Learning Bayesian Networks based on Optimization Approaches

Sona Taheri

This thesis is submitted in total fulfilment of the requirement for the degree of Doctoral of Philosophy

School of Science, Information Technology
and Engineering (SITE)
University of Ballarat

PO Box 663
University Drive, Mount Helen
Ballarat, VIC 3353, Australia.

Thesis Supervisors:
Dr. Musa Mammadov, Assoc Prof. Adil Bagirov

October 2012

Abstract

Learning accurate classifiers from preclassified data is a very active research topic in machine learning and artificial intelligence. There are numerous classifier paradigms, among which Bayesian Networks are very effective and well known in domains with uncertainty. Bayesian Networks are widely used representation frameworks for reasoning with probabilistic information. These models use graphs to capture dependence and independence relationships between feature variables, allowing a concise representation of the knowledge as well as efficient graph based query processing algorithms. This representation is defined by two components: structure learning and parameter learning. The structure of this model represents a directed acyclic graph. The nodes in the graph correspond to the feature variables in the domain, and the arcs (edges) show the causal relationships between feature variables. A directed edge relates the variables so that the variable corresponding to the terminal node (child) will be conditioned on the variable corresponding to the initial node (parent). The parameter learning represents probabilities and conditional probabilities based on prior information or past experience. The set of probabilities are represented in the conditional probability table. Once the network structure is constructed, the probabilistic inferences are readily calculated, and can be performed to predict the outcome of some variables based on the observations of others. However, the problem of structure learning is a complex problem since the number of candidate structures grows exponentially when the number of feature variables increases.

This thesis is devoted to the development of learning structures and parameters in Bayesian Networks. Different models based on optimization techniques are introduced to construct an optimal structure of a Bayesian Network. These models also consider the improvement of the Naive Bayes' structure by developing new algorithms to alleviate the independence assumptions.

We present various models to learn parameters of Bayesian Networks; in particular we propose optimization models for the Naive Bayes and the Tree Augmented Naive Bayes by considering different objective functions.

To solve corresponding optimization problems in Bayesian Networks, we develop new optimization algorithms. Local optimization methods are introduced based on the combination of the gradient and Newton methods. It is proved that the proposed methods are globally convergent and have superlinear convergence rates. As a global search we use the global optimization method, AGOP, implemented in the open software library GANSO. We apply the proposed local methods in the combination with AGOP.

Therefore, the main contributions of this thesis include (a) new algorithms for learning an optimal structure of a Bayesian Network; (b) new models for learning the parameters of Bayesian Networks with the given structures; and finally (c) new optimization algorithms for optimizing the proposed models in (a) and (b). To validate the proposed methods, we conduct experiments across a number of real world problems.

Statement of Authorship

Except where explicit reference is made in the text of the thesis, this thesis contains no material published elsewhere or extracted in whole or in part from a thesis by which I have qualified for or been awarded another degree or diploma. No other persons work has been relied upon or used without due acknowledgement in the main text and bibliography of the thesis.

Sona Taheri

October 2012

Acknowledgement

I would like to acknowledge and express sincere appreciation to a number of people without whom this thesis might not have been written, and to whom I am greatly indebted.

I wish to express my deepest gratitude to my principal supervisor, Dr Musa Mammadov, for his excellent guidance, caring and patience, and for providing me with a great atmosphere for doing research.

I am also grateful to my co supervisor, Associate Professor Adil Bagirov, for his kind advice, support and expert guidance throughout this research.

I offer my special gratitude to the Head of the School of Science, Information Technology and Engineering (SITE), Professor John Yearwood, all SITE staff members, the staff of the Research and Graduate Studies Office, the International Students Programs and the Financial, Academic and Technical support of the University of Ballarat.

I am lucky enough to have the support of my helpful friends; I greatly value their friendship and appreciate their belief in me.

Last but not least; my sincere thankfulness goes to my family. The unequivocal inspiration and guidance from them kept me focused and motivated. I cannot express my gratitude to my mother in words, whose unconditional love has been my greatest strength. I am grateful to my hard working father (who has sacrificed his life for his children), my gracious sisters and their dear families and my caring brother, for their constant love and compassion. I especially thank my precious in-laws who have given me their encouragement and support. This thesis could not have been accomplished without the love and constant devotion of my gentle husband, Sattar, who has always been there for me. He has selflessly endured all my fears and tears, and the countless hours of my necessary solitude. My heartfelt expression of gratitude does not suffice; it is simply beyond words.

Most of all I offer thanks for God's divine bounty that continues to make the impossible possible.

Dedication

In memory of my mother who supported my well being with her selfless love

To my father who has been a source of encouragement and inspiration to me

To my husband who has shown extraordinary patience, understanding

and resilience through my perpetual quest for further study

Contents

Chapter 1

Introduction

The task of data classification is to assign objects to several predefined categories. Currently, data classification is widely applied in numerous areas, and various techniques have been developed. It is divided to supervised data classification and unsupervised data classification. This research focuses on Bayesian Networks applied for supervised classifiers involving optimization approaches. Bayesian Networks are among the most commonly used methods in machine learning, and their use have received considerable attention [5, 6, 63, 68].

1.1 Background

The background of this research is presented in two subsections: Bayesian Networks and optimization.

1.1.1 Bayesian Networks

Bayesian Networks are also known as belief networks, causal probabilistic networks, and graphical probability networks. These networks have attracted much attention recently as a possible solution to complex problems related to decision support under uncertainty.

Let us first give a brief discerption to the conditional probability and the Bayes' Theorem as they are fundamental components of BNs. The probability of event A (an hypothesis) conditional on the occurrence of some event B (evidence) is denoted by $P(A|B)$. If we are counting sample points, we are interested in the fraction of events B for which A is also true, and we have

$$P(A|B) = \frac{P(A,B)}{P(B)},$$

this is often written in the form

$$P(A,B) = P(A|B)P(B),$$

and referred to as the product rule, this is in fact the simple form of the Bayes Theorem. It is important to realize that this form of the rule is not, as often stated, a definition. Rather, it is a theorem derivable from simpler assumptions. The Bayesian Theorem can be used to tell us how to obtain a posterior probability of a hypothesis A after observation of some evidence B, given the prior probability of A and the likelihood of observing B were A to be the case:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \tag{1.1}$$

Bayesian Networks (BNs) are directed acyclic graphical representations of probabilistic and conditional probabilistic relationships that are constructed by the set of variables. BNs are very successful in reasoning between the variables via conditional probabilities, and have long been used to encode expert knowledge about uncertain domains [54]. To augment available expert knowledge, many researches have been done to construct BNs. Constructing a BN from data is the learning process that is divided in two steps: structure learning and parameter learning. The structure learning process needs to select the arcs (edges) between variables (features) which connect child variables with the set of parent variables, and therefore construct a network from data. In addition to providing a network that will allow us to predict behavior under conditions that we have not seen, the structure can also incorporate domain expert knowledge to provide more reliable suggestions. Once the structure has been specified, then training the network is straightforward. It consists of computing probabilities and conditional probabilities called parameter learning. Given a set of variables, the more challenging problem is to learn a structure to present the connections among the feature variables.

The learning components of BNs, all together, define the joint probability distribution $P(\mathbf{X})$ for the set of variables $\mathbf{X} = \{X_1, X_2, ..., X_n\}$, where X_i denotes both the variable and its corresponding node. Let $Pa(X_i)$ denote the set of parents of the node X_i. When there is an arc from X_i to X_j, then X_j is called the child variable for the parent variable X_i. A conditional dependency $P(X_i|Pa(X_i))$ connects a child variable with a set of parent variables. In particular, given a structure, the joint probability distribution for $\mathbf{X}$ is given by

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|Pa(X_i))$$

Figure 1.1 illustrates a simple typical BN. It describes the causal relationships among the season of the year (X_1), whether rain falls (X_2) during the season, whether the sprinkler is on (X_3) during that season, whether the pavement would get wet (X_4), and whether the pavement would be slippery (X_5). The joint probability distribution for this sample is:

$$P(X_1, X_2, X_3, X_4, X_5) = P(X_1)P(X_2|X_1)P(X_3|X_1)P(X_4|X_2, X_3)P(X_5|X_4).$$

1.1.2 Optimization Problems

Finding parameters and structures in BNs leads to optimization problems. To solve these problems in this thesis, we apply local and global optimization methods. We introduce new globally convergent local optimization methods. The idea in these methods is based on the combination of the gradient and Newton methods. It is well known that the Newton methods have a quadratic convergence rate. However, they are very sensitive to initial points which

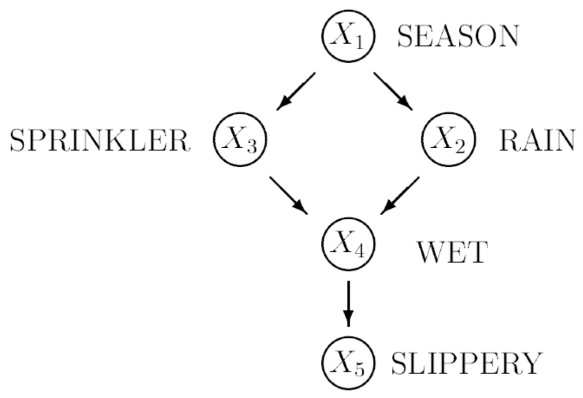


Figure 1.1: A Simple Bayesian Network

often lead to failure in practical applications. On the other hand, the gradient methods are globally convergent but they have a linear convergence rate. The proposed methods in this thesis are globally convergent and, at the same time, have high convergence rates. As a global optimization method, we apply the AGOP method introduced in [86, 87]. This algorithm was designed for optimization problems with box constraints. It uses a line search mechanism where the descent direction is obtained via a dynamic systems approach. It is applicable to a wide range of optimization problems requiring only function evaluations to work. We apply this global optimization method in conjunction with the newly suggested local optimization methods.

1.2 Motivation for the Research

The reasons for our choice of BNs are multiple: Firstly, they can encode dependencies among all variables; therefore they readily handle situations where some data entries are missing. BNs are also used to learn causal relationships, and hence can be applied to gain understanding about a problem domain and to predict the consequences of intervention. Moreover, since BNs in conjunction with Bayesian statistical techniques have both causal and probabilistic semantics, they are an ideal representation for combining prior knowledge and data. In addition, BNs in conjunction with Bayesian statistical methods offer an efficient and principal approach for avoiding the over fitting of data [92].

There are several difficulties when applying BNs, which are mainly related to learning process. Learning parameter with a given structure is one difficulty, whereas learning structure, itself, is another problem. The structural BN learning is much harder problem compared to parameter learning since the number of candidate networks grows exponentially when the number of variables increases. In fact, it has been proved that learning an optimal BN is an NP-hard problem [27, 55]. However, research in this direction is essential because of its enormous usefulness, as much for end user applications.

1.3 Outline of the Thesis

This thesis focuses on BN models; in particular structure learning and parameter learning. We find structures and parameters in BNs by introducing different strategies. Since the definition of structures in BNs is a very difficult problem, priori or manually defined structures are commonly used models for BNs. Naive Bayes (NB) [75] is the most commonly used model of BNs due to its simple structure, fast learning and at the same time being able to provide quite high accuracy in many data classification problems. However, the strong assumption in the NB that all features are conditionally independent given the class is often violated in many real world applications. In this research, we introduce different methods in order to improve the performance of the NB. The first one is alleviating the feature independence assumption. We propose two new algorithms to eliminate this assumption. In the first algorithm, each feature depends on the class and at most one other feature. The dependency between features in this algorithm is found by using conditional probabilities. The second algorithm finds unrestricted dependencies between features iteratively. Each feature in this algorithm has the class and several features as parents. Some features could have a large number of parents, whereas others just have a few. The number of these parents is defined by the algorithm internally.

Another alternative without violating the feature independence assumption in the NB is using feature (attribute) weights. We present a novel attribute weighted NB by assigning weights to conditional probabilities. An objective function has been constructed based on the NB structure and the attribute weights. The number of weights for each attribute is considered as the number of class labels. These weights are considered in the form of powers to conditional attribute class probabilities. The weights, then, are found by using a local optimization method.

We also propose a new algorithm to find an optimal structure in BNs based on global optimization. Although a BN can represent arbitrary feature dependencies, learning an optimal BN from data is an NP-hard problem. To discover better structure, the application of global optimization methods is natural. We apply global optimization method in conjunction with the proposed local methods to find an optimal structure in a BN.

Once the best structure has been specified, then the network is trained by learning parameters. In this research, we introduce three different optimization models to find optimal values of the NB's parameters. We construct different objective functions with some unknown variables corresponding to class probabilities and conditional feature class probabilities. To optimize these functions to find optimal solutions, we apply newly developed local optimization methods.

Tree Augmented Naive Bayes (TAN) is another important model in BNs. Unlike the NB, the TAN [42] allows additional edges between features that capture correlations among them. In fact each feature has the class and at most one other feature as parents. Friedman et al. [42], showed that the TAN maintains the robustness of the NB, and at the same time displays better accuracy. The TAN approximates the dependency between features by using a tree structure imposed on the NB structure. In this research, we apply a similar strategy to the NB's for learning parameters of the TAN. We consider an objective function involving the unknown variables for the class probabilities, and the optimal values of these variables are computed by applying the proposed local optimization method.

Finally, we introduce our novel local optimization algorithms to solve the models proposed for BNs efficiently. The proposed algorithms are based on the combination of the gradient and Newton based methods. Two different strategies are introduced in this research; in the first one, the step length is determined only along the anti-gradient direction, and in the second case the step length is considered along both directions.

Therefore, the following significant problems are formulated in this thesis:

1. To learn optimal structures in BNs by applying different strategies including optimization techniques.

2. To learn parameters in BNs; in particular the Naive Bayes and the Tree Augmented Naive Bayes, by using optimization formulation for finding the parameters of these models.

3. To develop new optimization methods to solve the optimization problems in 1 and 2 efficiently:

- Local methods: We mainly concentrate on the combination of the gradient method with the Newton

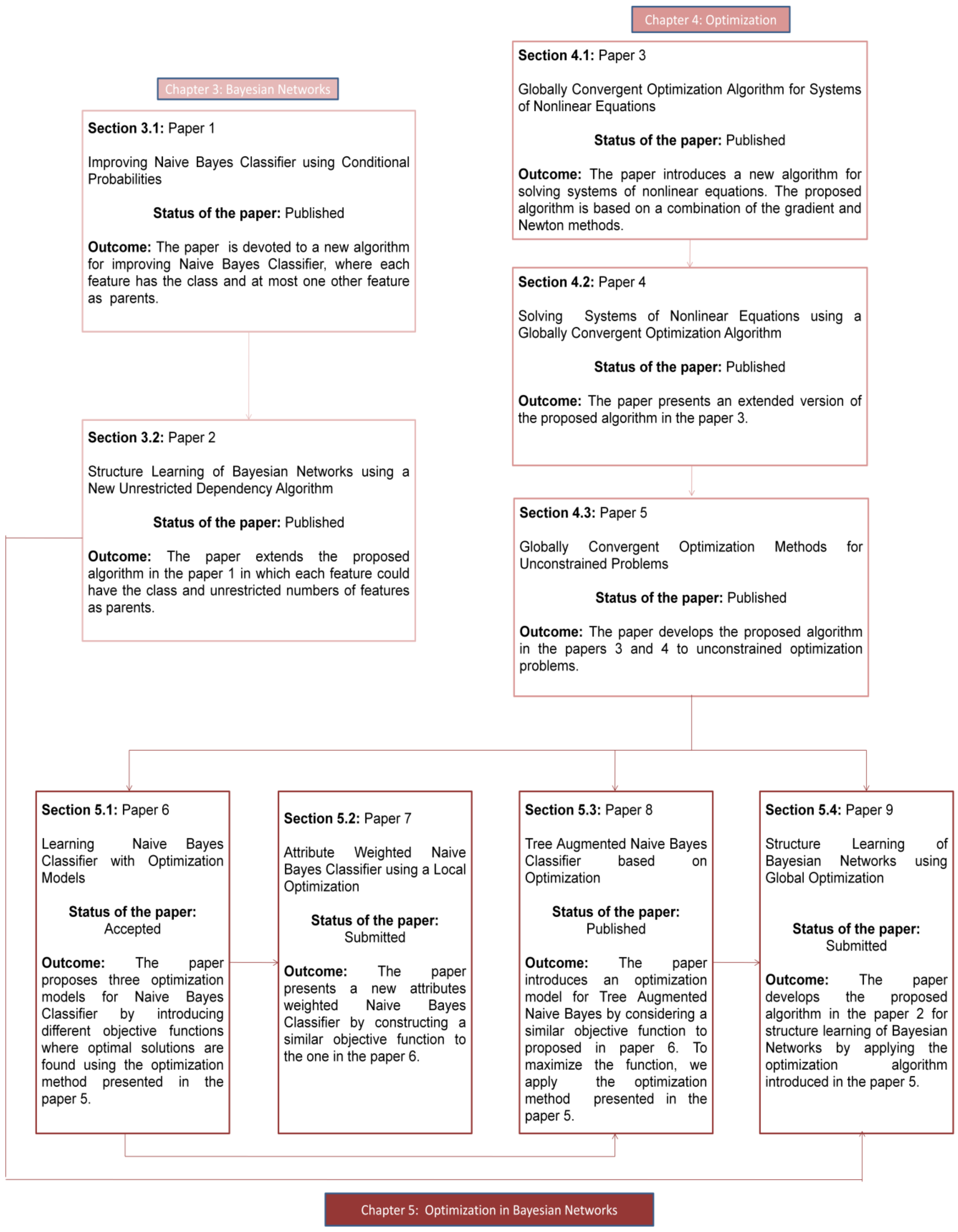
  based methods [88, 121, 122].

- Global methods: We apply the method AGOP introduced in [86, 87] in conjunction with the newly suggested local optimization methods.

4. Application of the developed models to the real world problems. We validate the proposed algorithms for BNs using real world data sets taken from the UCI machine learning repository and the LIBSVM.

1.4 Structure of the Thesis

In this section a brief description of the format of the presented thesis (PhD by publication) is given. An introduction is followed by explication of nine papers having different status of publication. A literature review of BNs, local and global optimization methods and application of the optimization in BNs are provided in next Chapter. Chapters 3 to 5 present the original formats of published, or submitted to publish of our research work related to BNs, optimization and application of optimization in BNs. The semantic structure for Chapters 3 to 5 with the links between the corresponding papers is shown on the next page in a flow chart. In Chapter 3, we introduce new algorithms for learning BNs. In Chapter 4, we propose new local optimization algorithms for solving systems of nonlinear equations and unconstrained optimization problems. Chapter 5 presents applications of the optimization in BNs. We conclude the thesis by providing final remarks and recommendation for future work.



Chapter 3: Bayesian Networks

Section 3.1: Paper 1

Improving Naive Bayes Classifier using Conditional Probabilities

Status of the paper: Published

Outcome: The paper is devoted to a new algorithm for improving Naive Bayes Classifier, where each feature has the class and at most one other feature as parents.

Section 3.2: Paper 2

Structure Learning of Bayesian Networks using a New Unrestricted Dependency Algorithm

Status of the paper: Published

Outcome: The paper extends the proposed algorithm in the paper 1 in which each feature could have the class and unrestricted numbers of features as parents.

Chapter 4: Optimization

Section 4.1: Paper 3

Globally Convergent Optimization Algorithm for Systems of Nonlinear Equations

Status of the paper: Published

Outcome: The paper introduces a new algorithm for solving systems of nonlinear equations. The proposed algorithm is based on a combination of the gradient and Newton methods.

Section 4.2: Paper 4

Solving Systems of Nonlinear Equations using a Globally Convergent Optimization Algorithm

Status of the paper: Published

Outcome: The paper presents an extended version of the proposed algorithm in the paper 3.

Section 4.3: Paper 5

Globally Convergent Optimization Methods for Unconstrained Problems

Status of the paper: Published

Outcome: The paper develops the proposed algorithm in the papers 3 and 4 to unconstrained optimization problems.

Section 5.1: Paper 6

Learning Naive Bayes Classifier with Optimization Models

Status of the paper: Accepted

Outcome: The paper proposes three optimization models for Naive Bayes Classifier by introducing different objective functions where optimal solutions are found using the optimization method presented in the paper 5.

Section 5.2: Paper 7

Attribute Weighted Naive Bayes Classifier using a Local Optimization

Status of the paper: Submitted

Outcome: The paper presents a new attributes weighted Naive Bayes Classifier by constructing a similar objective function to the one in the paper 6.

Section 5.3: Paper 8

Tree Augmented Naive Bayes Classifier based on Optimization

Status of the paper: Published

Outcome: The paper introduces an optimization model for Tree Augmented Naive Bayes by considering a similar objective function to proposed in paper 6. To maximize the function, we apply the optimization method presented in the paper 5.

Section 5.4: Paper 9

Structure Learning of Bayesian Networks using Global Optimization

Status of the paper: Submitted

Outcome: The paper develops the proposed algorithm in the paper 2 for structure learning of Bayesian Networks by applying the optimization algorithm introduced in the paper 5.

Chapter 5: Optimization in Bayesian Networks

Chapter 2

Literature Review

In this chapter, we present a literature review of Bayesian Networks as well as other concepts and methods related to them which we require for our discussion in the latter part of the Thesis. After a brief description of data classification in Section 2.1, we describe Bayesian Network models in Section 2.2. Subsection 2.2.1 presents approaches to structure learning of Bayesian Network models. In Subsection 2.2.2, we discuss learning the parameters of Bayesian Networks once the structure is known. Some commonly used Bayesian Network models are explained in Subsection 2.2.3 to Subsection 2.2.5. We present several advantages of Bayesian Networks over alternative methods and also applications of them in Subsections 2.2.6 and 2.2.7, respectively. In Section 2.3, we review briefly optimization methods and application of them to Bayesian Networks. Finally, in Section 2.4, we conduct a brief review of discretization methods which we apply in our experiments.

2.1 Data Classification

Data classification is divided into two types: supervised data classification where labeled objects as training sets are required to build the classifier, and unsupervised data classification (clustering), where unlabeled data are fed to the learning system which then chooses an internal organization on its own. The main distinguish between them is that supervised data classification require a list of predefined classes at the beginning, while unsupervised data classification is given only unlabeled examples. Throughout this research we assume the supervised data classification.

The task of supervised data classification is to assign objects to several predefined categories. For instance, documents can be categorized according to their contents. Examples of classification applications include image and pattern recognition, medical diagnosis, loan approval, detecting faults in industry applications, and classifying financial market trends. Estimation and prediction maybe viewed as types of classification.

According to the number of classes, labels, of a classification problem, there are three main classification categories [80].

1. Binary classification: In this category, there are two classes, such as Yes or No. All examples in this setting will be assigned a positive (1) or negative (-1) integer based on whether or not they belong to the corresponding class.

2. Multi-class classification: In the real world, examples generally have different topics or belong to different classes based on their content. As multi-class setting attempts to classify examples based on their main topic, so that examples can belong to one and only one class. In order to deal with multi-class problems, different approaches have been explored in the literature. There are two types of approaches for multi-class classification. The first one is constructing and combining several binary classifiers, called decomposition approaches. Another one is considering all data in one optimization formulation, called single machine approaches. Several methods have been proposed for decomposition approaches, for instance, one-vs-all approaches [9, 106], all-vs-all approaches [42, 50], and error-correcting code approaches [2, 29, 32]. Unlike the decomposition approaches, the single machine approach attempts to solve a single optimization problem to find q functions simultaneously rather than combine the solutions to a collection of binary problems [129] and [132].

3. Multi-label classification: If an example can belong to more than one class, then we have a set of multi-labeled classification problems. There exists a number of multi-label

classification methods. Similar to multi-class classification methods, [128] group the existing multi-label classification methods into two main categories: problem transformation methods and algorithm adaptation methods [80].

2.2 Bayesian Networks

Bayesian Networks (BNs) are high level representations of probability distributions over a set of variables that are used for building a model of the problem domain. The benefit of BNs lies in the way such a model (structure) can be used as a compact representation for many naturally occurring and complex problem domains.

BNs were developed in the late 1970's to model the semantic and perceptual combination of evidence in reading. The capability for bidirectional inferences, combined with a rigorous probabilistic foundation, led to the rapid emergence of BNs at the early 1980's as the method of choice for uncertain reasoning in artificial intelligence, expert system, statistic and data mining [24, 53, 61, 100, 113]. For an introductory overview of BNs, we refer the reader to [25, 61, 100] and for a detailed analysis, to [53, 62, 101].

A BN is associated with a directed acyclic graph. The nodes in the graph correspond to the feature variables in the domain, and the arcs (edges) show the causal relationships between feature variables. The direction of the arrow indicates the direction of causality. Edges also determine some qualifying terms for nodes. When two nodes are joined by an edge, the causal node is called the parent of the other node, and another one is called the child. Other terminology you might encounter includes the term root node for any node without parents and leaf node for any node without children. Therefore, a graph $G = (V, E)$ is simply a collection of variables V and edges E between variables.

How one node influences another is defined by the conditional probabilities for the nodes, that describes the relationship between it and its parents. Conditional probabilities represent likelihoods based on prior information or past experience. For each parent and each possible state of that parent, there is a row in the conditional probability table (CPT) that describes the likelihood that the child node will be in some state. Nodes with no parents also have CPTs, but they are simpler and consist only of the probabilities for each state of the node under consideration.

A BN for a set of variables $\mathbf{X} = \{X_1, X_2, ..., X_n\}$ consist of a network structure B that encodes a set of conditional dependencies assertions about variables in $\mathbf{X}$, and a set P of local probability distributions associated with each variable. Together, these components define the joint probability distribution for $\mathbf{X}$. The network structure B is constrained to be acyclic. The nodes in B are in one to one correspondence with the variable $\mathbf{X}$. We use X_i to denote both the variable and its corresponding node. Let $Pa(X_i)$ denote the set of parents of the node X_i in $\mathbf{X}$ as well as variables corresponding to those parents. The lack of possible arcs in B encode conditional independencies. In particular, given the structure B, the joint probability distribution for $\mathbf{X}$ is given by

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|Pa(X_i)). \tag{2.1}$$

In the light of the above information, properties of BNs can be summarized as below [109]:

1. It has a set of variables and a set of directed edges between variables.
2. Each variable contains a finite set of mutually exclusive states.
3. The variables coupled with the directed edges construct a directed acyclic graph (DAG).
4. Each variable X_i, $1 \leq i \leq n$, with its parents has a conditional probability table (CPT) associated with it.
5. It has a joint probability distribution for $\mathbf{X} = \{X_1, X_2, ..., X_n\}$, given by formula (2.1).

If variable X_i, $1 \leq i \leq n$, does not have any parents, then the conditional probability table can be replaced by the probability $P(X_i)$. A graph is acyclic if there is no directed path $X_1 \rightarrow X_2 ... \rightarrow X_n$ such that $X_1 = X_n$.

Most of BNs researches can be put into two main categories. First, given a BN model and its parameters, how can the values of hidden (unobservable) variables be inferred from a set of observed variables. This process is referred to as BN inference [62]. Second, given a collection of data, what is the most appropriate BNs model that describes it. In this research we concentrate on the second category. BNs model identification can be separated into two tasks: structure learning and parameter learning. Structure learning is related to a graph and not to the values of the probabilities. Given a structure, parameter learning is related to find probabilities and conditional probabilities among variables.

There are four classes of learning BNs from data:

1. Known structure and complete data

In this case, the problem is to calculate the conditional probability tables of each node in the network from the complete data (parameter learning). This is a relatively easier problem and has been studied extensively [117].

2. Known structure and incomplete data

The problem of learning parameters for a fixed network in the presence of missing values or hidden variables studied by extending and adapting expectation-maximization (EM) algorithm [43] and by Gibbs sampling. Both of these algorithms use a basic strategy that is to estimate the missing data on the basis of available data and information about the missing data. Another approach, called bound and collapse (BC) [110], first bounds the set of possible estimates consistent with the available information by computing the optima of estimates that are gathered from all possible completions of the database constraint by the given pattern of the missing data and then collapses these bounds to a point estimate using information about the assumed pattern of missing data. Genetic algorithm [91] is used to evolve both the missing values and the structures to find an optimal BN.

3. Unknown structure and complete data

Our problem falls into this category in which we are given a complete data set and asked to generate the structure of BNs that fits the data the best.

4. Unknown structure and incomplete data

Since it is generally not feasible to compute exact solutions to this problem, approximate algorithms generally employed. It is first attacked by a gradient-based algorithm [107] by using structural EM (SEM) that optimizes parameters with structure search for model selection. Convergence to a sub-optimal network and need for heavy computation during the learning are major problems with the structural SEM algorithm. Another approach is to use above mentioned BC for incomplete data [110]. The problem of learning an optimal structure of a BN from incomplete data is also considered in [134].

The problem we attack falls into the third class in which we try to learn the structure of BNs by using the observed data.

2.2.1 Structure Learning

Structure learning is the task of finding out one graph model that best characterizes the true density of given data. Perhaps the most challenging task in dealing with a BN is learning the structure. However, research in this direction is essential because of its enormous usefulness, as much for end-user applications.

Structure learning can be categorized into two levels: micro-level (quantitative part) and macro-level (qualitative part). In the micro-level, structure learning cares about whether one edge in the graph should be existed or not. In this case, researchers usually employ the conditional independence test to determine the importance of edges [26, 101, 118]. In the macro-level, several candidate graph structures are known, and we need choosing the best one out. In order to avoid over fitting, model selection methods, such as Bayesian scoring function [28, 52], entropy-based method [56], and minimum descriptive length (MDL), etc [22, 42] are often used. Therefore, learning structure of BNs can be divided in two main categories:

1. Constraint-based Methods: Finding dependencies between the random variables; some well known algorithms are IC [101], PC [118], and recently TPDA of Cheng et al. [26].

2. Score-based Methods: Using heuristic searching methods to construct a model and then

evaluates it using a scoring method; we can cite Akaike Information Criteria (AIC) [1], Bayesian Information Criteria (BIC) [112], Normalized Minimum Likelihood (NML) [72–74], Mutual Information Tests (MIT) [21, 126] and Minimum Description Length (MDL) principal [22, 42], Bayesian Dirichlet (BD) score [28], Bayesian Dirichlet equivalence (BDE) [53], Bayesian Dirichlet equivalence uniform (BDEU) [14], BDgamma [12], CCL [48] and ACL [16].

Both of these categories have their advantage and disadvantage. Generally the first one is asymptotically correct when the probability distribution of data satisfies certain assumption, but as Cooper et al. pointed out in [56], conditional independency tests with large condition sets may be unreliable. The second one has less time complexity in the worst case, but it may not find the best solution due to its heuristic nature.

2.2.2 Parameter Learning

Given a structure, learning the parameters are related to finding probabilities and conditional probabilities. There are two approaches of parameter learning: generative and discriminative. In generative parameter learning, a model of joint probability of the features and corresponding class label are learnt and then prediction is performed by using the Bayes' rule to determine the class posterior probability. Maximum likelihood (ML) estimation is usually used to learn a generative classifier. Discriminative approaches model the class posterior probability directly. Hence, the class conditional probability is optimized when we learn the classifier which is the most important for classification accuracy.

There are some methods for finding parameters such as ML, ECL, ACL, and CGCL parameter learning [102]. Learning parameters from data, in detail, is discussed in [15, 116] and [11, 46, 125].

Some well known BN models are Naive Bayes [36, 75], Tree Augmented Naive Bayes [42], Super Parent BNs [69], and k-dependence BN [108] which will be presented in the following.

2.2.3 Naive Bayes Classifier

Naive Bayes classifier [75] has the simplest structure among BN models. It assumes that all features are conditionally independent given the class; it means that all features have only the class as a parent. The Naive Bayes (NB) is attractive as it has an explicit and sound theoretical basis which guarantees optimal induction given a set of explicit assumptions. There is a drawback, however, in that some of these assumptions will be violated in many induction scenarios. In particular, one key assumption that is frequently violated is that the features are independent with respect to the class. The NB has been shown to be remarkably robust in the face of many such violations of its underlying assumptions [33].

Let us denote the class of an observation $\mathbf{X}$ by C, where $C \in \{C_1, \cdots, C_q\}$. To predict the class of a test observation $\mathbf{X}$ by using Bayes' rule, the highest probability of

$$P(C = c|\mathbf{X} = \mathbf{x}) = \frac{P(C = c)P(\mathbf{X} = \mathbf{x}|C = c)}{P(\mathbf{X} = \mathbf{x})}, \tag{2.2}$$

should be found, where c represents a particular class label and $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ stands for a particular observed feature value. Since in the NB, features $X_1, X_2, ..., X_n$ are conditionally independent given the class C, the formula (2.2) could be written as

$$P(C = c|\mathbf{X} = \ \mathbf{x}) = \frac{P(C = c)\prod_{i=1}^{n} P(X_i = x_i|C = c)}{P(\mathbf{X} = \mathbf{x})}. \tag{2.3}$$

Therefore, the NB classifies an observation $\mathbf{X}$ by selecting

$$\arg \max_{1 \le k \le q} P(C_k|\mathbf{X}) \propto \arg \max_{1 \le k \le m} P(C_k) \prod_{i=1}^{n} P(X_i|C_k). \tag{2.4}$$

A sample of the NB classifier with n features is depicted in Figure 2.1.

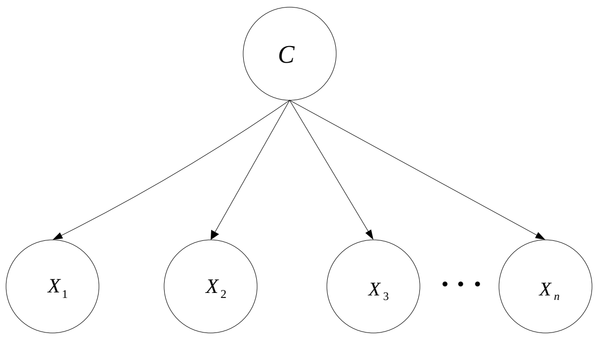


Figure 2.1: Naive Bayes

2.2.4 Tree Augmented Naive Bayes

Friedman et al. proposed Tree Augmented Naive Bayes (TAN) [42]. The TAN attempts to add edges to the NB. In fact each feature has the class and at most one other feature as parents. The TAN approximates the dependency between features by using a tree structure imposed on the NB structure. They showed that the TAN maintains the robustness and computational complexity of the NB, and at the same time displays better accuracy. Algorithm TAN consists of five main steps:

Algorithm. Tree Augmented Naive Bayes Algorithm

Step 1. Compute the conditional mutual information $I(X_i; Xj|C)$ for each pair of features $i \neq j$, using (2.6).

Step 2. Build a complete undirected graph in which the vertices are the features $X_1, ..., X_n$. Annotate the weight of an edge connecting X_i to X_j by $I(X_i; Xj|C)$.

Step 3. Build a maximum weighted spanning tree.

Step 4. Transform the resulting undirected tree to a directed one by choosing a root variable and setting the direction of all edges to be outward from it.

Step 5. Construct the TAN model by adding a vertex labled by C and adding an arc from C to each X_i.

This procedure reduces the problem of constructing a maximum likelihood tree to finding a maximal weighted spanning tree in a graph. The problem of finding such a tree is to select a subset of arcs from a graph such that the selected arcs constitute a tree and the sum of weights attached to the selected arcs is maximized.

The directions of edges in the TAN are crucial. In Step 4 of the TAN algorithm, a feature is randomly chosen as the root of the tree and the directions of all edges are set outward from it. Notice that the selection of the root feature actually determines the structure of the resulting TAN. Figure 2.2 shows a sample of the TAN with n features.

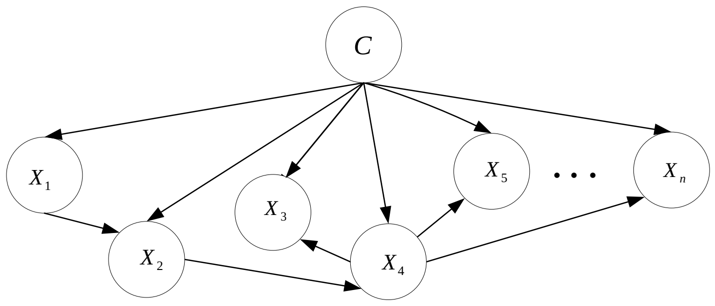


Figure 2.2: Tree Augmented Naive Bayes

In information theory, the mutual information of two nodes X_i, X_j is defined as

$$I(X_i; X_j) = \sum_{x_i \in X_i, x_j \in X_j} P(x_i, x_j) log \frac{P(x_i, x_j)}{P(x_i)P(x_j)} \tag{2.5}$$

and the conditional mutual information is defined as

$$I(X_i; X_j|C) = \sum_{x_i \in X_i, x_j \in X_j, c \in C} P(x_i, x_j, c) log \frac{P(c)P(x_i, x_j, c)}{P(x_i, c)P(x_j, c)}. \tag{2.6}$$

2.2.5 k-Dependency Bayesian Networks

In this subsection, we present an algorithm [108] which allows us to construct classifiers at arbitrary points (values of k) along the feature dependence, while also capturing much of the computational efficiency of the NB.

The k-dependence BN allows each feature X_i to have a maximum of k features as parents, i.e., the number of variables in $Pa(X_i)$ equals to $k + 1$; k features and 1 class. In the k-dependence BN, the number k is a priori chosen. According to the definition, the NB is a 0-dependence BN.

Algorithm. k-dependency Algorithm

Step 1. For each feature X_i, compute mutual information $I(X_i; C)$, using (2.5), where C is the class.

Step 2. Compute class conditional mutual information $I(X_i; Xj|C)$, using (2.6), for each pair of features X_i and X_j, where $i \neq j$.

Step 3. Let the used variable list, S, be empty.

Step 4. Let the BN being constructed begin with a single class node, C.

Step 5. Repeat until S includes all domain features:

5.1. Select feature X_{max}, which is not in S and has the largest value $I(X_{max}; C)$.

5.2. Add a node to the BN representing X_{max}.

5.3. Add an arc from C to X_{max} in BN.

5.4. Add $m = min(|S|, k)$ arcs from m distinct features X_j in S with the highest value for $I(X_{max}; Xj|C)$.

5.5. Add X_{max} to S.

Step 6. Compute the conditional probability tables inferred by the structure of the BN.

2.2.6 Advantages of Bayesian Networks

BNs offer several advantages over alternative modeling approaches [54]. The most important of these advantages are:

1. BNs encode dependencies among all variables, therefore, they readily handle situations where some data entries are missing.

For example, consider a classification or regression problem where two of the variables are strongly anti correlated. This correlation is not a problem for standard supervised learning, provided all inputs are measured in every case. When one of the inputs is not observed, however, many models will produce an inaccurate prediction, because they do not encode the correlation between the variables. BNs offer a natural way to encode such dependencies.

2. BNs can be used to learn causal relationships, and hence, can be used to gain understanding about a problem domain and to predict the consequences of intervention.

Learning about causal relationships are important for at least two reasons. The process is useful when we are trying to gain understanding about a problem domain, for example, during exploratory data analysis. In addition, knowledge of causal relationships allows us to make predictions in the presence of interventions. For example, a marketing analyst may want to know weather or not it is worthwhile to increase exposure of a particular advertisement in order to increase the sales of product. To answer this question, the analyst can determine whether or not the advertisement is a cause for increased sales, and to what degree. The use of BNs helps to answer such questions even no experiment about the effects of the increased exposure is available.

3. Because BNs in conjunction with Bayesian statistical techniques have both causal and probabilistic semantics, they are an ideal representation for combining prior knowledge (which often comes in causal form) and data.

Anyone who has performed a real world modeling task knows the importance of prior or domain knowledge, especially when data is scarce or expensive. The fact that some commercial systems can be built from prior knowledge alone is a testament to the power of prior knowledge. BNs have a causal semantics that makes the encoding of causal prior knowledge particularly straightforward. In addition, BNs encode the strength of causal relationships with probabilities.

4. BNs in conjunction with Bayesian statistical methods offer an efficient and principal approach for avoiding the over fitting of data.

2.2.7 Applications of Bayesian Networks

BNs have gained wide spread use in data mining [131]. They are offering a compact presentation of the interactions in a stochastic system by visualizing system variables and their dependencies, and therefore, they have been applied in electricity distribution system risk management [98]. BNs are risk modelings and analysis approaches that have been used for various types of analysis for different purposes in industrial sectors [4, 65, 98, 127].

BNs are hand-built by medical experts and later used to infer likelihood of different causes given observed symptoms,therefore, they have been used to build medical diagnostic systems [40]. Similar systems have also been built for diagnosing problems in factories and other mechanical systems [93]. In systems biology, a BN structure learning is used to infer different types of biological networks from data [95].

