learning algorithms.

While ensemble methods such as bagging, boosting, and random forest that, manipulate the training set or tuning parameters have been well studied, methods that combine different learning algorithms have lagged behind. Possible reasons for this can be given as: the large number of learning algorithms available today makes it difficult to chose a reliable algorithm, some algorithms are difficulties to implement while at the same time some are computational very expensive to deploy even as stand-alone. However, in most interesting real applications, there is either too much or insufficient training data to manipulate when multiple sources of information need to be combined. In addition, these sources are often heterogeneous with different underlying theories and assumptions, making it difficult or inconvenient to use combination methods that implements the same learning algorithm. The thesis therefore focused on developing simple, fast, scalable, and accurate ensemble methods with heterogeneous base members.

A primary motivation of this dissertation was in learning large-scale distributed datasets in a reasonable time. To this end, a number of large-scale learning algorithms were generated during the course of writing this thesis and discussed in chapter 2. These algorithms have been shown to be very simple and extremely fast to train on very large datasets compared to traditional methods like SVM and random forest (Ngufor and Wojtusiak, 2013,

2014a; Ngufor et al., 2014). At the same time, these algorithms are also very accurate as demonstrated in chapter 4, where they were ranked in the top 5 algorithms by the Bayes active data and algorithm selection method described in chapter 3. This shows that an ensemble method based only on the algorithms generated by this thesis can be very fast, scalable to large data set and above all, simple and accurate.

Modern datasets are often characterized by noisy data points, missing values, poor labeling, imbalance, sample bias, and many other irregularities. Given the large number of classification algorithms available today, it is often a daunting task for a domain expert to decide on the best algorithm to apply on such datasets. There exist a number of approaches to deal with these problems. The thesis carefully addressed the limitations of these methods and proposed an alternative solution. Two simple, fast and effective active learning techniques were introduced in chapter 3 for the purpose of training a collection of classification algorithms using a small sample of carefully chosen data points. By actively engaging the classifiers in determining the best data points to train on, these problems are maintained at a very minimal level. Further, the approach enables computationally expensive algorithms to be trained in a reasonable time which otherwise would be impossible (Ngufor and Wojtusiak, 2014b). With these favorable conditions, the thesis was able to fairly compare and rank the algorithms. Through a Bayes decision making process using a specified or arbitrary decision thresholds, several performance measures were computed for comparing and ranking the classifiers. The top k algorithms can then be selected for ensemble learning.

The literature on ensemble learning methods in machine learning is very extensive. Even though a large majority of the methods are based on combining classifiers of the same type, an appealing property lacking in these approaches is the ability to incorporate supplementary information to help correct for misclassifications by the the base classifiers. This thesis addressed this issue in chapter 4 by introducing a generative Bayesian graphical model and inference to simultaneously infer hidden information from the problem domain not available to the classifiers at training time and to combine the decisions of the classifiers.

Specifying the appropriate structure that models the domain problem is a critical step in Bayesian probabilistic networks. Two model structures were explored in chapter 4: an evidence graphical model for inference of the hidden information or classification evidence and an ensemble graphical model for integrating the decisions of the base classifiers. Because of the strong coupling between the two models, a joint inference may increase computational cost. The thesis therefore implements the computationally less expensive and simple decoupled model structure and reserved the coupled structure for future work.

A variational Bayesian factor common spatial pattern (VBFACSP) was proposed for the graphical evidence model. Common spatial pattern (CSP) (Wu et al., 2009) is a very popular feature extraction technique commonly used in brain-computer interface systems for discriminating between positive and negative classes in electroencephalogram data. It discriminates between the two classes by maximizing the variance of one class and minimizing the variance of the other. In the proposed VBFACSP, the classification evidence depends on the true class variable, but when this dependency is relaxed, the standard variational Bayesian factor analysis (VBFA) is derived. Though VBFACSP may produce more accurate results, a full implementation may be computationally more expensive, so the less expensive VBFA was implemented.

Once estimated, the classification evidence becomes an observed variable in the ensemble graphical model. Three Bayesian inference methods using the classification evidence were proposed for the ensemble model: an expectation-maximization (EM) optimization method for discrete class predictions, a variational Bayesian inference method for discrete class predictions (VBD) and a variational Bayesian inference method for continuous class posterior probability predictions (VBC). The two variational Bayesian ensemble methods implements automatic relevance determination (ARD) priors which helps to prune irrelevant classifiers. VBD and VBC were shown in a series of numerical experiments in chapters 4, 5, and 6 to significantly improve classification accuracy compared to other methods. Clearly the classification evidence contributed a large part to the accuracy of these methods. However, the same thing cannot be said for the EM method as its performance appeared to be unstable

and inferior to the other ensemble methods on most of the datasets. Its failure to make use of the true class variable during training was cited as one possible reason for the poor performance.

An interesting application of the classification evidence was discovered in chapter 5. Chapter 5 addressed the problem of learning form large-scale distributed datasets with possible different observed feature vectors. Thus the datsets may be homogeneous or heterogeneous or both. A collective learning system was proposed for this task. At the base of collective learning are data sites or agents and tightly integrated with the Bayes active data and algorithm selection method described in chapter 3 and the evidence and ensemble graphical models described in chapter 4. Two parallel programming models were proposed for collective learning: a single-pass MapReduce model with no communication between agents was proposed for learning homogeneous agents while a High Performance Computing (HPC) model with minimal communication was proposed for heterogeneous agents. A chained MapReduce programming model was also proposed for heterogeneous agents, however its implementation was reserved for future investigation.