Burnell and Horvitz [19] show how BNs and logical approaches can be married for program understanding and debugging. Fung and Del Falvero described an application of BNs to information retrieval [44]. Hekerman et al. [51] show how BNs can be used for troubleshooting system failures, including software and hardware problems as well as mechanical failures of cars and jet engines.

With the advent of small, powerful computers and GUI interfaces, modeling tools based on BNs are seeing frequent use in real world applications including diagnosis [3], forecasting [49], automated vision [81], sensor fusion [115], and manufacturing control [135]. BNs are general modeling frameworks which have been extensively applied in business and finance, capital equipment, causal modeling, natural language processing, planning, psychology, scheduling, speech recognition, vehicle control, forecasting, channel coding, and commonsense reasoning problems [31, 94, 130].

2.3 Optimization in Bayesian Networks

Finding parameters and structures in BNs leads to some optimization problems. In this section, we present a brief review about these optimization problems [8, 13, 97, 120], and then we will give a literature review of optimization methods in BNs.

Optimization is a very important tool in decision science. To use it, we must first identify some objective, a quantitative measure of the performance of the system under study. The objective depends on certain characteristics of the system, called variables or unknowns. The goal is to find values of the variables that optimize the objective. Often the variables are restricted, or constrained, in some way. The process of identifying objective, variables, and constraints for a given problem is known as modeling. Construction of an appropriate model is the first step in the optimization process. If the model is too simplistic, it will not give useful insights into the practical problem, but if it is too complex, it may become too difficult to solve. Methods for solving optimization problems take two different approaches; local optimization and global optimization. In local optimization, the compromise is to give up seeking the optimal point, which minimizes the objective over all feasible points. Instead we seek a point that is only locally optimal, which means that it minimizes the objective function among feasible points that are near it, but is not guaranteed to have a lower objective value than all other feasible points. Local optimization methods are fast, can handle large scale problems, and are widely applicable. However, there are several disadvantages of local optimization methods, beyond (possibly) not finding a globally optimal solution. The methods require an initial guess for variables. This initial guess or starting point is critical, and can greatly affect the objective value of the local solution obtained. To find a global solution of the optimization problem, one can use global optimization methods. Global optimization is usually used for problems with a small number of variables, where computing time is not critical. However, it is worthwhile to use it in cases where the value of finding a global solution is very high, or the cost of being wrong about the reliability or safety is high.

Optimization problems could be classified according to the nature of the objective function and constraints (linear, nonlinear, convex), the number of variables (large or small), the smoothness of the functions (smooth or non-smooth), and so on. Some models contain variables that are allowed to vary continuously and others that can attain only integer values; these models are called mixed integer programming problems. Possibly an important distinction between problems is that have constraints on variables and those that do not. Constrained optimization problems arise directly in many practical applications. Unconstrained problems arise also as reformulations of constrained optimization problems, in which the constraints are replaced by penalization terms in the objective function that have the effect of discouraging constraint violations.

Optimization models constructed for BNs are usually constrained problems. To transform these problems in to unconstrained cases, most of the researchers in this area used the Lagrange method [10], [59], [123], [102], [105], [58], [82], [90], [96], [47], [17], while others applied the penalty method [114], [35], [123], [45], [83], [71] and [66].

To solve unconstrained problems, one can use local or global optimization methods. Existing local methods used for optimizing BNs are gradient-based methods; for instance, Wilson et al.[133], Kitakoshi et al. [70], Jing et al. [63], Burge [16], Bang and Gil [7] applied the Gradient method. Pernkopf and Wohlmayr [102], Hinsbergen et al. [57] and Greiner et al. [47] used the Conjugate Gradient method. Quasi Newton method has been used by Zhang et al. [82].

Recently, researchers were more interested using global search methods for BNs' optimization. Various problems to learn a structure of a BN using global optimization have been defined [99], [77], [79], [78], [67], [111], [60], [140], [109], [30], [84], [22], [85]. Park and Cho

[99] used the Genetic algorithm to optimize the structure in BNs and make a model more efficient. A method of finding the most probable structure of BNs based on the intelligent search made by the Genetic algorithm has been introduced by Larranaga et al. [79]. The papers [77], [78], [103], [137], [136], and [37] present several new approaches based on the Genetic algorithm to find the best BNs' structure among alternative structures. Work by Kabli et al. [67] illustrates a novel method for finding the structure of BNs using the Genetic algorithm. They proposed a method that uses chain structures as a model for BNs that can be constructed from given node orderings. The simulated annealing approaches to learning the structure in BNs have been studied, for example, in [111], [60], [119]. Application of the Particle Swarm optimization to discover the best structure of BNs is studied in [140], [109]. Daly and Shen [30] applied the Ant Colony optimization to the problem of learning BNs' structure that provides a good fit to data. The papers [84], [20], [30] propose BNs' structure learning algorithms based on the Ant Colony optimization. Marinescu and Dechter [85] and Cano et al. [23] applied the Branch and Bound method to learn the structure of BNs.

2.4 Discretization of Continuous Features

BNs learning needs to estimate probabilities and conditional probabilities for each class and feature-class from data set. For a qualitative feature, its relevant probabilities can be estimated from the corresponding frequencies. For a quantitative feature, its relevant probabilities can be estimated if we know the probability distributions from which the quantitative values are drawn. However, those distributions are usually unknown for real world data. Thus how to deal with quantitative features is a key problem in BNs learning. Typically, there are two approaches to tackling this problem.

The first approach is probability density estimation that makes assumptions about the probability density function of a quantitative feature given a class. The relevant probabilities can then be estimated accordingly. For instance, a conventional approach is to assume that a quantitative features probability within a class has a normal distribution. This assumption is made because a normal distribution may provide a reasonable approximation to many real world distributions [64], or because the normal distribution is perhaps the most well studied probability distribution in statistics [89].

A second approach is discretization. Under discretization, a qualitative feature is created for a quantitative feature. Each value of the qualitative feature corresponds to an interval of values of the quantitative feature. The resulting qualitative features are used instead of the original quantitative features to train a classifier. Since the probabilities of a qualitative feature can be estimated from its frequencies, it is no longer necessary to assume any form of distributions for the quantitative features.

For BNs learning, discretization is more popular than assuming probability density function. The main reason is that the true density is usually unknown for real world data, and therefore, BN classifiers with discretization tend to achieve lower classification error than those with unsafe probability density assumptions [34].

A typical discretization process broadly consists of four steps:

1. Sorting the continuous values of the feature to be discretized.
2. Evaluating a cut-point for splitting or adjacent intervals for merging.
3. According to some criterion, splitting or merging intervals of continuous value.
4. Finally stopping at some point.

After sorting, the next step in the discretization process is to find the best cut-point to split a range of continuous values or the best pair of adjacent intervals to merge. One typical evaluation function is to determine the correlation of a split or a merge with the class label. A stopping criterion specifies when to stop the discretization process. It is usually governed by a trade-off between lower arty with a better understanding but less accuracy and a higher arty with a poorer understanding but higher accuracy. The number of inconsistencies caused by discretization should not be much higher than the number of inconsistencies of the original data before discretization. Two instances are considered inconsistent if they are the same in their feature values except for their class labels.

Generally, the discretization methods could be supervised or unsupervised. A distinction can be made dependent on whether the method takes class information into account to find proper intervals or not. The methods that do not make use of class membership information during the discretization process are referred to as unsupervised methods. In contrast, discretization methods that use class labels for carrying out discretization are referred to as supervised methods. There are several supervised and unsupervised methods to discretize quantitative features [138]. In following subsections, we present the most popular supervised discretization method, Fayyad and Irani's method, and the newest unsupervised discretization algorithm, using sub-optimal agglomerative clustering method (SOAC).

2.4.1 Fayyad and Irani's Discretization Method

In this subsection, we present Fayyad and Irani's Discretization algorithm [38]. The Fayyad and Irani's Discretization method is based on a minimal entropy heuristic, and it uses the class information entropy of candidate partitions to select bin boundaries for discretization.

Let us consider a given set of observations S, a feature X, and a partition boundary $\overline{T}$, the class information entropy of the partition induced by $\overline{T}$, denoted $E(X, \overline{T}; S)$ is given by

$$E(X, \overline{T}; S) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2),$$

where $S_1 \subset S$ be the subset of observations in S with X-values not exceeding $\overline{T}$ and $S_2 = S - S_1$. Let there be q classes $C_1, ..., C_q$, and $P(C_i, S)$ be the proportion of observations in S that have class C_i . The class entropy of a subset S is defined as:

$$Ent(S) = -\sum_{i=1}^{q} P(C_i, S) \lg(P(C_i, S)),$$

where the logarithm may be to any convenient base. When the base is 2, $Ent(S)$ measures the amount of information needed, in bits, to specify the classes in S.

For a given feature X, the boundary $\overline{T}_{min}$ which minimizes the entropy function over all possible partition boundaries is selected as a binary discretization boundary. This method can then be applied recursively to both of the partitions induced by $\overline{T}_{min}$ until some stopping condition is achieved, thus creating multiple intervals on the feature X.

Fayyad and Irani make use of the minimal description length principle to determine a stopping criteria for their recursive discretization strategy. Recursive partitioning within a set of values S stops if

$$Gain(X,\overline{T};S) < \frac{\lg_2(N-1)}{N} + \frac{\triangle(X,\overline{T};S)}{N},$$

where N is the number of observations in the set S, and

$$Gain(X,\overline{T};S) = Ent(S) - E(X,\overline{T};S),$$

$$\triangle(X,\overline{T};S) = \lg_2(3^q - 2) - [q.Ent(s) - q_1.Ent(S_1) - q_2 Ent(S_2)],$$

and q_i is the number of class labels represented in the set S_i. Since the partitions along each branch of the recursive discretization are evaluated independently using this criteria, some areas in the continuous spaces will be partitioned very finely whereas others (which have relatively low entropy) will be partitioned coarsely.

2.4.2 Discretization Algorithm using Sub-Optimal Agglomerative Clustering

In this section, we present the discretization algorithm using sub-optimal agglomerative clustering (SOAC) [139]. Let us consider a finite set of points A in the $n-$dimensional space

$\mathbb{R}^n$, that is $A = \{a^1, ..., a^m\}$, where $a^i \in \mathbb{R}^n,\ i = 1, ..., m$. Assume the sets $A^j,\ j = 1, ..., k$ be clusters, and each cluster A^j can be identified by its centroid $x^j \in \mathbb{R}^n,\ j = 1, ..., k$. The Algorithm SOAC proceeds as follows.

Algorithm. Discretization Algorithm SOAC

Step 1. Set $k = m$, and a small value of parameter $\theta,\ 0 < \theta < 1$. Sort values of the current feature in the ascending order. Each continuous feature requiring discretization is treated in turn.

Step 2. Calculate the center of each cluster:

$$x^j = \sum_{a \in A^j} \frac{a}{|A^j|},\ j = 1, ..., k$$

and the error E_k of the cluster system approximating set A:

$$E_k = \sum_{j=1}^{k} \sum_{a \in A^j} \|x^j - a\|^2.$$

Step 3. Merge in turn each cluster with the next tentatively. Calculate the error increase $E_{k-1} - E_k$ after each merge and choose the pair of clusters giving the least increase. Merge these two clusters permanently. Set $k = k - 1$.

Step 4. If $E_k \geq \theta E_1$, then stop, otherwise go to Step 2.

Chapter 3

Bayesian Networks

3.1 Improving Naive Bayes Classifier using Conditional Probabilities

This section is devoted to a new algorithm for improving Naive Bayes classifier. The Naive Bayes classifier is the simplest among Bayesian Networks, and it performs very well on a variety of data classification problems. However, the strong assumption that all features are conditionally independent given the class is often violated in many real world applications. The proposed algorithm, in this section, finds dependencies between features using conditional probabilities. The performance of the algorithm is empirically validated using real world data sets. The experimental results demonstrate that the proposed algorithm significantly improve the performance of the Naive Bayes classifier, yet at the same time maintains its robustness.

Paper:

Improving Naive Bayes Classifier using Conditional Probabilities

Published by:

9-th Australasian Data Mining Conference,

Ballarat, Australia, Volume 121, pp. 63-68, 2011

Authors: Sona Taheri^{a}, Musa Mammadov$^{a,\ b}$ and Adil M.Bagirov^{a}

[a]Centre for Informatics and Applied Optimization,

School of Science, Information Technology and Engineering,

University of Ballarat, VIC 3353, Australia

[b]National ICT Australia, VRL, VIC 3010, Australia

Corresponding author:

Sona Taheri

e-mail: sonataheri@students.ballarat.edu.au

Improving Naive Bayes Classifier Using Conditional Probabilities

SONA TAHERI[1] MUSA MAMMADOV[1,2] ADIL M. BAGIROV[1]

[1]Centre for Informatics and Applied Optimization, School of Science, Information Technology and Engineering, University of Ballarat, Victoria 3353, Australia
[2] National Information and Communications Technology Australia (NICTA)

Emails: sonataheri@students.ballarat.edu.au, m.mammadov@ballarat.edu.au, a.bagirov@ballarat.edu.au

Abstract

Naive Bayes classifier is the simplest among Bayesian Network classifiers. It has shown to be very efficient on a variety of data classification problems. However, the strong assumption that all features are conditionally independent given the class is often violated on many real world applications. Therefore, improvement of the Naive Bayes classifier by alleviating the feature independence assumption has attracted much attention. In this paper, we develop a new version of the Naive Bayes classifier without assuming independence of features. The proposed algorithm approximates the interactions between features by using conditional probabilities. We present results of numerical experiments on several real world data sets, where continuous features are discretized by applying two different methods. These results demonstrate that the proposed algorithm significantly improve the performance of the Naive Bayes classifier, yet at the same time maintains its robustness.

Keywords: Bayesian Networks, Naive Bayes, Semi Naive Bayes, Correlation

1 Introduction

Classification is the task to identify the class labels for instances based on a set of features, that is, a function that assigns a class label to instances described by a set of features. Learning accurate classifiers from pre classified data is an important research topic in machine learning and data mining. One of the most effective classifiers is Bayesian Networks (Shafer 1990, Heckerman 1995, Jensen 1996, Pearl 1996, Castillo 1997). A Bayesian Network (BN) is composed of a network structure and its conditional probabilities. The structure is a directed acyclic graph where the nodes correspond to domain variables and the arcs between nodes represent direct dependencies between the variables. Considering an instance $X = (X_1, X_2, ..., X_n)$ and a class C, the classifier represented by BN is defined as

$$\arg\max_{c\in C} P(c|x_1, x_2, ..., x_n) \propto \arg\max_{c\in C} P(c)P(x_1, x_2, ..., x_n|c), \quad (1)$$

where x_i, c are the values of X_i, C respectively.

However, accurate estimation of $P(x_1, x_2, ..., x_n|c)$ is non trivial. It has been proved that learning an optimal BN is NP-hard problem (Chickering 1996, Heckerman 2004). In order to avoid the intractable complexity for learning BN, the Naive Bayes classifier has been used. In the Naive Bayes (NB) (Langley 1992, Domingos 1997), features are conditionally independent given the class. The simplicity of the NB has led to its wide use, and to many attempts to extend it (Domingos 1997). Since NB assumes the strong assumption of independency between features, learning semi Naive Bayes has attracted much attention from researchers (Langley 1994, Kohavi 1996, Pazzani 1996, Friedman 1997, Kittler 1986, Zheng 2000, Webb 2005). The semi Naive Bayes classifiers are based on the structure of NB, requiring that the class variable be a parent of every feature. However, they allow additional edges between features that capture correlation among them. The main aim in this area of research has involved maximizing the accuracy of classifier predictions.

In this paper, we propose a new version of the Naive Bayes classifier (semi Naive Bayes) without assuming independence of features. The proposed algorithm finds dependencies between features using conditional probabilities. This algorithm is a new algorithm and different from the existing semi Naive Bayes methods (Langley 1994, Kohavi 1996, Pazzani 1996, Friedman 1997, Kittler 1986, Zheng 2000, Webb 2005).

Most of data sets in real world applications often involve continuous features. Therefore, continuous features are usually discretized (Lu 2006, Wang 2009, Ying 2009, Yatsko 2010). The main reason is that the classification with discretization tend to achieve lower error than the original one (Dougherty 1995). We apply two different methods to discretize continuous features. The first one, which is also the simplest one, transforms the values of features to $\{0, 1\}$ using their mean values. We also apply the discretization algorithm using sub-optimal agglomerative clustering algorithm from (Yatsko 2010) which allows us to get more than two values for discretized features. This leads to the design of a classifier with higher testing accuracy in most data sets used in this paper.

We organize the rest of the paper as follows. We give a brief review to the Naive Bayes and some semi Naive Bayes classifiers in Section 2. In Section 3, we present the proposed algorithm. Section 4 presents an overview of the discretization algorithm using sub-optimal agglomerative clustering. The numerical experiments are given in Section 5. Section 6 concludes the paper.

2 Naive Bayes and Semi Naive Bayes Classifiers

The Naive Bayes (NB) assumes that the features are independent given the class, it means that all features have only the class as a parent (Kononenko 1990, Langley 1992, Domingos 1997, Mitchell 1997). A sample of the NB with n features is depicted in Figure 1. The NB, classifies an instance $X = (X_1, X_2, ..., X_n)$ using Bayes rule, by selecting

$$\arg\max_{c \in C} P(c) \prod_{i=1}^{n} P(x_i|c). \tag{2}$$

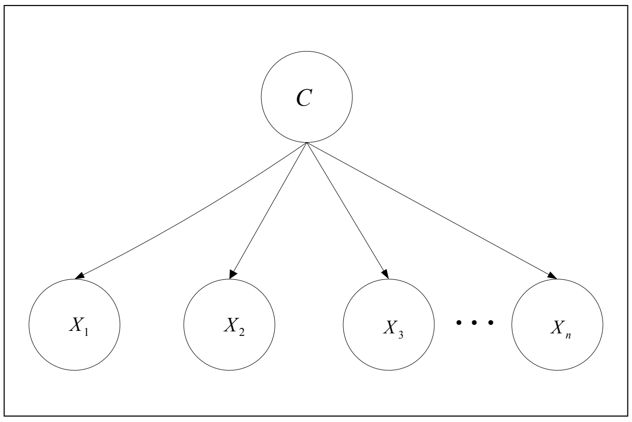


Figure 1: Naive Bayes

NB has been used as an effective classifier for many years. Unlike many other classifiers, it is easy to construct, as the structure is given a priori. Although the independence assumption is obviously problematic, NB has surprisingly outperformed many sophisticated classifiers, especially where the features are not strongly correlated (Domingos 1997). In spite of NB's simplicity, the strong independency assumption harms the classification performance of NB when it is violated. On the other hand, learning BN requires searching the space of all possible combinations of edges which is NP-hard problem (Chickering 1996, Heckerman 2004).

In order to relax the independence assumption of NB, a lot of effort has focussed on improving NB. The improved NB classifiers use exhaustive search to join features based on statistical methods. There are some improved algorithms of the NB. Langley and Sage (Langley 1994) considered Backwards Sequential Elimination (BSE) and Forward Sequential Selection (FSS) in which their methods select a subset of features using leave-one-out cross validation error as a selection criterion and establish a NB with these features. Starting from the full set of features, BSE successively eliminates the features whose elimination most improves accuracy, until there is no further accuracy improvement. FSS uses the reverse search direction, that is iteratively adding the features whose addition most improves accuracy, starting with the empty set of features. The work of Pazzani (Pazzani 1996) introduces Backward Sequential Elimination and Joining (BSEJ). It uses predictive accuracy as a merging criterion to create new Cartesian product features. The value set of a new compound features is the Cartesian product of the value sets of the two original features. As well as joining features, BSEJ also considers deleting features. BSEJ repeatedly joins the pair of features or deletes the features that most improves predictive accuracy using leave-one-out cross validation. This process terminates if there is no accuracy improvement. Kohavi (Kohavi 1996) proposed the NB Tree, a strategy that is a hybrid approach combining NB and decision tree learning. It partitions the training data using a tree structure and establishes a local NB in each leaf. It uses 5-fold cross validation accuracy estimate as the splitting criterion. A split is defined to be significant if the relative error reduction is greater than 5 percent and the splitting node has at least 30 instances. When there is no significant improvement, NB Tree stops the growth of the tree. As the number of splitting features is greater than or equals one, NB Tree is an x-dependence classifier. The classical decision tree predicts the same class for all the instances that reach a leaf. In NB Tree, these instances are classified using a local NB in the leaf, which only considers those non tested features. Friedman et al. (Friedman 1997) introduced Tree Augment Naive Bayes (TAN) based on tree structure. It approximates the interactions between features by using a tree structure imposed on the NB structure. In TAN, each feature has the class and at most one other feature as parents. Super Parent algorithm is proposed by Keogh and Pazzani (Keogh 1999). This algorithm uses the same representation as the Tree Augment Naive Bayes, but utilizes leave-one-out cross validation error as a criterion to add a link. The Super Parent is the feature that is the parent of all the other orphans, the features without a non-class parent. There are two steps to add a link: first selecting the best Super Parent that improves accuracy the most, and then selecting the best child of the Super Parent from orphans. This method stops adding links when there is no accuracy improvement. Zheng and Webb (Zheng 2000) developed Lazy Bayesian Rules (LBR), which adopts a lazy approach, and generates a new Bayesian rule for each test example. The antecedent of a Bayesian rule is a conjunction of feature-value pairs, and the consequent of the rule is a local NB, which uses those features that do not appear in the antecedent to classify. LBR stops adding feature value pairs into the antecedent if the outcome of a one tailed pairwise sign test of error difference is not better than 0.05. As the number of the feature value pairs in the antecedent is greater than or equals one, LBR is anx-dependence classifier. Webb et al. (Webb 2005) proposed Averaged One Dependence Estimators (AODE), which averages the predictions of all qualified 1-dependence classifiers. In each 1-dependence classifier, all features depend on the class and a single feature.

In the next section, we introduce a new version of the Naive Bayes classifier (semi Naive Bayes) without assuming independence of features. The proposed algorithm approximates the interactions between features by using conditional probabilities.

3 The Proposed Algorithm

In this section, we present a new algorithm that maintains the basic structure of the NB, and thus ensure that the class C is the parent of all features. The proposed algorithm, however, removes the strong assumption of independence in the NB by finding correlation between features, while also capturing much of the computational efficiency of the NB. In this algorithm, the class has no parents and each feature has the class and at most one other feature as parents. Therefore, each feature can have one augmenting edge pointing to it. The procedure for learning these edges is based on the Pearson's correlation and conditional probabilities. First, we construct a basic structure of the NB with n features $X_1, X_2, ..., X_n$ from the set X and the class C. After that, we find the Pearson's correlations between each feature X_i and the class C using the formula (3), $Corr(X_i, C)$. Then we reorder the set X as a set X^* in a descending order of $|Corr(X_i, C)|$. In the ordered set X^*, an arc from the first feature is added to the second one. Finally, for all remain features, we find the conditional probabilities of each feature with the previous features given the class values in the ordered set X^*, formula (4). The highest value of these conditional probabilities between features is used to recognize the parent of each feature. The conditional probabilities described in (4), first introduced by Quinn et al. (Quinn 2009) and called influence weights, have been used directly for data classification. However, here, we used them for finding the dependencies between features.

The correlation coefficient (Graham 2008) between two random variables X_i and X_j is defined as :

$$Corr(X_i, X_j) = \frac{N \sum_{i,j=1}^{N} X_i X_j - \sum_{i=1}^{N} X_i \sum_{j=1}^{N} X_j}{\sqrt{(N \sum_{i=1}^{N} X_i^2 - (\sum_{i=1}^{N} X_i)^2)(N \sum_{j=1}^{N} X_j^2 - (\sum_{j=1}^{N} X_j)^2)}}, \quad (3)$$

where N is the number of data points. This measure has the property of $|Corr(X_i, X_j)| \leq 1$. When this value is close to 1, it denotes the perfect linear correlation between X_i and X_j, and $Corr(X_i, X_j) = 0$ stands for no linear correlation.

The proposed algorithm consists of six main steps:

Algorithm. Proposed Algorithm

```
***Step 1.*** Construct a basic structure of the Naive Bayes with $n$ features,
    $X = \{X_1, X_2, ..., X_n\}$, and the class $C$.
***Step 2.*** Compute the correlation between each feature $X_i,\ i = 1, ..., n$ and the class $C$
    using the formula (3), $Corr(X_i, C)$.
***Step 3.*** Reorder $X$ as a set $X^* = \{X_1^*, X_2^*, ..., X_n^*\}$ in a descending order
    of $|Corr(X_i, C)|,\ i = 1, ..., n$.
***Step 4.*** Add an arc from $X_1^*$ to $X_2^*$.
***Step 5.*** For $j = 3, ..., n$:
    5.1 Find $X_i^*$ that has the highest value of
        $\sum_{k=1}^{N} |P(X_{ki}^*, X_{kj}^*|C) - P(X_{ki}^*, X_{kj}^*|\overline{C})|,\ i < j,$   (4)
        where $X_i^* = (X_{1i}^*, X_{2i}^*, ..., X_{Ni}^*)^T$, $N$ is
        the number of instances and $\overline{C} = -C$.
    5.2 Add an arc from $X_i^*$ to $X_j^*$.
***Step 6.*** Compute the conditional probability tables inferred by the new
    structure.
```

Figure 2 shows the structure of Svmguide1 data set, taken from LIBSVM, with four features (see Table 1) using the proposed algorithm. The solid lines are those edges required by the Naive Bayes classifier. The dashed lines are correlation edges between features found by our algorithm.

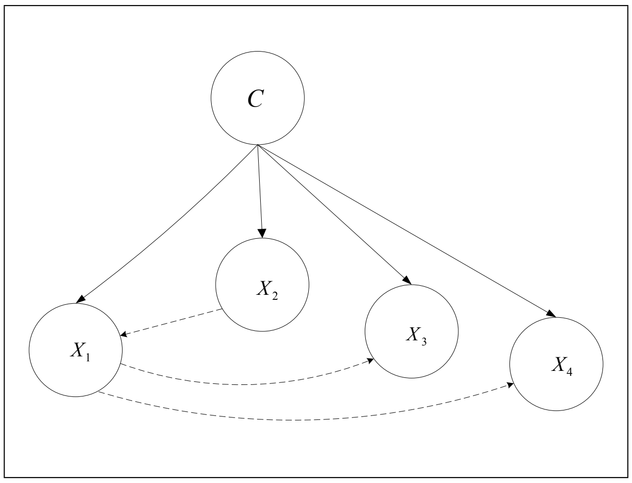

$$\begin{tikzcd}
 & & C \arrow[dll] \arrow[d] \arrow[dr] \arrow[drr] & & \\
X_1 \arrow[rrr, dashed, bend right] \arrow[rrrr, dashed, bend right] & & X_2 \arrow[ll, dashed] & X_3 & X_4
\end{tikzcd}$$

Figure 2: Proposed algorithm, Svmguide1

4 Discretization Algorithm Using Sub-Optimal Agglomerative Clustering (SOAC)

Discretization is a process which transform continuous numeric values into discrete ones. In this paper, we apply two different methods to discretize continuous features. The first one, which is also the simplest one, transforms the values of features to 0,1 using their mean values. We also apply the discretization algorithm using sub-optimal agglomerative clustering algorithm which allows us to get more than two values for discretized features. In this section, we introduce discretization algorithm SOAC which is an efficient discretization method for the NB learning. Details of this algorithm can be found in (Yatsko 2010).

Consider a finite set of points A in the n dimensional space R^n, that is $A = \{a^1, ..., a^m\}$, where $a^i \in R^n,\ i = 1, ..., m$. Assume that the sets $A^j,\ j = 1, ..., k$ be clusters, and each cluster A^j can be identified by its centroid $x^j \in R^n,\ j = 1, ..., k$. The discretization algorithm SOAC proceeds as follows.

Algorithm. Discretization Algorithm SOAC

```
***Step 1.*** Set $k = m$, and a small value of parameter $\theta$, $0 < \theta < 1$. Sort values of the
    current feature in the ascending order. Each feature requiring discretization is treated in
    turn.
***Step 2.*** Calculate the center of each cluster:
    $x^j = \sum_{a \in A^j} \frac{a}{|A^j|},\ j = 1, ..., k$
    and the error $E_k$ of the cluster system approximating set $A$:
    $E_k = \sum_{j=1}^{k} \sum_{a \in A^j} \|x^j - a\|^2.$
***Step 3.*** Merge in turn each cluster with the next tentatively. Calculate the error increase
    after each merge $E_{k-1} - E_k$ and choose the pair of clusters giving the least increase.
    Merge these two clusters permanently. Set $k = k - 1$.
```

Step 4. Once the error of the current cluster system is over the set fraction of the maximum error corresponding to the single cluster $E_k \geq \theta E_1$ stop, otherwise go to Step 2.

5 Numerical Experiments

To verify the efficiency of the proposed algorithm, numerical experiments with a number of real world data sets have been carried out. We use 10 real world data sets. The detailed description of the data sets used in this experiments can be found in the UCI machine learning repository, with the exception of "Fourclass", "Svmguide1" and "Svmguide3". These three data sets are downloadable on tools page of LIBSVM. A brief description of data sets is given in Table 1. We discritize the values of features in data sets using two different methods. In the first one, we apply a mean value of each feature variable to discritize the values to $\{0, 1\}$. The second one is the discrization algorithm SOAC (Yatsko 2010) which is presented in Section 4.

We conduct empirical comparison for the NB and the proposed algorithm in terms of test set accuracy using two different discritization methods. The results of the NB and the new algorithm on each data set were obtained via 1 run of 10-fold cross validation. Runs were carried out on the same training sets and evaluated on the same test sets. In particular, the cross validation folds were the same for all experiments on each data set.

The test set accuracy obtained by the NB and the proposed algorithm on 10 data sets using mean values for discretization summarized in Table 2. The results presented in this table demonstrate that the test set accuracy of the new algorithm is much better than that of the NB. The proposed algorithm works well in that it yields good classifier compared to the NB. Its performance was further improved by introducing some additional edges in the NB, using conditional probabilities. Improvement is noticed mainly in large data sets. In 8 cases out of 10, the new algorithm has higher accuracy than the NB. The accuracy of this algorithm is same with the NB in data sets Fourclass and Svmguide1.

Table 3 presents the test set accuracy obtained by the NB and the proposed algorithm on 10 data sets using discretization algorithm SOAC. The results from this table show that the accuracy obtained by the new algorithm in all data sets are higher than those obtained by the NB.

Figures 3 to 4 show the scatter plot comparing the proposed algorithm with the NB, using two different discritization methods. In these plots, each point represents a data set, where the x coordinate of a point is the percentage of miss classifications according to the NB and the y coordinate is the percentage of miss classification according the proposed algorithm. Therefore, points above the diagonal line correspond to data sets where the NB performs better, and points below the diagonal line correspond to data sets where the proposed algorithm performs better.

According to the results explained above, the proposed algorithm outperforms the NB, yet at the same time maintains its robustness. However, the proposed algorithm requires more computational effort than the NB since we need to compute conditional probabilities between features to recognize the parent of each feature in our algorithm.

Table 1: A brief description of data sets

Data sets	# Features	# Instances
Congres Voting Records	16	435
Credit Approval	14	690
Diabetes	8	768
Fourclass	2	862
Haberman Survival	3	306
Heart Disease	13	270
Phoneme CR	5	5404
Spambase	57	4601
Svmguide1	4	7089
Svmguide3	21	1284

Table 2: Test set accuracy of NB and the proposed algorithm using mean value for discretization

Data Sets	Naive Bayes	Proposed Algorithm
Congres Voting Records	90.11	91.47
Credit Approval	84.85	86.85
Diabetes	75.78	77.68
Fourclass	76.82	76.82
Haberman Survival	74.51	75.66
Heart Disease	84.14	85.18
Phoneme CR	75.96	78.30
Spambase	90.13	93.45
Svmguide1	92.17	92.17
Svmguide3	80.61	87.18

Table 3: Test set accuracy of NB and the proposed algorithm using discretization algorithm SOAC

Data Sets	Naive Bayes	Proposed Algorithm
Congres Voting Records	90.11	91.47
Credit Approval	84.85	86.85
Diabetes	75.78	77.68
Fourclass	78.58	79.70
Haberman Survival	74.66	75.33
Heart Disease	78.62	79.31
Phoneme CR	77.01	79.36
Spambase	89.30	92.30
Svmguide1	95.61	97.54
Svmguide3	77.25	80.85

6 Conclusion

In this paper, we have developed the new version of the Naive Bayes classifier without assuming independence of features. An important step in this algorithm is adding edges between features that capture correlation among them. The proposed algorithm finds dependencies between features using conditional probabilities. We have presented the results of numerical experiments on 10 data sets from UCI machine learning repository and LIBSVM. The values of features in data sets are discritized by using mean value of each feature and applying discretization algorithm SOAC. We have presented results of numerical experiments. These results clearly demonstrate that the proposed algorithm significantly improve the performance of the Naive Bayes classifier, yet at the same time maintains its robustness. Furthermore, this improvement becomes even more substantial as the size of the data sets increases.

7 References

References

Castillo, E., Gutierrez, J.M & Hadi, A.S. (1997), Expert Systems and Probabilistic Network Models, Springer Verlag, New York.

Chang, C., & Lin, C. (2001), A library for support vector machines, Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm.

Charniak, E. (1991), E. Bayesian Networks Without Tears. AI Magazine, 12 (4).

Chickering, D.M. (1996), Learning Bayesian Networks is NP-complete. In: Fisher, D., Lenz, H. Learning from data: Artificial Intelligence and statistics V, Springer, pp. 121–130.

Domingos, P., & Pazzani, M. (1997), On the optimality of the simple Bayesian classifier under zero-one loss. Mach Learn 29, pp. 103–130.

Dougherty, J., Kohavi, R., & Sahami, M., (1995), Supervised and unsupervised discretization of continuous features. In Proceedings of the 12th International Conference on Machine Learning, pp. 194–202.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997), Bayesian network classifiers. Machine Learning 29 , pp. 131–163.

Graham, E., (2008), CIMA Official Learning System Fundamentals of Business Maths. Burlington : Elsevier Science and Technology.

Heckerman, D., Geiger, D., & Chickering, D.M. (1995), Learning Bayesian Networks: the Combination of Knowledge and Statistical Data. Machine Learning, 20, pp. 197–243.

Heckerman, D., Chickering, D.M., & Meek.C. (2004), Large-Sample Learning of Bayesian Networks is NP-Hard. Journal of Machine Learning Research, pp. 1287–1330.

Jensen, F. (1996), An Introduction to Bayesian Networks. Springer, New York.

Kohavi, R. (1996), Scaling up the accuracy of naive-Bayes classifiers: a decision-tree hybrid. In: Proc. 2nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, pp. 202–207.

Keogh, E.J., & Pazzani, M.J., (1999), Learning augmented Bayesian classifers: A comparison of distribution-based and classification-based approaches. In: Proc. Int. Workshop on Artificial Intelligence and Statistics, pp. 225–230.

Kittler, J. (1986), Feature selection and extraction. In Young, T.Y., Fu, K.S., eds.: Handbook of Pattern Recognition and Image Processing. Academic Press, New York.

Kononenko, I. (1990), Comparison of Inductive and Naive Bayesian Learning Approaches to Autpmatic Knowledge Acquisition. In Wielinga, B., Boose, J., B.Gaines, Schreiber, G., van Someren, M., eds. Current Trends in Knowledge Acquisition. Amsterdam: IOS Press.

Langley, P., Iba, W., & Thompson, K. (1992), An Analysis of Bayesian Classifiers. In 10th International Conference Artificial Intelligence, AAAI Press and MIT Presspp, pp. 223–228.

Langley, P., & Saga, S. (1994), Induction of selective Bayesian classifiers. In: Proc. Tenth Conf. Uncertainty in Artificial Intelligence, Morgan Kaufmann , pp. 399–406.

Lu, J., Yang, Y., & Webb, G. I. (2006), Incremental Discretization for Naive-Bayes Classifier, Springer, Heidelberg, vol. 4093, pp. 223–238.

Mitchell, T.M. (1997), Machine Learning. McGraw-Hill, New York.

Pazzani, M.J. (1996), Constructive induction of Cartesian product attributes. ISIS: In- formation, Statistics and Induction in Science, pp. 66–77.

Pearl, J. (1988), Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann.

Quinn, A., Stranieri, A., Yearwood, J., & Hafen, G. (2009), A classification algorithm that derives weighted sum scores for insight into disease, Proc. of 3rd Australasian Workshop on Health Informatics and Knowledge Management, Wellington, New Zealand.

Shafer, G., & Pearl, J. (1990), Readings in Uncertain Reasoning. Morgan Kaufmann, San Mateo, CA.

Wang, S, Min, Z., Cao, T., Boughton, J., & Wang, Z. (2009), OFFD: Optimal Flexible Frequency Discretization for Naive Bayes Classification, Springer, Heidelberg, pp. 704–712.

Webb, G.I, Boughton, J., & Wang, Z. (2005), Not so naive Bayes: Aggregating one- dependence estimators. Machine Learning 58, pp. 5–24.

Yatsko, A., Bagirov, A. M., & Stranieri, A. (2010), On the Discretization of Continuous Features for Classification. School of Information Technology and Mathematical Sciences, University of Ballarat Conference, http://researchonline.ballarat.edu.au:8080/vital/access/manager/Repository.

Ying, Y., & Geoffrey, I., (2009), Discretization For Naive-Bayes Learning: Managing Discretization Bias And Variance, In Machine Learning, 74(1): pp. 39–74.

Ying, Y., (2009), Discretization for Naive-Bayes Learning, PhD thesis, school of Computer Science and Software Engineering of Monash University.

Zheng, Z., & Webb, G.I. (2000), Lazy learning of Bayesian rules. Machine Learning 41, pp. 53–84.

UCI repositary of machine learning databases (http://archive.ics.uci.edu/ml/datasets.html)

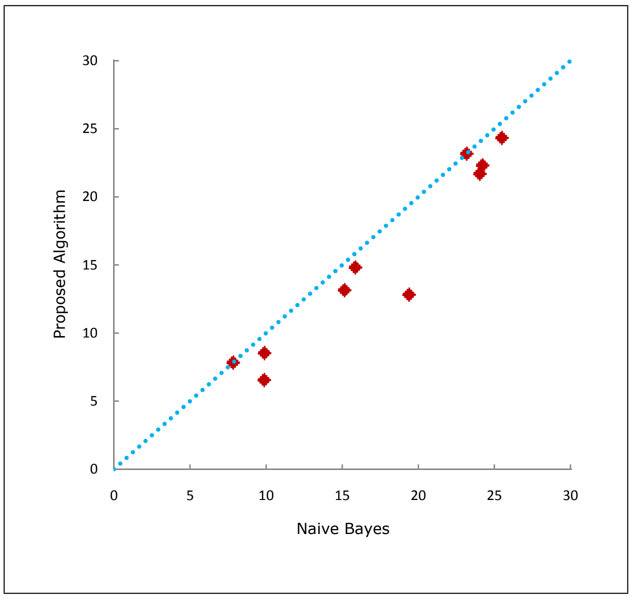


Figure 3: Scatter plot comparing miss classifications of the proposed algorithm (y coordinate) with Naive Bayes (x coordinate); using mean value for discritization

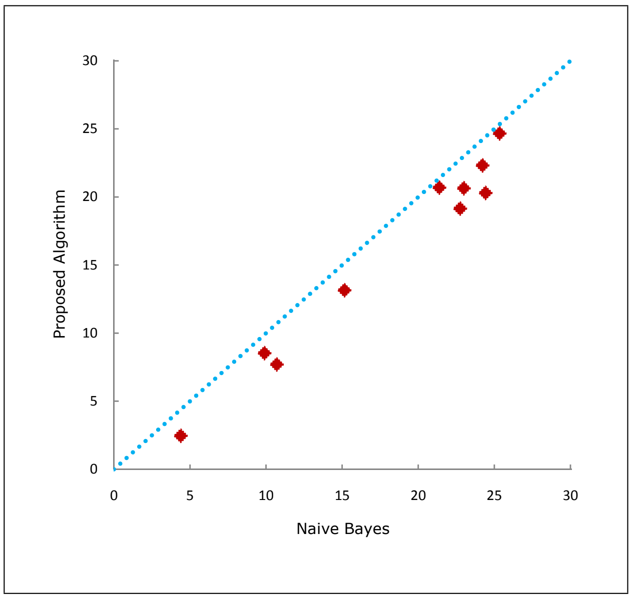


Figure 4: Scatter plot comparing miss classifications of the proposed algorithm (y coordinate) with Naive Bayes (x coordinate); using Algorithm SOAC for discritization

3.2 Structure Learning of Bayesian Networks using a New Unrestricted Dependency Algorithm

This section introduces a new algorithm on learning the structure of Bayesian Networks for binary classification problems. The structure of a Bayesian Network represents a set of conditional dependence and independence relations that hold in the domain. Learning the structure of Bayesian Networks that represents a domain can reveal insights into its underlying causal structure. The proposed algorithm is a novel iterative unrestricted dependency algorithm based on a combinatorial optimization model. The Algorithm is called UDBN meaning unrestricted dependency Bayesian Networks. The empirical results presented here show that the Algorithm UDBN produces networks with significantly higher structural accuracy.