For heterogeneous agents, the classification evidence was modeled by a Group Factor Analysis model (Virtanen et al., 2011). GFA enables the solution of a completely new type of problem which is discovered in the thesis. The factors or "groups" in GFA described a dataset as a whole in a sparse way instead of the individual variables as in traditional factor analysis. In collective learning, some distributed datasets may be irrelevant containing noisy observations or features. Using GFA group sparsity property, the influence of such datasets on the learning procedure is minimized. Thus by introducing factors in GFA as classification evidence in collective learning with heterogeneous agents, the many problems associated with learning from incomplete feature spaces can be minimized or eliminated since the evidence offers a higher level of information about the whole distributed datasets. Further, since the dimension of the evidence can be far less than the dimension of the distributed datasets, communication cost can be maintain at a low and acceptable level.

The thesis demonstrated that global models generated through this technique were approximately exact. Meaning that, in terms of performance, models generated on a distributed datset with only a subset of the feature vectors were approximately the same as models generated on the whole datset using all feature vectors. The thesis used two benchmark datasets to demonstrate this concept. The datasets are trained in a centralized site using all features and in a distributed fashion where each data site contained only a random subset of the feature space. The results showed only a relatively small percentage reduction in accuracy for VBD and VBC compared to larger values for other methods.

The thesis also discovered another important property of the classification evidence. Based on assumptions made in the thesis, the classification evidence being hidden variables are considered non-restrictive or public and can be shared in distributed environments where movement of data is restricted by privacy concerns. This therefore negates the need for difficult and expensive privacy preserving machine learning methods (Kargupta et al., 2003).

7.3 Applications

This section describes some applications of the work presented in this thesis.

7.3.1 Predicting Adverse Drug Reaction

The main case study of the work presented in the thesis was described in chapter 6. The timely and accurate prediction of adverse drug reactions (ADRs) is increasingly becoming important as ADRs accounts for more than two million injuries, hospitalizations and deaths each year in the US alone (Lazarou et al., 1998).

The thesis described a novel systematic approach to the prediction of ADR. The Bayesian multiclassifier integration framework is applied to predict ADR from two spontaneous reporting databases: FAERS and MedEffect Canada having very different observations and feature vectors but containing information about the same drugs. Each spontaneous site

contains demographic information of individuals or "patients" who consumed the drugs and the reported side-effects. MedEffect Canada contains a much richer set of demographic variables than FAERS, thus hidden information from one site can be used to enrich information on the other. Supplementary information such as biological and chemical properties and known side-effects about each drug available in the two sites are pulled from DrugBank (Knox et al., 2011) and SIDER (Kuhn et al., 2010) repectively.

Based on the proven fact that, drugs with similar chemical and or biological properties most often exhibit similar ADRs (Liu et al., 2012; Fliri et al., 2005; Harpaz et al., 2011), the thesis formulates a systematic and structures method to predict ADRs. The approach is based on the assumption that if patients were to be grouped into homogeneous groups based on demographic characteristics such as age, gender, height, weight, geographic location were side-effect occurred, duration of use, etc., then similar patients should have similar outcomes when administered the same treatment. With this idea, chemical and biological properties of drugs are combined with demographic information of patients who consumed the drugs to contruct clusters of similar drugs and patients from which ensemble models are generated.

Figure 6.1 illustrates the series of systematic steps to detection and prediction of ADRs. Three main features of the approach stand out:

1. Sharing Classification Evidence. Classification evidence inferred from the two sites are shared using the HPC programming model described in chapter 5. The evidence is computed only for reports that show significant pattern of association or "signals" between a drug and an ADR.

2. Clustering. Similar drugs often exhibit similar ADRs and similar individuals often responds similarly to the same treatment. Thus by clustering drug-ADRs signals, the sensitivity of the models can be dramatically improved. Clustering may also generate the following benefits:

    - Increase sensitivity to the detection of rare side-effects.
    - Minimize the imbalance problem in classification.

- Relationships between known side-effects and reported side-effects. This could lead to the discovery of potentially new side-effects.
- Interesting relationships can be discovered by studying the demographic characteristics of patients, biological and chemical properties of drugs, and the side-effects within a cluster.

3. Ensemble Models. The optimal Bayesian integration scheme described in the thesis is embedded within each cluster for prediction of ADRS.

In terms of sensitivity (and specificity), the application of the ensemble methods introduced in the thesis for ADR prediction was very effective in distinguishing the presence and absence of ADR for a drug based on patient demographic characteristics and the biological and chemical properties of the drug. Thus given a new patient whose demographic information matches the characteristics of the training set, a doctor can determine with high accuracy the likelihood of the patient experiencing a given side-effect or potentially new side-effect based on these features.

7.3.2 Other Applications

There are many other possible applications of the Bayesian multiple classifier integration framework presented in this thesis.

Even though the ensemble methods were generated using heterogeneous set of base classifiers, the appraoch can also be advantageously applied to combine base classifiers of the same type such as in bagging and booting methods. Thus the method can be used in many cases were a classification algorithm is required.

The concept of classification evidence can be applied to boost the performance of many systems: the evidence can be used to determine a higher level of item similarity in collaborative filtering and hence more accurate recommender systems. Equally, the classification evidence can be used to explain or predict relationships between items such as online dating. Some hidden characteristics about behaviors, interests, physical characteristics , etc. can

be inferred through the evidence model and used to improve dating relationship predictive models.

7.4 Suggestions for Future Work

A few research issues were stated in some of the chapters of the thesis but were reserved for future work. This section briefly expands on some of these issues and other research topics stemming out from this thesis and warranting future investigation.