Paper:

Structure Learning of Bayesian Networks using a New Unrestricted Dependency Algorithm

Published by:

Second International Conference on Social Eco-Informatics,

Venice, Italy, 2012

Authors: Sona Taheri[a] and Musa Mammadov[a, b]

[a]Centre for Informatics and Applied Optimization,

School of Science, Information Technology and Engineering,

University of Ballarat, VIC 3353, Australia

[b]National ICT Australia, VRL, VIC 3010, Australia

Corresponding author:

Sona Taheri

e-mail: sonataheri@students.ballarat.edu.au

Structure Learning of Bayesian Networks Using a New Unrestricted Dependency Algorithm

Sona Taheri
Centre for Informatics and Applied Optimization
School of Science, Information Technology and Engineering
University of Ballarat, VIC 3353, Australia
Email: sonataheri@students.ballarat.edu.au

Musa Mammadov
University of Ballarat, VIC 3353, Australia
Email: m.mammadov@ballarat.edu.au
National ICT Australia, VRL, VIC 3010, Australia
Email: musa.mammadov@nicta.com.au

***Abstract*—Bayesian Networks have deserved extensive attentions in data mining due to their efficiencies, and reasonable predictive accuracy. A Bayesian Network is a directed acyclic graph in which each node represents a variable and each arc a probabilistic dependency between two variables. Constructing a Bayesian Network from data is the learning process that is divided in two steps: learning structure and learning parameter. In many domains, the structure is not known a priori and must be inferred from data. This paper presents an iterative unrestricted dependency algorithm for learning structure of Bayesian Networks for binary classification problems. Numerical experiments are conducted on several real world data sets, where continuous features are discretized by applying two different methods. The performance of the proposed algorithm is compared with the Naive Bayes, the Tree Augmented Naive Bayes, and the $k-$Dependency Bayesian Networks. The results obtained demonstrate that the proposed algorithm performs efficiently and reliably in practice.**

Keywords-Data Mining; Bayesian Networks; Naive Bayes; Tree Augmented Naive Bayes; $k-$Dependency Bayesian Networks; Topological Traversal Algorithm.

I. Introduction

Data Mining is defined as the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data [6]. The whole process of data mining consists of several steps. Firstly, the problem domain is analyzed to determine the objectives. Secondly, data is collected and an initial exploration is conducted to understand and verify the quality of the data. Thirdly, data preparation is made to extract relevant data sets from the database. A suitable data mining algorithm is then employed on the prepared data to discover knowledge represented in different representations such as decision trees, neural networks, support vector machine and Bayesian Networks. Finally, the result of data mining is interpreted and evaluated. If the discovered knowledge is not satisfactory, these steps will be iterated. The discovered knowledge is then applied in decision making. Recently, there is an increasing interest in discovering knowledge represented in Bayesian Networks [13], [14], [17], [15], [19] and [28]. Bayesian networks (BNs), introduced by Pearl [21], can encode dependencies among all variables; therefore, they readily handle situations where some data entries are missing. BNs are also used to learn causal relationships, and hence can be used to gain understanding about a problem domain and to predict the consequences of intervention. Moreover, since BNs in conjunction with Bayesian statistical techniques have both causal and probabilistic semantics, they are an ideal representation for combining prior knowledge and data [10]. In addition, BNs in conjunction with Bayesian statistical methods offer an efficient and principal approach for avoiding the over fitting of data [20]. BNs have been applied widely for data mining, causal modeling and reliability analysis [29].

This paper presents a novel unrestricted dependency algorithm to learn knowledge represented in BNs from data. A BN is a graphical representation of probability distributions over a set of variables that are used for building a structure of the problem domain. The BN defines a network structure and a set of parameters, class probabilities and conditional probabilities. Once the network structure is constructed, the probabilistic inferences are readily calculated, and can be performed to predict the outcome of some variables based on the observations of others.

The main task of learning BNs from data is finding directed arcs between variables, or, in other words, the structure discovery, which is the more challenging, and thus, more interesting phase. Two rather distinct approaches have been used widely to structure discovery in BNs: the constraint-based approach [22], [27] and the score-based approach [1], [5], [26]. In the the constraint-based approach, structure learning cares about whether one arc in the graph should be existed or not. This approach relies on the conditional independence test to determine the importance of arcs [4]. In the score-based approach, several candidate graph structures are known, and we need choosing the best one out. In order to avoid over fitting, investigators often use model selection methods, such as Bayesian scoring function [5] and entropy-based method [12]. Several exact algorithms based on dynamic programming have recently been developed to learn an optimal BN [16], [24], [25] and [31]. The main idea in these algorithms is to solve small subproblems first and use the results to find solutions to larger problems until a global learning problem is solved. However, they might be inefficient due to their need to fully evaluate an exponential solution space.

It has been proved that learning an optimal BN is NP-hard [11]. In order to avoid the intractable complexity for learning BNs, the Naive Bayes [18] has been used. The Naive Bayes (NB) is the simplest among BNs. In the NB, features are conditionally independent given the class. It has shown to be very efficient on a variety of data mining problems. However, the strong assumption that all features are conditionally independent given the class is often violated on many real world applications. In order to relax this assumption of the NB while at the same time retaining its simplicity and efficiency, researchers have proposed many effective methods [7], [23] and [28]. Sahami [23] proposed the $k-$dependence BNs to construct the feature dependence with a given number, value of k. In this algorithm, each feature could have a maximum of k feature variables as parents, and these parents are obtained by using mutual information. The value of k in this algorithm is initially chosen before applying it, $k = 0, 1, 2, ...$ Friedman et al. [7] introduced the Tree Augment Naive Bayes (TAN) based on the tree structure. It approximates the interactions between features by using a tree structure imposed on the NB structure. In the TAN, each feature has the class and at most one other feature as parents.

Although the mentioned methods were shown to be efficient, the features in these methods depend on the class and a priori given number of features; $k = 0$ dependence for the NB, $k = 1$ dependence for the TAN, and an initially chosen k for the k-dependence BNs. In fact, by setting k, i.e., the maximum number of parent nodes that any feature may have, we can construct the structure of BNs. Since k is the same for all nodes, it is not possible to model cases where some nodes have a large number of dependencies, whereas others just have a few. In this paper, we propose a new algorithm to identify the limitations of each of these methods while also capturing much of the computational efficiency of the NB. In the proposed algorithm, the number k is defined by the algorithm internally, and it is an unrestricted dependency algorithm.

The rest of the paper is organized as follows. In the next section, we provide a brief description of BNs. In Section III, we introduce a new algorithm for structure learning of BNs from binary classification data. Section IV presents a brief review of the Topological Traversal algorithm. The results of numerical experiments are given in Section V. Section VI concludes the paper.

II. Representation of Bayesian Networks

A BN consists of a directed acyclic graph connecting each variables into a network structure and a collection of conditional probability tables, where each variable in the graph is denoted by a conditional probability distribution given its parent variables. The nodes in the graph correspond to the variables in the domain, and the arcs (edges) between nodes represent causal relationships among the corresponding variables. The direction of the arc indicates the direction of causality. When two nodes are joined by an arc, the causal node is called the parent of the other node, and another one is called the child. How one node influences another is defined by conditional probabilities for each node given its parents [21]. Suppose a set of variables $\mathbf{X} = \{X_1, X_2, ..., X_n\}$, where X_i denotes both the variable and its corresponding node. Let $Pa(X_i)$ denotes a set of parents of the node X_i in $\mathbf{X}$. When there is an edge from X_i to X_j, then X_j is called the child variable for a parent variable X_i. A conditional dependency connects a child variable with a set of parent variables. The lack of possible edges in the structure encodes conditional independencies.

In particular, given a structure, the joint probability distribution for $\mathbf{X}$ is given by

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|Pa(X_i)), \tag{1}$$

here, $P(X_i|Pa(X_i))$ is the conditional probability of X_i given its parents $Pa(X_i)$, where

$$P(X_i|Pa(X_i)) = \frac{P(X_i, Pa(X_i))}{P(Pa(X_i))} = \frac{n_{X_i,Pa(X_i)}}{n_{Pa(X_i)}},$$

where $n_{Pa(X_i)}$ denotes the number of items in the set $Pa(X_i)$, and $n_{X_i,Pa(X_i)}$ is the number of items in $X_i \cap Pa(X_i)$.

However, accurate estimation of $P(X_i|Pa(X_i))$ is non trivial. Finding such an estimation requires searching the space of all possible network structures for one that best describes the data. Traditionally, this is done by employing some search mechanism along with an information criterion to measure goodness and differentiate between candidate structures met while traversing the search space. The idea would be to try and maximize this information measure or score by moving from one structure to another. The associated structure is then chosen to represent and explain the data. Finding an optimal structure for a given set of training data is a computationally intractable problem. Structure learning algorithms determine for every possible edge in the network whether to include the edge in the final network and which direction to orient the edge. The number of possible graph structures grows exponentially as every possible subset of edges could represent the final model. Due to this exponential growth in graph structure, learning an optimal BNs has been proven to be NP-hard [11].

During the last decades a good number of algorithms whose aim is to induce the structure of the BN that better represents the conditional dependence and independence relationships underlying have been developed [4], [5], [7], [12], [16], [24] and [25]. In our opinion, the main reason for continuing the research in the structure learning problem is that mendelizing the expert knowledge has become an expensive, unreliable and time consuming job. We introduce a new algorithm for structure learning of BNs in the following section.

III. The Proposed Algorithm for Bayesian Networks

In this section, we propose a new algorithm to learn the structure of BNs for binary classification problems. Since the learning process in BNs is based on the correlations of children and parent nodes, we propose a combinatorial optimization model to find the dependencies between features. However, some features could be independent which is considered by intruding a threshold K. Let us consider an optimization model (2):

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n} (K_{ij} - K) w_{ij}, \quad j \neq i \tag{2}$$

$$\text{subject to} \quad w_{ij} + w_{ji} \leq 1,$$

where $1 \leq i, j \leq n,\ i < j$ and $w_{ij} \in \{0, 1\}$. w_{ij} is the association weight (to be found), given by

$$w_{ij} = \begin{cases} 1 & \text{if feature } X_i \text{ is the parent of feature } X_j, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

and for $1 \leq i, j \leq n,\ \ i \neq j$,

$$K_{ij} = \sum_{q_2=1}^{|X_j|} \sum_{q_1=1}^{|X_i|} \max\{P(X_{q_2 j}|C_1, X_{q_1 i}), P(X_{q_2 j}|C_2, X_{q_1 i})\}. \tag{4}$$

Here, $|X_j|$ and $|X_i|$ are the number of values of features X_j and X_i, respectively, and X_{ql} shows the qth value of the feature $X_l,\ 1 \leq l \leq n$. We assume binary classification; $C_1 = 1$ and $C_2 = -1$ are class labels. K is a threshold such that $K \geq 0$.

From the formula (2), $w_{ij} = 1$ if $K_{ij} > K_{ji}$ and $K_{ij} > K$, and therefore $w_{ji} = 0$ due to the constraint $w_{ij} + w_{ji} \leq 1$. It is clear that $w_{ii} = 0,\ 1 \leq i \leq n$. Thus problem (2) can be solved easily. Let us denote the solution of the problem (2) by $W(K) = [w_{ij}(K)]_{n \times n}$, where

$$w_{ij}(K) = \begin{cases} 1 & \text{if } K_{ij} > K_{ji} \text{ and } K_{ij} > K, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

and the set of arcs presented by

$$A(W) = \{(i, j) :\ if\ w_{ij} = 1,\ 1 \leq i, j \leq n,\ i \neq j\}, \tag{6}$$

(i, j) shows the arc from X_i to X_j. If we have set of arcs $A(W)$, then we have the corresponding matrix W that satisfies (6). It is clear that $A(W) \subset I$, where $I = \{(i, j),\ 1 \leq i, j \leq n\}$ is the set of all possible couples (i, j).

The best value for K will be found based on the maximum training accuracy for different values of $w_{ij}(K)$, where $0 \leq K \leq K^{max}$, and

$$K^{max} = max\{K_{ij},\ 1 \leq i, j \leq n,\ i \neq j\}. \tag{7}$$

More precisely, we find the values of $w_{ij}(K_r)$ for different $K_r = K^{max} - \varepsilon r,\ r = 0, 1, ...$ until $K_r < 0$, and we set $W(K_r) = [w_{ij}(K_r)]_{n \times n}$. With the matrix $W(K_r)$, the set of arcs $A(W(K_r))$ and, therefore, a network will be learnt. Based on the obtained network, the conditional probabilities will be found:

$$P(C|\mathbf{X}) \equiv \prod_{i=1}^{n} P(X_i|C, Pa(X_i)) P(C), \tag{8}$$

where $Pa(X_i)$ denotes the set of parents of the variable X_i to be found with $W(K_r)$. Now, based on these conditional probabilities, we calculate:

$$C(\mathbf{X}) = \begin{cases} 1 & \text{if } P(C_1 = 1|\mathbf{X}) > P(C_2 = -1|\mathbf{X}), \\ -1 & \text{otherwise,} \end{cases}$$

and then the maximum training accuracy will be found using the following formula:

$$accuracy(A(W(K_r))) = \frac{100}{ntr} \sum_{i=1}^{ntr} \delta(C(\mathbf{X}_i), C_i), r = 0, 1, ... \tag{9}$$

where

$$\delta(\alpha, \beta) = \begin{cases} 1 & \text{if } \alpha = \beta \\ 0 & \text{otherwise.} \end{cases}$$

We will choose that value of r corresponding to the highest training accuracy. Here, ntr stands for the number of instances in the training set.

Since BNs are directed acyclic graphs, we should not have any cycle in the structure obtained by $A(W(K_r))$. Therefore, the maximum training accuracy subject to no cycles will give the best value of K_r, denoted by K^*, and consequently, the best structure $A(W(K^*))$. Here, we apply the topological traversal algorithm to test if the corresponding graph to the obtained network is acyclic.

According to explanations above, the proposed algorithm constructs unrestricted dependencies between features based on the structure of the NB. The proposed algorithm eliminates the strong assumptions of independencies between features in the NB, yet at the same time maintains its robustness. It is clear that $r = 0$ in the proposed algorithm gives the structure of the NB. In our algorithm, some features could have a large number of dependencies, whereas others just have a few. The number of these dependencies will be defined by the algorithm internally. The steps of our algorithm is presented in the following:

```
Step 1. Compute $\{K_{ij},\ 1 \le i,j \le n,\ i \ne j\}$ using (4).
Step 2. Determine $K^{max}$ using (7). Set $r = 0$, and $p_0 = 0$.
Step 3. while $K^{max} - \varepsilon r \ge 0$ do
    3.1. Calculate $K_r = K^{max} - \varepsilon r$.
    3.2. Compute $w_{ij}(K_r),\ 1 \le i,j \le n,\ (i \ne j)$ using (5), and let $W(K_r) = [w(K_r)_{ij}]_{n \times n}$.
    3.3. Find dependencies between features by a set of arcs $A(W(K_r))$ using (6).
    3.4. Apply the topological traversal algorithm to test the network obtained by $A(W(K_r))$ for possible cycles. If any cycle is founds, then go to Step 4.
    3.5. Compute the training accuracy, $p = accuracy(A(W(K_r))$, using (9). If $p > p_0$ then set $p_0 = p$, $K^* = K_r$, $r = r + 1$.
end
Step 4. Construct the optimal structure based on the basic structure of the NB and applying the set of arcs $A(W(K^*))$ between features.
Step 5. Compute the conditional probability tables inferred by the new structure.
```

Algorithm 1: Unrestricted Dependency BNs Algorithm

In this paper, we limit ourselves to binary classification, though a brief discussion on multiple class classification is warranted. The most straightforward approach in these classification problems is finding maximum of m conditional probabilities in the formula (4), where m is the number of classes. Moreover, the one-versus-all classification paradigm will be used to find either in the training accuracy, (9), or the test accuracy in the experiments.

IV. Topological Traversal Algorithm

The topological traversal algorithm [8] is applied for testing a directed graph if there exists any cycle. The degree of a node in a graph is the number of connections or edges the node has with other nodes. If a graph is directed, meaning that edges point in one direction from one node to another node. Then nodes have two different degrees, the in-degree, which is the number of incoming edges to this node, and the out-degree, which is the number of outgoing edges from this edge.

The topological traversal algorithm begins by computing the in-degrees of the nodes. At each step of the traversal, a node with in-degree of zero is visited. After a node is visited, the node and all the edges emanating from that node are removed from the graph, reducing the in-degree of adjacent nodes. This is done until the graph is empty, or no node without incoming edges exists. The presence of the cycle prevents the topological order traversal from completing. Therefore, the simple way to test whether a directed graph is cyclic is to attempt a topological traversal of its nodes. If all nodes are not visited, the graph must be cyclic.

V. Experiments

We have employed 12 well-known binary classification data sets. A brief description of the data sets is given in Table I. The detailed description of the data sets used in this experiments are downloadable in the UCI repository of machine learning databases [2] and the tools page of the LIBSVM [3]. The reason that we have chosen these data sets is: they are the most frequently binary classification data sets considered in the literature.

All continue features in data sets are discretized using two different methods. In the first one, we apply a mean value of each feature to discretize values to binary, $\{0, 1\}$. In the second one, we use the discretization algorithm using sub-optimal agglomerative clustering (SOAC) [30] to get more than two values for discretized features.

We conduct an empirical comparison for the Naive Bayes (NB), the Tree Augmented Naive Bayes (TAN), the $k-$Dependency Bayesian Networks ($k-$DBN), and the proposed algorithm (UDBN) in terms of test set accuracy. We have compared our algorithm with the mentioned algorithms because the basic structure of all, the TAN, the $k-$DBN and the UDBN, is based on the the structure of the NB. In all the cases we have used $10-$fold cross validation. We report the averaged accuracy over the ten test folds.

Table II presents the averaged test set accuracy obtained by the NB, the TAN, the $k-$DBN and the UDBN on 12 data sets, where continuous features are discretized using mean values for discretization. The results presented in this table demonstrate that the accuracy of the proposed algorithm (UDBN) is much better than that of the NB, and the TAN in all data sets. The UDBN also works better than the $k-$DBN in most of data sets. In 10 cases out of 12, the UDBN has higher accuracy than the $k-$DBN. The accuracy of this method almost ties with the $k-$DBN in data sets Phoneme CR and German.numer.

The averaged test set accuracy obtained by the NB, the TAN, the $k-$DBN and the UDBN on 12 data sets using discretization algorithm SOAC summarized in Table III. The results from this table show that the accuracy obtained by the proposed algorithm in all data sets are higher than those obtained by the NB, the TAN, and the $k-$DBN.

According to the results explained, the proposed algorithm, UDBN, works well. It yields good classification compared to the NB, the TAN and the $k-$DBN. In addition, our algorithm is more general than the $k-$DBN. In the $k-$DBN, the number k is a priori chosen. In fact, by setting k, i.e., the maximum number of parent nodes that any feature may have, the structure of BNs could be constructed. Since k is the same for all nodes, it is not possible to model cases where some nodes have a large number of dependencies, whereas others just have a few. However, in the proposed algorithm, the number k is defined by the algorithm internally, and it is an unrestricted dependency algorithm. It might be various for different data sets, and even for each fold in the calculations. The computational times are not presented in Tables II and III. It is clear that the proposed algorithm needs more computational time than the others, since for example, the NB appears as a special case of UDBN when $r = 0$.

Table I
A BRIEF DESCRIPTION OF DATA SETS

Data sets	# Instances	# Features
Breast Cancer	699	10
Congressional Voting Records	435	16
Credit Approval	690	15
Diabetes	768	8
Haberman's Survival	306	3
Ionosphere	351	34
Phoneme CR	5404	5
Spambase	4601	57
Fourclass	862	2
German.numer	1000	24
Svmguide1	7089	4
Svmguide3	1284	21

Table II
TEST SET ACCURACY AVERAGED OVER $10-$FOLD CROSS VALIDATION FOR DATA SETS USING MEAN VALUES FOR DISCRETIZATION. NB STANDS FOR NAIVE BAYES, TAN FOR TREE AUGMENTED NAIVE BAYES, $k-$DBN FOR $k-$DEPENDENCY BAYESIAN NETWORKS, $k = 2$, AND UDBN FOR THE PROPOSED ALGORITHM

Data sets	NB	TAN	$k-$DBN	UDBN
Breast Cancer	97.18	96.52	97.31	97.66
Congressional Voting Records	90.11	93.21	94.62	95.48
Credit Approval	86.10	84.78	86.87	87.46
Diabetes	74.56	75.14	75.03	75.98
Haberman's Survival	75.09	74.41	76.43	77.86
Ionosphere	88.62	89.77	88.35	89.98
Phoneme CR	77.56	78.31	80.58	80.16
Spambase	90.41	89.78	89.27	92.37
Fourclass	77.46	77.61	77.94	79.06
German.numer	74.50	73.13	76.35	76.27
Svmguide1	92.39	91.61	92.98	94.17
Svmguide3	81.23	82.47	83.64	85.41

Table III
TEST SET ACCURACY AVERAGED OVER $10-$FOLD CROSS VALIDATION FOR DATA SETS USING DISCRETIZATION ALGORITHM SOAC. NB STANDS FOR NAIVE BAYES, TAN FOR TREE AUGMENTED NAIVE BAYES, $k-$DBN FOR $k-$DEPENDENCY BAYESIAN NETWORKS, $k = 2$, AND UDBN FOR THE PROPOSED ALGORITHM

Data Sets	NB	TAN	$k-$DBN	UDBN
Breast Cancer	96.12	95.60	96.76	97.65
Congressional Voting Records	90.11	91.42	92.61	94.16
Credit Approval	85.85	84.98	86.53	87.17
Diabetes	75.78	75.90	75.82	76.22
Haberman's Survival	74.66	73.78	75.64	77.31
Ionosphere	85.92	86.18	85.94	88.62
Phoneme CR	77.01	78.53	80.41	81.01
Spambase	89.30	89.04	90.69	92.54
Fourclass	78.58	79.52	78.97	79.96
German.Numer	74.61	74.01	75.31	76.15
Svmguide1	95.61	94.91	96.32	97.54
Svmguide3	77.25	79.99	80.75	82.92

VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a new algorithm for learning of the structure in Bayesian Networks. An important property of this algorithm is adding some numbers of arcs between features that captures unrestricted dependency among them. The number of arcs has been defined by the proposed algorithm internally. We have carried out a number of experiments on some binary classification data sets from the UCI machine learning repository and LIBSVM. The values of features in data sets are discritized by using mean value of each feature and applying discretization algorithm using sub-optimal agglomerative clustering. We have presented results of numerical experiments. These results clearly demonstrate that the proposed algorithm achieves comparable or better performance in comparison with traditional Bayesian Networks.

Our future work is applying the proposed algorithm to more complicated problems for learning BNs, e.g., problems with incomplete data, hidden variables, and multi class data sets.

REFERENCES

[1] H. Akaike, *Analysis of Cross Classified Data by AIC.* Ann. Inst. Statist. pp. 185-197, 1978.

[2] A. Asuncion and D. Newman, *UCI machine learning repository.* School of Information and Computer Science, University of California, 2007.

http://www.ics.uci.edu/mlearn/MLRepository.html, accessed May 2012.

[3] C. Chang and C. Lin, *LIBSVM: A library for support vector machines*. http://www.csie.ntu.edu.tw/cjlin/libsvm, accessed May 2012.

[4] J. Cheng, D. Bell, and W. Liu, *Learning Belief Networks from Data*. An Information Theory Based Approach. Artificial Intelligence, 137. pp. 43-90, 2002.

[5] G. F. Cooper and E. Herskovits, *A Bayesian Method for Constructing Bayesian Belief Networks from Databases*. Conference on Uncertainty in AI. pp. 86-94, 1990.

[6] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, *From data mining to knowledge discovery: An overview*. In Advances in Knowledge Discovery in Data Mining. AAAI Press, Menlo Park, CA. pp. 1-34, 1996.

[7] N. Friedman, D. Geiger, and M. Goldszmidt, *Bayesian network classifiers*. Machine Learning 29. pp. 131-163, 1997.

[8] B. Haeupler, T. Kavitha, R. Mathew, S. Sen, and R. E. Tarjan, *Incremental Cycle Detection, Topological Ordering, and Strong Component Maintenance*. 35th ACM Transactions on Algorithms (TALG). pp. 1-33, 2012.

[9] D. Heckerman, A. Mamdani, and W. Michael, *Real-World Applications of Bayesian Networks*. Communications of the ACM. pp. 38-68, 1995.

[10] D. Heckerman, *Bayesian Networks for Data Mining*. Data Mining and Knowledge Discovery1. pp. 79-119, 1997.

[11] D. Heckerman, D. Chickering, and C. Meek, *Large-Sample Learning of Bayesian Networks is NP-Hard*. Journal of Machine Learning Research. pp. 1287-1330, 2004.

[12] E. Herskovits and G. F. Cooper, *An Entropy-Driven System for Construction of Probabilistic Expert Systems from Databases*. 6th Internatial Conference on Uncertainty in Artificial Intelligence (UAI90), Cambridge, MA, USA, Elsevier Science, New York. pp. 54-62, 1991.

[13] Y. Jing, V. Pavlovic, and J. Rehg, *Efficient discriminative learning of Bayesian network classifier via Boosted Augmented Naive Bayes*. The 22 nd International Conference on Machine Learning, Bonn, Germany. pp. 369 - 376, 2005.

[14] A. Jonsson and A. Barto, *Active Learning of Dynamic Bayesian Networks in Markov Decision Processes*. Springer-Verlag Berlin Heidelberg. pp. 273284, 2007.

[15] D. Kitakoshi, H. Shioya, and R. Nakano, *Empirical analysis of an on-line adaptive system using a mixture of Bayesian networks*. Information Sciences, Elsevier. pp. 2856-2874, 2010.

[16] M. Koivisto and K. Sood, *Exact Bayesian structure discovery in Bayesian networks*. Journal of Machine Learning 5. pp. 549573, 2004.

[17] P. Kontkanen, T. Silander, T. Roos, and P. Myllymki, *Bayesian Network Structure Learning Using Factorized NML Universal Models*. Information Theory and Applications Workshop (ITA-08), IEEE Press. pp. 272 - 276, 2008.

[18] P. Langley, W. Iba, and K. Thompson, *An Analysis of Bayesian Classifiers*. In 10th International Conference Artificial Intelligence, AAAI Press and MIT Press. pp. 223-228, 1992.

[19] W. Liaoa and Q. Ji, *Learning Bayesian network parameters under incomplete data with domain knowledge*. Pattern Recognition, Elsevier. pp. 3046-3056, 2009.

[20] P. Myllymaki, *Advantages of Bayesian networks in data mining and knowledge discovery*. http://www.bayesit.com/docs/advantages.html, 2005, acessed April 2012.

[21] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. Networks of Plausible Inference, Morgan Kaufmann, 1988.

[22] J. Pearl, *Causality: Models, Reasonings and Inference*. Cambridge University Press. pp. 675685, 2003.

[23] M. Sahami, *Learning Limited Dependence Bayesian Classifiers*. In the 2nd International Conference. Knowledge Discovery and Data mining (KKD). pp. 335-338, 1996.

[24] T. Silander and P. Myllymaki, *A simple approach for finding the globally optimal Bayesian network structure*. In Proceedings of UAI-06. pp. 445-452, 2006.

[25] A. Singh and A. Moore, *Finding optimal Bayesian networks by dynamic programming*. Technical Report CMU-CALD-05-106, Carnegie Mellon University, 2005.

[26] G. Schwarz, *Estimating the Dimension of a Model*. Annals of Stastics. pp. 461-464, 1978.

[27] P. Spirtes, C. Glymour, and R. Sheines, *Causation, Prediction, and Search*. 1993.

[28] S. Taheri, M. Mammadov, and A. M. Bagirov, *Improving Naive Bayes Classifier Using Conditional Probabilities*. In the proceedings of Ninth Australasian Data Mining Conference (AusDM), Ballarat, Australia. Vol. 125. pp. 63-68, 2011.

[29] P. Weber, G. Medina-Oliva, C. Simon, and B. Iung, *Overview on Bayesian networks applications for dependability*. Risk-analysis and maintenance areas, Engineering Applications of Artificial Intelligence. pp. 671-682, 2010.

[30] A. Yatsko, A. M. Bagirov, and A. Stranieri, *On the Discretization of Continuous Features for Classification*. In the proceedings of Ninth Australasian Data Mining Conference (AusDM 2011), Ballarat, Australia. Vol. 125, 2011.

[31] C. Yuan, B. Malone, and X. Wu, *Learning Optimal Bayesian Networks Using A* Search*. In the proceedings of twenty second international joint conferenceon Artificial intelligence. pp. 2186-2191, 2011.

Chapter 4

Optimization Methods

4.1 A Globally Convergent Optimization Algorithm for Systems of Nonlinear Equations

In this section, a new algorithm to solve systems of nonlinear equations has been introduced. The idea in this algorithm is the combination of the gradient and Newton methods. We use the Gradient method due to its global convergence property, and the Newton method to improve the convergence rate. We consider two different combinations in this algorithm. In the first one, the step length is determined only along the gradient direction. In the second one, we find the step length along both the gradient and the Newton directions. The performance of the proposed algorithm is tested using some well known systems of nonlinear equations. The results provide evidence that this combination algorithm is robust and efficient.

Paper:

A Globally Convergent Optimization Algorithm for Systems of Nonlinear Equations

Published by:

Third Global Conference on Power Control and Optimization,

Gold Coast, Australia, 2010

Authors: Sona Taheri[a] and Musa Mammadov[a, b]

[a]Centre for Informatics and Applied Optimization,

School of Science, Information Technology and Engineering,

University of Ballarat, VIC 3353, Australia

[b]National ICT Australia, VRL, VIC 3010, Australia

Corresponding author:

Sona Taheri

e-mail: sonataheri@students.ballarat.edu.au

A GLOBALLY CONVERGENT OPTIMIZATION ALGORITHM FOR SYSTEMS OF NONLINEAR EQUATIONS

M. Mammadov, S. Taheri
School of Information Technology and Mathematical Science, Ballarat University, Ballarat VIC 3350, Australia
m.mammadov@ballarat.edu.au
sonataheri@students.ballarat.edu.au

Abstract
In this paper, a new algorithm is proposed for the solutions of system of nonlinear equations. This algorithm uses a combination of the gradient and Newton's methods. A novel dynamic combinator is developed to determine the contribution of the methods in the combination. Also, by using some parameters in the proposed algorithm, this contribution is adjusted. The efficiency of the algorithms is studied in solving system of nonlinear equations.

Keywords: System of nonlinear equations, Line search, Descent direction, Newton's Method

1. Introduction

We consider the problem of finding solutions to a system of nonlinear equations of the form

$$F(x) = \Theta, \tag{1}$$

where $\Theta = (0, \dots, 0)$, $x = (x_1, x_2, \dots, x_n)$ and $F = (f_1, f_2 \dots f_n)$ is a twice continuously differentiable function. At present, nonlinear systems of equations, due to frequently arise from various area of scientific and engineering computations, is still a topic research. These systems often arises when solving initial or boundary value problems in ordinary or partial differential equations ([1] and [2]). The application of nonlinear systems in load flow calculation in power system has been done by Spong and et all [3] in which their results of block Guass-Sidel iteration are compared with those of Newton-Raphson iteration.

Most of the methods for solving (1) are optimization-based methods in which Eq. (1) is reformulated as an optimization problem as follows:

$$f(x) = \frac{1}{2}\|F(x)\|^2, \tag{2}$$

where, here and throughout the paper, $\|\cdot\|$ stands for the Euclidean norm. Obviously, global optimal solutions of problem (2) with the zero value of the objective function correspond to solutions of system (1). In the last decades, many publications in this area have been done both in theoretical and especially numerical issues. Many search direction methods such as gradient method, Newton's method, quasi-Newton methods, conjugate gradient and coordinate direction methods, have been applied to find a minimizer of (2).

The steepest descent method (or gradient method) is a commonly used method. However, this method suffers from the slow speed and is easy plunging into local minima. In order to accelerate these difficulties, many methods have been used [4]. One way is the use of combination of different local optimization methods. It has been found that these methods show significant reduction in the number of iterations and the expense of function evaluations. In recent years, there has been a growing interest in applying these combination methods. Buckley [5] proposed a strategy of using conjugate gradient search direction for most iterations and using periodically a quasi-Newton step to improve the convergence. This algorithm offers the user the opportunity to specify the amount of available storage. Wang and et al. [6] proposed a revised conjugate gradient projection method, that is, a combination of the conjugate projection gradient and the quasi-Newton for nonlinear inequality constrained optimization problems. Recently, Y. Shi [7] proposed a combined method of Newton's and steepest descent methods for solving nonlinear system of equation within each iteration. Further in [8], in order to deal with an unconstrained problem, the combination of steepest descent with Newton and quasi-Newton methods were developed and compared with some traditional and existing methods. It is shown that this method is global convergent and at the same time has a high convergence rate.

Our procedure here for solving system of nonlinear equations is based on the gradient method and Newton's method which are combined into an integrated procedure, and especially the dynamic combination is of our interest challenge. The combined algorithms proposed in this paper is different from the existing algorithms [5,6,7,8]. In the other words, we propose a novel algorithm with a new combination which offers the user the opportunity to specify the amount contribution of the methods.

2. Descent Methods

Many techniques have been devoted for solving (2) as well as (1). These problems are usually carried out using iterative methods due to the fact that there are generally no analytical methods to solve these problems. Among the variety of the exiting methods, the descent direction methods are the most popular techniques because of their fast convergence property.

Denote $\nabla f(x)$ by $g(x)$ and $\nabla f(x_k)$ by g_k. Given an initial point $x_1 \in R^n$ and an error tolerance $\epsilon > 0$, each iteration $k = 1,2,\dots$ of a descent direction method contains the following steps:

1. If $\|g_k\| < \epsilon$, then stop;
2. compute a descent direction d_k at x_k satisfying
$$g_k d_k < 0 \tag{3}$$
3. determine an appropriate step length $\alpha_k > 0$;
4. set $x_{k+1} = x_k + \alpha_k d_k$, and go to the next iteration.

Depending on the choice of d_k and α_k, where d_k is a descent direction and α_k is a line search factor, different descent direction methods have been developed. There are some criterions for accepting α_k as an admissible step length such as backtracking method, Armijo, Goldestain and Wolfe line search rules. In Wolfe's case, the step length α_k is determined by an inexact line search along the direction d_k satisfying

$$f(x_k + \alpha d_k) \leq f(x_k) + \rho\alpha g_k^T d_k \tag{4}$$
$$g_{k+1}^T d_k \geq \sigma g_k^T d_k \tag{5}$$

where $\rho \in (0,1)$ and $\sigma \in (\rho, 1)$ are fixed parameters. Denote Ω by the level set $\{x | f(x) \leq f(x_1)\}$ and consider the Wolfe conditions (4) and (5) to determine α in the above algorithm, then the global convergence of the above algorithm is given by the following Theorem.

Theorem 1 ([9], Theorem 2.5.4]). Let α_k in the above descent algorithm be defined by (4) and (5). Let also d_k satisfy

$$cos\,\theta_k \geq \delta \tag{6}$$

for some $\delta > 0$ and for all k, where θ_k is the angle between d_k and $-g_k$. If g(x) exists and is uniformly continuous on the level set Ω, then either $g_k = 0$ for some k, or $g_k \to 0$.

One of the most widely used methods satisfying Theorem 1 is the gradient method, in which $d_k = -g_k$, for all k. Although the method is global convergent and usually works well in some early steps, as a stationary point is approached, it may descend very slowly. In fact, it is shown that the convergence rate of the gradient method is at least linear, and the following bound holds

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} \leq \frac{\lambda_{max} - \lambda_{min}}{\lambda_{max} + \lambda_{min}}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest eigenvalues of the Hessian matrix respectively.

In order to cope with the above-mentioned difficulties, one can use Newton's method with superlinear convergence property. At the k-th iteration, the classical Newton's direction is the solution of the following system

$$H_k d = -g_k \tag{7}$$

where H_k is the Hessian matrix at x_k. If H is positive definite then the Newton's direction is a descent direction and consequently the system has a unique solution. Even when H is positive definite, it is not guaranteed that Newton's method will be globally convergent. Although Newton's method generally converges faster than the gradient method, it depends on a starting point. On the other hand, the application of Newton's method to the solving of nonlinear equations is expensive for a large scale problem. A number of techniques avoiding the direct computation of H may be used and upon different approximation there are different methods. In this category are the quasi-Newton methods which approximate second derivatives in a most subtle and efficient way. Another alternative is the use of a combined method of different local optimization methods which lead naturally to powerful algorithms and has been attracted extensive attention in recent years. One of the most successful methods of this category, introduced by Shi [8], uses a combination of the gradient method and Newton's method. Another algorithm in [8] uses quasi-Newton method instead of Newton's method in this combination. Here, we just compare our results with the first one introduced in [8]. We refer this algorithm by ShA. This algorithm is efficient algorithm for solving problem (2) due to its global convergence property and superlinear convergence rate. The direction in algorithm ShA is very close to Newton's direction. However, practical implementations show that, in some cases the gradient method can be a more suitable choice than Newton's method. For instance, when the difference of the function values, in two previous iterations, and also the value of the gradient in the previous iteration is large enough, the gradient method may work better than Newton's method.

3. Proposed Algorithm

Our aim here is to present an algorithm with two different combinations for solving problem (2), and as well as problem (1). The first case is the usual combination which has been developed in some research works. Another one is a combination of the gradient and standard Newton's methods. In this case, in each iteration α_k is determined only along the gradient direction. Both the proposed combinations are constructed so that they satisfy in the condition of descent methods and as well as Theorem 1.

Let δ_0, η, ρ and σ be four parameters so that $0 < \delta < 1$, $0 < \eta < 1$, $0 < \rho < \frac{1}{2}$ and $\rho < \sigma < 1$. Take any positive constants γ_1, γ_2 and b_i, $i = 1,2,3$, such that $\gamma_i > 1$, $0 < b_1 < 1$, $1 < b_2 < 1/\delta$ and $b_3 > 1$ and initialize Λ_0 by 1. By taking $f(x_0) = f(x_1)$, the steps of our algorithm are as follows.

0. Choose a starting point $x_1 \in R^n$, and an error tolerance $\epsilon > 0$. Set $k = 1$ and go to the next step.
1. If $\|f(x_k)\| < \epsilon$, then stop.
2. If Newton's direction d_1 is not computable, due to the singularity of the Hessian, then compute the gradient direction $d_2 = -g_k$ at x_k, and go to step 8.
3. Compute the gradient direction d_2 and Newton's direction d_1 at x_k that satisfies (7).
4. Set $\delta = \delta_0$ and $\Lambda = \Lambda_0$. If $|f(x_k) - f(x_k)| > \gamma_1$ and $\|g_k\| > \gamma_2$ then set $\delta \leftarrow b_2\delta_0$ and go to step 7.
5. If $k = 1$ or if $\|g_k\| \leq \|g_{k-1}\|$, set $\overline{x} = x_k + d_1$ and go to step 6.
6. If $|f(\overline{x})| < |f(x_k)|$ and $\|g(\overline{x})\| \leq \eta\|g_k\|$, then $\delta \leftarrow b_1\delta_0$.
7. If $d_1^T d_2 \geq 0$ go to step 9, otherwise go to the next step.
8. Use rules (4) and (5) to determine a step length $\alpha_k > 0$ along the direction $d_k = d_2$, set $s_k = \alpha_k d_k$ and go to step 12.
9. Compute ξ as follows:
$$\xi = \frac{1}{\Lambda + |f(x_k) - f(x_k)|} \tag{8}$$
and set $d(\xi) = (1-\xi)d_2 + \xi d_1$.
10. If $d(\xi)^T d_2 < \delta\|d(\xi)\|\|d_2\|$, set $\Lambda \leftarrow b_3\Lambda$ and and go back to step 9.
11. Consider one of the following two versions to calculate s_k:
a. Use rules (4) and (5) to determine a step length $\alpha_k > 0$ along the direction $d_k = d_2$ and set $s_k = \alpha_k(1-\xi)d_2 + \xi d_1$.
b. Use rules (4) and (5) to determine a step length $\alpha_k > 0$ along the direction $d_k = d(\xi)$ and set $s_k = \alpha_k d_k$.
12. Set $x_{k+1} = x_k + \alpha_k d_k$ and go to step 1.

Parameters b_1, b_2 and b_3 are positive constants so that they offer the user the opportunity to specify the amount contribution of the methods. More precisely, when the slope of the function is slight, the algorithm tends to Newton's method, otherwise it is considered close to the gradient method. In (8), by considering Λ is enough close to , when difference between two previous values of the function is high then is close to , and $\xi \to 1$ as $|f(x_k) - f(x_k)| \to 0$. Moreover, this equation is a dynamic form and has a crucial rule in the algorithm so that it specifies the amount contribution of the methods. This guaranteed that, near the solution, we can get the optimal point with a superlinear convergence rate.

In step 11 of the above Algorithm, we use two different strategies by means of the combination. Step 11.a is a new combination and different from the existing methods in the literature. In this combination, the step length is determined only along the gradient direction. In the other words, finally, we use a novel combination of pure Newton's method (with $\alpha_k = 1$) and the gradient method.

4. Global convergence Theorem

Here, we establish the global convergence of the above algorithm based on the global convergence property of Theorem 1.

Theorem 2. Consider using Algorithm 1 to solve problem (2). Assume that g(x) exists and is uniformly continuous on Ω. Then either $g_k = 0$ for some k, or $g_k \to 0$.

Proof. Let assume that $g_k \neq 0$ for all k. Here, we show that the direction d_k obtained by the algorithm satisfies condition (6) of Theorem 1. Denote $\delta^* \leftarrow b_1\delta_0$. Clearly, $\delta^* \in (0,1)$. We show that condition (6) holds for δ^*; that is,
$$\cos\theta_k \geq \delta^* \tag{9}$$
for all k, where θ_k is the angle between d_k and $d_2 = -g_k$.

Take any integer k. We have one of the following cases.

Case 1. The direction d_k is obtained at step 8. In this case $d_k = -g_k$ and, consequently, it is easy to see that $\cos\theta_k = 1 > \delta^*$; that is, (6) holds.

Case 2. The direction d_k is obtained at step 11 in the form $d_k = d(\xi)$. According to steps 9-10, the number ξ is chosen so that the inequality $d(\xi)^T d_2 \geq \delta\|d(\xi)\|\|d_2\|$, holds, where $\delta \leftarrow b_1\delta_0$ or $\delta \leftarrow b_2\delta_0$. The existence of such a number ξ follows from the fact that $d(\xi) \to d_2$ as $\xi \to 0$ (or $\Lambda \to 0$). Then, we have
$$\cos\theta_k \geq \frac{d(\xi)^T d_2}{\|d(\xi)\|\|d_2\|} \geq \delta \geq \delta^* \tag{10}$$
that is, (9) holds.

In the above cases, the obtained direction, d_k, satisfy in the assumption of Theorem 1, hence the remainder proof is similar to the proof of Theorem 1 given in [9].

5. Test results

We have tested our algorithm described above on a several test problems given in [10] and [11]. The group of methods we have compared includes our proposed algorithms, the algorithm presented in [8] and also the gradient and Newton's method.

In the proposed algorithm, we use two different combination as described in steps 11.a and 11.b; we refer these cases as 'Alg-a' and 'Alg-b', respectively. The group of methods we have compared includes Alg-a, Alg-b, the gradient method (GM), Newton's method (NM), and ShA presented in [8]. In all algorithms we used the wolfe line search rules to find acceptable step length.

All the algorithms are terminated if the iteration number exceeds 500. Table 1 lists some functions to be tested the algorithms. Table 2 lists the performance of the above-mentioned algorithms relative to the number of iterations used. Table 3 shows the summary of convergence results for the Table 2. Also, the summary of the convergence results of the algorithms with considering 50 initial random points is given in Table 4. In these tables, notations 'convergence', 'almost convergence (AC)' and 'not convergence (NoC)' mean as follows: 'convergence' if $|f(x_k)| < 10^{-5}$; 'almost convergence' if $10^{-5} < |f(x_k)| \leq 10^{-2}$; otherwise 'not convergence'.

The parameters δ_0, η, ρ, σ, γ_i and b_i used in this paper are: $\delta_0 = 0.001$, $\eta = 0.99$, $\rho = 0.001$, $\sigma = 0.9$, $\gamma_1 = \gamma_2 = n$, $b_1 = 0.01$, $b_2 = \frac{1}{b_1}$ and $b_3 = 1.1$.

Table 1. Functions for testing the performance

Problem	n
P1. $F(x) = \left(x_1 + x_2, 2(x_1^2 + x_2^2 - 1) - \frac{1}{3}\right)$	2
P2. Powell Singular [10]	4
P3. Extended Kearfott [11]	7
P4. Extended Rosenbrock [10]	8
P5. Extended Eiger-Sikorski-Stenger[11]	10
P6. Trigonometric [10]	10
P7. Discrete Boundary Value [10]	20

Table 2. Functions for testing the performance

	Point	Alg-a	Alg-b	ShA	NM	GM
P1	x_0	4	5	4	5	17
	$10x_0$	8	8	11	12	20
	10^2x_0	24	12	16	18	13
	10^3x_0	11	19	22	24	23
P2	x_0	7	10	11	11	135
	$10x_0$	9	15	17	19	AC
	10^2x_0	31	39	21	NoC	NoC
	10^3x_0	38	73	27	NoC	NoC
P3	x_0	13	10	9	NoC	10
	$10x_0$	8	11	15	NoC	13
	10^2x_0	13	13	24	NoC	15
	10^3x_0	26	22	21	NoC	18
P4	x_0	15	27	16	17	NoC
	$10x_0$	69	382	38	39	NoC
	10^2x_0	138	NoC	80	NoC	NoC
	10^3x_0	498	NoC	195	NoC	NoC
P5	x_0	10	5	7	NoC	6
	$10x_0$	17	11	14	NoC	8
	10^2x_0	20	17	18	NoC	17
	10^3x_0	20	29	23	NoC	25
P6	x_0	6	12	8	9	7
	$10x_0$	11	AC	NoC	NoC	22
	10^2x_0	10	9	NoC	NoC	16
	10^3x_0	11	23	NoC	NoC	14
P7	x_0	4	5	3	4	AC
	$10x_0$	12	14	7	8	NoC
	10^2x_0	20	22	20	23	NoC
	10^3x_0	29	31	29	NoC	NoC

Table 3. Summary of convergence results for Table 2

Algorithm	Convergence	AC	NoC
Alg-a	100.00	0.00	0.00
Alg-b	89.29	3.57	7.14
ShA	89.29	0.00	10.71
NM	42.86	0.00	54.14
GM	60.71	7.14	32.15

Table 4. Convergence results with 50 initial random points for each problem

Algorithm	Convergence	AC	NoC
Alg-a	96.86	2.29	0.85
Alg-b	88.57	1.43	10.00
ShA	89.71	0.57	9.70
NM	43.71	0.57	54.70
GM	57.71	3.71	38.58

6. Conclusion

A combined algorithm of the gradient and Newton's methods has been presented for solving system of nonlinear equations. We have considered two different combinations. On of them is a simple case which has been recently introduced in some research works. Another one is a new combination and different from others in the literature. According to the above results, it is clear that the combination algorithms, especially the proposed algorithm with the new combination, are more efficient than others.

7. References

[1] H. K. Gummel, "A self-consistent iterative scheme for one-dimensional steady state transistor calculations", IEEE Trans. Electron Devices ED-11, 1964, pp. 455-465.

[2] T. Kerkhoven, "A proof of convergence of Gummel's algorithm for realistic boundary condition", SIAM J. Numer. Anal., 23(6), 1986, pp. 1121-1137.

[3] M. Ili-Spong, I. Norman Katz, H. Dai, and J. Zaborszky, "Block diagonal dominance for systems of nonlinear equations with application to load flow calculations in power systems", Mathematical Modelling, 5, 1984, pp. 275-297.

[4] R. Fletcher, 'Practical methods of optimization", Second Edition, Wiley, New York, 1987.

[5] A. G. Buckley, "A combined conjugate-gradient quasi-Newton minimization algorithm", Mathematical Programming, 15, 1978, pp. 200-210.

[6] W. Wang, Y.F. Xu, "A revised conjugate gradient projection algorithm for inequality constrained optimization", J. of Computational Mathematics, 23, 2005, pp. 217-224.

[7] Y. Shi, "A globalization procedure for solving nonlinear systems of equations", Numerical Algorithms, 12, 1996, pp. 273-286.

[8] Y. Shi, "Globally convergent algorithms for unconstrained optimization", Computational Optimizatin and Applications, 16, 2000, pp. 295-308.

[9] W. Sun, and Y.X. Yuan, "Optimization theory and methods: nonlinear programming", Springer, 2006.

[10] J.J. More, B.S. Garbow, and K.E. Hillstrom, "Testing unconstrained optimization software", ACM Transactions on Mathematical Software, 7, 1981, pp. 17-41.

[11] W M.N. Vrahatis, "Solving systems of nonlinear equations using the nonzero value of the topological degree", ACM Transactions on Mathematical Software, 14, 1988, pp. 312-329.

4.2 Solving Systems of Nonlinear Equations using a Globally Convergent Optimization Algorithm

This section presents an extended version of the proposed algorithm in Section 4.1. The proposed algorithm is based on a combination of the gradient and Newton methods, and it can be applied for solving systems of nonlinear equations. To validate the proposed algorithm, numerical experiments with a number of well known nonlinear equations systems have been carried out. The results demonstrate the high efficiency of the algorithm.

Paper:

Solving Systems of Nonlinear Equations using a Globally Convergent Optimization Algorithm

Published by:

Global Journal of Technology and Optimization,

Volume 3, pp. 132-138, 2012

Authors: Sona Taheri[a] and Musa Mammadov[a, b]

[a]Centre for Informatics and Applied Optimization,

School of Science, Information Technology and Engineering,

University of Ballarat, VIC 3353, Australia

[b]National ICT Australia, VRL, VIC 3010, Australia

Corresponding author:

Sona Taheri

e-mail: sonataheri@students.ballarat.edu.au

Transaction on Evolutionary Algorithm and Nonlinear Optimization
ISSN: 2229-8711 Online Publication, June 2012
www.pcoglobal.com/gjto.htm
NG-O21/GJTO

GJTO Global Journal of Technology & Optimization Volume 3, 2012

SOLVING SYSTEMS OF NONLINEAR EQUATIONS USING A GLOBALLY CONVERGENT OPTIMIZATION ALGORITHM

Sona Taheri, Musa Mammadov

Centre for Informatics and Applied Optimization, School of Science, Information Technology and Engineering, University of Ballarat, Victoria, Australia
Email: sonataheri@students.ballarat.edu.au, m.mammadov@ballarat.edu.au

Received February 2010, Revised September 2011, Accepted May 2012

Abstract
Solving systems of nonlinear equations is a relatively complicated problem for which a number of different approaches have been presented. In this paper, a new algorithm is proposed for the solutions of systems of nonlinear equations. This algorithm uses a combination of the gradient and the Newton's methods. A novel dynamic combinatory is developed to determine the contribution of the methods in the combination. Also, by using some parameters in the proposed algorithm, this contribution is adjusted. We use the gradient method due to its global convergence property, and the Newton's method to speed up the convergence rate. We consider two different combinations. In the first one, a step length is determined only along the gradient direction. The second one is finding a step length along both the gradient and the Newton's directions. The performance of the proposed algorithm in comparison to the Newton's method, the gradient method and an existing combination method is explored on several well known test problems in solving systems of nonlinear equations. The numerical results provide evidence that the proposed combination algorithm is generally more robust and efficient than other mentioned methods on some important and difficult problems.

Keywords: Systems of nonlinear equations, Newton's Method, Gradient method, Line search, Global convergence

1. Introduction

The solutions of systems of equations have a well-developed mathematical and computational theory when solving linear systems, or a single nonlinear equation. The situation is much more complicated when the equations in the system do not exhibit nice linear or polynomial properties. In this general case, both the mathematical theory and computational practices are far from complete understanding of the solution process.
Systems of nonlinear equations arise in various domains of practical importance such as engineering, medicines, chemistry, and robotics [15, 21, 37]. They appear also in many geometric computations such as intersections, minimum distance, creation of centenary curves, and when solving initial or boundary value problems in ordinary or partial differential equations [13] and [16]. The application of nonlinear systems in load flow calculation in power system has been done by Spong and et. all [32] in which their results of block Guass-Sidel iteration are compared with those of Newton-Raphson iteration. Solving such a system involves finding all the solutions of equations contained in the mentioned system.
In this paper, we consider the problem of finding solutions to a system of nonlinear equations of the form

$$F(x) = \Theta, \qquad (1)$$

where $F: R^n \to R^n, \Theta = (0, \dots, 0)$, and x refers to n variables, $x = (x_1, \dots, x_n)$. We denote the i-th component of F by f_i, where $f_i: R^n \to R$ is a nonlinear function and twice continuously differentiable on a convex set $D \subset R^n$.
There is a class of methods for the numerical solutions of the system (1), which arises from iterative procedure used for systems of linear equations [12]. These methods use reduction to simpler one-dimensional nonlinear equations for the components $f_1, f_2 \dots f_n$.
There are some iterative methods for solving systems of nonlinear equations in the book written by Kelley [15]. A wide range class of iterative methods for solving systems of nonlinear equations has been suggested in the papers [2, 11, 25, 26].
Most of the methods for solving (1) are optimization-based methods [1, 4, 6, 11, 17, 22, 37]. In the approach proposed in [22], the system (1) is transformed in to a constraint optimization problem. At each step, some equations that are satisfied at the current point are treated as constraints and the other ones as objective functions. In a strategy based on optimization methods, at each iteration, a quadratic function is minimized to determine the next feasible point to step to. The quadratic function is the squared norm of the original system.
To find a solution of (1), one can transform the system (1) into an unconstrained optimization problem and then solving the new unconstrained problem instead by applying an optimization method. The transformed problem is formulated as:

$$f(x) = \frac{1}{2}\|F(x)\|^2, \qquad (2)$$

where, here and throughout the paper, $\|\cdot\|$ stands for the Euclidean norm. Obviously, optimal solutions of problem (2) with the zero value of the objective function correspond to global solutions of system (1).
In the last decades, many publications, both in theoretical and especially numerical issues, have been done for solving the problem (2) [3, 5, 9, 10, 18, 24, 27, 31, 33, 35]. Many search

direction methods such as the gradient method, the Newton's method, the quasi-Newton methods, the conjugate gradient and coordinate direction methods have been applied to find a minimizer of (2).

The steepest descent method (or gradient method) is a commonly used method. It has the globally convergence property, however, this method suffers from the slow speed and is easy plunging into local minima. In order to accelerate these difficulties, many methods have been used [10]. One way is the use of combination of different local optimization methods. It has been found that these methods show significant reduction in the number of iterations and the expense of function evaluations. In recent years, there has been a growing interest in applying these combination methods [7, 29, 30, 36]. Buckley [7] proposed a strategy of using a conjugate gradient search direction for most iterations and using periodically a quasi-Newton step to improve the convergence. This algorithm offers the user the opportunity to specify the amount of available storage. Wang and et al. [36] proposed a revised conjugate gradient projection method, that is, a combination of the conjugate projection gradient and the quasi-Newton methods for nonlinear inequality constrained optimization problems. Recently, Y. Shi [29] proposed a combined method of the Newton's and the steepest descent methods for solving nonlinear systems of equations within each iteration. Further in [30], in order to deal with an unconstrained problem, the combination of the steepest descent with the Newton and the quasi-Newton methods were developed and compared with some traditional and existing methods.

Our procedure here for solving systems of nonlinear equations is based on the combination of local optimization methods. We apply the gradient and the Newton's methods for our combination algorithm. They are combined into an integrated procedure, and especially the dynamic combination is of our interest challenge. The combined algorithms proposed in this paper are different from the existing algorithms [7, 29, 30, 36]. In the other words, we propose a novel algorithm with a new combination which offers the user the opportunity to specify the amount contribution of the methods.

The rest of the paper is organized as follows: Section 2 gives a brief review to preliminaries about optimization. In Section 3, we review the descent methods. We present the proposed combination algorithm in Section 4. The global convergence property of this algorithm has been proved in Section 5. We have demonstrated the efficiency of the proposed algorithm with some experiments in Section 6. Section 7 concludes the paper.

2. Preliminaries

Usually, optimization methods are iterative. The basic idea is that, with an initial guess of the optimal values of the variables, $\{x_0\}$, an optimization method generates a sequence $\{x_k\}$ of improved estimates until it reaches a solution. When $\{x_k\}$ is a finite sequence, the last point is the optimal solution; when $\{x_k\}$ is infinite, it has a limit point which is the optimal solution of the problem. The strategy used to move from one iterate to the next distinguishes one algorithm from another. A typical behavior of an algorithm which is regarded as acceptable is that the iterates $\{x_k\}$ move steadily towards the neighborhood of a point local minimizer, and then rapidly converge to that point. When a given convergence rule is satisfied, the iteration will be terminated. In general, the most natural stopping criterion is

$$\|g_k\| < \epsilon, \tag{3}$$

where g_k stands for $\nabla f(x)$ at x_k and f is defined by (2). $\epsilon > 0$ is a prescribed error tolerance.

Let x_k be the k-th iterate, d_k k-th search direction, and α_k k-th step length, then the k-th iteration is

$$x_{k+1} = x_k + \alpha_k d_k. \tag{4}$$

There are two fundamental strategies for moving from the current point x_k to a new state x_{k+1}: Trust region [24, 38] and Line search [24, 28, 31, 33].

In the trust region strategy, the information gathered about f is used to construct a model function whose behavior near the current point x_k is similar to that of the actual objective function f. When x is far from x_k, the model may not be a good approximation of f. Therefore, the search for a minimizer of the model is restricted to some region around x_k.

In the line search strategy, the algorithm chooses a direction d_k and searches along this direction from the current iterate x_k for a new iterate with a lower function value.

The line search and trust-region approaches differ in the order in which they choose the direction and distance of the move to the next iterate. Line search starts by fixing the direction d_k and then identifying an appropriate distance, namely the step length α_k. In trust region, firstly a maximum distance is chosen, the trust region radius, and then a direction and a step that attain the best possible improvement subject to this distance constraint is found. If this step proves to be unsatisfactory, the distance measure will be reduced and tried again [24].

A trust region method is effective since it limits the step to a region of greater confidence in the local model and attempts to utilize more information from the local model for finding a shortened step. However, trust region models are more difficult to formulate and solve than a line search strategy [31]. In this paper, we will focus on line search strategies.

2.1 Line Search

Line search methods are traditional and efficient methods for solving unconstrained minimization problems. Its convergence has attracted more attention in recent years [3, 19, 35].

The success of a line search method depends on effective choices of both the direction d_k and the step length α_k. It is clarified that the search direction plays a main role in the algorithm and that step length guarantees the global convergence in some cases.

There are two alternatives for finding the distance to move along d_k namely the exact line search and inexact line search [19, 28, 31, 33]. In the exact line search, the following one-dimensional minimization problem will be solved to find a step length α:

$$min_\alpha f(x_k + \alpha d_k). \tag{5}$$

If we choose α_k such that the objective function has acceptable descent amount, i.e., it means the descent $f(x_k) - f(x_k + \alpha_k d_k) > 0$ (6)

is acceptable by users, such a line search is called inexact line search. Since, in practical computation, exact optimal step length generally cannot be found, and it is also expensive to find almost exact step length, therefore the inexact line search with less computation load is highly popular.

A simple condition we could impose on α_k in an inexact line search is to require a reduction in f:

$$f(x_k + \alpha_k d_k) < f(x_k). \tag{7}$$

It has been shown that this requirement is not enough to produce convergence to optimal point [24, 33]. The difficulty is that there is not always a sufficient reduction in f at each step, a concept we discuss next.

There are several inexact line search rules for choosing an appropriate step length α_k, for example the Armijo rule, the

Goldstein rule, and the Wolfe-Powell rules [24, 31, 33], which are described briefly in the following.

Armijo Rule and Goldstein Rule

Armijo rule is as follows:

$f(x_k) - f(x_k + \beta^m \tau d_k) \geq -\rho \beta^m \gamma g_k^T d_k$,

where $\beta \in (0,1)$, $\rho \in \left(0, \frac{1}{2}\right)$, and $\gamma > 0$, (8)

$m = 0, 1, \ldots$, are tried successively until the above inequality is satisfied for $m = m_k$.

Goldstein presented the following rule. Let

$$I = \{\alpha > 0 : f(x_k + \alpha d_k) < f(x_k)\}$$

be an interval. In order to guarantee the function decreases sufficiently, we want to choose α such that it is away from the two end points of the interval I.

The Goldstein conditions are

$$f(x_k + \alpha d_k) \leq f(x_k) + \rho \alpha g_k^T d_k, \quad (9)$$

and

$$f(x_k + \alpha d_k) \geq f(x_k) + (1-\rho) \alpha g_k^T d_k, \quad (10)$$

which exclude those points near the right end point and the left end point.

Wolfe-Powell Rule

It is possible that the rule (10) excludes the minimizing value of α outside the acceptable interval. Instead, the Wolfe-Powell gives another rule to replace (10):

$$g_{k+1}^T d_k \geq \sigma g_k^T d_k, \sigma \in (\rho, 1).$$

Therefore, the step length α_k in the Wolfe-Powell rule will be determined along the direction d_k satisfying:

$$f(x_k + \alpha d_k) \leq f(x_k) + \rho \alpha g_k^T d_k, \quad (11)$$

and

$$g_{k+1}^T d_k \geq \sigma g_k^T d_k, \sigma \in (\rho, 1). \quad (12)$$

The Wolfe-Powell rule is a popular inexact line search rule. We will use it in our algorithm and all experiments in this paper.

2.2. Search Directions

The search direction in gradient-based methods often has the form

$$d_k = -B_k^{-1} g_k, \quad (13)$$

where B_k is a symmetric and nonsingular matrix. For example, in the gradient method B_k is simply the identity matrix, $d_k = -g_k$ [3, 24, 33]. $d_k = -H_k^{-1} g_k$ corresponds to the Newton's method with H_k^{-1} being available, where H_k is an exact Hessian of f [24, 33]. In quasi-Newton methods, B_k is an approximation to the Hessian H_k that is updated at every iteration by means of a low-rank formula [5, 9, 24, 33]. In the conjugate gradient method, d_k is defined by

$d_k = -g_k + \beta_k d_{k-1}$, $k \geq 2$, and $d_1 = -g_1$; β_k is a parameter [8,18, 24, 33].

When d_k is defined by (13) and B_k is positive definite, we have $d_k^T g_k = -g_k^T B_k^{-1} g_k < 0$, and therefore d_k is a descent direction.

The search direction d_k is generally required to satisfy the descent condition:

$$g_k d_k < 0. \quad (14)$$

The condition (14) guarantees that d_k is a descent direction of f at x_k [24, 33].

3. Descent Methods

Many techniques have been devoted for solving (2), as well as (1). These problems are usually carried out using iterative methods due to the fact that there are generally no analytical methods to solve these problems. Among the variety of the exiting methods, the descent direction methods are the most popular techniques because of their fast convergence property. A general descent direction algorithm is given in the Algorithm 1.

Algorithm 1. A General Descent Framework

```
0. Lets $x_1 \in R^n$ be a given initial point, and $\epsilon > 0$ an error
   tolerance. Each iteration $k = 1,2,\ldots$ of a descent direction
   method contains the following steps:
1. If $\|g_k\| < \epsilon$, then stop.
2. Compute a descent direction $d_k$ at $x_k$ satisfying (14).
3. Determine an appropriate step length $\alpha_k > 0$.
4. Set $x_{k+1} = x_k + \alpha_k d_k$, and go to the next iteration.
```

Let $\Omega = \{x | f(x) \leq f(x_1)\}$ be the level set, and consider the Wolfe-Powell conditions (11) and (12) to determine α_k, then the global convergence of the Algorithm 1 is given by the following Theorem [33].

Theorem 1. Let α_k in the above descent direction algorithm be defined by (11) and (12). Let also d_k satisfies

$$\cos \theta_k \geq \delta, \quad (15)$$

for some $\delta > 0$ and for all k, where θ_k is the angle between d_k and $-g_k$. If $g(x)$ exists and is uniformly continuous on the level set Ω, then either $g_k = 0$ for some k, or $f_k \to -\infty$, or $g_k \to 0$.

Proof can be found in [33], Theorem 2.5.4.

One of the most widely used methods satisfying Theorem 1 is the gradient method, in which $d_k = -g_k$ for all k. Although the method is globally convergent and usually works well in some early steps, as a stationary point is approached, it may descend very slowly. In fact, it is shown that the convergence rate of the gradient method is at least linear, and the following bound holds

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} \leq \frac{\lambda_{max} - \lambda_{min}}{\lambda_{max} + \lambda_{min}}, \quad (16)$$

where λ_{max} and λ_{min} are the largest and the smallest eigenvalues of the Hessian matrix, respectively.

In order to cope with the above-mentioned difficulties, one can use the Newton's method with the quadratic convergence property. At the k-th iteration, the classical Newton's direction is the solution of the following system:

$$H_k d_k = -g_k, \quad (17)$$

where H_k is the Hessian matrix at x_k. If H is positive definite, then the Newton's direction is a descent direction and consequently the system has a unique solution. Even when H is positive definite, it is not guaranteed that Newton's method will be globally convergent. Although the Newton's method generally converges faster than the gradient method, it depends strongly on a starting point. On the other hand, the application of the Newton's method for solving the nonlinear equations is expensive due to the direct calculations of second order derivatives of the function, H. A number of techniques avoiding the direct computation of H may be used. Upon different approximation there are different methods. In this category are the quasi-Newton methods which approximate second derivatives in a most subtle and efficient way. Another alternative is the use of a fusion of different local optimization methods which lead naturally to powerful algorithms and has been attracted extensive attention in recent years. One of the most successful methods of this category, introduced by Shi [30], uses a combination of the gradient method and the Newton's method. This algorithm is an efficient algorithm for solving problem (2) due to its global convergence property. In our experiments, we compare our results with this combination algorithm and refer it by ShA. The direction in algorithm ShA is very close to the Newton's direction. However, practical implementations show that, in some cases the gradient method can be a more suitable choice than the Newton's method. For instance, when the difference of the function values, in two previous iterations, and also the value of the gradient in the previous iteration is large enough, the gradient method may work better than the Newton's method.

4. Proposed Algorithm

Our aim here is to present an algorithm with two different combinations for solving the problem (2), as well as the problem (1). Both proposed combinations are constructed so that they satisfy in the condition of descent methods and as well as the Theorem 1.

Let δ_0, η, ρ and σ be four parameters so that $0 < \delta < 1$, $0 < \eta < 1$, $0 < \rho < \frac{1}{2}$ and $\rho < \sigma < 1$. Take any positive constants γ_1, γ_2 and b_i, $i = 1,2,3$, such that $\gamma_i > 1$, $0 < b_1 < 1$, $1 < b_2 < 1/\delta$ and $b_3 > 1$ and initialize Λ_0 by 1. Let also T, and τ be very large and small positive numbers, respectively, and let $f(x_0) = f(x_1)$. The steps of the proposed algorithm are as follows.

Algorithm 2. A Combination of the Gradient and Newton's Methods

```
0.  Choose a starting point $x_1 \in R^n$, and an error tolerance $\epsilon > 0$. For $k = 1,2,\ldots$ do
1.  If $\|g(x_k)\| < \epsilon$, then stop.
2.  If the Newton's direction $d_1$ is not computable, due to the singularity of the Hessian, then compute the gradient direction $d_2 = -g_k$ at $x_k$, and go to step 8.
3.  Compute the gradient direction $d_2$ and the Newton's direction $d_1$ at $x_k$ that satisfies (17).
4.  Set $\delta = \delta_0$ and $\Lambda = \Lambda_0$. If $|f(x_k) - f(x_{k-1})| > \gamma_1$ and $g_k > \gamma_2$, set $\delta \leftarrow b_2\delta_0$ and go to step 7.
5.  If $k = 1$ or if $\|g_k\| \le \|g_{k-1}\|$, set $\bar{x} = x_k + d_1$ and go to step 6.
6.  If $f(\bar{x}) < f(x_k)$ and $g(\bar{x}) \le \eta g_k$ then $\delta \leftarrow b_1\delta_0$.
7.  If $d_1^T d_2 \ge 0$ go to step 9, otherwise go to the next step.
8.  Use rules (11) and (12) to determine a step length $\alpha_k > 0$ along the direction $d_k = d_2$, set $s_k = \alpha_k d_k$ and go to step 12.
9.  Compute $\xi$ as follows:
        $$\xi = \frac{1}{\Lambda + |f(x_k) - f(x_{k-1})|} \qquad (18)$$
    and set $d(\xi) = (1-\xi)d_2 + \xi d_1$.
10. If $d(\xi)^T d_2 < \delta\|d(\xi)\|\|d_2\|$, set $\Lambda \leftarrow b_3\Lambda$ and and go back to step 9.
11. Consider one of the following two versions to calculate $s_k$:
    a. Use rules (11) and (12) to determine a step length $\alpha_k > 0$ along the direction $d_k = d_2$ and set $\widetilde{s_k} = \alpha_k(1-\xi)d_2 + \xi d_1$. If $f(x_k + \widetilde{s_k}) \le f(x_k) - \tau\|\widetilde{s_k}\|$ and $\alpha_k\|d_2\| \le T\|d_1\|$, set $s_k = \widetilde{s_k}$; otherwise set $s_k = \alpha_k d_2$.
    b. Use rules (11) and (12) to determine a step length $\alpha_k > 0$ along the direction $d_k = d(\xi)$ and set $s_k = \alpha_k d_k$.
12. Set $x_{k+1} = x_k + s_k$.
```

Parameters b_1, b_2 and b_3 are positive constants so that they offer the user the opportunity to specify the amount contribution of the methods. More precisely, when the slope of the function is slight, the algorithm tends to the Newton's method, otherwise the contribution of gradient is increased and is considered close to the gradient method. In (18), when a difference between two previous values of the function is high then ξ is close to 0, and $\xi \to 1$ as $|f(x_k) - f(x_k)| \to 0$. Moreover, this equation is a dynamic form and has a crucial rule in the algorithm so that it specifies the amount contribution of the methods. It, also, guaranties that, near the solution, we get the optimal point with a super-linear convergence rate.

In step 11 of the above algorithm, we use two different strategies by means of the combination. Step $11.a$ is a new combination and different from the existing methods in the literature. In this combination, the step length α_k is determined only along the gradient direction. In other words, we use a novel combination of the pure Newton's method (i.e., $\alpha_k = 1$) and the gradient method. The second one is the usual combination which has been developed in some research works. The step length in this case is found along a combination of the gradient and the Newton's directions.

5. Global convergence Theorem

Here, we establish the global convergence of the proposed combination algorithm based on the global convergence property of the Theorem 1.

Theorem 2. Consider using the Algorithm 2 to solve the problem (2). Assume that $g(x)$ exists and is uniformly continuous on the level set Ω. Then either $g_k = 0$ for some k, or $f_k \to -\infty$, or $g_k \to 0$.

Proof. Let assume that $g_k \ne 0$ and f_k is bounded below for all k. It is clear that in this case, $f_k < f_{k-1}$ for all k. Denote $\delta^* \leftarrow b_1\delta_0$, clearly $\delta^* \in (0,1)$. We will show that the direction d_k obtained by the algorithm satisfies condition (15) of the Theorem 1 for δ^*; that is,

$$\cos\theta_k \ge \delta^*, \qquad (19)$$

for all k, where θ_k is the angle between d_k and $d_2 = -g_k$.

Suppose s_k is obtained at Step 8. Then $d_k = -g_k$ (Step 8), and it is easy to see that $\cos\theta_k = 1 > \delta^*$; it means (15) holds. Now, we consider other cases: case 1: s_k is obtained via Step 11.b and case 2: s_k is obtained via Step 11.a. We will proof each case separately as follows:

Take any integer k.

1. In this case, we assume that s_k is obtained at Step 11.b. Then $d_k = d(\xi)$ is chosen as a descent direction and according to steps 9-10, the number ξ can be chosen so that the inequality $d(\xi)^T d_2 \ge \delta\|d(\xi)\|\|d_2\|$ holds, where $\delta \leftarrow b_1\delta_0$ or $\delta \leftarrow b_2\delta_0$. Therefore, we have

$$\cos\theta_k \ge \frac{d(\xi)^T d_2}{\|d(\xi)\|\|d_2\|} \ge \delta \ge \delta^*, \qquad (20)$$

that is, (19) holds and therefore the obtained direction, d_k, satisfies in the assumption of the Theorem 1, hence the remainder proof is similar to the proof of the Theorem 1 in [33].

2. In this case, we assume s_k is obtained at Step 11.a, i.e. $s_k = \alpha_k(1-\xi)d_2 + \xi d_1$.

If the number of cases in s_k obtained by Step 11.a is finite, then it means s_k is defined by the gradient direction, d_2, for all sufficiently large k, and therefore the proof will be easily obtained.

Now suppose it is not finite, i.e., there is a subsequence $k_m \to \infty$ such that s_{k_m} is obtained via Step 11.a.

By considering the first condition in 11.a, since f_k is bounded below we have

$$\|s_{k_m}\| \to 0 \text{ as } k_m \to \infty.$$

In addition, from $d_1 d_2 \ge 0$ we obtain

$$\|s_{k_m}\|^2 \ge \alpha_{k_m}^2(1-\xi)^2\|d_2\|^2 + \xi^2\|d_1\|^2.$$

Now, we are going to show $g_k \to 0$. Suppose it is not true. Then there exist a subsequence $\{k_m\}$ such that $\|g_{k_m}\| \ge \bar{\epsilon} > 0, \forall k_m$.

Here we consider two cases: (i) $\|d_1\| \to 0$ as $k_m \to \infty$, and (ii) $\|d_1\|$ does not converge to zero. The case (i) leads to contradiction by applying the second condition in Step 11.a. In the case (ii), let us consider $\|d_{1,k_m}\| \geq \tilde{\epsilon} > 0, \forall k_m$ which is contradiction by $\|s_{k_m}\| \to 0$. Therefore, the proof is complete, i.e., $g_k \to 0$

6. Experiments and Results

We have evaluated the performance of the proposed algorithm for several well known benchmark test problems given in [20, 34].In the proposed algorithm, we use two different combination as described in steps 11.a and 11.b; we refer these cases as 'Ala' and 'Alb', respectively. The group of methods we have compared includes Ala, Alb, the gradient method (GM), the Newton's method (NM), and ShA presented in [8]. In all algorithms we use the Wolfe-Powell line search rules to find an acceptable step length.

The calculations were carried out using MATLAB. The comparison of the methods is based on the following criteria: all methods are terminated if the gradient converges to a predefined tolerance, $\|g(x_k)\| < \epsilon$, $\epsilon = 10^{-6}$, or the iteration number exceeds 500.

The parameters δ_0, Λ_0, η, ρ, σ, γ_i, b_i, τ and T used in this paper are: $\delta_0 = 0.001$, $\Lambda_0 = 1$, $\eta = 0.99$, $\rho = 0.001$, $\sigma = 0.9$, $\gamma_1 = \gamma_2 = n$, $b_1 = 0.01$, $b_2 = \frac{1}{b_1}$, $b_3 = 1.1$, $\tau = 10^{-10}$and $T = 10^{10}$.

We have listed the following ten test problems used in the experiments. To define the test functions, the general formats 1 to 3 have been adopted [20, 34]:

1. Dimension, n .
2. Function definition, $f_1, f_2 \dots f_m$.
3. Standard initial point, x_0.

Problem 1. Helical Valley function

$n = m = 3$

$f_1(x) = 10[x_3 - 10\theta(x_1, x_2)]$

$f_2(x) = 10[(x_1^2+, x_2^2)^{0.5} - 1]$

$f_3(x) = x_3$

where

$$\theta(x_1, x_2) = \begin{cases} \frac{1}{2\pi}\arctan\left(\frac{x_2}{x_1}\right), & if\ x_1 > 0 \\ \frac{1}{2\pi}\arctan\left(\frac{x_2}{x_1}\right) + 0.5, & if\ x_1 < 0 \end{cases}$$

$x_0 = (-1,0,0)$.

Problem 2. Powell Singular function

$n = m = 4$

$f_1(x) = x_1 + 10x_2$

$f_2(x) = 5^{0.5}(x_3 - x_4)$

$f_3(x) = (x_2 - 2x_3)^2$

$f_4(x) = 10^{0.5}(x_1 - x_4)^2$

$x_0 = (3, -1, 0, 1)$.

Problem 3. Wood function

$n = 4, \quad m = 6$

$f_1(x) = 10(x_2 - x_1^2)$

$f_2(x) = 1 - x_1$

$f_3(x) = 90^{0.5}(x_4 - x_3^2)$

$f_4(x) = 1 - x_3$

$f_5(x) = 10^{0.5}(x_2 + x_4 - 2)$

$f_6(x) = 10^{-0.5}(x_2 - x_4)$

$x_0 = (-3, -1, -3, -1)$.

Problem 4. Watson function

$2 \leq n \leq 31, \qquad m = 31$

$$f_i(x) = \sum_{j=2}^{n}(j-1)\, x_j\, t_i^{j-2} - \left(\sum_{j=1}^{n} x_j\, t_i^{j-1}\right)^2 - 1$$

$t_i = \frac{i}{29}, \qquad 1 \leq i \leq 29$

$f_{30}(x) = x_1,$

$f_{31}(x) = x_2 - x_1^2 - 1$

Problem 5. Extended Kearfott function

$n = m = 7$

$f_i(x) = x_i^2 - x_{i+1}$

$f_n(x) = x_n^2 - x_1$

$x_0 = (0.1, 0.1, \dots, 0.1)$.

Problem 6. Extended Eiger-Sikorski-Stenger

$n = m = 9$

$f_i(x) = (x_i - 0.1)^2 + x_{i+1} - 0.1$

$f_n(x) = (x_n - 0.1)^2 + x_1 - 0.1$

$x_0 = (-2000, \dots, -2000)$.

Problem 7. Variably dimensional function

n variable, $\qquad m = n + 2$

$f_i(x) = x_i - 1, i = 1, \dots, n$

$$f_{n+1}(x) = \sum_{j=1}^{n} j(x_j - 1)$$

$f_{n+2}(x) = \left(\sum_{j=1}^{n} j(x_j - 1)\right)^2$

$x_0 = \gamma_j$, where $\gamma_j = 1 - \left(\frac{j}{n}\right)$.

Problem 8. Discrete Boundary Value function

n variable, $\qquad m = n$

$$f_i(x) = 2x_i - x_{i-1} - x_{i+1} + \frac{h^2(x_i + t_i + 1)^3}{2}$$

where $h = \frac{1}{n+1}, t_i = ih, and\ x_0 = x_{n+1} = 0$

$x_0 = \gamma_j$ where $\gamma_j = t_j(t_j - 1)$.

Problem 9. Extended Rosenbrock function

n variable but even, $\qquad m = n$

$f_{2i-1}(x) = 10(x_{2i} - x_{2i-1}^2)$

$f_{2i}(x) = 1 - x_{2i-1}$

$x_0 = \gamma_j$ where $\gamma_{2j-1} = -1.2, \gamma_{2j} = 1$.

Problem 10. Trigonometric function

n variable, $\quad m = n$

$$f_i(x) = n - \sum_{j=1}^{n} cos\, x_j + i(1 - cos x_i) - sin x_i$$

$x_0 = \left(\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}\right)$.

Table 1 lists the performance of the above-mentioned algorithms relative to the number of iterations used. We have multiplied the given initial points by 10 to have an additional initial point. In this table, "TP" and "IP" stand for test problem and initial point, respectively. Table 2 shows the summary of convergence results for the Table 1. In order to compare the algorithms with more initial points, we have generated 50 random initial points uniformly distributed from their domains with the intersection of $[-10,10]$. The summary of the convergence results of the algorithms considering these random initial points is given in Table 3. In these tables, notations "AC" and "NC" stand for the almost convergence and not convergence, respectively. Convergence means that the method finds the solution and $\|g(x_k)\| < 10^{-6}$, almost convergence means that the method finds a solution almost close to the optimal local solution and $10^{-6} < \|g(x_k)\| \le 10^{-2}$; otherwise not convergence.