Chapter 2 presented a number of large-scale parallel efficient learning algorithms developed in the course of writing this thesis. To better understand how they work, a study of their theoretical performance is needed. Thus rigorous theoretical investigations are required to establish convergence, stability, generalization bounds and general asymptotic properties of the methods. This may be pursued through the development and application of concentration inequalities and empirical processes theory (Bousquet, 2002).

Chapter 4 presented different versions of the classification evidence and ensemble graphical models. Some are simple and computationally less expensive to implement while others are complex and expensive. For reasons explained, the thesis implemented only the simple and less expensive models. However, these algorithms may be sub-optimal and in some non-standard conditions, results from these models may be misleading (overfitting). Some of the methods assumed linear relationships while others failed to model any dependencies that may exist among some components of the system. For example, a VBFACSP model is assumed for the classification evidence model. This is a linear model with a Gaussian distributional assumption. This may be impractical in some real applications. A better and optimal approach is to consider robust kernel based methods which makes no distributional assumptions and model non-linear relationships. Promising directions for a more robust evidence model is to investigate kernel based VBFACSP or kernel based non-negative matrix factorization methods for feature extraction (Lee et al., 2009) as has been done for principal components (Lawrence, 2005). The price paid for adopting kernel methods is that they

are computationally very expensive to deploy. However, based on techniques used in the thesis generated articles Ngufor and Wojtusiak (2014a); Ngufor et al. (2014), it is possible to transform or approximate the kernels to reduce the computational cost.

The EM ensemble method presented in chapter 4 did not perform very well compared to VBD or VBC even though it uses the classification evidence. Its performance was very unstable. This indicates that a poorly designed ensemble scheme cannot produce good results. Based on limitations that were addressed in the thesis, there are ways to improve the EM ensemble method. The EM method can be re-implemented by introducing hidden variables or using Bayes theorem to somehow transform the algorithm to depend directly on the true class variable. Another possibility is to investigate how to use class posterior probabilities instead of the discrete class predictions.

In learning heterogeneous data sites in collective learning presented in chapter 5, the classification evidence model was approximated by GFA (Virtanen et al., 2011). The factors or groups were simply computed independently for each distributed dataset. Thus the group wise sparsity property may be lost and the dependencies between the dataset may not be sufficiently modeled. A full distributed implementation of GFA is required to take advantage of these properties. The main difficulty in computing GFA for a distributed dataset is how to represent each factor as a distributed sparse matrix and how to compute the covariance matrix updates in Algorithm 4 for distributed datasets with different feature spaces. The later case may be resolved by modifying the single-pass covariance matrix formula presented in section 2.3.2 to handle distributed datasets with different features. Group wise sparsity is just a direct consequence of ARD priors and can be equally modified for distributed datasets.

Finally, the collective learning system presented in chapter 5 artificially modeled the distributed data sites as "agents", the data sites were called agents for simplicity. The obvious next step is to give these "agents" their desired artificial intelligence properties like autonomy, sociability, reactivity and pro-activeness (Stone and Veloso, 2000). This is the case of distributed data mining supported by artificial intelligence.

Appendix A: Proof of Single-Pass covariance matrix formula

Proof. The sample mean of Proposition 2 is a classical result and the proof will be omitted. The proof for the scatter matrix is carried out by induction on k the number of data-blocks, however, some of the algebra involved will be omitted.

For the base case when $k = 2$, i.e, $D = D_1 \cup D_2$, by the definition of the scatter matrix, the left hand side of equation 2.16 can be written as

$$S = \sum_{X \in D} (X - \bar{X})(X - \bar{X})^T = S_{D_1} + S_{D_2},$$

where $S_{D_i} = \sum_{X \in D_i} (X - \bar{X})(X - \bar{X})^T,\ i = 1, 2$ and $\bar{X} = (n_1 \bar{X}_1 + n_2 \bar{X}_2)/n$.

Now expand each S_{D_i} and substitute for the mean. After some simplification this gives

$$S_{D_1} = \sum_{X \in D_1} (X - \bar{X}_1)(X - \bar{X}_1)^T$$

$$+ \frac{n_1 n_2^2}{n^2} (\bar{X}_1 - \bar{X}_2)(\bar{X}_1 - \bar{X}_2)^T .$$

The derivation of the above expression makes use of the fact that

$$\frac{n_1 n_2^2}{n^2} \sum_{X \in D_1} (X - \bar{X}_1)(\bar{X}_1 - \bar{X}_2)^T = 0 \ \text{ and}$$

$$\frac{n_1 n_2^2}{n^2} \sum_{X \in D_1} (\bar{X}_1 - \bar{X}_2)(X - \bar{X}_1)^T = 0 .$$

A similar expression for S_{D_2} can also be derived. Thus

$$S = S_1 + S_2 + \frac{n_1 n_2}{n} (\bar{X}_1 - \bar{X}_2)(\bar{X}_1 - \bar{X}_2)^T .$$

Now for $k = 2$, the total number of distinct pairs of elements selected from the set (1,2) is

one. Hence $(\pi_1, \pi_2) = (1, 2)$ and the right hand side of equation 2.16 becomes

$$S = S_1 + S_2 + \frac{n_1 n_2}{n}(\bar{X}_1 - \bar{X}_2)(\bar{X}_1 - \bar{X}_2)^T$$

and so both sides of equation 2.16 are equal for $k = 2$.

Now, suppose that the proposition is true for $j = k \geq 2$ partitions of D, it remains to proof that the formula also holds for $j + 1$ partitions of D. For $j + 1$ partitions, S can be written as

$$\begin{aligned} S &= \sum_{i=1}^{j} \sum_{X \in D_i} (X - \bar{X})(X - \bar{X})^T \\ &\quad + \sum_{X \in D_{j+1}} (X - \bar{X})(X - \bar{X})^T \\ &= \sum_{i=1}^{j} S_{D_i} + S_{D_{j+1}} \qquad \text{(A.1)} \\ &= S_{D_j} + S_{D_{j+1}} \,. \end{aligned}$$

Another similar expansion is carried out for the $j + 1$'th scatter matrix to give

$$\begin{aligned} S_{D_{j+1}} &= \sum_{X \in D_{j+1}} (X - \bar{X}_{j+1})(X - \bar{X}_{j+1})^T \\ &\quad + \frac{n_{j+1}}{n^2} \left[\sum_{i=1}^{j} n_i (\bar{X}_{j+1} - \bar{X}_i) \cdot \sum_{i=1}^{j} n_i (\bar{X}_{j+1} - \bar{X}_i)^T \right] . \qquad \text{(A.2)} \end{aligned}$$

The product in the square bracket is made up of a total of j^2 terms of which j are of the form

$$V_{j+1,i} = \frac{n_{j+1} n_i^2}{n^2}(\bar{X}_{j+1} - \bar{X}_i)(\bar{X}_{j+1} - \bar{X}_i)^T \qquad \text{(A.3)}$$

and the remaining $j^2 - j = j(j-1)$ terms are of the form

$$V_{j+1,i,l} = \frac{n_{j+1} n_i n_l}{n^2} (\bar{X}_{j+1} - \bar{X}_i)(\bar{X}_{j+1} - \bar{X}_l)^T \quad i \neq l\,. \tag{A.4}$$

A similar expansion for each of the summands of the first term on the right hand side of equation A.1 gives

$$\begin{aligned} S_{D_i} = & \sum_{X \in D_i} (X - \bar{X}_i)(X - \bar{X}_i)^T \\ & + \frac{n_i}{n^2} \left[\sum_{\substack{l=1 \\ l \neq i}}^{j+1} n_l (\bar{X}_i - \bar{X}_l) \cdot \sum_{\substack{l=1 \\ l \neq i}}^{j+1} n_l (\bar{X}_i - \bar{X}_l)^T \right]. \end{aligned} \tag{A.5}$$

Since by Pascal's identity $\binom{j+1}{2} - \binom{j}{2} = \binom{j}{1} = j$, this indicates that each S_{D_i} contains j terms involving the $j+1$ mean $\bar{X}_{j+1}$. Precisely, these terms are: exactly one term of the form

$$V_{i,j+1} = \frac{n_i n_{j+1}^2}{n^2} (\bar{X}_i - \bar{X}_{j+1})(\bar{X}_i - \bar{X}_{j+1})^T \tag{A.6}$$

and $j-1$ terms of the form

$$V_{i,j+1,l} = \frac{n_i n_{j+1}^2 n_l}{n^2} (\bar{X}_i - \bar{X}_{j+1})(\bar{X}_i - \bar{X}_l)^T, \quad i \neq l\,. \tag{A.7}$$

This gives a total of j^2 terms of this form. By taking out all these terms from $S_{D_j} = \sum_{i=1}^{j} S_{D_i}$ and adding them to $S_{D_{j+1}}$, the remaining terms in S_{D_j} is simply the expansion of S for j partitions of D. But by the induction hypothesis, the formula holds for S_{D_j}.

At this point, only further algebra is required to complete the proof by showing that the updated $S_{D_{j+1}}$

$$S^*_{D_{j+1}} = S_{D_{j+1}} + \sum_{i=1}^{j} V_{i,j+1} + \sum_{i=1}^{j} \sum_{\substack{l=1 \\ l \neq i}}^{j} V_{i,j+1,l}$$

can be written in as

$$S_k + \sum_{\pi^*} \frac{n_{\pi_1} n_{\pi_2}}{n} (\bar{X}_{\pi_1} - \bar{X}_{\pi_2})(\bar{X}_{\pi_1} - \bar{X}_{\pi_2})^T \tag{A.8}$$

where $\sum_{\pi^*}$ is the summation over the selection of distinct pairs (π_1, π_2) with $\pi = i,\ i = 1, \ldots j$ and $\pi_2 = j + 1$. □

Appendix B: Expression for Kullback-Leibler (KL) divergence

The KL divergence between two K variate gaussian distributions $q(\boldsymbol{\theta}) = \mathcal{N}_k(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$ and $p(\boldsymbol{\theta}) = \mathcal{N}_K(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)$ is given by

$$KL\Big(q(\boldsymbol{\theta})\|p(\boldsymbol{\theta})\Big) = \int \log\left(\frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta})}\right) q(\boldsymbol{\theta})\, d\boldsymbol{\theta} \tag{B.1}$$

$$= \frac{1}{2}\left(\mathrm{tr}\left(\boldsymbol{\Sigma}_p^{-1}\boldsymbol{\Sigma}_q\right) + (\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)^\top \boldsymbol{\Sigma}_p^{-1}(\boldsymbol{\mu}_p - \boldsymbol{\mu}_q) - K - \log\left(\frac{|\boldsymbol{\Sigma}_q|}{|\boldsymbol{\Sigma}_p|}\right)\right). \tag{B.2}$$