Table 1. Number of iterations for 10 test problems

TP	n	IP	Ala	Alb	ShA	NM	GM
P1	3	x_0	11	11	13	25	AC
		$10x_0$	13	21	21	NC	NC
P2	4	x_0	7	10	11	11	135
		$10x_0$	9	15	17	19	AC
P3	4	x_0	20	AC	AC	NC	NC
		$10x_0$	19	AC	NC	NC	NC
P4	6	x_0	10	12	12	75	AC
		$10x_0$	11	18	21	77	AC
P5	7	x_0	13	10	9	NC	10
		$10x_0$	8	11	15	NC	13
P6	10	x_0	10	5	7	NC	6
		$10x_0$	17	11	14	NC	8
P7	10	x_0	14	17	23	24	56
		$10x_0$	27	34	37	38	AC
P8	20	x_0	4	5	3	4	AC
		$10x_0$	12	14	7	8	NC
P9	100	x_0	15	17	26	NC	AC
		$10x_0$	57	73	78	NC	NC
P10	100	x_0	19	21	22	AC	129
		$10x_0$	23	24	27	34	472

Te numerical results in Tables 1 to 3, demonstrate the high performance of the proposed combination algorithm compared to other mentioned methods. This is confirmed by the number of iterations obtained, and the convergence properties. For example, the proposed algorithm, Ala, converges in all test problems for two different initial points. Alb converges in nine test problems out of ten. This algorithm finds the solution in the Wood function almost near the optimal solution. Although the algorithm proposed by Shi, ShA, convergences in nine test problems out of ten, but it fails to find the solution in the problem 3. Also, the number of iterations obtained by ShA is more than the proposed algorithms, in average. This is worse for the Newton's and the gradient methods with more AC and NC properties.

Table 2. Summary of convergence results for Table 1

Algorithm	Convergence	AC	NC
Ala	100.00	0.00	0.00
Alb	90.00	10.00	0.00
ShA	90.00	5.00	5.00
NM	50.00	5.00	45.00
GM	40.00	35.00	25.00

Table 3. Convergence results, by considering 50 initial random points for each test problem

Algorithm	Convergence	AC	NC
Ala	96.40	2.80	0.80
Alb	88.80	9.40	1.80
ShA	87.80	4.40	7.80
NM	54.60	2.20	43.20
GM	51.70	30.40	17.90

7. Conclusion

A combined algorithm of the gradient and the Newton's methods has been presented for solving systems of nonlinear equations. We have considered two different combinations. One of them is a usual case which has been recently introduced in some research works. Another one is a new combination and different from others in the literature. According to the numerical experiments, it is clear the proposed algorithm, especially the proposed algorithm with the new combination, is more efficient than others.

References

[1] S. Abbasbandy, Y. Tan, and S.J. Liao, "Newton-homotopy analysis method for nonlinear equations", Appl. Math. Comput. 188, 2007, pp. 1794–1800.

[2] F. Awawdeh, "On new iterative method for solving systems of nonlinear equations", Springer Science, Business Media, LLC 2009.

[3] D.P. Bertsekas, J.N. Tsitsiklis, "Gradient convergence in gradient methods with errors", SIAM J. Optim. 10, 2000, pp. 627–642.

[4] J. Biazar, and B. Ghanbari, "A modification on Newtons method for solving systems of nonlinear equations", World Academy of Science, Engineering and Technology, Vol. 58, 2009, pp. 897-901.

[5] C. G. Broyden, "Quasi-Newton methods and their application to function minimization", Math. Comput. 21, 1967, pp. 368–381.

[6] A. G. Buckley, "Modified Quasi-Newton methods for solving systems of linear Equations", Int. J. Contemp. Math. Sci., Vol. 2, no. 15, 2007, pp. 737-744.

[7] A. G. Buckley, "A combined conjugate-gradient quasi-Newton minimization algorithm", Mathematical Programming, 15, 1978, pp. 200-210.

[8] Y.H. Dai, J. Han, G. Liu, D. Sun, H. Yin, Y.-X. Yuan, "Convergence properties of nonlinear conjugate gradient methods", SIAM J. Optim. 10, 1999, pp. 345–358.

[9] J. E. Dennis, and J. More, "Quasi-Newton methods: motivation and theory", SIAM Rev. 19, 1977, pp. 46–84.

[10] R. Fletcher, 'Practical methods of optimization" , Second Edition, Wiley, New York, 1987.

[11] A. Golbabai, and M. Javidi, "Newton-like iterative methods for solving system of non-linear equations", Appl. Math. Comput. 192, 2007, pp. 546–551.

[12] C. Grosan, and A. Abraham, "A new approach for solving non linear equations systems", IEEE transactions on systems, Vol. 38, No. 3, 2008

[13] H. K. Gummel, "A self-consistent iterative scheme for one-dimensional steady state transistor calculations", IEEE Trans. Electron Devices ED-11, 1964, pp. 455-465.

[14] D. Kaya, and S. M. El-Sayed, "Adomian's decomposition method applied to systems of nonlinear algebraic equations". Appl. Math. Comput. 154, 2004, pp. 487–493.

[15] C. T. Kelley, “Iterative methods for linear and nonlinear equations”, Society for Industrial and Applied Mathematics, Philadelphia, 1995.

[16] T. Kerkhoven, “A proof of convergence of Gummel’s algorithm for realistic boundary condition”, SIAM J. Numer. Anal., 23(6), 1986, pp. 1121-1137.

[17] J. Kou, “A third-order modification of Newton method for systems of nonlinear equations”, Appl. Math. Comput, Vol. 191, 2007, pp. 117-121.

[18] G. H. Liu, L.L. Jing, L.X. Han, D. Han, “A class of nonmonotone conjuagte gradient methods for unconstrained optimization”, J. Optim. 101, 1999, pp. 127–140.

[19] J. M. Moguerza, and F.J. Prieto, “Combining search directions using gradient flows”, Math Program. 96, 2003, pp. 529–559.

[20] J. J. More, B.S. Garbow, and K.E. Hillstrom, “Testing unconstrained optimization software”, ACM Transactions on Mathematical Software, 7, 1981, pp. 17-41.

[21] G. H. Nedzhibov, “A family of multi-point iterative methods for solving systems of nonlinear equations”, Comput. Appl. Math. 222, 2008, pp. 244–250.

[22] P. Y. Nie, “A null space methods for solving system of equations”, Appl. Math. Comput., Vol. 149, No. 1, 2004, pp. 215-226.

[23] J. Nocedal, “Theory of algorithms for unconstrained optimization”, Acta Numer. 1, 1992, pp. 199–242.

[24] J. Nocedal, and J. W. Stephen, “Numerical optimization”, Springer-Verlag, New York, 1999

[25] M. A. Noor, and K. I. Noor, “Iterative schemes for solving nonlinear equations”, Appl. Math. Comput. 183, 2006, pp. 774–779.

[26] M. A. Noor, “New classes of iterative methods for nonlinear equations”, Applied Mathematics and Computation 191, 2007, pp. 128–131

[27] F. A. Potra, and Y. Shi, “Efficient line search algorithm for unconstrained optimization”, J. Optim. 85,1995, pp. 677–704.

[28] M. J. D. Powell, “Direct search algorithms for optimization calculations”, Acta Numer. 7, 1998, pp. 287–336.

[29] Y. Shi, “A globalization procedure for solving nonlinear systems of equations”, Numerical Algorithms, 12, 1996, pp. 273-286

[30] Y. Shi, “Globally convergent algorithms for unconstrained optimization”, Computational Optimization and Applications, 16, 2000, pp. 295-308

[31] Z. J. Shi, “Convergence of line search methods for unconstrained optimization”, Applied Mathematics and Computation 157, 2004, pp. 393–405.

[32] M. I. Spong, I. Norman Katz, H. Dai, and J. Zaborszky, “Block diagonal dominance for systems of nonlinear equations with application to load flow calculations in power systems”, Mathematical Modelling, 5, 1984, pp. 275-297.

[33] W. Sun, and Y. X. Yuan, “Optimization theory and methods: nonlinear programming”, Springer, 2006.

[34] M. N. Vrahatis, “Solving systems of nonlinear equations using the nonzero value of the topological degree”, ACM Transactions on Mathematical Software, 14, 1988, pp. 312-329

[35] M. N. Vrahatis, G.S. Androulakis, J. N. Lambrino, and G.D. Magoulas, “A class of gradient unconstrained minimization algorithms with adaptive stepsize”, Computer and Applied Math. 114, 2000, pp. 367–386.

[36] W. Wang, and Y. F. Xu, “A revised conjugate gradient projection algorithm for inequality constrained optimization”, J. of Computational Mathematics, 23, 2005, pp. 217-224

[37] Y. Xiao, and E. King-Wah Chu, “A nonmonotone inexact Newton algorithm for nonlinear systems of equations”, J. Austral. Math. Soc. Ser. B 36, 1995, pp. 460-492.

[38] F. Zhou, and Y. Xiao, “A class of nonmonotone stabilization Trust Region methods”, Computing 53, Springer-Verlag, 1994, pp.119-136

4.3 Globally Convergent Optimization Methods for Unconstrained Problems

In this section, the algorithm developed in the previous sections is extended to unconstrained optimization problems. The algorithms are based on the combination of different local optimization methods. The first one is the the gradient method due to its global convergence property. The second one is chosen, either the Newton method or the Quasi-Newton method to improve the convergence rate. The performance of proposed algorithms are evaluated by applying them to several well known unconstrained test problems. The numerical experiments demonstrate the efficiency of proposed algorithms, and they have the global and superlinear convergence properties which have been proved.

Paper:

Globally Convergent Optimization Methods for Unconstrained Problems

Published by:

Optimization Journal, 2012

Authors: Sona Taheri[a], Musa Mammadov[a, b] and Sattar Seifollahi[c]

[a]Centre for Informatics and Applied Optimization, SITE,
University of Ballarat, VIC 3353, Australia

[b]National ICT Australia, VRL, VIC 3010, Australia

[c]School of Civil, Environmental and Mining Engineering,
University of Adelaide, SA 5005, Australia

Corresponding author:

Sona Taheri

e-mail: sonataheri@students.ballarat.edu.au

Optimization
Vol. 00, No. 00, March 2012, 1–13

RESEARCH ARTICLE

Globally Convergent Algorithms for Solving Unconstrained Optimization Problems

Sona Taheri[a*] & Musa Mammadov[ab] and Sattar Seifollahi[c]

[a]*Centre for Informatics and Applied Optimization, School of Science, Information Technology and Engineering, University of Ballarat, VIC 3353, Australia*; [b]*National ICT Australia*; [c]*School of Civil, Environmental and Mining Engineering, University of Adelaide, SA 5005, Australia*

(*released March 2012*)

New algorithms for solving unconstrained optimization problems are presented based on the idea of combining two types of descent directions: the direction of anti-gradient and either the Newton or Quasi-Newton directions. The use of latter directions allows one to improve the convergence rate. Global and superlinear convergence properties of these algorithms are established. Numerical experiments using some unconstrained test problems are reported. Also the proposed algorithms are compared with some existing similar methods using results of experiments. This comparison demonstrates the efficiency of the proposed combined methods.

Keywords: Unconstrained optimization; Gradient method; Newton method; Quasi-Newton method; Global convergence; Superlinear convergence

AMS Subject Classifications: 90C53; 49M15; 65K05

1. Introduction

Consider the following unconstrained minimization problem

$$\min \ f(x) \text{ subject to } x \in \mathbb{R}^n \tag{1}$$

where the function f is twice continuously differentiable. Let $g(x) = \nabla f(x)$ and $H(x) = \nabla^2 f(x)$ be the gradient and the Hessian matrix of the function f, respectively.

Numerical methods have been developed extensively for solving the minimization problem (1). The gradient method is one of the simplest and commonly used methods. Although this method is globally convergent, it suffers from the slow convergence rate as a stationary point is approached. In order to improve the convergence rate, one can use the Newton method. This method is one of the most popular methods due to its attractive quadratic convergence, but it depends on the initial point and sometimes the computation of the inverse of the Hessian could be time consuming [18]. A number of different modified Newton methods have been introduced to improve the performance of the Newton method [1, 8–10, 12]. However, the global convergence of these methods are not always guaranteed.

In order to avoid these difficulties, one way is the use of a combination of different local optimization methods. In recent years, there has been a growing interest

*Corresponding author. Email: sonataheri@students.ballarat.edu.au

ISSN: 0233-1934 print/ISSN 1029-4945 online

DOI: 10.1080/0233193YYxxxxxxxx
http://www.informaworld.com

in applying such combined methods. De Luca et al. [5] considered a combination of the gradient and the Newton methods for the solution of nonlinear complementarity problems. The work of Malik Hj et al. [13], employs a hybrid descent direction strategy which uses a convex combination of the anti-gradient and the Quasi-Newton as a search direction. Buckley [2, 3] proposed a strategy of using the Conjugate gradient search direction for the most iterations and periodically using the Quasi-Newton direction to improve the convergence. Wang et al. [19] proposed a revised Conjugate gradient projection method, that is, a combination of the Conjugate gradient projection and the Quasi-Newton methods for nonlinear inequality constrained optimization problems. Shi [16] introduced a method based on the combination of the gradient and Newton methods for solving a system of nonlinear equations. In [4], he proposed a combination of the modified Quasi-Newton and the gradient method to find a solution for systems of linear equations. Furthermore, Shi developed methods based on the combinations of the gradient method with Newton and Quasi-Newton methods for solving unconstrained optimization problems [17]. Recently, the idea of combining the gradient method with the Newton and the Quasi-Newton methods has been developed in [7, 11, 21]. These combinations are also applied for minimizing the cost function during the training of Neural Networks [7]. More recently Yang [20] applied the Newton-Conjugate gradient method for solitary wave computations.

In this paper, we propose new algorithms based on the idea of combining the anti-gradient direction with either the Newton direction or the Quasi-Newton direction for solving the problem (1). We call the algorithm involving combination of the gradient and Newton methods as Algorithm CGN, and the combination of the gradient and Quasi-Newton methods as Algorithm CGQN. These algorithms are different from the existing combination algorithms [2–5, 7, 11, 13, 16, 17, 19–21]. We introduce a special parameter which allows us to control contribution from each component method. We also define two different combinations. The first one is a novel combination in which the step length, α_k, is determined only along the anti-gradient direction. The second one is similar to those developed in [4, 16, 17]. Under some assumptions we prove that the proposed methods are globally convergent and they have superlinear convergence rate.

The rest of the paper is organized as follows. In the next section, we present a general scheme of the descent methods and some theorems which are used to establish the convergence of the proposed methods. In Section 3, we describe the proposed algorithms in details. The global and superlinear convergence properties of our algorithms are proved in Sections 4 and 5, which is followed by some numerical experiments in Section 6, demonstrating the efficiency of the proposed algorithms. Finally, some concluding remarks are made in Section 7.

2. Preliminaries

Consider the problem (1) and denote by $g_k = \nabla f(x_k)$, the gradient of the function f at a point x_k. A general descent method for solving Problem (1) proceeds as follows:

Algorithm 1: A Descent Method

Initialization. Select a starting point $x_0 \in \mathbb{R}^n$, and a tolerance $\varepsilon > 0$, set k:=0.

Step 1. If $\|g_k\| < \varepsilon$, then stop.

Step 2. Compute a descent direction d_k at x_k satisfying

$$g_k^T d_k < 0. \tag{2}$$

Step 3. Determine an appropriate step length $\alpha_k > 0$.

Step 4. Set $x_{k+1} := x_k + \alpha_k d_k, k := k + 1$ and go to Step 1.

Depending on the choice of d_k and α_k, where d_k is a descent direction and α_k is a step length, different descent direction methods have been developed. There are two alternatives for finding α_k, namely using the exact and inexact line search. In practical implementations, the finding an exact optimal step length is, in general, difficult or expensive [18], therefore, the inexact line search with less computational load is highly popular. There are some inexact line search techniques such as Armijo, Goldstein and Wolfe-Powell rules. Given descent direction d_k, the Wolfe-Powell rule suggests the following relations to find the step length $\alpha_k > 0$ [18]

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \rho \alpha_k g_k^T d_k, \tag{3}$$

$$g_{k+1}^T d_k \geq \sigma g_k^T d_k, \tag{4}$$

where $\rho \in (0, 1/2)$ and $\sigma \in (\rho, 1)$.

Let us consider the Wolfe-Powell conditions (3) and (4) to determine α_k in the descent direction algorithm. The global convergence of the general descent direction algorithm is given by the following theorem [18].

Theorem 2.1 : *Let α_k in the descent direction algorithm be defined by (3) and (4). Let also d_k satisfy*

$$\cos(\theta_k) \geq \delta \tag{5}$$

for some $\delta > 0$ and for all k, where θ_k is the angle between d_k and $-g_k$. If $g(x)$ exists and is uniformly continuous on the level set $\{x \in \mathbb{R}^n |\ f(x) \leq f(x_0)\}$, then either $g_k = 0$ for some k, or $f_k \to -\infty$, or $g_k \to 0$.

One of the simplest and the most fundamental minimization methods satisfying Theorem 2.1 is the gradient method, in which $d_k = -g_k$, for all k. Although this method is globally convergent and usually works well in some early steps, as a stationary point is approached, it may descend very slowly.

In order to improve the convergence rate, one can use the Newton method. At the k-th iteration, the classical Newton direction d_k is the solution to the following system:

$$H_k d_k = -g_k, \tag{6}$$

where H_k is the Hessian matrix at x_k. In general, the Newton method is not globally convergent. Moreover, this method requires the computation of the inverse of the Hessian in order to find descent directions which can be time consuming. One technique, for instance, is the Quasi-Newton method which uses approximations with a positive definite matrix. However, these approximations still will not guarantee the global convergence. A common strategy that is recently applied to guarantee the global convergence is the use of methods based on the combination of different local optimization methods [2–5, 7, 11, 13, 16, 17, 19–21]. Most of these methods are efficient for solving the problem (1) due to their global convergence property and high local convergence rate. We propose new combined methods in the next section.

The following theorems will be used to prove the convergence of the proposed methods. Their proofs can be found in [18].

Theorem 2.2: *Let $g : \mathbb{R}^n \to \mathbb{R}^m$ be continuously differentiable in the open convex set $D \subset \mathbb{R}^n$. Assume that H is Lipschitz continuous in D with a Lipschitz constant $\gamma \geq 0$. Then for any $u, v, x \in D$, we have*

$$\|g(u) - g(v) - H(x)(u-v)\| \leq \gamma \frac{\|u-x\| + \|v-x\|}{2} \|u-v\|. \tag{7}$$

Theorem 2.3: *Let g and H satisfy the conditions of Theorem 2.2. Assume that $H^{-1}(x)$ exists. Then there exist $\varepsilon > 0$ and $\mu > \beta > 0$ such that for all $u, v \in D$, when $max\{\|u-x\|, \|v-x\|\} \leq \varepsilon$, we have*

$$\beta\|u-v\| \leq \|g(u) - g(v)\| \leq \mu\|u-v\|. \tag{8}$$

3. The Proposed Algorithms

In this section, we introduce our new algorithms, called CGN and CGQN, for solving the unconstrained optimization problem (1). Algorithm CGN is based on the idea of combining anti-gradient and Newton directions. In Algorithm CGQN, we use the Quasi-Newton direction in the combination with the anti-gradient direction.

Throughout the paper $d_{1,k}$ denotes the anti-gradient direction at x_k and $d_{2,k}$ stands for the second direction at x_k to be used in the combination with $d_{1,k}$. The steps of our combined methods are presented in Algorithm **2**.

Algorithm 2: Algorithms CGN and CGQN

Initialization. Select a starting point $x_0 \in \mathbb{R}^n$, and a tolerance $\varepsilon > 0$, η and δ be small positive numbers and $\vartheta > 1$, ω and L are two fixed numbers. Set k:=0.

Step 1. If $\|g(x_k)\| < \varepsilon$, then stop.
Step 2. Compute the direction $d_{1,k}$ at x_k, $d_{1,k} = -g_k$.

Step 3. Compute a second direction $d_{2,k}$ at x_k . If the direction $d_{2,k}$ at x_k is not computable, then go to Step 5.

Step 4. If $d_{2,k}^T d_{1,k} \geq 0$, go to Step 6.

Step 5. Use rules (3) and (4) to determine a step length $\alpha_k > 0$ along the direction $d_k = d_{1,k}$, set $s_k := \alpha_k d_k$ and go to Step 10.

Step 6. Set $j := 0,\ \eta_0 := \eta$.

Step 7. Compute ξ_k as follows:

$$\xi_k = \begin{cases} \frac{1}{1+\eta_j\|g_0\|} & \text{if } k = 0, \\ \\ \frac{1}{1+\eta_j|f_k - f_{k-1}|} & \text{if } k > 0, \end{cases} \tag{9}$$

and set $d(\xi_k) := (1-\xi_k)d_{1,k} + \xi_k d_{2,k}$.

Step 8. If $d(\xi_k)^T d_{1,k} < \delta \|d(\xi_k)\|\|d_{1,k}\|$, set $\eta_{j+1} := \vartheta \eta_j$ and $j = j+1$, and go to Step 7.

Step 9. Compute s_k using one of the following two approaches:

9.1. Use rules (3) and (4) to determine a step length $\alpha_k > 0$ along the direction $d_{1,k}$ and set $\overline{s}_k := \alpha_k(1-\xi_k)d_{1,k} + \xi_k d_{2,k}$. If $f(x_k + \overline{s}_k) \leq f(x_k) - \omega\|\overline{s}_k\|$ and $\alpha_k\|d_{1,k}\| \leq L\|d_{2,k}\|$, set $s_k := \overline{s}_k$; otherwise set $s_k := \alpha_k d_{1,k}$.

9.2. Use rules (3) and (4) to determine a step length $\alpha_k > 0$ along the direction $d_k = d(\xi_k)$ and set $s_k := \alpha_k d_k$.

Step 10. Set $x_{k+1} := x_k + s_k,\ k := k+1$ and go to Step 1.

The direction $d_{2,k}$ can be either the Newton direction or the Quasi-Newton direction. In Algorithm CGN, when the Hessian at x_k is singular or $d_{2,k}$ is not computable then we use the anti-gradient direction. Moreover, if $d_{2,k}^T d_{1,k} < 0$ then the Newton direction tends to increase the function value. In this case, again, we take the anti-gradient direction as indicated in Step 5.

In Algorithm CGQN, we use the Quasi-Newton direction as the second direction in the combination with the anti-gradient direction. In the Quasi-Newton method, an approximation B_k is used instead of the Hessian H_k. At the k-th iteration, the Quasi-Newton direction is the solution to the following system:

$$B_k d_k = -g_k, \tag{10}$$

where B_k is a positive definite matrix. There are some well known formulas for updating B_k in the Quasi-Newton method [18]. In this paper, B_k is updated by the BFGS formula as follows:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k}, \tag{11}$$

where $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$.

In practical implementations, when $s_k^T y_k = 0$ (or $s_k^T y_k$ is too small), then B_k and consequently $d_{2,k}$ may not be computable and we use only the anti-gradient direction as indicated in Step 5.

In equation (9), parameter ξ_k is in the interval $[0,1]$ that weights two different directions in the combination. When the slope of the function is slight, the algorithm tends to the second direction, $d_{2,k}$, otherwise it is close to the anti-gradient direction, $d_{1,k}$. More precisely, when the difference between function values is a large number, and consequently ξ_k is close to 0, the gradient method may work better. Also, it is clear from (9) that, near the solution, we can get the optimal point with a high convergence rate.

In Step 7, we consider two different conditions for choosing ξ_k. At the first step, $k = 0$, the value of $|f(x_k) - f(x_{k-1})|$ is not defined, so we will use $\|g_0\|$ instead.

In Step 9, we use two different strategies for the combination. Step 9.1 is a new combination and is different from the existing methods in the literature [4, 16, 17]. In this combination, the step length α_k is determined only along the anti-gradient direction. In this strategy, we define two conditions that make the new direction to be a descent direction. In Step 9.2, the step length α_k is determined along the combination of both directions $d_{1,k}$ and $d_{2,k}$.

4. Global Convergence

The following theorem shows that Algorithms CGN and CGQN are globally convergent.

Theorem 4.1 : *Consider using Algorithm* **2** *to solve the problem (1). Assume that $g(x)$ exists and is uniformly continuous on the level set $\{x \in \mathbb{R}^n | \ f(x) \le f(x_0)\}$. Then either $g_k = 0$ for some k, or $f_k \to -\infty$, or $g_k \to 0$.*

Proof: Let us assume that $g_k \neq 0$ and f_k is bounded below for all k. Clearly in this case $f_k < f_{k-1}$ for all k. We need to show that $g_k \to 0$.

Denote by θ_k the angle between d_k and $-g_k$. Consider iterations $x_{k+1} = x_k + s_k$. There are two versions to be considered. In the first version s_k is calculated using Steps 5 and 9.2 of Algorithm **2**, in the second version s_k is calculated using Steps 5 and 9.1.

Version 1. If s_k is obtained at Step 5, then $d_k = d_{1,k} = -g_k$ and consequently

$$\cos(\theta_k) = \frac{-d_k g_k}{\|d_k\| \|g_k\|} = 1 > \delta. \tag{12}$$

Now suppose s_k is obtained at Step 9.2. Then $d_k = d(\xi_k)$ is chosen as a descent direction and according to Steps 7-8, the number ξ_k can be chosen so that the inequality $d(\xi_k)^T d_{1,k} \ge \delta \|d(\xi_k)\| \|d_{1,k}\|$ holds. Then we have

$$\cos(\theta_k) = \frac{d(\xi_k)^T d_{1,k}}{\|d(\xi_k)\| . \|d_{1,k}\|} \ge \delta, \tag{13}$$

Therefore, in this version for all k the inequality $\cos(\theta_k) \ge \delta > 0$ holds and the proof of the theorem follows from Theorem 2.1.

Version 2. Suppose s_k is obtained at Steps 5 and 9.1. In this case we have $x_{k+1} = x_k + s_k$ where s_k is defined by

$$s_k = \alpha_k d_{1,k} \tag{14}$$

or

$$s_k = \alpha_k (1 - \xi_k) d_{1,k} + \xi_k d_{2,k}. \tag{15}$$

Here we note that according to Step 9.1 the step length $\alpha_k > 0$ is determined by the Wolfe-Powell rule along the direction $d_{1,k}$; that is, the following two inequalities are satisfied:

$$f(x_k + \alpha_k d_{1,k}) \le f(x_k) + \rho \alpha_k g_k^T d_{1,k}, \tag{16}$$

$$g(x_k + \alpha_k d_{1,k})^T d_{1,k} \ge \sigma g_k^T d_{1,k}, \tag{17}$$

where $d_{1,k} = -g_k = -g(x_k)$.
If the number of cases when s_k is defined using (15) is finite, that is, s_k is defined by (14) for all sufficiently large k, then the proof follows from Theorem 2.1 in view

of (12).
Assume that there is a subsequent $k_m \to \infty$ such that

$$s_{k_m} = \alpha_{k_m}(1-\xi_{k_m})d_{1,k_m} + \xi_k d_{2,k_m}. \tag{18}$$

According to Step 9.1, this in particular means that the following two relations hold:

$$f(x_{k_m} + s_{k_m}) \le f(x_{k_m}) - \omega \|s_{k_m}\|, \quad \text{for all } k_m \tag{19}$$

and

$$\alpha_{k_m}\|d_{1,k_m}\| < L\|d_{2,k_m}\|, \quad \text{for all } k_m. \tag{20}$$

Since sequence f_k is bounded below, it follows from (19) that it follows

$$\|s_{k_m}\| \to 0 \quad \text{as } k_m \to \infty. \tag{21}$$

Moreover, since $d_{1k}^T d_{2k} \ge 0, \ \forall k$, from (18) we have

$$\|s_{k_m}\|^2 \ge \alpha_{k_m}^2(1-\xi_{k_m})^2\|d_{1,k_m}\|^2 + \xi_{k_m}^2\|d_{2,k_m}\|^2. \tag{22}$$

We need to show that

$$d_{1,k_m} = -g_{k_m} = -g(x_{k_m}) \to 0.$$

Assume the contrary, that is this is not true. For the sake of simplicity, assume that there exists $\widetilde{\varepsilon}$ such that

$$\|g_{k_m}\| \ge \widetilde{\varepsilon} > 0, \ \forall k_m. \tag{23}$$

We will show that this leads to a contradiction by considering two possible cases with respect to the sequence $\|d_{2,k_m}\|$. In the first case we assume that this sequence converges to zero, in the second case it does not.

(i) Let

$$\|d_{2,k_m}\| \to 0 \quad \text{as } k_m \to \infty.$$

In this case from (20) we have

$$\alpha_{k_m}\|d_{1,k_m}\| = \alpha_{k_m}\|g(x_{k_m})\| \to 0 \quad \text{as } k_m \to \infty. \tag{24}$$

Then from uniformly continuity of g we obtain that

$$\|g(x_{k_m} - \alpha_{k_m} g(x_{k_m})) - g(x_{k_m})\| \to 0 \quad \text{as } k_m \to \infty. \tag{25}$$

From (17) it follows

$$g(x_{k_m} + \alpha_{k_m} d_{1,k_m})^T d_{1,k_m} \geq \sigma g(x_{k_m})^T d_{1,k_m}.$$

Letting $d_{1,k_m} = -g(x_{k_m})$ from the last enequality we obtain

$$\left[g(x_{k_m} - \alpha_{k_m} g(x_{k_m})) - g(x_{k_m})\right]^T g(x_{k_m}) \leq (\sigma - 1) g(x_{k_m})^T g(x_{k_m});$$

or

$$\sigma \geq 1 + \frac{\left[g(x_{k_m} - \alpha_{k_m} g(x_{k_m})) - g(x_{k_m})\right]^T g(x_{k_m})}{\|g(x_{k_m})\|^2}.$$

Then from (23) and (25) we have $\sigma \geq 1$ that is a contradiction.

(ii) Now we assume that the sequence $\|d_{2,k_m}\|$ does not converge to zero. For the sake of simplicity assume that $\|d_{2,k_m}\| \geq \mu > 0$ for all k_m. Then from (21) and (22) it follows that $\xi_{k_m} \to 0$ and therefore (24) is satisfied. Then we get a contradiction as in the case of **(i)**.

Therefore (23) leads to a contradiction; that is, $g(x_{k_m}) \to 0$. □

5. Superlinear Convergence

Theorem 4.1 in the previous section establishes the convergence of gradients $g(x_{k_m})$ that is the stoping criterion for Algorithm **2**. In this section, we assume that the sequence of points $\{x_k\}$ generated by the algorithm also converges to some point x^*. In this case we aim to investigate the convergence rate of Algorithm **2**.

Denote by $D \subset \mathbb{R}^n$ some convex neighborhood of x^* that contains all elements x_k for sufficiently large k. Since we are interested in the convergence rate of $\{x_k\}$ to x^*, we assume that $x_k \in D$ for all $k = 0, 1, 2, \ldots$.

We recall that the function f is assumed to be twice continuously differentiable. In addition we will use the following assumptions.

(AS1) $x^* \in D$ is a strong local minimizer of the function f, (for definition see [18]), with $H(x^*)$ symmetric and positive definite.
(AS2) There is a constant $\gamma \geq 0$ such that

$$\|H(\overline{x}) - H(x)\| \leq \gamma \|\overline{x} - x\|, \forall x, \overline{x} \in D.$$

In the following theorem, we show that under some assumptions Algorithm CGQN, with Step 9.1, is superlinearly convergent; that is,

$$\lim_{k \to \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0.$$

Theorem 5.1 : *Let the function f be twice continuously differentiable and Assumptions AS1-AS2 be satisfied. Consider a sequence $x_k \to x^*$, $x_k \in D$, $x_{k+1} = x_k + s_k$, that is generated by Algorithm CGQN with a sequence of symmetric bounded and positive definite matrices B_k. Moreover, suppose there is k_0 such that for all $k \geq k_0$: $\cos(\theta_k^0) \geq \delta > 0$, where θ_k^0 is the angle between $d_{1,k}$ and*

$d_{2,k}$; and the iterations s_k in Step 9.1 utilize the formula

$$s_k = \alpha_k(1-\xi_k)d_{1,k} + \xi_k d_{2,k}.$$

Then $\{x_k\}$ converges superlinearly to x^* if and only if

$$\lim_{k\to\infty} \frac{\|[B_k - H(x^*)](x_{k+1}-x_k)\|}{\|x_{k+1}-x_k\|} = 0. \tag{26}$$

Proof: The proof of the theorem is based on the following equivalence:

$$\lim_{k\to\infty} \frac{\|[B_k - H(x^*)](x_{k+1}-x_k)\|}{\|x_{k+1}-x_k\|} = 0 \Longleftrightarrow \lim_{k\to\infty} \frac{\|g_{k+1}\|}{\|x_{k+1}-x_k\|} = 0$$

$$\Longleftrightarrow \lim_{k\to\infty} \frac{\|x_{k+1}-x^*\|}{\|x_k - x^*\|} = 0. \tag{27}$$

1. First we prove that $\{x_k\}$ converges to x^* superlinearly if the relation (26) holds. If $g(x_k) = 0$ for some k then the theorem is true. Assume that $\|g(x_k)\| > 0$ and $g(x_k) \to 0$ as $k \to \infty$. Denote $d(\xi_k) = (1-\xi_k)d_{1,k} + \xi_k d_{2,k}$, with $d_{1,k} = -g_k$, and let θ_k be the angle between $d(\xi_k)$ and $d_{1,k}$.

Since $\xi_k < 1$ and $d_{1,k}^T d_{2,k} \geq 0$ for all $k \geq k_0$, it is clear that $\theta_k < \theta_k^0$ and therefore by the assumption of the theorem $\cos(\theta_k) > \cos(\theta_k^0) \geq \delta > 0$.

This in particular means that the required inequality in Step 8 is achieved at the first round for η_0; that is, ξ_k has the form

$$\xi_k = \frac{1}{1+\eta_0|f_k - f_{k-1}|}, \quad \forall k \geq k_0.$$

Since $|f_k - f_{k-1}| \to 0$ we obtain that

$$\xi_k \to 1. \tag{28}$$

By assumption, for all $k \geq k_0$, the increments s_k are obtained in Step 9.1 of Algorithm CGQN; that is,

$$x_{k+1} = x_k - \alpha_k(1-\xi_k)g_k - \xi_k B_k^{-1} g_k. \tag{29}$$

From (29), we have

$$[B_k - H(x^*)](x_{k+1}-x_k) =$$

$$-\alpha_k(1-\xi_k)B_k g_k - \xi_k g_k - H(x^*)(x_{k+1}-x_k) =$$

$$g_{k+1} - g_k - H(x^*)(x_{k+1}-x_k) - g_{k+1} + (1-\xi_k)(g_k - \alpha_k B_k g_k). \tag{30}$$

Taking here the norm and dividing by $\|x_{k+1} - x_k\|$, we obtain

$$\frac{\|g_{k+1}\|}{\|x_{k+1} - x_k\|} \le \frac{\|[B_k - H(x^*)](x_{k+1} - x_k)\|}{\|x_{k+1} - x_k\|}$$

$$+\frac{\|g_{k+1} - g_k - H(x^*)(x_{k+1} - x_k)\|}{\|x_{k+1} - x_k\|} + \frac{\|(1-\xi_k)(g_k - \alpha_k B_k g_k)\|}{\|x_{k+1} - x_k\|}. \tag{31}$$

We note that all the assumptions of Theorems 2.2 and 2.3 are satisfied on the open convex set D. By applying Theorem 2.2 we have

$$\frac{\|g_{k+1} - g_k - H(x^*)(x_{k+1} - x_k)\|}{\|x_{k+1} - x_k\|} \le \frac{\gamma}{2}(\|x_{k+1} - x^*\| + \|x_k - x^*\|) \to 0. \tag{32}$$

Now consider the third term on right hand side of (31). Denoting by I the unit matrix, we have

$$\frac{\|(1-\xi_k)(g_k - \alpha_k B_k g_k)\|}{\|x_{k+1} - x_k\|} = \frac{\|g_k - \alpha_k B_k g_k\|}{\|\alpha_k g_k + \frac{\xi_k}{1-\xi_k} B_k^{-1} g_k\|} \le$$

$$\frac{\|I - \alpha_k B_k\|}{\|\alpha_k \frac{g_k}{\|g_k\|} + \frac{\xi_k}{1-\xi_k} B_k^{-1} \frac{g_k}{\|g_k\|}\|}.$$

Since $\|B_k\|$ is assumed to be bounded for all $k \ge k_0$ the relation $\xi_k \to 1$ from (28) yields

$$\left\| \frac{\xi_k}{1-\xi_k} B_k^{-1} \frac{g_k}{\|g_k\|} \right\| \to \infty \quad \text{as } k \to \infty.$$

Indeed, if this is not true, then $\|B_{k_m}^{-1} \frac{g_{k_m}}{\|g_{k_m}\|}\| \to 0$ for some $k_m \to \infty$ that contradicts

$$\left\| B_{k_m} \left(B_{k_m}^{-1} \frac{g_{k_m}}{\|g_{k_m}\|} \right) \right\| = 1, \quad \forall k_m.$$

Thus

$$\frac{\|(1-\xi_k)(g_k - \alpha_k B_k g_k)\|}{\|x_{k+1} - x_k\|} \to 0 \quad \text{as } k \to \infty. \tag{33}$$

Therefore, it follows from (26), (32) and (33) that

$$\lim_{k\to\infty} \frac{\|g_{k+1}\|}{\|x_{k+1} - x_k\|} = 0. \tag{34}$$

The remaining part of the proof is similar to the proof of Theorem 5.4.3 from [18] which yeilds $\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0$. This means that the sequence $\{x_k\}$ is convergent to x^* superlinearly.

2. Now suppose that x_k converges to x^* superlinearly. Clearly $g(x^*) = 0$. From the proof of Theorem 5.4.3 [18] it follows that the relation (34) is true. Therefore, from (30) we obtain

$$\frac{\|[B_k - H(x^*)](x_{k+1} - x_k)\|}{\|x_{k+1} - x_k\|} \leq \frac{\|g_{k+1} - g_k - H(x^*)(x_{k+1} - x_k)\|}{\|x_{k+1} - x_k\|}$$

$$+\frac{\|g_{k+1}\|}{\|x_{k+1} - x_k\|} + \frac{\|(1-\xi_k)(g_k - \alpha_k B_k g_k)\|}{\|x_{k+1} - x_k\|}. \tag{35}$$

which gives (26) by using (32), (33) and (34). □

Similar to Theorem 5.1, for Algorithm CGN with Step 9.1 we have the following theorem.

Theorem 5.2 : *Let the function f be twice continuously differentiable and Assumptions AS1-AS2 be satisfied. Consider a sequence $x_k \to x^*$, $x_k \in D$, $x_{k+1} = x_k + s_k$, that is generated by Algorithm CGN. Moreover, suppose there is k_0 such that for all $k \geq k_0$: $\cos(\theta_k^0) \geq \delta > 0$, where θ_k^0 is the angle between $d_{1,k}$ and $d_{2,k}$; and the iterations s_k in Step 9.1 utilize the formula*

$$s_k = \alpha_k(1-\xi_k)d_{1,k} + \xi_k d_{2,k}.$$

Then $\{x_k\}$ converges to x^* at a superlinear rate.

Proof : From assumption AS2, there is a constant $\gamma \geq 0$ such that

$$\frac{\|[H_k - H(x^*)](x_{k+1} - x_k)\|}{\|x_{k+1} - x_k\|} \leq \frac{\|H_k - H(x^*)\|\|x_{k+1} - x_k\|}{\|x_{k+1} - x_k\|}$$

$$= \|H_k - H(x^*)\| \leq \gamma\|x_k - x^*\|. \tag{36}$$

Since $\{x_k\}$ converges to x^*, we have

$$\lim_{k\to\infty} \frac{\|[H_k - H(x^*)](x_{k+1} - x_k)\|}{\|x_{k+1} - x_k\|} = 0. \tag{37}$$

The remaining of the proof follows from the proof of Theorem 5.1 with considering H_k instead of B_k. □

6. Numerical Experiments

In this section, the performance of the proposed algorithms are evaluated by applying them to some unconstrained test problems taken from [15]. Out of 18 unconstrained minimization problems we use 15 problems in the numerical implementations excluding the 3 problem that are global optimization problems. Table 1 gives a brief description about each test problem, where n is a given integer number by a user. More details can be found in [15].