A similar expression can be written for the KL of two gamma distributions. Using the distributions of the variational posteriors $q(\boldsymbol{\theta})$ given in Algorithm 4 and the true posteriors $p(\boldsymbol{\theta})$ (equations 4.8, 4.9, 4.10 and 4.11), the various components of the KL in equation 4.16 can be derived as:

$$\Big\langle KL\Big(q(\mathbf{e})\|p(\mathbf{e}|\boldsymbol{\Lambda})\Big)\Big\rangle_{q(\boldsymbol{\Lambda})} = -\frac{n}{2}\left(\log|\boldsymbol{\Sigma}_e| + \mathrm{tr}\left[\mathbf{I} - \boldsymbol{\Sigma}_e\right] - \frac{1}{n}\left(\mathbf{m}_e^T\mathbf{m}_e\right)\right)$$

$$\Big\langle KL\Big(q(\mathbf{A})\|p(\mathbf{A}|\boldsymbol{\alpha})\Big)\Big\rangle_{q(\boldsymbol{\alpha})} = -\frac{p}{2}\sum_{k=1}^{K}\left(\psi(\hat{a}_k^\alpha) - \log(\hat{b}_k\right) - \frac{1}{2}\left(\sum_{j=1}^{p}\log\left|\boldsymbol{\Sigma}_a^j\right| + p\,K\right.$$

$$\left. - \mathrm{tr}\left[\langle\boldsymbol{\alpha}\rangle\left(\sum_{j=1}^{p}\boldsymbol{\Sigma}_a^j + \mathbf{m}_a^T\mathbf{m}_a\right)\right]\right)$$

$$KL\Big(q(\boldsymbol{\alpha})\|p(\boldsymbol{\alpha})\Big) = \sum_{k=1}^{K}\left[\hat{a}_k^\alpha\log\hat{b}_k^\alpha - a_k^\alpha\log b_k^\alpha - \log\left(\frac{\Gamma(\hat{a}_k^\alpha)}{\Gamma(a_k^\alpha)}\right)\right.$$

$$\left. + b_k^\alpha\langle\alpha_k\rangle - \hat{a}_k^\alpha + (\hat{a}_k^\alpha - \alpha_k^\alpha)\left(\psi(\hat{a}^\alpha) - \log\hat{b}_k^\alpha\right)\right] \tag{B.3}$$

where

- $\mathbf{m}_e = \left\{m_e^i\right\}_i^n = \mathbf{X}\left\langle\boldsymbol{\phi}\right\rangle\left\langle\mathbf{A}\right\rangle\boldsymbol{\Sigma}_e$ is a $n \times K$ matrix whose rows are the means of the $q(e_i)$ distributions,
- $\mathbf{m}_a = \left\{m_a^j\right\}_j^p = \left\langle\mathbf{A}\right\rangle$ is a $p \times K$ matrix whose rows are the means of the $q(a_j)$ distributions,
- $\left\langle\boldsymbol{\alpha}\right\rangle = \mathbf{diag}\{\left\langle\alpha_k\right\rangle, k = 1, \ldots, K\}$,
- $\Gamma(\cdot)$ is the gamma function.

The expressions of $KL\Big(q(\boldsymbol{\phi})\|p(\boldsymbol{\phi})\Big)$ and $KL\Big(q(\boldsymbol{\Lambda})\|p(\boldsymbol{\Lambda})\Big)$ are very similar to $KL\Big(q(\boldsymbol{\alpha})\|p(\boldsymbol{\alpha})\Big)$.

Bibliography

J Alcalá-Fdez, L Sánchez, S García, MJ Del Jesus, S Ventura, JM Garrell, J Otero, C Romero, J Bacardit, VM Rivas, et al. Keel: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, 13(3):307–318, 2009.

Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2004.

E. Alpaydm. Combined 5× 2 cv f test for comparing supervised classification learning algorithms. *Neural computation*, 11(8):1885–1892, 1999.

John Robert Anderson, Ryszard Spencer Michalski, Ryszard Stanisław Michalski, Thomas Michael Mitchell, et al. *Machine learning: An artificial intelligence approach*, volume 2. Morgan Kaufmann, 1986.

Stuart Bailey, Robert Grossman, Harimath Sivakumar, and A Turinsky. Papyrus: a system for data mining over local and wide area clusters and super-clusters. In *Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM)*, page 63. ACM, 1999.

A Bate and SJW Evans. Quantitative signal detection using spontaneous adr reporting. *Pharmacoepidemiology and drug safety*, 18(6):427–436, 2009.

Andrew Bate, Marie Lindquist, IR Edwards, S Olsson, R Orre, A Lansner, and R Melhado De Freitas. A bayesian neural network method for adverse drug reaction signal generation. *European journal of clinical pharmacology*, 54(4):315–321, 1998.

Ron Bekkerman, Mikhail Bilenko, and John Langford. *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press, 2011.

Janine Bennett, Ray Grout, Philippe Pébay, Diana Roe, and David Thompson. Numerically stable, single-pass, parallel statistics algorithms. In *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pages 1–8. IEEE, 2009.

Peter J Bickel and Bo Li. Mathematical statistics. In *Test*. Citeseer, 1977.

Christopher M Bishop. Variational principal components. 1999.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

Zalán Bodó, Zsolt Minier, and Lehel Csató. Active learning with clustering. *Active Learning Challenge Challenges in Machine Learning, Volume 6*, page 141, 2011.

Olivier Bousquet. *Concentration inequalities and empirical processes theory applied to the analysis of learning algorithms*. PhD thesis, Ecole Polytechnique, 2002.

A.P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

P. Brazdil, C.G. Carrier, C. Soares, and R. Vilalta. *Metalearning: Applications to data mining*. Springer, 2008.

Gavin Brown. Ensemble learning. In *Encyclopedia of Machine Learning*, pages 312–320. Springer, 2010.

Yingyi Bu, Bill Howe, Magdalena Balazinska, and Michael D Ernst. Haloop: Efficient iterative data processing on large clusters. *Proceedings of the VLDB Endowment*, 3(1-2): 285–296, 2010.

Tung Bui and Jintae Lee. An agent-based framework for building decision support systems. *Decision Support Systems*, 25(3):225–237, 1999.

R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.

Tony F Chan, Gene H Golub, and Randall J LeVeque. *Updating formulae and a pairwise algorithm for computing sample variances.* Department of Computer Science, Stanford University, 1979.

Bing Cheng and D Michael Titterington. Neural networks: A review from a statistical perspective. *Statistical science*, pages 2–30, 1994.

Cheng Chu, Sang Kyun Kim, Yi-An Lin, YuanYuan Yu, Gary Bradski, Andrew Y Ng, and Kunle Olukotun. Map-reduce for machine learning on multicore. *Advances in neural information processing systems*, 19:281, 2007.

A. Cichocki and S. Amari. Families of alpha-beta-and gamma-divergences: Flexible and robust measures of similarities. *Entropy*, 12(6):1532–1568, 2010.

David C Classen, Stanley L Pestotnik, R Scott Evans, James F Lloyd, and John P Burke. Adverse drug events in hospitalized patientsexcess length of stay, extra costs, and attributable mortality. *Jama*, 277(4):301–306, 1997.

David Roxbee Cox and E Joyce Snell. *Analysis of binary data*, volume 32. Chapman & Hall/CRC, 1989.

Frank Critchley and Ian Ford. Interval estimation in discrimination: the multivariate normal equal covariance case. *Biometrika*, 72(1):109–116, 1985.

J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

Saso Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? *Machine learning*, 54(3):255–273, 2004a.

Saso Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? *Machine learning*, 54(3):255–273, 2004b.

Bradley Efron. The efficiency of logistic regression compared to normal discriminant analysis. *Journal of the American Statistical Association*, 70(352):892–898, 1975.

Jaliya Ekanayake, Hui Li, Bingjing Zhang, Thilina Gunarathne, Seung-Hee Bae, Judy Qiu, and Geoffrey Fox. Twister: a runtime for iterative mapreduce. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 810–818. ACM, 2010.

Seyda Ertekin, Jian Huang, Leon Bottou, and Lee Giles. Learning on the border: active learning in imbalanced data classification. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 127–136, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-803-9. doi: 10.1145/1321440.1321461. URL `http://doi.acm.org/10.1145/1321440.1321461`.

SJW Evans, Patrick C Waller, and S Davis. Use of proportional reporting ratios (prrs) for signal generation from spontaneous adverse drug reaction reports. *Pharmacoepidemiology and drug safety*, 10(6):483–486, 2001.

Kai Fan, Xingzhi Sun, Ying Tao, Linhao Xu, Chen Wang, Xianling Mao, Bo Peng, and Yue Pan. High-performance signal detection for adverse drug events using mapreduce paradigm. In *AMIA Annual Symposium Proceedings*, volume 2010, page 902. American Medical Informatics Association, 2010.

Anton F Fliri, William T Loging, Peter F Thadeio, and Robert A Volkmann. Analysis of drug-induced effect patterns to link structure and side effects of medicines. *Nature chemical biology*, 1(7):389–397, 2005.

Andrew Frank and Arthur Asuncion. Uci machine learning repository. 2010.

Juihsi Fu, Singling Lee, and Wangping Wu. Efficient active learning based on uncertain clusters. In *Technologies and Applications of Artificial Intelligence (TAAI), 2012 Conference on*, pages 157–164. IEEE, 2012.

A.E. Gelfand, B.K. Mallick, and D.K. Dey. Modeling expert opinion arising as a partial probabilistic specification. *Journal of the American Statistical Association*, 90(430):598–604, 1995.

Zoubin Ghahramani and Matthew J Beal. Variational inference for bayesian mixtures of factor analysers. In *NIPS*, pages 449–455, 1999.

Kathleen M Giacomini, Ronald M Krauss, Dan M Roden, Michel Eichelbaum, Michael R Hayden, and Yusuke Nakamura. When good drugs go bad. *Nature*, 446(7139):975–977, 2007.

Horacio González-Vélez, Mariola Mier, Margarida Julià-Sapé, Theodoros N Arvanitis, Juan M García-Gómez, Montserrat Robles, Paul H Lewis, Srinandan Dasmahapatra, David Dupplaw, Andrew Peet, et al. Healthagents: distributed multi-agent brain tumor diagnosis and prognosis. *Applied Intelligence*, 30(3):191–202, 2009.

Somesh Das Gupta. Some aspects of discrimination function coefficients. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 387–400, 1968.

J.A. Hanley. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 743:29–36, 1982.

Rave Harpaz, Hector Perez, Herbert S Chase, Raul Rabadan, George Hripcsak, and Carol Friedman. Biclustering of adverse drug events in the fda's spontaneous reporting system. *Clinical Pharmacology & Therapeutics*, 89(2):243–250, 2011.

Trevor Hastie, Robert Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction: with 200 full-color illustrations*. New York: Springer-Verlag, 2001.

Christian Hennig. Cluster-wise assessment of cluster stability. *Computational Statistics & Data Analysis*, 52(1):258–271, 2007.

Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006.

Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang. Extreme learning machine for regression and multiclass classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 42(2):513–529, 2012.

J. Huang and C.X. Ling. Using AUC and accuracy in evaluating learning algorithms. *Knowledge and Data Engineering, IEEE Transactions on*, 17(3):299–310, 2005.

T Jaakkola and M Jordan. A variational approach to bayesian logistic regression models and their extensions. In *Sixth International Workshop on Artificial Intelligence and Statistics*. Citeseer, 1997.

R.A. Jacobs. Methods for combining experts' probability assessments. *Neural computation*, 7(5):867–888, 1995.