The group of methods we compare includes our algorithms, algorithms presented by Shi [17], the Newton method (NM), the Quasi-Newton method (QNM) and the gradient method (GM). In the CGN and CGQN algorithms we use two different versions, as described in Steps 9.1 and 9.2. We refer algorithms using Step 9.1 as CGN_1 and $CGQN_1$, and algorithms using Step 9-2 as CGN_2 and $CGQN_2$. From [17], we apply Algorithms 2 and 4, and we refer them as Shi_1 and Shi_2 that are the Newton and Quasi-Newton based methods, respectively.
The termination criteria are the same for all algorithms. Algorithms terminate when either $\|g(x)\| \leq \varepsilon$ or the number of iterations exceeds 500. Parameters in Algorithms CGN and CGQN are chosen as follows: $\varepsilon = 10^{-6}$, $\eta = 10^{-3}$, $\delta = 10^{-3}$, $\vartheta = 1.1$, $\omega = 10^{-10}$, $L = 10^{10}$. Also we select $\rho = 10^{-3}$, $\sigma = 0.9$ for the Wolfe-Powell rule.

The number of iterations (to find the local optimal solutions) used by the algorithms for given initial points are reported in Table 2. In this table, TP stands for test problems, Dim for dimention and IP for initial points. The initial points are taken from [15, 17]. "AC" and "NC" stand for the "almost convergent" and "not convergent", respectively. Convergence means that the method finds the solution x_k where $\|g_k\| < 10^{-6}$, almost convergence means that the method finds a solution x_k where $10^{-6} \leq \|g_k\| \leq 10^{-2}$; otherwise we accept that a method fails to find a solution, that is, it is not convergent.

Based on Table 2, the number of iterations used by Algorithm CGN is, overall, less than those used by other Newton based methods. Especially CGN_1 (using Step 9.1) has the lowest iteration numbers in comparison with other Newton based methods. In the Quasi-Newton based methods, our algorithm ($CGQN$) has found the local optimal solutions using the lowest number of iterations. The gradient method is almost convergent or not convergent in most of the cases.

In order to compare the algorithms with more initial points, we generate 50 random initial points uniformly distributed in $[-10, 10]^n \subset \mathbb{R}^n$ for each test problem 1. Table 3 presents the summary of convergence results in percentage for all test problems. Results from this table demonstrate that Algorithm CGN has the highest convergence rate among Newton based methods. Similarly, Algorithm $CGQN$ has the highest convergence rate among Quasi-Newton based methods.

New algorithms for solving unconstrained optimization problems are presented based on the idea of combining two types of descent directions: the direction of anti-gradient and either the Newton or Quasi-Newton directions. The use of latter directions allows one to improve the convergence rate. Global and superlinear convergence properties of these algorithms are established. Numerical experiments using some unconstrained test problems are reported. Also the proposed algorithms are compared with some existing similar methods using results of experiments. This comparison demonstrates the efficiency of the proposed combined methods.

7. Conclusion

In this paper, we have developed new algorithms which combine the anti-gradient direction with either the Newton direction or the Quasi-Newton direction. We have proved that under some conditions, the first version is both globally and superlinear convergent, while the second version is only globally convergent.

We have carried out a number of experiments using fifteen unconstrained test problems. The numerical results clearly demonstrate the efficiency of proposed algorithms.

References

[1] J. Biazar, and B. Ghanbari, *A modification on Newtons method for solving systems of nonlinear equations*, World Academy of Science, Engineering and Technology, Vol. 58(4), (2009), pp. 897-901.

[2] A. G. Buckley, *A combined Conjugate-Gradient Quasi-Newton minimization Algorithm*, Mathematical Programming 15, (1978), pp. 200-210.

[3] A. G. Buckley, *Extended the relationship between the Conjugate Gradient and BFGS Algorithms*, Mathematical Programming 15, (1978), pp. 343-348.

[4] A. G. Buckley, *Modified Quasi-Newton methods for solving systems of linear equations*, International Journal of Contemp Mathematics and Science, Vol. 2, no. 15, (2007), pp. 737-744.

[5] T. De Luca, F. Facchinei, C. Kanzow, *A Semismooth equation approach to the solution of nonlinear complementarity problems*, Mathematical Programming 75, (1996), pp. 407-439.

[6] R. Fletcher, *Practical methods of optimization*, second edition, Wiley, New York, 1987.

[7] X. He, and S. Xu, *Process Neural Networks*, Guocheny Shejing Yuan Wangluo, Science Press, 2007

[8] H.H.H. Homeier, *A modified Newton method with cubic convergence: the multivariate case*, Journal of Computer and Applied Mathematics 169, (2004), pp. 161-169.

[9] A.E. Kostopoulos, G.S. Androulakis, and T.N. Grapsa, *A new nonmonotone Newton's modification for unconstrained optimization*, Tech. Rep. 09-05, Department of Mathematics, University of Patras, Patras, Greece, 2009.

[10] J. Kou, *A third-order modification of Newton method for systems of nonlinear equations*, Appl. Math. Comput, Vol. 191, (2007), pp. 117-121.

[11] D. Luenberger, and Y. Ye, *Linear and Non Linear Programming*, Springer Science+ Business Media, LLc, 2008.

[12] E.N. Malihoutsaki, I.A. Nikas, and T.N. Grapsa, *Improved Newton's method without direct function evaluations*, J. Comput. Appl.Math. 227 (2009), pp. 206-212.

[13] A. Malik Hj, M. Mansor, and W. Leong, *A switching criterion in hybrid Quasi-Newton BFGS-steepest descent direction.*, Pertanika J. Sci and Technol 7(2), (1999), pp. 111-123.

[14] M. Mammadov, and S. Taheri, *A Globally convergent optimization Algorithm for systems of nonlinear equations*, in *In Proceedings of the third global conference on Power Control and Optimization, Gold Coast, Australia*, 2010.

[15] J.J. More, B.S. Garbow, K.E. Hillstrom, *Testing unconstrained optimization software*, ACM Transactions on Mathematical Software 7(1), (1981), pp. 17-41.

[16] Y. Shi, *A globalization procedure for solving nonlinear systems of equations*, Numerical Algorithms 12, (1996), pp. 273-286.

[17] Y. Shi, *Globally convergent Algorithms for unconstrained optimization*, Computational Optimizatin and Applications 16, (2000), pp. 295-308.

[18] W. Sun, Y.X. Yuan, *Optimization theory and methods*, nonlinear programming, Springer, 2006.

[19] W. Wang, and Y.F. Xu, *A revised Conjugate Gradient projection Algorithm for inequality constrained optimization*, Journal of Computational Mathematics 23, (2005), pp. 217-224.

[20] J. Yang, *Newton-Conjugate-Gradient methods for solitary wave computations*, Journal of Computational Physics 228, (2009), pp. 7007-7024.

[21] N. Ye, *The hand book of Data Mining*, Lawrence Erlbaum Associates, Inc, 2003.

Table 1. Some test problems taken from [15], n is a user given integer

Problem	Function name	Dimension
P1	Freudenstein and Roth function	2
P2	Box three demential function	3
P3	Gaussian function	3
P4	Gulf research and development function	3
P5	Helical valley function	3
P6	Brown and Dennis function	4
P7	Wood function	4
P8	Biggs EXP6 function	6
P9	Watson function	n
P10	Extended Powell singular function	n
P11	Penalty function1	n
P12	Penalty function2	n
P13	Trigonometric function	n
P14	Variably dimensioned function	n
P15	Extended Rosenbrock function	n

Table 2. Number of iterations for 15 test problems obtained by combination of gradient and Newton methods (CGN_1 and CGN_2), combination of gradient and Quasi Newton methods ($CGQN_1$ and $CGQN_2$), Shi's methods [17] (Shi_1, Shi_2), Newton method (NM), Quasi-Newton method (QNM) and gradient method (GM)

TP	Dim	IP	Newton based methods				Quasi-Newton based methods				
			CGN_1	CGN_2	Shi_1	NM	$CGQN_1$	$CGQN_2$	Shi_2	QNM	GM
P1	2	$(0.5, -2)$	6	8	8	14	9	9	24	17	AC
P1	2	$(5, -20)$	7	10	13	24	17	17	26	27	253
P2	3	$(0, 10, 20)$	7	7	9	10	23	23	25	41	AC
P2	3	$(0, 100, 200)$	12	15	36	NC	14	20	29	31	37
P3	3	$(0.4, 1, 0)$	2	3	4	5	2	2	6	6	10
P3	3	$(4, 10, 0)$	9	15	12	NC	18	20	22	27	180
P4	3	$(5, 2.5, 0.15)$	4	8	11	11	19	34	48	48	66
P4	3	$(-5, -2.5, -0.15)$	3	5	14	14	13	13	13	13	18
P5	3	$(-1, 0, 0)$	10	10	13	24	17	17	28	22	AC
P5	3	$(-10, 0, 0)$	14	20	19	NC	20	20	37	35	NC
P6	4	$(25, 5, -5, -1)$	10	12	13	19	39	44	44	44	320
P6	4	$(500, 100, -100, -20)$	16	16	18	18	64	74	67	69	430
P7	4	$(-3, -1, -3, -1)$	21	AC	AC	NC	31	37	81	62	NC
P7	4	$(3, 1, 3, 1)$	9	10	11	11	18	27	64	57	AC
P8	6	$(1, 2, 1, 1, 1, 1)$	20	23	24	NC	64	64	102	87	AC
P8	6	$(10, 20, 10, 10, 10, 10)$	27	30	66	NC	24	26	33	32	AC
P9	6	$(0, 0, 0, 0, 0, 0)$	6	11	14	14	28	34	36	38	AC
P9	6	$(0.1, 0.1, 0.1, 0.1, 0.1, 0.1)$	8	9	9	NC	29	34	37	35	AC
P9	30	$(0, 0, ..., 0)$	7	10	10	14	45	54	59	58	AC
P9	30	$(0.1, 0.1, ..., 0.1)$	55	57	66	72	73	94	95	99	AC
P10	4	$(3, -1, 0, 1)$	12	12	13	18	20	23	23	25	AC
P10	4	$(30, -10, 0, 10)$	17	19	29	18	24	27	36	35	AC
P10	40	$(3, -1, 0, 1, ..., 3, -1, 0, 1)$	13	14	20	24	68	73	77	93	AC
P10	40	$(30, -10, 0, 10, ..., 30, -10, 0, 10)$	25	28	33	35	53	55	55	59	AC
P11	10	$(1, 2, ..., 9, 10)$	5	7	12	10	5	8	14	15	17
P11	10	$(10, 20, ..., 90, 100)$	7	9	15	13	13	13	16	18	24
P11	100	$(1, 2, ..., 90, 100)$	8	11	26	NC	15	18	19	22	NC
P11	100	$(10, 20, ..., 900, 1000)$	14	14	15	NC	21	23	23	28	NC
P12	10	$(0.5, 0.5, ..., 0.5)$	6	6	6	8	10	11	14	14	71
P12	10	$(5, 5, ..., 5)$	9	12	17	14	22	22	32	27	48
P12	100	$(0.5, 0.5, ..., 0.5)$	22	27	AC	NC	73	89	94	174	AC
P12	100	$(5, 5, ..., 5)$	29	30	AC	NC	80	91	102	127	NC
P13	10	$(0.1, 0.1, ..., 0.1, 0.1)$	10	15	14	AC	18	19	20	21	104
P13	10	$(1, 1, ..., 1, 1)$	11	11	14	14	29	39	46	57	123
P13	100	$(0.01, 0.01, ..., 0.01, 0.01)$	16	17	20	AC	39	56	54	135	137
P13	100	$(0.1, 0.1, ..., 0.1, 0.1)$	24	23	29	30	440	452	468	480	489
P14	10	$(0.9, 0.8, ..., 0.1, 0)$	10	11	14	21	16	20	21	22	27
P14	10	$(9, 8, ..., 1, 0)$	9	12	12	26	11	12	22	31	AC
P14	100	$(0.99, 0.98, ..., 0.01, 0)$	15	19	24	26	24	24	29	38	57
P14	100	$(9.9, 9.8, ..., 0.1, 0)$	29	32	32	36	14	17	20	31	AC
P15	10	$(-1.2, 1, ..., -1.2, 1)$	10	18	25	NC	81	87	104	95	AC
P15	10	$(-12, 10, ..., -12, 10)$	41	52	71	NC	97	108	115	114	NC
P15	100	$(-1.2, 1, ..., -1.2, 1)$	16	18	24	NC	327	355	383	411	AC
P15	100	$(-12, 10, ..., -12, 10)$	51	58	62	NC	217	238	241	295	NC

Table 3. Summary of average convergence results over 15 test problems given in Table 1 with 50 random initial points

	Algorithm	Convergence	Almost convergence	Non convergence
Newton based methods	CGN_1	98.27	1.36	0.37
	CGN_2	95.72	2.82	1.46
	Shi_1	94.91	3.55	1.54
	NM	56.18	4.45	39.37
Quasi Newton based methods	$CGQN_1$	96.72	1.91	1.37
	$CGQN_2$	95.54	2.36	2.10
	Shi_2	94.82	2.18	3.00
	QNM	93.12	2.09	4.79
	GM	53.45	26.27	20.28

Chapter 5

Optimization in Bayesian Networks

5.1 Learning Naive Bayes Classifier with Optimization Models

In this section, three optimization models for Naive Bayes classifier are proposed. These models are constructed by introducing different objective functions, where class probabilities and conditional probabilities are considered as unknown variables. Optimal values of these variables can be found by applying optimization techniques. The performance of proposed models are evaluated on some real world binary classification problems. The results obtained demonstrate that the proposed models achieve high accuracy than the Naive Bayes classifier.

Paper:

Learning Naive Bayes Classifier with Optimization Models

Accepted in:

International Journal of Applied Mathematics and Computer Science, 2012

Authors: Sona Taheri[a] and Musa Mammadov[a, b]

[a]Centre for Informatics and Applied Optimization,

School of Science, Information Technology and Engineering,

University of Ballarat, VIC 3353, Australia

[b]National ICT Australia, VRL, VIC 3010, Australia

Corresponding author:

Sona Taheri

e-mail: sonataheri@students.ballarat.edu.au

Int. J. Appl. Math. Comput. Sci., , Vol. , No. , –
DOI:

LEARNING NAIVE BAYES CLASSIFIER WITH OPTIMIZATION MODELS

SONA TAHERI [*], MUSA MAMMADOV [*,**]

[*] Centre for Informatics and Applied Optimization,
School of Science, Information Technology and Engineering,
University of Ballarat, Victoria 3353, Australia
e-mail: `sonataheri@students.ballarat.edu.au`

[**] National ICT Australia, VRL, VIC 3010, Australia

Naive Bayes is among simplest probabilistic classifiers. It has shown to be very efficient on many real world applications. The Naive Bayes classifier often performs surprisingly well in practice, even though the strong assumption that all features are conditionally independent given the class. In the learning process of this classifier with the known structure, class probabilities and conditional probabilities are calculated using training data, and then values of these probabilities are used to classify new observations. In this paper, we introduce three novel optimization models for the Naive Bayes classifier where both class probabilities and conditional probabilities are considered as variables. Values of these variables are found by solving the corresponding optimization problems. Numerical experiments are conducted on several real world binary classification data sets, where continuous features are discretized by applying three different methods. The performances of these models are compared with the Naive Bayes classifier, Tree Augmented Naive Bayes, SVM and C4.5. The results obtained demonstrate that the proposed models can significantly improve the performance of the Naive Bayes classifier, yet at the same time maintains its simple structure.

Keywords: Bayesian Networks, Naive Bayes Classifier, Optimization, Discretization

1. Introduction

Bayesian Networks (BNs) introduced by Pearl (Pearl, 1988) are high level representations of probability distributions over a set of variables $\mathbf{X} = \{X_1, X_2, ..., X_n\}$ that are used for learning process. The learning of BNs are divided in two steps: structure learning and parameter learning. The structure learning is constructing a directed acyclic graph from the set **X**. In the graph, each node corresponds to the variable and each arc denotes the causal relationship between two variables, the direction of the arc indicates the direction of the causality. When two nodes are joined by an arc, the causal node is called the parent of the other node, and another one is called the child. We use X_i to denote both the variable (feature) and its corresponding node, and $Pa(X_i)$ to denote the set of parents of the node X_i. Given a structure, finding local probability distributions, class probabilities and conditional probabilities, associated with each variable is called parameter learning (Campos *et al.*, 2002).

In particular, the joint probability distribution for **X** is given by

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|Pa(X_i)), \qquad (1)$$

however, accurate estimation of $P(X_i|Pa(X_i))$ needs to find the structure which is non trivial. It has been proved that learning an optimal structure of a BN is an NP-hard problem (Chickering, 1996; Heckerman *et al.*, 2004). In order to avoid the intractable complexity of the structure learning in BNs, the Naive Bayes classifier (Langley *et al.*, 1992) with the known structure has been used. In the Naive Bayes (NB), features are conditionally independent given the class. It means that each feature has the class as an only parent. The efficiency of the NB has witnessed its widespread development in real world applications including medical diagnosis, recommender systems, email filtering, web page perfecting and fraud detection (Crawford *et al.*, 2002; Kononenko, 2001; Miyahara and Pazzani, 2000; Zupan *et al.*, 2001).

In this paper, our aim is to improve the performance of the NB by applying optimization techniques, yet at the same time maintains its simple structure. We consider class probabilities and conditional probabilities as unknown variables, whose optimal values can be computed applying optimization techniques. We introduce three different optimization models for the NB using different definitions of unknown variables.

Most of data sets in real world applications involve continuous features. The most well known attempt for improving the performance of the NB with continuous features is the discretization of the features into intervals, instead of using the default option to utilize the normal distribution to calculate probabilities. The main reason is that the NB with discretization tends to achieve lower classification error than the original one (Dougherty *et al.*, 1995). It has been shown that the performance of the NB classifier significantly improves when features are discretized using an entropy based method (Dougherty *et al.*, 1995). In this paper, therefore, we use the Fayyad and Irani's method (Fayyad and Irani, 1993), a method based on a minimal entropy heuristic to discretize continuous features. We also apply two other different discretization methods. The first one, which is also the simplest one, transforms the values of features to $\{0, 1\}$ using their mean values. The second one is the discretization algorithm recently introduced by Yatsko et al. Yatsko *et al.* (2011), using sub-optimal agglomerative clustering algorithm (SOAC).

The structure of the paper is as follows. In the next section, we provide a brief description of the NB classifier. In Section 3 with the preliminaries, we briefly describe the globally convergent optimization method, a combination of the Gradient and the Newton's methods (CGN), and discretization algorithms, respectively, which we require for our discussion in the latter part of the paper. In Section 4, we introduce three different optimization models for the NB classifier. The results of numerical experiments are given in Section 5. Section 6 concludes the paper.

2. Naive Bayes Classifier

The Naive Bayes classifier (Domingos and Pazzani, 1997; Langley *et al.*, 1992) assumes that each feature only depends on the class as depicted in Figure 1. It means that each feature has only the class as a parent. The NB is attractive as it has an explicit and sound theoretical basis which guarantees optimal induction given a set of explicit assumptions. There is a drawback in which the independency assumptions of features with respect to the class is violated in some real world problems. However, it has been shown that the NB is remarkably robust in the face of such violations (Domingos and Pazzani, 1996; Friedman *et al.*, 1997). Domingos has found that this limitation has less impact than might be expected Domingos and Pazzani (1997). The NB is fast, easy to implement with the simple structure, and an effective classifier. It is also useful for high dimensional data as probability of each feature is estimated independently. The NB is one of the 10 top algorithms in data mining as listed in (X. *et al.*, 2008).

Let C denotes the class of an observation $\mathbf{X}$. To predict the class of the observation $\mathbf{X}$ by using the Bayes rule, the highest posterior probability of

$$P(C|\mathbf{X}) = \frac{P(C)P(\mathbf{X}|C)}{P(\mathbf{X})}, \tag{2}$$

should be found.

In the NB classifier, using the assumption that features $X_1, X_2, ..., X_n$ are conditionally independent of each other given the class, we get

$$P(C|\mathbf{X}) = \frac{P(C)\prod_{i=1}^{n} P(X_i|C)}{P(\mathbf{X})}. \tag{3}$$

In classification problems, Equation (3) is sufficient to predict the most probable class given a test observation.

To estimate class probabilities $P(C)$ and conditional probabilities $P(X_i|C),\ i = 1, ..., n,$ in the formula (3), we introduce three different optimization models in this paper.

3. Preliminaries

In this section, we briefly review the optimization method, CGN, and discretization algorithms, the Fayyad and Irani's method and the Algorithm SOAC which we use in the latter part of the paper.

3.1. Combination of the Gradient and the Newton's methods (CGN).

In this section, we present the recently introduced optimization algorithm, a combination of the Gradient and the Newton's methods (CGN) (Taheri and Mammadov, 2012). The CGN is a new globally convergent optimization algorithm for solving systems of nonlinear equations and unconstrained optimization problems. The idea in this algorithm is combining two directions from different local optimization methods. The first direction is the gradient direction due to its global convergence property. The second one is the Newton's direction to speed up the convergence rate. Two different combinations are considered in this algorithm. The first one is a novel combination in which the step length is determined only along the gradient direction. In the second one, the step length is considered along both directions.

Let us consider the following unconstrained minimization problem

$$\min \quad f(x) \tag{4}$$

where $x \in \mathbb{R}^n$. Let $g(x) = \nabla f(x)$ and $H(x) = \nabla^2 f(x)$ be correspondingly the gradient and the Hessian matrix of the function f. Let $\delta,\ \eta,\ \rho$, and σ be four parameters so that $0 < \delta < 1$, $0 < \eta < 1$, $0 < \rho < 1/2$ and $\rho < \sigma < 1$.

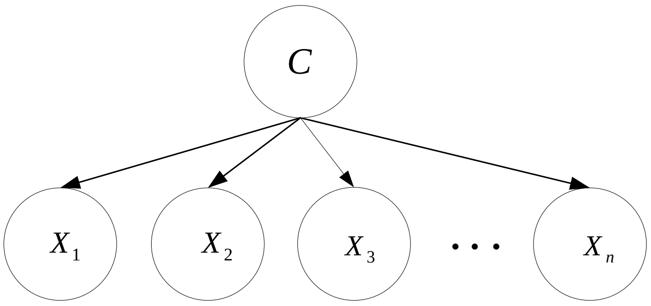


Fig. 1. Naive Bayes

Take any positive constants γ_1, γ_2 and $b_i, i = 1, 2, 3$ such that $\gamma_1,\ \gamma_2 > 1$, $0 < b_1 < 1$, $1 < b_2 < 1/\delta$ and $b_3 > 1$ and initialize Λ by Λ_0. Let also T, and τ be very large and small positive numbers, respectively, and let $f(x_0) = f(x_1)$. The steps of the Algorithm CGN are as follows.

Algorithm. Optimization Algorithm CGN

Choose an initial point $x_1 \in R^n$, and an error tolerance $\epsilon > 0$.
For $k = 1, 2, ...$ do:
Step 1. If $\|g(x_k)\| < \epsilon$, then stop.
Step 2. If the Newton's direction d_1 is not computable, due to the singularity of the Hessian, then compute the gradient direction $d_2 = -g_k$ at x_k, and go to step 8.
Step 3. Compute the gradient direction d_2 and the Newton's direction d_1 at x_k.
Step 4. Set $\delta = \delta_0$ and $\Lambda = \Lambda_0$. If $|f(x_k) - f(x_{k-1})| > \gamma_1$ and $\|g_k\| > \gamma_2$, set $\delta \leftarrow b_2\delta_0$ and go to step 7.
Step 5. If $k = 1$ or if $\|g_k\| \leq \|g_{k-1}\|$, set $\overline{x} = x_k + d_1$ and go to step 6.
Step 6. If $|f(\overline{x})| < |f(x_k)|$ and $\|g(\overline{x})\| \leq \eta\|g_k\|$ then $\delta \leftarrow b_1\delta_0$.
Step 7. If $d_1^T d_2 \geq 0$ go to step 9.
Step 8. Use Wolfe-Powell rules to determine a step length $\alpha_k > 0$ along the direction $d_k = d_2$, set $s_k = \alpha_k d_k$ and go to step 12.
Step 9. Compute ξ as follows:

$$\xi = \frac{1}{\Lambda + |f(x_k) - f(x_{k-1})|}, \tag{5}$$

and set $d(\xi) = (1 - \xi)d_2 + \xi d_1$.
Step 10. If $d(\xi)^T d_2 < \delta\|d(\xi)\|\|d_2\|$, set $\Lambda \leftarrow \Lambda b_3$ and go back to step 9.
Step 11. Consider one of the following two versions to calculate s_k :
a. Use Wolfe-Powell rules to determine a step length $\alpha_k > 0$ along the direction $d_k = d_2$ and set $\widetilde{s_k} = \alpha_k(1-\xi)d_2 + \xi d_1$. If $f(x_k + \widetilde{s_k}) \leq f(x_k) - \tau\|s_k\|$ and $\alpha_k\|d_2\| \leq T\|d_1\|$, set $s_k = \widetilde{s_k}$; otherwise set $s_k = \alpha_k d_2$.
b. Use Wolfe-Powell rules to determine a step length $\alpha_k > 0$ along the direction $d_k = d(\xi)$ and set $s_k = \alpha_k d_k$.
Step 12. Set $x_{k+1} = x_k + s_k$.

Parameters $b_1,\ b_2$ and b_3 are positive constants so that they offer the user the opportunity to specify the amount contribution of the methods. In this algorithm, when the slope of the function is slight, the algorithm tends to the Newton's method, otherwise the contribution of the Gradient method is increased. Global convergence as well as superlinear convergence rate of this algorithm are established under some conditions in (Taheri and Mammadov, 2012; Taheri *et al.*, 2012).

3.2. Fayyad and Irani's Discretization Method.

In order to apply the NB classifier to data sets with continuous features, one should first discretize those features. Discretization is a process which transform continuous numeric values into discrete ones. In this paper, we apply three different methods to discretize continuous features. The first one, which is also the simplest one, transforms the values of features to $\{0, 1\}$ using their mean values. We also apply two other methods which allows us to get more than two values for discretized features.

In this section, we give a brief description of the Fayyad and Irani's Discretization method (Fayyad and Irani, 1993) which is the most applied discretization method in the literature. The Fayyad and Irani's Discretization method is based on a minimal entropy heuristic, and it uses the class information entropy of candidate partitions to select bin boundaries for discretization.

Let us consider a given set of observations S, a feature X, and a partition boundary T, the class information entropy of the partition induced by T, denoted $E(X, T; S)$ is given by

$$E(X, T; S) = \frac{|S_1|}{|S|}Ent(S_1) + \frac{|S_2|}{|S|}Ent(S_2),$$

where $S_1 \subset S$ be the subset of observations in S with X-values not exceeding T and $S_2 = S - S_1$. Let there be m classes $C_1, ..., C_m$, and $P(C_i, S)$ be the proportion of observations in S that have the class C_i . The class entropy of a subset S is defined as:

$$Ent(S) = -\sum_{i=1}^{m} P(C_i, S) \lg(P(C_i, S)),$$

where the logarithm may be to any convenient base. When the base is 2, $Ent(S)$ measures the amount of information needed, in bits, to specify the classes in S.

For a given feature X, the boundary T_{min} which minimizes the entropy function over all possible partition boundaries is selected as a binary discretization boundary. This method can then be applied recursively to both of the partitions induced by T_{min} until some stopping condition is achieved, thus creating multiple intervals on the feature X.

Fayyad and Irani make use of the minimal description length principle to determine a stopping criteria for their recursive discretization strategy. Recursive partitioning within a set of values S stops if

$$Gain(X, T; S) < \frac{\lg_2(N-1)}{N} + \frac{\triangle(X, T; S)}{N},$$

where N is the number of observations in the set S, and

$$Gain(X, T; S) = Ent(S) - E(X, T; S),$$

$$\triangle(X, T; S) =$$

$$\lg_2(3^m - 2) - [m.Ent(s) - m_1.Ent(S_1) - m_2 Ent(S_2)],$$

and m_i is the number of class labels represented in the set S_i. Since the partitions along each branch of the recursive discretization are evaluated independently using this criteria, some areas in the continuous spaces will be partitioned very finely whereas others (which have relatively low entropy) will be partitioned coarsely.

3.3. Discretization Algorithm SOAC. In this section, we give a brief description of the discretization algorithm SOAC which is recently introduced in (Yatsko *et al.*, 2011).

Let consider a finite set of points A in the n-dimensional space R^n, that is

$$A = \{a^1, ..., a^m\},$$

where $a^i \in R^n$, $i = 1, ..., m$. Assume the sets A^j, $j = 1, ..., k$ be clusters, and each cluster A^j can be identified by its centroid $x^j \in R^n$, $j = 1, ..., k$. The Algorithm SOAC proceeds as follows.

Algorithm. Discretization Algorithm SOAC

Step 1. Set $k = m$, and a small value of parameter θ, $0 < \theta < 1$. Sort values of the current feature in the ascending order. Each continuous feature requiring discretization is treated in turn.
Step 2. Calculate the center of each cluster:

$$x^j = \sum_{a \in A^j} \frac{a}{|A^j|}, \; j = 1, ..., k$$

and the error E_k of the cluster system approximating set A:

$$E_k = \sum_{j=1}^{k} \sum_{a \in A^j} \|x^j - a\|^2.$$

Step 3. Merge in turn each cluster with the next tentatively. Calculate the error increase $E_{k-1} - E_k$ after each merge and choose the pair of clusters giving the least increase. Merge these two clusters permanently. Set $k = k - 1$.
Step 4. If $E_k \geq \theta E_1$, then stop, otherwise go to Step 2.

In the next section, we introduce three optimization models to improve the performance of the NB classifier. In the proposed models, class probabilities and conditional probabilities are considered as unknown variables, and the optimal values of these variables are computed by applying optimization techniques.

4. Optimization Models

Consider a data set $D = \{(\mathbf{X}_1, C_1), (\mathbf{X}_2, C_2), ..., (\mathbf{X}_N, C_N)\}$, where N is the number of observations; $\mathbf{X}_i = \{X_{i1}, X_{i2}, ..., X_{in}\}$, n is the number of features. We assume binary classification; that is $C_i \in \{-1, 1\}$, $i = 1, ..., N$, and we use the notation $\overline{C} = -C$.

From the Bayes rule, we know that

$$P(C_i|\mathbf{X}_i) = \frac{P(\mathbf{X}_i|C_i)P(C_i)}{P(\mathbf{X}_i)}, \tag{6}$$

where $P(\mathbf{X}_i) = P(\mathbf{X}_i|C_i)P(C_i) + P(\mathbf{X}_i|\overline{C_i})P(\overline{C_i})$. Since C_i and $\overline{C_i}$ are complimentary to each other, and $P(C_i), P(\overline{C_i})$ are probabilities, we have

$$P(C_i) + P(\overline{C_i}) = 1, \;\; 0 \leq P(C_i), P(\overline{C_i}) \leq 1. \tag{7}$$

In the NB classifier, it is assumed that all features are independent of each other given the class. This means that:

$$P(\mathbf{X}_i|C_i) = \prod_{j=1}^{n} P(X_{ij}|C_i). \tag{8}$$

Therefore the formula (6) for the NB can be rewritten as $P(C_i|\mathbf{X}_i) =$

$$\frac{\prod_{j=1}^{n} P(X_{ij}|C_i)P(C_i)}{\prod_{j=1}^{n} P(X_{ij}|C_i)P(C_i) + \prod_{j=1}^{n} P(X_{ij}|\overline{C_i})P(\overline{C_i})}. \quad (9)$$

Similarly $P(\overline{C}_i|\mathbf{X}_i) =$

$$\frac{\prod_{j=1}^{n} P(X_{ij}|\overline{C}_i)P(\overline{C}_i)}{\prod_{j=1}^{n} P(X_{ij}|C_i)P(C_i) + \prod_{j=1}^{n} P(X_{ij}|\overline{C_i})P(\overline{C_i})}. \quad (10)$$

Using definition of the conditional probability

$$P(X_{ij}|C_i) = \frac{P(X_{ij}, C_i)}{P(C_i)}, \quad (11)$$

(9) and (10) can be represented as

$$P(C_i|\mathbf{X}_i) = \frac{\frac{\prod_{j=1}^{n} P(X_{ij},C_i)}{\left(P(C_i)\right)^{n-1}}}{\frac{\prod_{j=1}^{n} P(X_{ij},C_i)}{\left(P(C_i)\right)^{n-1}} + \frac{\prod_{j=1}^{n} P(X_{ij},\overline{C}_i)}{\left(P(\overline{C}_i)\right)^{n-1}}}, \quad (12)$$

and

$$P(\overline{C}_i|\mathbf{X}_i) = \frac{\frac{\prod_{j=1}^{n} P(X_{ij},\overline{C}_i)}{\left(P(\overline{C}_i)\right)^{n-1}}}{\frac{\prod_{j=1}^{n} P(X_{ij},C_i)}{\left(P(C_i)\right)^{n-1}} + \frac{\prod_{j=1}^{n} P(X_{ij},\overline{C}_i)}{\left(P(\overline{C}_i)\right)^{n-1}}}. \quad (13)$$

Considering that C_i is the class of the observation $\mathbf{X}_i$, the value of $P(C_i|\mathbf{X}_i)$ is expected to be greater than the value of $P(\overline{C_i}|\mathbf{X}_i)$ for the majority of observations, $i = 1, ..., N$.

In the next three subsections, we present three different optimization models for the NB classifier by considering class probabilities and conditional probabilities as unknown variables.

4.1. Model 1: Optimization Model based on Class Probabilities.

4.1. Model 1: Optimization Model based on Class Probabilities. In this subsection, we consider class probabilities as variables. We introduce the variable v_1 for the probability $P(1)$, and v_2 for the probability $P(-1)$. Since $v_1 + v_2 = 1$, we take the variable $w = v_1$ and so $v_2 = 1 - w$. Let us consider

$$\xi(w; C) = \begin{cases} w & \text{if } C = 1, \\ 1 - w & \text{if } C = -1, \end{cases} \quad (14)$$

then the objective functions for (12) and (13) can be written as

$$f_1(w) = \sum_{i=1}^{N} \frac{\frac{\prod_{j=1}^{n} P(X_{ij},C_i)}{\left(\xi(w;C_i)\right)^{n-1}}}{\frac{\prod_{j=1}^{n} P(X_{ij},C_i)}{\left(\xi(w;C_i)\right)^{n-1}} + \frac{\prod_{j=1}^{n} P(X_{ij},\overline{C}_i)}{\left(\xi(w;\overline{C}_i)\right)^{n-1}}}, \quad (15)$$

and

$$f_2(w) = \sum_{i=1}^{N} \frac{\frac{\prod_{j=1}^{n} P(X_{ij},\overline{C}_i)}{\left(\xi(w;\overline{C}_i)\right)^{n-1}}}{\frac{\prod_{j=1}^{n} P(X_{ij},C_i)}{\left(\xi(w;C_i)\right)^{n-1}} + \frac{\prod_{j=1}^{n} P(X_{ij},\overline{C}_i)}{\left(\xi(w;\overline{C}_i)\right)^{n-1}}}. \quad (16)$$

By considering C_i as the class of $\mathbf{X}_i$, it is quite natural that the value of $f_1(w)$ should be maximized, meanwhile the value of $f_2(w)$ to be minimized. Therefore, the NB classifier leads to a multi-criteria optimization problem:

$$maximize: \ f(w) \doteq \Big(f_1(w), \frac{1}{f_2(w)}\Big) \quad (17)$$

$$\text{subject to} \quad 0 \le w \le 1.$$

In this paper, instead of multi-criteria optimization problem, we consider a single objective maximization problem by taking $\psi(w) = \frac{f_1(w)}{f_2(w)}$:

$$maximize: \ \psi(w) =$$

$$\sum_{i=1}^{N} \frac{\prod_{j=1}^{n} P(X_{ij}, C_i)\big(\xi(w;\overline{C}_i)\big)^{n-1}}{\prod_{j=1}^{n} P(X_{ij}, \overline{C}_i)\big(\xi(w;C_i)\big)^{n-1}}, \quad (18)$$

$$\text{subject to} \quad 0 \le w \le 1.$$

4.2. Model 2: A Simplified Version of Model 1.

4.2. Model 2: A Simplified Version of Model 1. In this subsection, for simplicity we consider only the variable w for the probability $P(1)$, and $1 - w$ for the probability $P(-1)$ in formulas (9) and (10). Therefore, the second optimization model for the NB classifier under these assumptions and using (14) can be described by the following objective functions:
$\widetilde{f}_1(w) =$

$$\sum_{i=1}^{N} \frac{\prod_{j=1}^{n} P(X_{ij}|C_i)\xi(w;C_i)}{\prod_{j=1}^{n} P(X_{ij}|C_i)\xi(w;C_i) + \prod_{j=1}^{n} P(X_{ij}|\overline{C_i})\xi(w;\overline{C}_i)}, \quad (19)$$

and $\widetilde{f}_2(w) =$

$$\sum_{i=1}^{N} \frac{\prod_{j=1}^{n} P(X_{ij}|\overline{C}_i)\xi(w;\overline{C}_i)}{\prod_{j=1}^{n} P(X_{ij}|C_i)\xi(w;C_i) + \prod_{j=1}^{n} P(X_{ij}|\overline{C_i})\xi(w;\overline{C}_i)}. \quad (20)$$

Then we can consider an optimization problem in a similar way to (17), and a single objective maximization problem can be formulated by taking $\phi(w) = \frac{\widetilde{f}_1(w)}{\widetilde{f}_2(w)}$:

$$maximize: \ \phi(w) =$$

$$\sum_{i=1}^{N} \frac{\prod_{j=1}^{n} P(X_{ij}|C_i)\xi(w;C_i)}{\prod_{j=1}^{n} P(X_{ij}|\overline{C_i})\xi(w;\overline{C}_i)}, \quad (21)$$

$$\text{subject to} \quad 0 \le w \le 1.$$

Since problems in Models 1 and 2 are univariate optimization problems, we partition the constraint $0 \leq w \leq 1$ in to 1000 intervals and we find the maximum value of the objective function in each model.

4.3. Model 3: Optimization Model based on Class Probabilities and Conditional Probabilities. In the third optimization model for the NB classifier, we discretize the values of all features to binary values, $\{0, 1\}$, by applying mean value of each feature. Since we have binary classification $1, -1$, we consider not only $P(1)$ and $P(-1)$, but also the conditional probabilities $P(1|1)$, $P(0|1)$, $P(1|-1)$ and $P(0|-1)$ as variables. For each feature j, $j = 1, ..., n$, we introduce four variables: v_{1j} for $P(jth\ feature\ is\ 1|class\ is\ 1)$, v_{2j} for $P(jth\ feature\ is\ 0|class\ is\ 1)$, v_{3j} for $P(jth\ feature\ is\ 1|class\ is\ -1)$ and v_{4j} for $P(jth\ feature\ is\ 0|class\ is\ -1)$. As a result we have a matrix of $4n$ variables

$$V = \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \\ v_{31} & v_{32} & \dots & v_{3n} \\ v_{41} & v_{42} & \dots & v_{4n} \end{pmatrix}. \tag{22}$$

Since we have constraints $v_{1j} + v_{2j} = 1$ and $v_{3j} + v_{4j} = 1,\ j = 1, ..., n$, the matrix V can be rewritten as

$$W = \begin{pmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \end{pmatrix}, \tag{23}$$

where $w_{1j} = v_{1j}$ and $w_{2j} = v_{3j},\ j = 1, ..., n$. Clearly $v_{2j} = 1 - w_{1j}$ and $v_{4j} = 1 - w_{2j},\ j = 1, ..., n$.

Similar to (14) we introduce

$$\zeta(\alpha, \beta; X, C) = \begin{cases} \alpha & \text{if } X = 1, C = 1, \\ 1-\alpha & \text{if } X = 0, C = 1, \\ \beta & \text{if } X = 1, C = -1, \\ 1-\beta & \text{if } X = 0, C = -1. \end{cases} \tag{24}$$

Then using (14) and (24), the formulas (19) and (20) will be

$$\widehat{f}_1(w, W) = \sum_{i=1}^{N} \frac{\prod_{j=1}^{n} \zeta(w_{1j}, w_{2j}; X_{ij}, C_i)\xi(w; C_i)}{\mathfrak{P}}, \tag{25}$$

and

$$\widehat{f}_2(w, W) = \sum_{i=1}^{N} \frac{\prod_{j=1}^{n} \zeta(w_{1j}, w_{2j}; X_{ij}, \overline{C}_i)\xi(w; \overline{C}_i)}{\mathfrak{P}}, \tag{26}$$

where
$$\mathfrak{P} = \prod_{j=1}^{n} \left(\zeta(w_{1j}, w_{2j}; X_{ij}, C_i)\xi(w; C_i) + \zeta(w_{1j}, w_{2j}; X_{ij}, \overline{C}_i)\xi(w; \overline{C}_i)\right).$$

Therefore, the maximization problem for this model is

$$maximize: \ \ \varphi(w, W) = \sum_{i=1}^{N} \frac{\prod_{j=1}^{n} \zeta(w_{1j}, w_{2j}; X_{ij}, C_i)\xi(w; C_i)}{\prod_{j=1}^{n} \zeta(w_{1j}, w_{2j}; X_{ij}, \overline{C}_i)\xi(w; \overline{C}_i)}, \tag{27}$$
$$\text{subject to } \ 0 \leq w, w_{1j}, w_{2j} \leq 1, \ \ j = 1, ..., n.$$

The problem (27) is a constrained optimization problem. We apply the penalty method to reduce this problem to an unconstrained one. The unconstrained problem is as follows:
$maximize:$

$$\varpi(w, W, \mu) = \varphi(w, W) - \mu\big\{[\max(0, -w, w-1)]^2 + \sum_{j=1}^{n} \left([\max(0, -w_{1j}, w_{1j} - 1)]^2 + [\max(0, -w_{2j}, w_{2j} - 1)]^2\right)\big\}, \tag{28}$$

where $\mu > 0$ is a penalty parameter.

For solving the nonlinear and nonconvex unconstrained optimization problem (28), we use the new globally convergent optimization method, CGN, (Taheri and Mammadov, 2012) which is briefly introduced in Section 3. The reason for choosing this method is that it has a better performance than some well known local optimization methods such as the Gradient method and the Newton's method. We showed this fact in the numerical section.

The Model 3 can be generalized to any discrete features. We have not considered such a model in this paper that could be a topic for a separate research paper.

5. Numerical Experiments

To verify the efficiency of the proposed models, numerical experiments with a number of real world data sets have been carried out. We have chosen 14 binary classification data sets in which they are the most frequently binary classification data sets considered in the literature. A brief description of these data sets is given in Table 1. The detailed description of them can be found in the UCI machine learning repository (Asuncion and Newman, 2007), and tools page of the LIBSVM (Chang and Lin, 2001).

We discretize the values of features in data sets using three different methods. In the first one which is the simplest method, we apply a mean value of each feature to discretize the values to $\{0, 1\}$. In the second one, we apply the well known method, Fayyad and Irani's discretization method. The third one is the recently introduced discretization method, Algorithm SOAC; the parameter θ in this algorithm is chosen as 0.2. This parameter has not been fitted by preliminary experimentation, and is similar to the one used for other problems in (Yatsko *et al.*, 2011).

The efficiency of the CGN method when applied to the Model 3 have been tested on some data sets randomly chosen from Table 1: such as credit approval, diabetes and heart disease, and we have compared this method with the Gradient method (GM) and the Newton's method (NM). Tables 2 to 4 show the average maximum value of the objective function of the Model 3, $\varpi_{max}(w, W, \mu)$, over 10 fold cross validation for three different data sets. Fail (fail) occurs when the constraints are not satisfied. All the optimization methods are terminated if the number of iterations to find the solution exceeds 100 or $\|g_k\| < 10^{-3}$ or $|f_k - f_{k-1}| < 10^{-3}$. The penalty parameter in the problem (28) is chosen as $\mu = 10^6$. We have chosen 9 initial points for w from the box $(0, 1)$, and therefore for $1 - w \in (0, 1)$, and conditional probabilities are taken as initial points for W. The parameters of the Algorithm CGN are chosen: $\delta_0 = 0.001, \Lambda_0 = 1, \eta = 0.99, \rho = 0.001, \sigma = 0.9, \gamma_1 = \gamma_2 = n, b_1 = 0.01, b_2 = 1/b_1, b_3 = 1.1, \tau = 10^{-10}$ and $T = 10^{10}$; these parameters have the same values as chosen in the paper (Taheri and Mammadov, 2012).

From Tables 2 to 4, it is shown that the values of the objective function for the maximization problem (28) obtained by the Algorithm CGN are higher than those obtained by the Gradient and the Newton's methods. In the credit approval data set, the values of the objective function are 410.326 at different initial points using the CGN method, while they vary using the Gradient method and the Newton's method. The Gradient method has even less value at initial points $(0.1, 0.9)$ and $(0.9, 0.1)$ which is 350.1. In addition, the values of the objective function when applying the Newton's method fails at some initial points. The values obtained by this method are the same as the values for the CGN method only at few initial points. This means that the Newton's method is heavily dependent on initial points.

For diabetes data set, the CGN method gives the values of the objective function 249.570 at all initial points which is the highest value for this data set. However, the values vary using the Gradient and the Newton's methods. It is notable that in the Gradient method, the objective function value is 152.3 at the initial points $(0.2, 0.8)$ and $(0.8, 0.2)$. Even, the Gradient method fails at the point $(0.1, 0.9)$. Moreover, it can be observed that the Newton's method has approximately the same objective function values at some initial points as the CGN method, but it fails in some others. In summary, in the heart disease data set we have similar results to data sets credit approval and diabetes, where the CGN method outperforms the Gradient method and the Newton's method. Therefore, the CGN method is more robust and efficient local optimization method to be applied in the Model 3 to find a better solution.

We have also compared the test set accuracy of the proposed models with the NB, the Tree Augmented Naive Bayes (TAN), the SVM and the C4.5 using different discretization methods. The Model 3 is not suitable when applying the discretization methods Fayyad and Irani and SOAC as it needs binary values for features. The results in Tables 5 to 7 are the average values of test set accuracy over 10 fold cross validation. Run with the various methods are carried out on the same training sets and evaluated on the same test sets. In particular, the cross validation folds are the same for all classifiers on each data set.

Table 5 demonstrates the test set accuracy obtained by the NB, the TAN, the SVM, the C4.5, Models 1, 2 and 3 on 14 data sets using mean values for discretization. The results presented in this table demonstrate that the test set accuracy of Models 2 and 3 in all data sets are better than those obtained by the NB. The Model 1 has also the higher accuracy than the NB in the majority of data sets. In 12 cases out of 14, the Model 1 shows better accuracy than the NB. In data sets credit approval and spam base, the accuracy of this model almost ties with those obtained by the NB. Table 5 also shows that the proposed models have much better accuracy than the TAN in all data sets. Observe from this table, the Model 2 has better accuracy than the SVM in 11 cases out of 14, and the Model 1 performs better than the SVM in 9 cases out of 14. The Model 2 has the greater accuracy than C4.5 in 12 data sets, and the accuracy obtained by the Model 1 is higher than C4.5 in 10 cases out of 14. The results from this table also show that the Models 3 outperform the SVM and C4.5 in all data sets.

Table 6 presents the test set accuracy obtained by the NB, the TAN, the SVM, the C4.5, Models 1 and 2 on 14 data sets, where continuous features are discretized by applying the Fayyad and Irani's method (FaI). The results presented in this table demonstrate that the accuracy of Models 1 and 2 are significantly better than those of the NB in all data sets. On 12 data sets out of 14, these models perform better than the TAN, where as the TAN has slightly higher accuracy in data sets Ionosphere and Sonar. The results from Table 6 also indicate that Models 1 and 2 have the greater accuracy than the SVM in 11 data sets. Compared to the C4.5, Table 6 shows higher accuracy for Models 1 and 2 in 10 data sets.

The test set accuracy obtained by the NB, the TAN, the SVM, the C4.5, Models 1 and 2 on 14 data sets using discretization algorithm SOAC is summarized in Table 7. The results from this table show that the accuracy obtained by Models 1 and 2 in all data sets are higher than those obtained by the NB. The accuracy of Models 1 and 2 are better than those of the TAN on most of data sets. In 13 cases out of 14, Models 1 and 2 have greater accuracy than the TAN. The results from Table 7 also demonstrate that the Model 2 has the greater accuracy than the SVM in 12 data sets out of 14, and the later method outperforms the Model 1 in diabetes, german.numer and sonar data sets. Table 7 indicates higher accuracy for the Model 2 in 13 data sets, and 11 cases for the Model 1 when compared to the C4.5.

The numerical results generally demonstrate that the proposed models can significantly improve the performance of the Naive Bayes classifier, yet at the same time maintains its simple structure. Especially the Model 3 has a dramatic increase in the test set accuracy, and it is reasonable due to considering more variables replacing class probabilities and conditional probabilities which allows to build more accurate model. However, these models require more training time than the Naive Bayes classifier due to applying optimization techniques.

6. Conclusion

In this paper, we have introduced three different optimization models for the Naive Bayes classifier by considering class probabilities and conditional probabilities as unknown variables. Then, we have applied optimization techniques to find the optimal values for these variables. We have compared the proposed models with the NB, the TAN, the SVM, and the C4.5 on 14 real world binary classification data sets. The values of features in data sets are discretized by using mean value of each feature and applying two different discretization algorithms, the Fayyad and Irani's method and the Algorithm SOAC. We have presented results of numerical experiments. The results demonstrate that the proposed models perform better than the NB, the TAN, the SVM, and the C4.5 in terms of accuracy, yet at the same time maintains the simple structure of the NB. How these optimization models for the Naive Bayes classifier perform in multi class data sets, and also generalizing the Model 3 to any discrete features remain

Table 1. A brief description of data sets.

Data sets	# Observations	# Features
Breast Cancer	699	10
Congres Vote	435	16
Credit Approval	690	15
Diabetes	768	8
German.numer	1000	24
Haberman	306	3
Heart Disease	303	14
Hepatitis	155	19
Ionosphere	351	34
Liver Disorders	345	6
Sonar	208	60
Spambase	4601	57
Svmguide1	7089	4
Svmguide3	1284	21

Table 2. Average maximum value of the objective function in Model 3 over 10 fold cross validation for **Credit Approval** data set.

Initial point	CGN	GM	NM
$(0.1, 0.9)$	410.326	350.128	410.326
$(0.2, 0.8)$	410.326	398.232	408.322
$(0.3, 0.7)$	410.326	367.914	fail
$(0.4, 0.6)$	410.326	407.562	406.453
$(0.5, 0.5)$	410.326	409.525	410.326
$(0.6, 0.4)$	410.326	407.578	fail
$(0.7, 0.3)$	410.326	398.765	fail
$(0.8, 0.2)$	410.326	381.209	408.276
$(0.9, 0.1)$	410.326	350.119	410.323

Table 3. Average maximum value of the objective function in Model 3 over 10 fold cross validation for **Diabetes** data set.

Initial point	CGN	GM	NM
$(0.1, 0.9)$	249.570	fail	249.569
$(0.2, 0.8)$	249.570	152.392	fail
$(0.3, 0.7)$	249.570	207.392	fail
$(0.4, 0.6)$	249.570	207.426	249.569
$(0.5, 0.5)$	249.570	207.561	249.568
$(0.6, 0.4)$	249.570	207.269	fail
$(0.7, 0.3)$	249.570	239.962	249.570
$(0.8, 0.2)$	249.570	152.302	fail
$(0.9, 0.1)$	249.570	247.292	249.569

Table 4. Average maximum value of the objective function in Model 3 over 10 fold cross validation for **Heart Disease** data set.

Initial point	CGN	GM	NM
$(0.1, 0.9)$	170.472	162.844	170.472
$(0.2, 0.8)$	170.472	161.274	167.001
$(0.3, 0.7)$	170.472	168.441	170.472
$(0.4, 0.6)$	170.472	168.441	fail
$(0.5, 0.5)$	170.472	168.441	170.472
$(0.6, 0.4)$	170.472	168.441	fail
$(0.7, 0.3)$	170.472	161.238	fail
$(0.8, 0.2)$	170.472	168.441	170.472
$(0.9, 0.1)$	170.472	142.305	170.472

important questions for the future work.

Table 5. Average test set accuracy over 10 fold cross validation for 14 data sets using mean value for discretization.

Data Sets	NB	TAN	SVM	C4.5	Model1	Model2	Model3
Breast	96.10	95.71	95.15	91.06	97.18	97.61	97.94
Congress	90.31	91.42	96.02	95.51	96.43	96.81	96.92
Credit	84.95	82.88	85.31	87.47	84.62	88.37	89.21
Diabetes	75.90	76.48	76.72	75.98	76.72	77.16	78.54
German	75.41	74.13	76.11	72.43	75.74	75.81	78.18
Haberman	75.01	73.85	73.34	71.60	75.80	77.84	78.46
Heart	81.12	84.12	80.14	81.53	86.61	87.52	88.49
Hepatitis	83.61	83.14	83.61	82.97	83.97	84.12	84.53
Ionosphere	82.90	84.02	85.94	86.20	84.12	85.89	88.17
Liver	61.86	61.89	60.16	60.79	63.19	65.84	65.94
Sonar	75.18	75.44	76.98	76.65	76.19	76.41	79.92
Spambase	90.03	89.69	90.17	91.89	89.98	92.18	92.97
Svmguide1	92.57	91.99	93.24	93.62	93.71	94.79	96.84
Svmguide3	81.51	83.04	80.16	81.24	83.17	83.81	86.78

Table 6. Average test set accuracy over 10 fold cross validation for 14 data sets using discretization Method FaI.

Data Sets	NB	TAN	SVM	C4.5	Model1	Model2
Breast Cancer	97.18	96.52	96.52	94.11	97.72	97.78
Congress Vote	90.11	93.21	95.04	95.32	95.81	96.12
Credit Approval	86.10	84.78	85.03	84.87	86.93	87.71
Diabetes	74.56	75.14	75.51	73.83	76.42	75.84
German.numer	74.50	73.13	76.41	71.92	75.95	76.81
Haberman	75.09	74.41	73.20	71.24	77.14	76.87
Heart Disease	82.93	81.23	81.67	82.85	83.56	85.62
Hepatitis	84.56	83.91	85.16	83.87	85.76	85.81
Ionosphere	88.62	89.77	89.67	89.98	88.96	89.57
Liver Disorders	63.26	63.18	62.03	62.15	64.83	65.72
Sonar	76.32	76.47	77.96	77.31	76.40	76.37
Spambase	90.41	89.78	90.43	92.97	92.84	92.51
Svmguide1	92.39	91.61	94.31	95.99	94.88	93.98
Svmguide3	81.23	82.47	80.37	81.38	84.90	86.12

Table 7. Average test set accuracy over 10 fold cross validation for 14 data sets using discretization Algorithm SOAC.

Data Sets	NB	TAN	SVM	C4.5	Model1	Model2
Breast Cancer	96.12	95.60	95.31	91.16	97.85	97.99
Congress Vote	90.11	91.42	96.75	95.12	96.81	96.97
Credit Approval	85.85	84.98	86.11	87.54	86.94	88.51
Diabetes	75.78	75.90	76.68	75.63	76.17	78.12
German.Numer	74.61	74.01	76.35	72.21	76.08	76.19
Haberman	74.66	76.08	73.36	72.15	75.61	75.32
Heart Disease	78.62	77.37	77.96	79.17	79.44	84.11
Hepatitis	82.93	81.54	84.24	82.34	85.83	86.37
Ionosphere	85.92	86.18	86.15	86.71	86.98	88.23
Liver Disorders	65.82	65.73	63.69	64.98	66.51	66.94
Sonar	75.09	75.76	77.74	76.41	76.13	76.81
Spambase	89.30	89.04	91.56	93.73	92.89	93.43
Svmguide1	95.81	94.91	95.94	96.91	97.38	97.75
Svmguide3	77.25	79.99	78.32	78.49	82.11	82.27

References

Asuncion, A. and Newman, D. (2007). Uci machine learning repository, `http://www.ics.uci.edu/mlearn/mlrepository`.

Campos, M., Fernandez-Luna, Gamez, A. and Puerta, M. (2002). Ant colony optimization for learning bayesian networks, *International Journal of Approximate Reasoning* **31**: 291–311.

Chang, C. and Lin, C. (2001). Libsvm: A library for support vector machines, `http://www.csie.ntu.edu.tw/cjlin/libsvm`.

Chickering, D. M. (1996). Learning bayesian networks is np-complete, *Artificial Intelligence and statistics* pp. 121–130.

Crawford, E., Kay, J. and Eric, M. (2002). The intelligent email sorter, *Proceedings of the 19th International Conference on Machine Learning*, pp. 83–90.

Domingos, P. and Pazzani, M. (1996). Beyond independence: Conditions for the optimality of the simple bayesian classifier, *Proceedings of the Thirteenth International Conference on Machine Learning, Bari, Italy, Morgan Kaufmann*.

Domingos, P. and Pazzani, M. (1997). Learning bayesian networks is np-complete, *Macine Learning* pp. 103–130.

Dougherty, J., Kohavi, R. and Sahami, M. (1995). Supervised and unsupervised discretization of continuous features, *Proceedings of the 12th International Conference on Machine Learning*, pp. 194–202.

Fayyad, U. M. and Irani, K. (1993). On the handling of continuous-valued attributes in decision tree generation, *Macine Learning* **8**: 87–102.

Friedman, N., Geiger, D. and Goldszmidti, M. (1997). Bayesian network classifiers, *Macine Learning* **29**(2): 131–163.

Heckerman, D., Chickering, D. and Meek, C. (2004). Large sample learning of bayesian networks is np-hard, *Journal of Machine Learning Research* pp. 1287–1330.

Kononenko, I. (2001). Machine learning for medical diagnosis: History, state of the art and perspective, *Artificial Intelligence in Medicine* **23**: 89–109.

Langley, P., Iba, W. and Thompson, K. (1992). An analysis of bayesian classifiers, *In 10th International Conference Artificial Intelligence, AAAI Press and MIT Press* pp. 223–228.

Miyahara, K. and Pazzani, M. J. (2000). Collaborative filtering with the simple bayesian classifier, *Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence*, pp. 679–689.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann.

Taheri, S. and Mammadov, M. (2012). Solving systems of nonlinear equations using a globally convergent optimization algorithm, *Global Journal of Technology and Optimization* **3**: 132–138.

Taheri, S., Mammadov, M. and Seifollahi, S. (2012). Globally convergent optimization methods for unconstrained problems, *Submitted to Optimization* .

X., W., Kumar, J. V., Quinlan, R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, J., Ng, A., Liu, B., Yu, P. S., Zhou, Z., Steinbach, M., Hand, D. J. and Steinberg, D. (2008). Top 10 algorithms in data mining, *Springer-Verlag London Limited, Knowledge Information System* pp. 1–37.

Yatsko, A., Bagirov, A. M. and Stranieri, A. (2011). On the discretization of continuous features for classification, *Proceedings of Ninth Australasian Data Mining Conference (AusDM 2011), Ballarat, Australia*, Vol. 125.

Zupan, B., Demsar, J., Kattan, M. W., Ohori, M., Graefen, M., Bohanec, M. and Beck, J. R. (2001). Orange and decisions-at-hand: Bridging predictive data mining and decision support, *Proceedings of ECML/PKDD Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning*, pp. 151–162.

Received:

5.2 Attribute Weighted Naive Bayes Classifier using a Local Optimization

This section presents a new attribute weighted Naive Bayes classifier method. The method assigns some weights to conditional attribute class probabilities. These weights are considered in the form of powers to conditional probabilities. An objective function is constructed based on the Naive Bayes' structure and the attribute weights. The optimal values of weights are computed by using a local optimization method. The numerical results on several well known real world data sets demonstrate that the proposed method can efficiently improve the performance of the Naive Bayes and yield more accurate probability prediction than it.

Paper:

Attribute Weighted Naive Bayes Classifier using a Local Optimization

Submitted to:

Neural Computing and Applications Journal, February 2012

Authors: Sona Taheri[a], John Yearwood[a], Musa Mammadov[a, b] and Sattar Seifollahi[c]

[a]Centre for Informatics and Applied Optimization, SITE,
University of Ballarat, VIC 3353, Australia

[b]National ICT Australia, VRL, VIC 3010, Australia

[c]School of Civil, Environmental and Mining Engineering,
University of Adelaide, SA 5005, Australia

Corresponding author:

Sona Taheri

e-mail: sonataheri@students.ballarat.edu.au

Neural Computing and Applications manuscript No.
(will be inserted by the editor)

Attribute Weighted Naive Bayes Classifier Using a Local Optimization

Sona Taheri · John Yearwood · Musa Mammadov · Sattar Seifollahi

Received: date / Accepted: date

Abstract The Naive Bayes classifier is a popular classification technique for data mining and machine learning. It has been shown to be very effective on a variety of data classification problems. However, the strong assumption that all attributes are conditionally independent given the class is often violated in real world applications. Numerous methods have been proposed in order to improve the performance of the Naive Bayes classifier by alleviating the attribute independence assumption. However, violation of the independence assumption can increase the expected error. Another alternative is assigning the weights for attributes. In this paper, we propose a novel attribute weighted Naive Bayes classifier by considering weights to the conditional probabilities. An objective function is modeled and taken into account, which is based on the structure of the Naive Bayes classifier and the attribute weights. The optimal weights are determined by a local optimization method using the quasisecant method. In the proposed approach, the Naive Bayes classifier is taken as a starting point. We report the results of numerical experiments on several real world data sets in binary classification, which show the efficiency of the proposed method.

Sona Taheri
School of Science, Information Technology and Engineering, University of Ballarat, VIC 3353, Australia
E-mail: sonataheri@students.ballarat.edu.au

John Yearwood
School of Science, Information Technology and Engineering, University of Ballarat, VIC 3353, Australia

Musa Mammadov
School of Science, Information Technology and Engineering, University of Ballarat, VIC 3353, Australia

Sattar Seifollahi
School of Civil, Environmental and Mining Engineering, University of Adelaide, SA 5005, Australia

Keywords Classification · Naive Bayes · Attribute Weighting · Local Optimization

1 Introduction

Classification is the task of identifying the class labels for instances based on a set of attributes. Learning accurate classifiers from pre-classified data is a very active research topic in machine learning and data mining. Classification learning is the process of predicting a discrete class label $C \in \{C_1, \cdots, C_m\}$ for a test instance $\mathcal{X} = (X_1, \cdots, X_n)$.

One of the most effective classifiers is the Bayesian Network (BN) introduced by Pearl [20]. A BN is composed of a network structure and its conditional probabilities. The structure is a directed acyclic graph where the nodes correspond to domain variables and the arcs between nodes represent direct dependencies between the variables. The classifier represented by the BN can be expressed as:

$$\arg\max_{1\le k\le m} P(C_k|\mathcal{X}) = \arg\max_{1\le k\le m} \frac{P(C_k)P(\mathcal{X}|C_k)}{P(\mathcal{X})}; \tag{1}$$

this rule is called Bayes rule. We can see that for each class, the denominator of equation (1) is the same and it will not interfere in classification. So, the BN classifier can be rewritten as:

$$\arg\max_{1\le k\le m} P(C_k|\mathcal{X}) \propto \arg\max_{1\le k\le m} P(C_k)P(\mathcal{X}|C_k). \tag{2}$$

However, accurate estimation of $P(\mathcal{X}|C_k)$ is non trivial. It has been proved that learning an optimal BN is NP-hard [4] [10]. In order to avoid the intractable complexity for learning the BN, the Naive Bayes (NB) classifier has been used. In the NB [15] [22], attributes are conditionally independent given the class. Compared to other supervised machine learning methods, the NB classifier is perhaps one of the simplest, yet surprisingly powerful, techniques to construct predictive models from labeled training sets. The NB classifier is important for several reasons. It is easy to construct and implement because the structure is given a priori (no structure learning procedure is required) and it needs only to compile a table of class probabilities and conditional probabilities from the training instances. Therefore, it may be readily applied to huge data sets. It is easy to interpret, and even unskilled users in classifier technology can understand why it is making the classification it makes. Finally, it may not be the best possible classifier in any particular application, but it can usually be relied on to be robust and to do quite well [5].

A sample of the NB classifier with n attributes is depicted in Figure 1. The NB classifies an instance $\mathcal{X} = (X_1, \cdots, X_n)$ by selecting

$$\arg\max_{1\le k\le m} P(C_k|\mathcal{X}) \propto \arg\max_{1\le k\le m} P(C_k)\prod_{i=1}^{n} P(X_i|C_k). \tag{3}$$

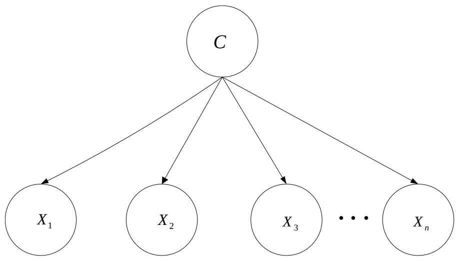


Fig. 1 Naive Bayes

However, the attribute independence assumption made by the NB classifier harms its classification performance when it is violated in reality. In order to relax the attribute independence assumption of the NB classifier while at the same time retaining its simplicity and efficiency, researchers have proposed many effective methods. These methods have been proposed in order to improve the performance of the Naive Bayes classifier by alleviating the attribute independence assumption. Among the variety of works, semi NB classifiers [14] [16] [19] show significant improvements in the NB classifier by using the selected subset of attributes. The Tree Augmented Naive Bayes (TAN) [8] utilizes the tree structure to find relations between attributes. The Super Parent [13] uses the same representation as the TAN, but utilizes leave-one-out cross validation error as a criterion to add a link. The Improved Naive Bayes (INB), proposed by Taheri et al. [22], uses conditional probabilities for finding the dependencies between attributes.

Another way to mitigate its attributes independence assumption is assigning weights to important attributes in the classification. Since attributes do not play the same role in many real world applications, some of them are more important than others. A natural way to extend the NB classifier is to assign a weight to each attribute. This is the main idea of the algorithm called attribute weighted NB. Much work to evaluate the importance of attributes has been done in recent years [9] [11] [18] [26] [33] [31]. Jiang and Zhang [11] developed the improved NB called weightily averaged one-dependence estimators based on the idea of a model introduced by Webb et al. [25]. Hall [9] presented a simple filter method for setting attribute weights to use in the NB classifier. The assumption made is that the weight assigned to a predictive attribute should be inversely related to the degree of dependency it has on other attributes. More recently Wu and Cai [26] used differential evolution algorithms to determine the weights of attributes in the model introduced by Hall [9] and then they used these weights in the developed weighted NB classifier. The paper [31] investigates how to learn a weighted NB classifier with accurate ranking from data, or more precisely, how to learn the weights of a weighted NB classifier to produce accurate ranking.

In this paper, we propose a new attribute weighted NB classifier, called AWNB, which assigns more than one weight for each attribute. The number of weights for each attribute is considered as the number of class labels. These weights are written in the form of powers to the conditional attribute-class probabilities. An objective function is constructed based on the NB structure and the attribute weights. The weights, then, are determined by using a local optimization method, which here is the quasisecant method [2]. The initial weights for the quasisecant method are set to unity; this means that the NB classifier is taken as an initial point. More precisely, our aim is improving the NB classifier by modelling a proper objective function and optimizing the attribute weights. To find a global solution, one can also apply a global optimization, however the complexity of the problem will increase.

Most of data sets in real world applications often involve continuous attributes. The most well known attempt for improving the performance of the NB with continuous attributes is the discretization of the attributes into intervals, instead of using the default option to utilize the normal distribution to calculate probabilities. Numerous discretization methods have been examined for the NB learning [17] [24] [28] [29] [30]. The performance of the NB classifier significantly improves when attributes are discretized using an entropy based method [6]. In this paper, we use Fayyad and Irani's discretization method [7]; a method based on a minimal entropy heuristic. We also apply the discretization algorithm using sub-optimal agglomerative clustering algorithm which is an efficient discretization method, recently introduced in [28].

The rest of the paper is organized as follows. In the next section, we present a brief review of the quasisecant method. Section 3 reviews briefly two different discretization methods, Fayyad and Irani's method and sub-optimal agglomerative clustering based method, respectively. The leaning of the proposed method is illustrated in Section 4, which follows by the experiments and discussion on the experiments in Section 5. Section 6 concludes the paper followed by a few directions for future work.

2 A Brief Review of the Quasisecant Method

The quasisecant method [2] is a local method for solving nonsmooth, nonconvex optimization problems. In general, this method is applicable for solving the following unconstrained minimization problem:

$$minimize \quad f(x) \tag{4}$$

where $x \in \mathbb{R}^d$, and the objective function f is assumed to be locally Lipschitz.

Formally, quasisecants are defined as follows. Let $S = \{x \in R^d : \ \|x\| = 1\}$ be the unit sphere. A vector $v \in R^d$ is called a quasisecant of the function f at the point x in the direction $g \in S$ with the length $h > 0$ iff

$$f(x + hg) - f(x) \leq h\langle v, g\rangle.$$

Here $\langle v, g\rangle$ is the inner product of vectors $v, g \in R^d$. The above inequality is called a quasisecant inequality. Quasisecants provide overestimation to the function f in some neighborhood of a point x. There are many vectors v

satisfying the quasisecant inequality. We consider only those which provide approximation to the function. Subgradient-related quasisecants introduced in [2] provide such approximations and they converge to tangents of the graph of the function f.

Any quasisecant is defined with respect to a given direction $g \in S$ and with given length $h > 0$. The choice of h allows one to compute descent directions with different lengths. Therefore, one can compute descent directions even from some shallow local minimizers using quasisecants. This observation makes the quasisecant method applicable to nonconvex problems and compute a "deep" local minimizers.

On the other hand, the quasisecant method uses a bundle of quasisecants at a given point to compute descent directions which makes it similar to the well-known bundle methods in nonsmooth optimization. Therefore, it is applicable to solve nonsmooth optimization problems. Results presented in [2] demonstrate that the quasisecant method is efficient and robust method for solving nonsmooth, nonconvex optimization problems.

3 Discretization Methods

In order to apply the NB classifier to data sets with continuous attributes, one should first discretize the attributes. Discretization is a process which transforms continuous numeric values into discrete ones. In this paper, we apply two different methods in our experiments to discretize the attributes. The first one is the Fayyad and Irani's discretization method, and the second one is discretization algorithm using sub-optimal agglomerative clustering proposed by Yatsko et al. [28].

3.1 Fayyad and Irani's Method

The Fayyad and Irani's Discretization method is based on a minimal entropy heuristic, and it uses the class information entropy of candidate partitions to select bin boundaries for discretization. In this subsection, we give a brief review to this method, and details can be found in [7].

Let us consider a given set of instances $\mathbf{X}$, an attribute X, and a partition boundary T, the class information entropy of the partition induced by T, denoted $E(X, T; \mathbf{X})$ is given by

$$E(X, T; \mathbf{X}) = \frac{|\mathbf{X}^{(1)}|}{|\mathbf{X}|} Ent(\mathbf{X}^{(1)}) + \frac{|\mathbf{X}^{(2)}|}{|\mathbf{X}|} Ent(\mathbf{X}^{(2)}),$$

where $\mathbf{X}^{(1)} \subset \mathbf{X}$ be the subset of instances in $\mathbf{X}$ with X-values not exceeding T and $\mathbf{X}^{(1)} = \mathbf{X} - \mathbf{X}^{(1)}$. Let there be m classes $C_1, ..., C_m$. Let $P(C_i, \mathbf{X})$ be the proportion of instances in $\mathbf{X}$ that have the class C_i . The class entropy of a subset $\mathbf{X}$ is defined as:

$$Ent(\mathbf{X}) = -\sum_{i=1}^{m} P(C_i, \mathbf{X}) \lg(P(C_i, \mathbf{X})),$$

where the logarithm may be to any convenient base. When the base is 2, $Ent(\mathbf{X})$ measures the amount of information needed, in bits, to specify the classes in $\mathbf{X}$.

For a given attribute X, the boundary T_{min} which minimizes the entropy function over all possible partition boundaries is selected as a binary discretization boundary. This method can then be applied recursively to both of the partitions induced by T_{min} until some stopping condition is achieved, thus creating multiple intervals on the attribute X.

Fayyad and Irani make use of the minimal description length principle to determine a stopping criteria for their recursive discretization strategy. Recursive partitioning within a set of values $\mathbf{X}$ stops if

$$Gain(X, T; \mathbf{X}) < \frac{\lg_2(N-1)}{N} + \frac{\triangle(X, T; \mathbf{X})}{N},$$

where N is the number of instances in the set $\mathbf{X}$, and

$$Gain(X, T; \mathbf{X}) = Ent(\mathbf{X}) - E(X, T; \mathbf{X}),$$

$$\triangle(X, T; \mathbf{X}) = \lg_2(3^m - 2) - [m.Ent(\mathbf{X}) - m_1.Ent(\mathbf{X}^{(1)}) - m_2 Ent(\mathbf{X}^{(2)})],$$

and m_i is the number of class labels represented in the set $\mathbf{X}^{(i)}$. Since the partitions along each branch of the recursive discretization are evaluated independently using this criteria, some areas in the continuous spaces will be partitioned very finely whereas others (which have relatively low entropy) will be partitioned coarsely.

3.2 Sub-Optimal Agglomerative Clustering based Method(SOAC)

In this section, we give a brief description of the discretization algorithm SOAC. Details of this algorithm can be found in [28]. Consider a finite set of points $\mathbf{X}$ in the $n-$dimensional space R^n, that is $\mathbf{X} = \{\mathcal{X}_1, ..., \mathcal{X}_N\}$, where $\mathcal{X}_i \in R^n,\ i = 1, ..., N$. Assume the sets $A_j,\ j = 1, ..., k$ be clusters, and each cluster A_j can be identified by its centroid $\mathcal{X}_j \in R^n,\ j = 1, ..., k$. The discretization algorithm SOAC proceeds as follows.

Step 1. Set $k = N$, and a small value of parameter θ, $0 < \theta < 1$. Sort values of the current feature in the ascending order. Each continuous feature requiring discretization is treated in turn.

Step 2. Calculate the center of each cluster, $\mathcal{X}_j = \sum_{\mathcal{X}\in A_j} \frac{\mathcal{X}}{|A_j|}$, $j = 1, ..., k$ and the error E_k of the cluster system approximating set $\mathbf{X}$, $E_k = \sum_{j=1}^{k} \sum_{\mathcal{X}\in A_j} \|\mathcal{X}_j - \mathcal{X}\|^2$.

Step 3. Merge in turn each cluster with the next tentatively. Calculate the error increase $E_{k-1} - E_k$ after each merge and choose the pair of clusters giving the least increase. Merge these two clusters permanently. Set $k = k - 1$.

Step 4. If $E_k \geq \theta E_1$, then stop, otherwise go to Step 2.

Algorithm 1: Discretization Algorithm SOAC

4 Learning the Proposed Attribute Weighted Naive Bayes using Optimization

Good attribute weighting can eliminate the effects of noisy or irrelevant attributes. In this section, we propose a weighting procedure, in which each conditional attribute-class probability has its own power as a weight. The number of weights for each attribute is equal to the number of class labels. The idea of our weighting method is similar to the works in [26] [33], however constructing a proper objective function and utilizing the new weighting procedure are different from the existing methods.

Let us consider $D = \{\mathcal{X}_i, C_i\}$, $1 \leq i \leq N$, where N is the number of instances and $C_i \in \{C_1, ..., C_m\}$. $\mathcal{X}_i$ is an n-dimensional vector, $\mathcal{X}_i = (X_{i1}, X_{i2}, ..., X_{in})$, n is the number of attributes, and C_i is the class label. In this paper, we consider the binary classification and assume that the two classes are, 1 and -1. Then, for each attribute, we define two weights, one corresponding to the class $C_1 = 1$ and another to the class $C_2 = -1$. By considering two weights for each attribute, the attribute weighted NB classifies an instance $\mathcal{X}_i$ by selecting:

$$\arg \max_{1\leq k\leq 2} P(C_k) \prod_{j=1}^{n} P(X_{ij}|C_k)^{w_{jk}}. \tag{5}$$

In equation (5), there are two alternatives for k in w_{jk}. We denote these cases by w_j and $\overline{w}_j$ if $\mathcal{X}_i$ is allocated to the real class and its counterpart, respectively. Considering that C_k is the real class of $\mathcal{X}_i$, the value of $P(C_k|\mathcal{X}_i)$ is expected to be greater than the value of $P(\overline{C}_k|\mathcal{X}_i)$ for the majority of instances, $i = 1, ..., N$, where $\overline{C}_k = -C_k$. Then, it is quite natural that the value of

$$P(C_k) \prod_{j=1}^{n} P(X_{ij}|C_k)^{w_j} \tag{6}$$

should be maximized, while the value of

$$P(\overline{C}_k) \prod_{j=1}^{n} P(X_{ij}|\overline{C}_k)^{\overline{w}_j} \tag{7}$$

to be minimized. Therefore one possible objective function for the NB classifier, by considering the weights for attributes, can be written as follows:

$$\text{maximize } f(w) = \sum_{i=1}^{N} \frac{P(C_k) \prod_{j=1}^{n} P(X_{ij}|C_k)^{w_j} - P(\overline{C}_k) \prod_{j=1}^{n} P(X_{ij}|\overline{C}_k)^{\overline{w}_j}}{P(C_k) \prod_{j=1}^{n} P(X_{ij}|C_k)^{w_j} + P(\overline{C}_k) \prod_{j=1}^{n} P(X_{ij}|\overline{C}_k)^{\overline{w}_j}}, \tag{8}$$

where $w = (w_1, \overline{w}_1, w_2, \overline{w}_2, ..., w_n, \overline{w}_n)$ is a set of unknown variables (attribute weights). The objective function (8) is similar to the objective function presented in [23]. The weights in (8) are considered as positive numbers. Also, we put an upper limit for these weights to prevent large numbers. So, we maximize the above objective function over a hyper box $[a, b]$. Therefore, the problem (8) can be formulated as a constrained optimization problem:

$$\begin{aligned} &\text{minimize} \quad -f(w) \\ &\text{subject to} \quad w_i, \overline{w}_i \in [a, b], \ 1 \leq i \leq n. \end{aligned} \tag{9}$$

Different methods can be applied to transfer the problem (9) to an unconstrained optimization. One of the well-known methods is the penalty method, which is used here. To find the weights in (9), a local optimization method is applied, which here is the quasisecant method presented in Section 2. The NB classifier is taken as a starting point for the quasisecant method. More precisely, we initialize all the weights to unity, then we use the quasisecant method to find the attribute weights for further improvement. In other words, we search for an optimal classifier starting with the NB classifier. It is noted that a global optimization is also applicable to find the global solution of the problem (9), but the complexity of the problem will increase.

5 Experiments

5.1 Data collections

This paper studies 16 benchmark data sets taken from the literature. A brief description of the data sets is given in Table 1. The detailed description of the first eleven data sets used in this experiments can be found in the UCI repository of machine learning databases [1], and the last five data sets are downloadable on the tools page of the LIBSVM [3]. These data sets have been analyzed quite frequently by the current data mining approaches. Another reason for selecting these data sets were that conventional approaches have analyzed them with variable success.

5.2 Results and discussion

We conduct empirical comparison for the Naive Bayes (NB), the Tree Augmented Naive Bayes (TAN), the improved Naive Bayes (INB) proposed by Taheri et al. [22], and the attribute weighted Naive Bayes (AWNB) in terms of accuracy. The structure of the TAN and the INB are originated from the structure of the NB, in which each attribute has at most one augmenting edge pointing to it. The relations between attributes in the TAN are found by using the tree procedure [8], while the INB uses conditional probabilities for finding the correlations [22].

We discretize the values of continuous attributes in data sets using two different methods. In the first one, we apply Fayyad and Irani's discretization method [7]. The second one is the discretization algorithm SOAC [28].

For each method, we run 50 trials and then the average accuracy over the 50 runs are calculated. The accuracy of the methods in each run is calculated using 10-fold cross validation with random orders of data records in partitioning training and test data sets to have more reliable results. More precisely, each fold contained 10% of the data set randomly selected (without replacement). For consistent comparison, the same folds, including the same training and test data sets, are used in implementing the methods.

The penalty parameter is chosen as $\mu = 10^6$. We set the lower and upper limits in (9) as $a = 0.1, b = 10$.

Table 2 presents the average accuracy obtained by the NB, the TAN, the INB and the AWNB on 16 data sets, where continuous attributes are discretized by applying Fayyad and Irani's method [7]. The results presented in this table demonstrate that the accuracy of the proposed method (AWNB) is much better than that of the NB in all data sets. It is also shown a higher accuracy of the AWNB, in general, compared to the results obtained by the TAN and the INB. The proposed method outperforms the both methods (the TAN and the INB) in most of data sets, and the accuracy of this method slightly less or almost ties with the TAN and the INB in a few cases.