Hillol Kargupta and Philip Chan. *Advances in distributed and parallel knowledge discovery*. MIT Press, 2000.

Hillol Kargupta, Ilker Hamzaoglu, and Brian Stafford. Scalable, distributed data mining-an agent architecture. In *KDD*, pages 211–214, 1997.

Hillol Kargupta, B Park, Daryl Hershberger, and Erik Johnson. Collective data mining: A new perspective toward distributed data mining. *Advances in Distributed and Parallel Knowledge Discovery*, (part II):131–174, 1999.

Hillol Kargupta, Kun Liu, and Jessica Ryan. Privacy sensitive distributed data mining from multi-party data. In *Intelligence and Security Informatics*, pages 336–342. Springer, 2003.

R.D. King, C. Feng, and A. Sutherland. Statlog: comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence an International Journal*, 9(3): 289–333, 1995.

Craig Knox, Vivian Law, Timothy Jewison, Philip Liu, Son Ly, Alex Frolkis, Allison Pon, Kelly Banco, Christine Mak, Vanessa Neveu, et al. Drugbank 3.0: a comprehensive resource for omics research on drugs. *Nucleic acids research*, 39(suppl 1):D1035–D1041, 2011.

Tim Kraska, Ameet Talwalkar, John C Duchi, Rean Griffith, Michael J Franklin, and Michael I Jordan. Mlbase: A distributed machine-learning system. In *CIDR*, 2013.

R. Krzysztofowicz and D. Long. Fusion of detection probabilities and comparison of multisensor systems. *Systems, Man and Cybernetics, IEEE Transactions on*, 20(3):665–677, 1990.

Michael Kuhn, Monica Campillos, Ivica Letunic, Lars Juhl Jensen, and Peer Bork. A side effect resource to capture phenotypic effects of drugs. *Molecular systems biology*, 6(1), 2010.

Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

T.C.W. Landgrebe, P. Paclik, and R.P.W. Duin. Precision-recall operating characteristic (p-roc) curves in imprecise environments. In *Proceedings of the 18th International Conference on Pattern Recognition-Volume 04*, pages 123–127. IEEE Computer Society, 2006.

J Langford, L Li, and A Strehl. Vowpal wabbit online learning project, 2007.

Harri Lappalainen and Antti Honkela. Bayesian non-linear independent component analysis by multi-layer perceptrons. In *Advances in independent component analysis*, pages 93–121. Springer, 2000.

Neil Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *The Journal of Machine Learning Research*, 6:1783–1816, 2005.

Jason Lazarou, Bruce H Pomeranz, and Paul N Corey. Incidence of adverse drug reactions in hospitalized patients: a meta-analysis of prospective studies. *Jama*, 279(15):1200–1205, 1998.

Hyekyoung Lee, Andrzej Cichocki, and Seungjin Choi. Kernel nonnegative matrix factorization for spectral eeg feature extraction. *Neurocomputing*, 72(13):3182–3190, 2009.

D.D. Lewis and W.A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, Inc., 1994.

Mei Liu, Yonghui Wu, Yukun Chen, Jingchun Sun, Zhongming Zhao, Xue-wen Chen, Michael Edwin Matheny, and Hua Xu. Large-scale prediction of adverse drug reactions using chemical, biological, and phenotypic properties of drugs. *Journal of the American Medical Informatics Association*, 19(e1):e28–e35, 2012.

Andrew W Lo. Logit versus discriminant analysis: A specification test and application to corporate bankruptcies. *Journal of Econometrics*, 31(2):151–178, 1986.

Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M Hellerstein. Graphlab: A new framework for parallel machine learning. *arXiv preprint arXiv:1006.4990*, 2010.

Jaakko Luttinen and Alexander Ilin. Transformations in variational bayesian factor analysis to speed up learning. *Neurocomputing*, 73(7):1093–1102, 2010.

David JC MacKay. The evidence framework applied to classification networks. *Neural computation*, 4(5):720–736, 1992.

David JC MacKay et al. Bayesian nonlinear modeling for the prediction competition. *Ashrae Transactions*, 100(2):1053–1062, 1994.

James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, and Angela Hung Byers. Big data: The next frontier for innovation, competition, and productivity. *McKinsey Global Institute*, pages 1–137, 2011.

G. McLachlan and D. Peel. *Finite mixture models*, volume 299. Wiley-Interscience, 2000.

T. Menzies, J. DiStefano, A. Orrego, and R. Chapman. Assessing predictors of software defects. In *Proc. Workshop Predictive Software Models*, 2004.

Che Ngufor and Janusz Wojtusiak. Learning from large-scale distributed health data: An approximate logistic regression approach. *ICML 13: Role of Machine Learning in Transforming Healthcare*, 2013.

Che Ngufor and Janusz Wojtusiak. Extreme logistic regression. *Advances in Data Analysis and Classification (ADAC), Springer (Accepeted with Revisions)*, 2014a.

Che Ngufor and Janusz Wojtusiak. Learning from large distributed data: A scaling down sampling scheme for efficient data processing. *International Journal of Machine Learning and Computing (IJMLC)*, 4(3):216–224, 2014b.

Che Ngufor, Janusz Wojtusiak, Andrea Hooker, Talha Oz, and Jack Hadley. Extreme logistic regression: A large scale learning algorithm with application to prostate cancer mortality prediction. In *The Twenty-Seventh International Flairs Conference*, 2014.

G. Ostrouchov, W.-C. Chen, D. Schmidt, and P. Patel. Programming with big data in r, 2012. URL `http://r-pbd.org/`.

Sean Owen, Robin Anil, Ted Dunning, and Ellen Friedman. *Mahout in action*. Manning, 2011.

Andreas Prodromidis, Philip Chan, and Salvatore Stolfo. Meta-learning in distributed data mining systems: Issues and approaches. *Advances in distributed and parallel knowledge discovery*, 3, 2000.

Foster John Provost and Bruce G Buchanan. Inductive policy: The pragmatics of bias selection. *Machine Learning*, 20(1-2):35–61, 1995.

John Ross Quinlan. *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann, 1993.

Alvin C Rencher and G Bruce Schaalje. *Linear models in statistics*. Wiley-Interscience, 2008.

Lior Rokach. Ensemble methods for classifiers. In Oded Maimon and Lior Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 957–980. Springer US, 2005. ISBN 978-0-387-24435-8. doi: 10.1007/0-387-25465-X_45. URL `http://dx.doi.org/10.1007/0-387-25465-X_45`.

Lior Rokach. *Pattern classification using ensemble methods*, volume 75. World Scientific, 2010.

N. Roy and A. McCallum. Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, 2001.

Pabitra Mitra Sankar K. Pal. *Pattern recognition algorithms for data mining: scalability, knowledge discovery and soft granular computing*. CRC Press, 2004.

Andrew Ian Schein. *Active learning for logistic regression*. PhD thesis, University of Pennsylvania, 2005.

CE Schumer and Carolyn B Maloney. Your flight has been delayed again: flight delays cost passengers, airlines, and the us economy billions. *The US Senate Joint Economic Committee*, 2008.

Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 2010.

H.S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM, 1992.

Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

Ana Szarfman, Stella G Machado, and Robert T ONeill. Use of screening algorithms and computer systems to efficiently signal higher-than-expected combinations of drugs and events in the us fdas spontaneous reports database. *Drug Safety*, 25(6):381–392, 2002.

Kai Ming Ting and Ian H Witten. Issues in stacked generalization. *arXiv preprint arXiv:1105.5466*, 2011.

Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.

Jan Tozicka, Michael Rovatsos, and Michal Pechoucek. A framework for agent-based distributed machine learning and data mining. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 96. ACM, 2007.

Eugene P van Puijenbroek, Andrew Bate, Hubert GM Leufkens, Marie Lindquist, Roland Orre, and Antoine CG Egberts. A comparison of measures of disproportionality for signal detection in spontaneous reporting systems for adverse drug reactions. *Pharmacoepidemiology and drug safety*, 11(1):3–10, 2002.

Vladimir N Vapnik. Statistical learning theory. 1998.

Seppo Virtanen, Arto Klami, Suleiman A Khan, and Samuel Kaski. Bayesian group factor analysis. *arXiv preprint arXiv:1110.3204*, 2011.

Z. Wang and Y.C.I. Chang. Marker selection via maximizing the partial area under the roc curve of linear risk scores. *Biostatistics*, 12(2):369–385, 2011.

Gerhard Weiß. A multiagent perspective of parallel and distributed machine learning. In *International Conference on Autonomous Agents: Proceedings of the second international conference on Autonomous agents*, volume 10, pages 226–230, 1998.

Janusz Wojtusiak, Ryszard S Michalski, Kenneth A Kaufman, and Jaroslaw Pietrzykowski. The aq21 natural induction program for pattern discovery: initial version and its novel features. In *Tools with Artificial Intelligence, 2006. ICTAI'06. 18th IEEE International Conference on*, pages 523–526. IEEE, 2006.

Janusz Wojtusiak, Tobias Warden, and Otthein Herzog. Machine learning in agent-based stochastic simulation: Inferential theory and evaluation in transportation logistics. *Computers & Mathematics with Applications*, 64(12):3658–3665, 2012.

David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996.

Wei Wu, Zhe Chen, Shangkai Gao, and Emery N Brown. A probabilistic framework for learning robust common spatial patterns. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 4658–4661. IEEE, 2009.

Jerry Ye, Jyh-Herng Chow, Jiang Chen, and Zhaohui Zheng. Stochastic gradient boosted distributed decision trees. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 2061–2064. ACM, 2009.

Chengqi Zhang, Zili Zhang, and Longbing Cao. Agents and data mining: Mutual enhancement by integration. In *Autonomous Intelligent Systems: Agents and Data Mining*, pages 50–61. Springer, 2005.

Harry Zhang. The optimality of naive bayes. *AA*, 1(2):3, 2004.

BIOGRAPHY

Che Ngufor received his B.sc in Mathematics and Computer Sciences from the University of Dschang, Cameroon in 2004. He taught mathematics and physics at Our Lady of Lourdes College Mankon, Bamenda for two years before moving to the US where he obtained his M.sc in Mathematics from Tennessee Technological University, Cookevile, TN in 2008. In his masters thesis, he investigated the design of efficient and robust preconditioners for groundwater flow.

In the fall of 2009, Che joined the PhD program in Computational Sciences and Informatics then offered by the Department of Computational and Data Sciences (CDS) at George Mason University, Fairfax, VA. In 2011, CDS was merged with the Department of Physics and Astronomy to form the School of Physics, Astronomy, and Computational Sciences. His initial concentration in the program was Computational Mathematics, but quickly switched to Computational Statistics. Currently his main areas of research include: Computational Mathematics and Statistics, Medical Informatics, Distributed and Parallel Machine Learning, and Big Data Analystics.

Che plans to continue doing research and in particular on innovative ideas on Big Data. He is passionate about designing, improving, and using state of the art machine learning techniques to solve outstanding Big Data problems in areas such as healthcare, drug discovery, computational biology, operations research, and many other areas where he believes machine learning can make a difference. In this direction, after graduation, his immediate plan is to find a position as a data scientist in an environment that promotes scientific research.