The results of the average accuracy obtained by the methods on 16 data sets using discretization algorithm SOAC are reported in Table 3. The results show that the accuracy obtained by the proposed method (AWNB) in all data sets are higher than those of the NB. The accuracy of the AWNB is also higher than those of the TAN and the INB in most of data sets, and the accuracy of the AWNB almost ties with those of the TAN and the INB in a few cases.

Figure 2 shows the scatter plots comparing the average miss-classifications of the proposed attribute weighted Naive Bayes, AWNB, with those of the NB, the TAN and the INB using two different discretization methods. In these plots, each point represents a data set, where the horizontal axis shows the percentage of miss-classifications according to the NB, the TAN and the INB and the vertical axis is the percentage of miss-classification according to the proposed method, AWNB. Therefore, points below the diagonal line correspond to data sets where the AWNB performs better, and points above the diagonal line correspond to data sets where the other mentioned methods perform better.

According to the results explained above, the proposed attribute weighted Naive Bayes, AWNB, works well in that it improves the results of the NB classifier. Moreover, in general, it outperforms the TAN and the INB so that it's accuracy in most of the data sets are higher than those of the the TAN and the INB. In a few cases, the TAN and the INB perform slightly better than the proposed method, and the results are acceptable as the two methods are also developments on the NB classifier.

The complexities of the methods are not compared in this work, since different softwares are used to implement the methods. The proposed method is coded in Matlab, while others are coded in Fortran. It is clear that the complexity of the proposed method is higher than the others due to the complexity of the optimization procedure. A global optimization is also applicable to determine the weights for the attributes. Although it may cause a better accuracy, a higher level of computational effort is required.

6 Conclusions

In this paper, we proposed a classifier based on attribute weighted Naive Bayes, AWNB. A novel weighting method for attribute weighted NB classifier was introduced, in which for each attribute we used more than one weight depending on the number of class labels. An objective function consisting of the attribute weights based on the structure of the NB classifier was then modeled to optimize the attribute weights. This objective function was optimized by a local optimization using the quasisecant method. The initial values in the quasisecant method were chosen as one; meaning that the NB classifier was taken as a starting point.

We carried out a number of experiments on some data sets obtained from the UCI machine learning repository and LIBSVM. The numerical results demonstrated that the proposed method has positive impact on the NB accuracy as expected. How this attribute weighting for the NB classifier performs in multi class data sets remains an important question for future work.

References

1. A. Asuncion, D. Newman, UCI machine learning repository. School of Information and Computer Science, University of California (2007) http://www.ics.uci.edu/mlearn/MLRepository.html.
2. A. M. Bagirov, A. Nazari Ganjehlou, A quasisecant method for minimizing nonsmooth functions. Optimization Methods and Software, Vol. 25, No. 1, 3-18 (2010)
3. C. Chang, C. Lin, LIBSVM: A library for support vector machines (2001) Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm.
4. D.M. Chickering, Learning Bayesian Networks is NP-complete, In: Fisher, D., Lenz, H. Learning from data: Artificial Intelligence and statistics V, Springer, 121-130 (1996)
5. P. Domingos, M. Pazzani, On the optimality of the simple Bayesian classifier under zero-one loss, Mach Learn 29, 103-130 (1997)

6. J. Dougherty, R. Kohavi, M. Sahami, Supervised and unsupervised discretization of continuous features, In Proceedings of the 12th International Conference on Machine Learning, 194-202 (1995)
7. U.M. Fayyad, K.B. Irani, On the Handling of Continuous-Valued Attributes in Decision Tree Generation, Machine Learning 8, 87-102 (1993)
8. N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, Machine Learning 29, 131-163 (1997)
9. M. Hall. A Decision Tree-Based Attribute weighting Filter for Naive Bayes, In Knowledge-Based Systems, Vol 20, 120-126 (2007)
10. D. Heckerman, D.M. Chickering, C. Meek, Large-Sample Learning of Bayesian Networks is NP-Hard, Journal of Machine Learning Research, 1287-1330 (2004)
11. L. Jiang, H. Zhang, Weightily Averaged One-Dependence Estimators, Proceedings of the 9th Biennial Pacific Rim International Conference on Artificial Intelligence, Guilin, China, 970-974 (2006)
12. L. Jiang, D. Wang, Z. Cai, X. Yan, Survey of improving naive Bayes for classification, In Proceedings of the 3rd International Conference on Advanced Data Mining and Applications, Springer, vol. 4632, 134-145 (2007)
13. E.J. Keogh, M.J. Pazzani, Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches, In: Proc. Int. Workshop on Artificial Intelligence and Statistics, 225-230 (1999)
14. R. Kohavi, Scaling up the accuracy of naive-Bayes classifiers: a decision-tree hybrid, In: Proc. 2nd ACM SIGKDD Int, Conf, Knowledge Discovery and Data Mining, 202-207 (1996)
15. P. Langley, W. Iba, K. Thompson, An Analysis of Bayesian Classifiers, In 10th International Conference Artificial Intelligence, AAAI Press and MIT Press, 223-228 (1992)
16. P. Langley, and S. Saga, Induction of selective Bayesian classifiers, In: Proc. Tenth Conf, Uncertainty in Artificial Intelligence, Morgan Kaufmann, 399-406 (1994)
17. J. Lu, Y. Yang, G. I. Webb, Incremental Discretization for Naive-Bayes Classifier, Springer, Heidelberg, Vol. 4093, 223-238 (2006)
18. S. Ozsen, S. Gunecs, Attribute weighting via genetic algorithms for attribute weighted artificial immune system (AWAIS) and its application to heart disease and liver disorders problems, Expert Systems with Applications, 36, 386-392 (2009)
19. M.J. Pazzani, Constructive induction of Cartesian product attributes, ISIS: In- formation, Statistics and Induction in Science, 66-77 (1996)
20. J. Pearl, Probabilistic Reasoning in Intelligent Systems: networks of plausible inference, Morgan Kaufmann (1988)
21. W. Sun, Y.X. Yuan, Optimization theory and methods: nonlinear programming, Springer (2006)
22. S. Taheri, M. Mammadov, A. M. Bagirov, Improving Naive Bayes Classifier Using Conditional Probabilities, In the proceedings of Ninth Australasian Data Mining Conference (AusDM 2011), Ballarat, Australia. Vol. 125, (2011)
23. S. Taheri, M. Mammadov, Tree Augmented Naive Bayes Based On Optimization, Proceeding of 42nd Annual Iranian Mathematics Conference, Vali-e-Asr University of Rafsanjan, Iran (2011)
24. S. Wang, F. Min, Z. Wang, T. Cao, OFFD: Optimal Flexible Frequency Discretization for Naive Bayes Classification, Springer, Heidelberg, 704-712 (2009)
25. . G. I. Webb, J. Boughton, Z. Wang, Not so naive bayes: Aggregating one dependence estimators, Machine Learning, Vol. 58, 5-24 (2005)
26. J. Wu, Z. Cai, Attribute Weighting via Differential Evolution Algorithm for Attribute Weighted Naive Bayes (WNB), Journal of Computational Information Systems 7:5, 1672-1679 (2011)
27. W. Xindong et al., Top 10 algorithms in data mining, Knowl Inf Syst, 14, 1-37 (2008)
28. A. Yatsko, A. M. Bagirov, A, Stranieri, On the Discretization of Continuous Features for Classification, School of Information Technology and Mathematical Sciences, University of Ballarat Conference, (2010)
(http://researchonline.ballarat.edu.au:8080/vital/access/manager/Repository)
29. Y. Ying, I. Geoffrey, Discretization For Naive-Bayes Learning: Managing Discretization Bias And Variance, In Machine Learning, 74(1), 39-74 (2009)

30. Y. Ying, Discretization for Naive-Bayes Learning, PhD thesis, school of Computer Science and Software Engineering of Monash University (2003)
31. H. Zhang, S. Sheng, Learning weighted naive Bayes with accurate ranking, In Proceeding of the 4th IEEE International Conference on Data Mining, 567-570 (2005)
32. F. Zheng, G.I. Webb, Semi-naive Bayesian Classification, Journal of Machine Learning Research (2008)
33. Y. Zhou T. S. Huang, Weighted Bayesian Network for Visual Tracking, Proceedings of the 18th International Conference on Pattern Recognition (ICPR'O6), 0-7695-2521-0106 (2006)

Table 1 A brief description of data sets

Data sets	# Instances	# Attributes
Breast Cancer	699	10
Congressional Voting Records	435	16
Credit Approval	690	15
Diabetes	768	8
Haberman's Survival	306	3
Heart Disease	303	14
Ionosphere	351	34
Liver Disorders	345	6
Phoneme CR	5404	5
Sonar	208	60
Spambase	4601	57
Fourclass	862	2
German.numer	1000	24
Splice	3175	60
Svmguide1	7089	4
Svmguide3	1284	21

Table 2 Test set accuracy averaged over 50 runs for data sets using Fayyad and Irani's discretization method. NB stands for Naive Bayes, TAN for Tree Augmented Naive Bayes, INB for improved Naive Bayes and AWNB for attribute weighted Naive Bayes

Data sets	NB	TAN	INB	AWNB
Breast Cancer	97.18	96.52	97.63	**97.74**
Congressional Voting Records	90.11	93.21	93.47	**94.24**
Credit Approval	86.10	84.78	86.72	**86.91**
Diabetes	74.56	75.14	**76.06**	75.98
Haberman's Survival	75.09	74.41	**77.03**	76.83
Heart Disease	82.93	81.23	83.36	**85.57**
Ionosphere	88.62	**89.77**	88.98	89.61
Liver Disorders	63.26	63.18	64.89	**65.79**
Phoneme CR	77.56	**78.31**	77.71	78.22
Sonar	76.32	**76.47**	76.41	76.39
Spambase	90.41	89.78	**92.87**	92.43
Fourclass	77.46	77.61	78.61	**78.91**
German.numer	74.50	73.13	75.91	**76.79**
Splice	95.43	94.87	**95.91**	95.88
Svmguide1	92.39	91.61	**94.04**	93.97
Svmguide3	81.23	82.47	84.98	**87.44**

Table 3 Test set accuracy averaged over 50 runs for data sets using discretization algorithm SOAC. NB stands for Naive Bayes, TAN for Tree Augmented Naive Bayes, INB for improved Naive Bayes and AWNB for attribute weighted Naive Bayes

Data Sets	NB	TAN	INB	AWNB
Breast Cancer	96.12	95.60	96.45	**96.56**
Congressional Voting Records	90.11	91.42	91.47	**94.52**
Credit Approval	85.85	84.98	**86.85**	86.79
Diabetes	75.78	75.90	**77.68**	77.53
Haberman's Survival	74.66	**76.08**	75.33	75.91
Heart Disease	78.62	77.37	79.31	**82.41**
Ionosphere	85.92	**86.18**	85.97	86.11
Liver Disorders	65.82	65.73	66.51	**66.85**
Phoneme CR	77.01	78.53	79.36	**79.65**
Sonar	75.09	75.76	**75.83**	75.69
Spambase	89.30	89.04	92.30	**92.44**
Fourclass	78.58	79.52	79.70	**79.76**
German.Numer	74.61	74.01	75.23	**75.81**
Splice	92.12	**93.04**	92.39	92.87
Svmguide1	95.61	94.91	**97.54**	97.43
Svmguide3	77.25	79.99	80.85	**81.23**

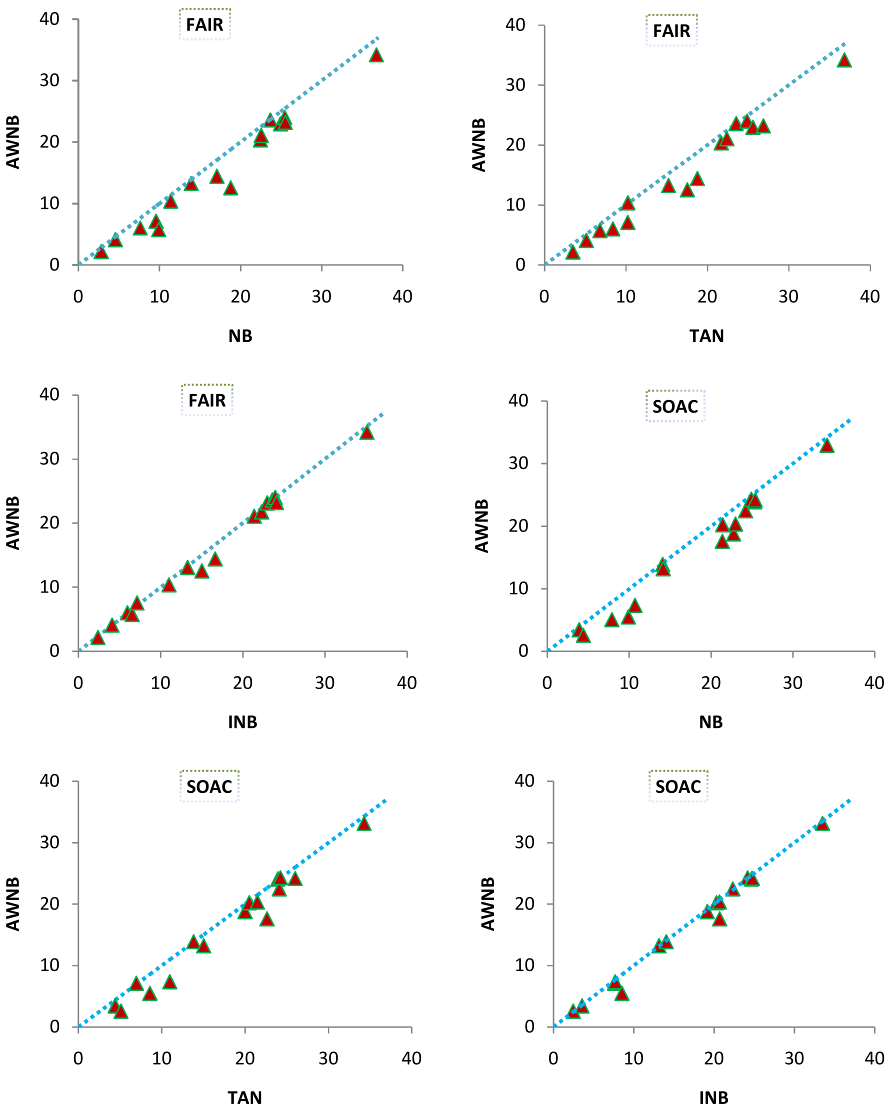


Fig. 2 Scatter plot comparing the average miss-classifications of the proposed method (AWNB) with Naive Bayes (NB), Tree Augmented Naive Bayes (TAN), Improved Naive Bayes (INB); Using Fayyad and Irani's discretization method (FAIR) and Sub-Optimal Agglomerative Clustering based Method (SOAC)

5.3 Tree Augmented Naive Bayes based on Optimization

In this section, a new optimization model for Tree Augmented Naive Bayes is introduced. The model is constructed by using an objective function depending on unknown variables for class distributions. We apply a local optimization method to find optimal values of theses variables. The efficiency of the proposed optimization model is verified by applying it to some real world data sets.

Paper:

Tree Augmented Naive Bayes Classifier based on Optimization

Published by:

42nd Annual Iranian Mathematics Conference,

Rafsanjan, Iran, pp. 594-597, 2011

Authors: Sona Taheri[a] and Musa Mammadov[a, b]

[a]Centre for Informatics and Applied Optimization,

School of Science, Information Technology and Engineering,

University of Ballarat, VIC 3353, Australia

[b]National ICT Australia, VRL, VIC 3010, Australia

Corresponding author:

Sona Taheri

e-mail: sonataheri@students.ballarat.edu.au

CS-T2

42nd Annual Iranian Mathematics Conference

5-8 September 2011, Vali-e-Asr University of Rafsanjan, Iran, pp 594-597

TREE AUGMENTED NAIVE BAYES BASED ON OPTIMIZATION

SONA TAHERI AND MUSA MAMMADOV[1]*

Abstract. Tree Augmented Naive Bayes is an efficient form of Bayesian Network. It is factored representations of probability distributions that generalize the Naive Bayes, yet at the same time maintains the robustness of Naive Bayes. In this paper, we propose a new optimization model for Tree Augmented Naive Bayes by considering some of class probabilities as unknown variables. Results of numerical experiments on 10 real world data sets demonstrate that the proposed model can significantly improve the classification accuracy of Tree Augmented Naive Bayes.

1. Introduction and Preliminaries

Bayesian Network (BN) [4] has long been a popular medium for graphically representing the probabilistic dependencies which exist in a domain. It is associated with a directed acycle graph, in which the nodes correspond to the variables in the domain and causal relationships are denoted by an arrow, called an edge. When two nodes are joined by an edge, the causal node is called the parent of the other node. Every node has a conditional probability table that describes the relationship between it and its parents. A BN for a set of variables $\mathbf{X} = \{X_1, X_2, ..., X_n\}$ consist of a structure B that encodes a set of conditional independency assertions about variables in $\mathbf{X}$, and a set of probability distributions associated with each variable. All together, these

2000 *Mathematics Subject Classification.* Primary 00X00; Secondary 00X00, 00X00.
Key words and phrases. Bayesian Network, Tree Augmented Naive Bayes, Optimization.
* Sona Taheri.

components define the joint probability distribution for $\mathbf{X}$. In particular, given structure B, the joint probability distribution for $\mathbf{X}$ is given by

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|Pa(X_i)),$$

where X_i and $Pa(X_i)$ stand for the variable and its parents, respectively.

One of the most effective and simple form of BN is Naive Bayes (NB). NB [2] assumes that all features have only the class as a parent. Unlike NB, Tree Augmented Naive Bayes (TAN) [1] allows additional edges between features that capture correlations among them. In fact each feature has the class and at most one other feature as parents. Friedman et al. [1], showed that TAN maintains the robustness of NB, and at the same time displays better accuracy. TAN approximates the dependency between features by using a tree structure imposed on NB structure [1].

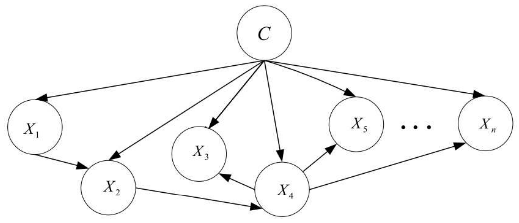


FIGURE 1. Tree Augmented Naive Bayes

2. Optimization Model for Tree Augmented Naive Bayes

In this section we present an optimization model for TAN. Consider data set $D = \{(\mathbf{X}_1, C_1), (\mathbf{X}_2, C_2), ..., (\mathbf{X}_N, C_N)\}$, where N is the number of instances. Each $\mathbf{X}_i$ is a n-dimensional vector: $\mathbf{X}_i = (X_{i1}, X_{i2}, ..., X_{in})$, and n is the number of features. We consider binary classification, that is $C_i \in \{-1, 1\}$, $i = 1, ..., N$. We introduce $\overline{C} = -C$.

Considering that C_i is the class of $\mathbf{X}_i$, the value of $P(C_i|\mathbf{X}_i)$ is expected to be greater than the value of $P(\overline{C_i}|\mathbf{X}_i)$ for the majority of instances, $i = 1, ..., N$.

From the structure of TAN, we have

$$P(X_i|C_i) = \prod_{j=1}^{n} P(X_{ij}|C_i, Pa(X_{ij})) = \prod_{j=1}^{n} \Big(\frac{P(X_{ij}, C_i, Pa(X_{ij}))}{P(C_i, Pa(X_{ij}))}\Big). \tag{2.1}$$

From the Bayes rule and (2.1) we have

$$P(C_i|X_i) = \frac{\prod_{j=1}^{n} \Big(\frac{P(X_{ij},C_i,Pa(X_{ij}))}{P(C_i,Pa(X_{ij}))}\Big) P(C_i)}{\prod_{j=1}^{n} \Big(\frac{P(X_{ij},C_i,Pa(X_{ij}))}{P(C_i,Pa(X_{ij}))}\Big) P(C_i) + \prod_{j=1}^{n} \Big(\frac{P(X_{ij},\overline{C}_i,Pa(X_{ij}))}{P(\overline{C}_i,Pa(X_{ij}))}\Big) P(\overline{C}_i)}. \tag{2.2}$$

Similarly

$$P(\overline{C}_i|X_i) = \frac{\prod_{j=1}^{n}\left(\frac{P(X_{ij},\overline{C}_i,Pa(X_{ij}))}{P(\overline{C}_i,Pa(X_{ij}))}\right)P(\overline{C}_i)}{\prod_{j=1}^{n}\left(\frac{P(X_{ij},C_i,Pa(X_{ij}))}{P(C_i,Pa(X_{ij}))}\right)P(C_i)+\prod_{j=1}^{n}\left(\frac{P(X_{ij},\overline{C}_i,Pa(X_{ij}))}{P(\overline{C}_i,Pa(X_{ij}))}\right)P(\overline{C}_i)}, \tag{2.3}$$

where $P(C_i)+P(\overline{C_i}) = 1,\ 0 \le P(C_i), P(\overline{C_i}) \le 1$. We introduce two variables w_1, w_2 that will stand for probabilities $P(C_i), P(\overline{C}_i)$ in (2.2) and (2.3). We can write (2.2) and (2.3) as

$$f_1(w_1,w_2) = \sum_{i=1}^{N}\frac{\prod_{j=1}^{n}\left(\frac{P(X_{ij},C_i,Pa(X_{ij}))}{P(C_i,Pa(X_{ij}))}\right)\xi(w_1,w_2;C_i)}{\prod_{j=1}^{n}\left(\frac{P(X_{ij},C_i,Pa(X_{ij}))}{P(C_i,Pa(X_{ij}))}\right)\xi(w_1,w_2;C_i)+\prod_{j=1}^{n}\left(\frac{P(X_{ij},\overline{C}_i,Pa(X_{ij}))}{P(\overline{C}_i,Pa(X_{ij}))}\right)\xi(w_1,w_2;\overline{C}_i)},$$

and

$$f_2(w_1,w_2) = \sum_{i=1}^{N}\frac{\prod_{j=1}^{n}\left(\frac{P(X_{ij},\overline{C}_i,Pa(X_{ij}))}{P(\overline{C}_i,Pa(X_{ij}))}\right)\xi(w_1,w_2;\overline{C}_i)}{\prod_{j=1}^{n}\left(\frac{P(X_{ij},C_i,Pa(X_{ij}))}{P(C_i,Pa(X_{ij}))}\right)\xi(w_1,w_2;C_i)+\prod_{j=1}^{n}\left(\frac{P(X_{ij},\overline{C}_i,Pa(X_{ij}))}{P(\overline{C}_i,Pa(X_{ij}))}\right)\xi(w_1,w_2;\overline{C}_i)},$$

where $\xi(w_1,w_2;C) = w_1$ if $C = 1$ and $\xi(w_1,w_2;C) = w_2$ if $C = -1$.

It is quite natural that the value of $f_1(w_1,w_2)$ should be maximized, while the value of $f_2(w_1,w_2)$ to be minimized. Therefore TAN leads to a multi-criterion optimization problem:

$$maximize:\ f(w_1,w_2) \doteq \Big(f_1(w_1,w_2), -f_2(w_1,w_2)\Big)$$

$$subject\ to\ w_1+w_2 = 1,\ 0 \le w_1,w_2 \le 1. \tag{2.4}$$

In this paper, instead of multi-criterion optimization problem (2.4), we consider a single criterion maximization problem by taking $\psi(w_1,w_2) = f_1(w_1,w_2) - f_2(w_1,w_2)$:

$maximize:\ \psi(w_1,w_2) =$

$$\sum_{i=1}^{N}\frac{\prod_{j=1}^{n}\left(\frac{P(X_{ij},C_i,Pa(X_{ij}))}{P(C_i,Pa(X_{ij}))}\right)\xi(w_1,w_2;C_i)-\prod_{j=1}^{n}\left(\frac{P(X_{ij},\overline{C}_i,Pa(X_{ij}))}{P(\overline{C}_i,Pa(X_{ij}))}\right)\xi(w_1,w_2;\overline{C}_i)}{\prod_{j=1}^{n}\left(\frac{P(X_{ij},C_i,Pa(X_{ij}))}{P(C_i,Pa(X_{ij}))}\right)\xi(w_1,w_2;C_i)+\prod_{j=1}^{n}\left(\frac{P(X_{ij},\overline{C}_i,Pa(X_{ij}))}{P(\overline{C}_i,Pa(X_{ij}))}\right)\xi(w_1,w_2;\overline{C}_i)},$$

$$subject\ to\ w_1+w_2 = 1,\ 0 \le w_1,w_2 \le 1. \tag{2.5}$$

The problem (2.5) is a nonlinear and nonconvex constrained optimization problem. We apply the penalty method to reduce this problem to an unconstrained one. For solving the unconstrained optimization problem, we use the globally convergent optimization method [3] which is a combination of the Gradient method and the Newton's method.

Table 1. Test set accuracy

Data Sets	# Instances	# Features	TAN	Optimization Model
Breast Cancer	683	10	96.09	96.82
Congressional Voting	435	16	93.50	94.21
Credit Approval	690	14	84.89	86.61
Chess	3196	36	92.40	93.37
Diabetes	768	8	75.38	76.41
German	1000	24	73.82	75.61
Heart Disease	297	13	83.14	84.13
Hepatitis	155	19	91.04	91.89
Ionosphere	351	34	85.74	86.32
Pima	768	8	75.61	76.11

The efficiency of the proposed optimization model was verified by applying it to 10 real world data sets taken from UCI machine learning repository.

We conduct empirical comparison for TAN and the proposed model in terms of test set accuracy via 1 run of 10-fold cross validation. The experimental results of comparing test set accuracy obtained by the proposed optimization model and TAN are presented in Table 1. The results from this table show that the accuracy obtained by the proposed model in all data sets are higher than those obtained by TAN.

References

[1] Friedman, N., Geiger, D., and Goldszmidt, M., *Bayesian Network Classifiers*, Machine Learning 29 (2). 131-163, 1997.

[2] Langley, P., Iba, W., Thompson, K., *An Analysis of Bayesian Classifiers.*, In 10th International Conference Artificial Intelligence. 223-228, 1992.

[3] Mammadov, M., Taheri, S., *A Globally Convergent Optimization Algorithm for Systems of Nonlinear Equations. In Proceedings of the Third Global Conference on Power Control And Optimization.*, 2-4 Febrauary, 2010, Gold Coast, Australia., 2010.

[4] Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.*, Morgan Kaufmann, 1988.

[1] Center for Informatics and Applied Optimization, School of Information Technology and Mathematical Sciences, University of Ballarat, Victoria 3353, Australia.

E-mail address: sonataheri@students.ballarat.edu.au, m.mammadov@ballarat.edu.au

5.4 Structure Learning of Bayesian Networks using Global Optimization

The focus of this section is development of structure learning algorithm for Bayesian Networks that produces actionable models where the structure of the model accurately captures the causal relationships in the data. We apply optimization techniques to have the best acyclic structure in a Bayesian Network. Empirical evaluations on several well known real world data sets are performed to evaluate the effectiveness of the proposed algorithm. The computational results indicate that this algorithm is robust and accurate for classification problems.

Paper:

Structure Learning of Bayesian Networks using Global Optimization

Submitted to:

IMA Journal of Applied Mathematics, October 2012

Authors: Sona Taheri[a] and Musa Mammadov[a, b]

[a]Centre for Informatics and Applied Optimization,

School of Science, Information Technology and Engineering,

University of Ballarat, VIC 3353, Australia

[b]National ICT Australia, VRL, VIC 3010, Australia

Corresponding author:

Sona Taheri

e-mail: sonataheri@students.ballarat.edu.au

IMA Journal of Applied Mathematics (2012) Page 1 of 20
doi: 10.1093/imanum/dri017

Structure Learning of Bayesian Networks using Global Optimization

SONA TAHERI[1] AND MUSA MAMMADOV[2]
[1]*Centre for Informatics and Applied Optimization*
School of Science, Information Technology and Engineering
University of Ballarat, VIC 3353, Australia
[1,2]*National ICT Australia, VRL, VIC 3010, Australia*

[Received on October 2012]

Bayesian Networks are increasingly popular methods of modeling uncertainty in artificial intelligence and machine learning. A Bayesian Network consists of a directed acyclic graph in which each node represents a variable and each arc represents probabilistic dependency between two variables. Constructing a Bayesian Network from data is a learning process that consists of two steps: learning structure and learning parameter. Learning a network structure from data is the most difficult task in this process. This paper presents a new algorithm for constructing better structures based on optimization techniques. The main advantage of the proposed method is that the maximal number of parents for variables is not fixed a priory, it is defined during the optimization procedure. The efficiency of the proposed algorithm is demonstrated on several real world data sets taken from the UCI machine learning repository and the LIBSVM, where continuous features are discretized by applying three different methods. Several closely related algorithms, as well as, benchmarks algorithms SVM and C4.5 are employed for comparison.

Keywords: Data Classification, Bayesian Networks, Global Optimization, Discretization

1. Introduction

Classification is a basic task in data mining that requires the construction of a classifier, that is, a function which assigns a class label to observations described by a set of feature variables. Learning accurate classifiers from preclassified data is a very active research topic in machine learning and artificial intelligence. One of the most effective classifiers is Bayesian Networks.

Bayesian Networks (BNs) are widely used representation frameworks for reasoning with probabilistic information Castillo *et al.* (1997); Heckerman D *et al.* (1995); Jensen1 (1996); Pearl (1988); Shafer & Pearl (1990). These models use graphs to capture dependence and independence relationship between feature variables, allowing a concise representation of the knowledge as well as efficient graph based query processing algorithms. This representation is defined by two components: structure learning and parameter learning. The structure of this model represents a directed acyclic graph. The nodes in the graph correspond to the feature variables in the domain, and the arcs (edges) show the causal relationships between feature variables. A directed edge relates the variables so that the variable corresponding to the terminal node (child) will be conditioned on the variable corresponding to the initial node (parent). More incoming edges into a node result in a conditional probability of the corresponding variable with conjunctional condition containing all its parents. The parameter learning represents probabilities and conditional probabilities based on prior information or past experience. The set of probabilities are represented in the conditional probability table (CPT). For each parent and each possible state of that parent, there is a row in the CPT that describes the likelihood that the child node will be in some state. Nodes with no parents also have CPT, but they are simpler and consist only of the probabilities for each state of the node under consideration. Once the network structure is constructed, the probabilistic inferences are readily calculated, and can be performed to predict the outcome of some variables based on observations of others. However, the problem of structure learning is a much more complex problem since the number of candidate structures grows exponentially when the number of features increases Robinson (1997).

In recent years, the search for the structure of a BN able to reflect all existing relations of dependence in a data base has constituted a research topic of fundamental importance. Given a set of features and a data set composed of all features, the problem is to build a structure to present the connections among the features. This structure learning process needs to select the arcs between them, and therefore construct a network from data. Developing a structure is very useful for a variety of applications in general, for example, where there are masses of data available and we want to understand what underlies the knowledge or which features are correlated. In addition to providing a network that will allow us to predict behavior under conditions that we have not seen, the structure can also incorporate domain expert knowledge to provide more reliable suggestions. Nevertheless, there still remains the problem of building such a network structure. It is an important task, therefore, to develop some methods capable of learning a network structure directly from data.

Nowadays, the problem of learning structure of a BN from data is receiving increasing attention within the community of researchers into uncertainty in artificial intelligence and machine learning. Learning such a structure is a global optimization problem. Various optimization problems for finding a structure of a BN have been defined Campos *et al.* (2002); Daly & Shen (2009); Janzura &Nielsen (2006); Kabli *et al.* (2007); Larranaga *et al.* (1996, 1997, 1996); Marinescu & Dechter (2009); Park & Cho (2006); Sahin *et al.* (2007); Schleip *et al.* (2009); Taheri & Mammadov (2012); Zhao *et al.* (2009); Ji *et al.* (2009). Park and Cho Park & Cho (2006) used the Genetic algorithm to optimize the structure in BNs and make a model more efficient. A method of finding the most probable structure of BNs based on the intelligent search made by the Genetic algorithm has been introduced by Larranaga et al. Larranaga *et al.* (1997). The paper Larranaga *et al.* (1996) presents new approaches based on the Genetic algorithm to find the best BNs' structure among alternative structures. Work by Kabli et al. Kabli *et al.* (2007) illustrates a novel method for finding the structure of BNs using the Genetic algorithm. The Simulated Annealing approaches to learn the best structure in BNs have been studied, for example, in Janzura &Nielsen (2006); Schleip *et al.* (2009). Application of the Particle Swarm optimization to discover the best structure of BNs is studied in Sahin *et al.* (2007); Zhao *et al.* (2009). Daly and Shen Daly & Shen (2009) applied the Ant Colony optimization to the problem of learning a BN structure that provides a good fit to a set of data. The papers Campos *et al.* (2002); Ji *et al.* (2009) propose BNs' structure learning algorithms based on the Ant Colony optimization. In Marinescu & Dechter (2009), the Branch and Bound method is applied for constructing a structure in a BN. More recently, we have introduced the unrestricted dependency Bayesian Networks algorithm based on the combinatorial optimization Taheri & Mammadov (2012).

In this paper, we propose a new algorithm to learn an optimal structure of a BN by extending the idea considered in Taheri & Mammadov (2012). The proposed algorithm is based on optimization approaches. It has two main phases; first we define an optimization model similar to the one introduced in Taheri & Mammadov (2012) to find the most probable networks. Then, we apply optimization techniques to delete the possible cycles to have an acyclic network structure. We apply two different methods for the second phase. The first one is a simple case when we have a small number of cycles. In this case, we chose an optimal network from the all possible combinations of deleting some arcs in the existing cycles. In the second case, when we have a large number of cycles, we apply the global optimization algorithm AGOP introduced in Mammadov *et al.* (2004, 2005) in combination with the recently developed local optimization algorithm CGN Taheri & Mammadov (2012); Taheri *et al.* (2012).

The paper is structured as follows: we begin in Section 2 with the preliminaries, where we briefly describe the concepts and methods related to BNs, discretization, the global and local optimization methods. In Section 3, we develop a new algorithm based on optimization for structure learning in BNs. In Section 4, we present some experimental results to compare the proposed algorithm with some well-known classification methods. Finally, Section 5 contains the concluding remarks.

2. Preliminaries

In this section, we briefly review some basic concepts related to BNs, as well as other concepts about discretization and optimization.

2.1 *Bayesian Networks*

A BN is a directed acyclic graph containing nodes and edges and a set of conditional probability distributions. Suppose a set of variables $\mathbf{X} = \{X_1, X_2, ..., X_n\}$, where X_i denotes both the feature variable and its corresponding node. Let $Pa(X_i)$ stand for the set of parents of the node X_i as well as the feature variables corresponding to those parents. When there is an edge from X_i to X_j, then X_j is called the child variable for the parent variable X_i. A conditional dependency connects a child variable with a set of parent variables. The lack of possible edges encode conditional independencies.

Throughout this paper, we will refer to the collection of edges (arcs), the conditional dependence and independence relations among the variables, as the structure of BNs. In particular, given a structure, the joint probability distribution for $\mathbf{X}$ is given by

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|Pa(X_i)). \tag{2.1}$$

However, accurate estimation of $P(X_i|Pa(X_i))$ is non trivial. Finding such an estimation requires searching the space of all possible network structures for one that best describes the data. Structure learning algorithms determine for every possible edge in the network whether to include the edge in the final network and which direction to orient the edge. The number of possible graph structures grows exponentially as every possible subset of edges could represent the final model. Due to this exponential growth in graph structures, even a restricted form of structure learning has been proven to be an NP-hard problem Chickering (1996); Heckerman *et al.* (2004).

k-dependence BNs introduced by Sahami Sahami (1996) is one of the restricted models in BNs. In this algorithm, each feature variable could have a maximum of k feature variables as parents, and these parents are obtained by using mutual information. The value of k is initially chosen before applying the k-dependence BNs, $k = 0, 1, 2, ...$ Naive Bayes (NB) Langley *et al.* (1992) is a very simple form of this algorithm when $k = 0$. In the NB, feature variables are conditionally independent given the class. Although the NB is a very efficient method on a variety of data mining problems, the strong assumption that all features are conditionally independent given the class is often violated on many real world applications. Friedman et al. Friedman *et al.* (1997) introduced Tree Augment Naive Bayes (TAN). The TAN is a special form of the k-dependence BNs when $k = 1$. In the TAN, each feature variable has the class and at most one other feature variable as parents.

Although the mentioned methods were shown to be efficient, the features in these methods depend on the class and a priori given number of features; $k = 0$ dependence for the NB, $k = 1$ dependence for the TAN, and an priory given k for the k-dependence algorithm. In fact, by setting k, i.e., the maximum number of parent nodes that any feature may have, the best structure of BNs have been constructed. Since k is the same for all nodes, it is not possible to model cases where some nodes have a large number of dependencies, whereas others just have a few. In the paper Taheri & Mammadov (2012), we have developed an unrestricted dependency algorithm, where the number k is defined by the algorithm internally. Although this algorithm performs well and the results are promising, it does not involve all possible networks. The algorithm considers only acyclic networks and choose a network structure with the maximum training accuracy. However, there might be a cyclic network that results an optimal solution. In the present paper, we address this challenge by developing a new algorithm based on optimization approaches. The aim of optimization is to remove some edges in the cyclic networks to obtain acyclic ones. The best structure is a network with the highest accuracy among all proposed networks.

2.2 *Detection of Cycles in a Directed Graph*

There are many algorithms that can be applied to detect cycles in a directed graph. One algorithm to detect the presence of cycles in a directed graph is the topological traversal algorithm Bender *et al.* (2009); Haeupler *et al.* (2008). Another one which determines also edges of cycles, is the DFS algorithm Sedgewick (1983); Tucker (2006).

The degree of a node in a graph is the number of connections or edges the node has with other nodes. If a graph is directed, meaning that edges point in one direction from one node to another node. Then each node has two different degrees, the in-degree, which is the number of incoming edges to this node, and the out-degree, which is the number of outgoing edges from this edge. The topological traversal algorithm begins by computing the in-degrees of the nodes. At each step of the traversal, a node with in-degree of zero is visited. After a node is visited, the node and all the edges emanating from that node are removed from the graph, reducing the in-degree of adjacent nodes. This is done until the graph is empty, or no node without incoming edges exists. The presence of the cycle prevents the topological order traversal from completing. Therefore, the simple way to test whether a directed graph is cyclic is to attempt a topological traversal of its nodes. If all nodes are not visited, the graph must be cyclic.

In the DFS algorithm, all nodes are initially marked white. When a node is encountered, it is marked grey, and when its descendants are completely visited, it is marked black. If a grey node is ever encountered, then there is a cycle.

2.3 *Fayyad and Irani's Discretization Method*

In order to apply the BNs to data sets with continuous features, one should first discretize those features. Discretization is a process which transform continuous numeric values into discrete ones. In this paper, we apply three different methods to discretize continuous features. The first one, which is also the simplest one, transforms the values of features to $\{0,1\}$ using their mean values. We also apply two other methods which allows us to get more than two values for discretized features.

In this section, we give a brief description of the Fayyad and Irani's Discretization method Fayyad & Irani (1993) which is the most applied discretization method in the literature. The Fayyad and Irani's Discretization method is based on a minimal entropy heuristic, and it uses the class information entropy of candidate partitions to select bin boundaries for discretization.

Let us consider a given set of observations S, a feature X, and a partition boundary $\overline{T}$, the class information entropy of the partition induced by $\overline{T}$, denoted $E(X,\overline{T};S)$ is given by

$$E(X,\overline{T};S) = \frac{|S_1|}{|S|}Ent(S_1) + \frac{|S_2|}{|S|}Ent(S_2),$$

where $S_1 \subset S$ be the subset of observations in S with X-values not exceeding $\overline{T}$ and $S_2 = S - S_1$. Let there be q classes $C_1,...,C_q$, and $P(C_i,S)$ be the proportion of observations in S that have class C_i . The class entropy of a subset S is defined as:

$$Ent(S) = -\sum_{i=1}^{q} P(C_i,S)\lg(P(C_i,S)),$$

where the logarithm may be to any convenient base. When the base is 2, $Ent(S)$ measures the amount of information needed, in bits, to specify the classes in S.

For a given feature X, the boundary $\overline{T}_{min}$ which minimizes the entropy function over all possible partition boundaries is selected as a binary discretization boundary. This method can then be applied recursively to both of the partitions induced by $\overline{T}_{min}$ until some stopping condition is achieved, thus creating multiple intervals on the feature X.

Fayyad and Irani make use of the minimal description length principle to determine a stopping criteria for their recursive discretization strategy. Recursive partitioning within a set of values S stops if

$$Gain(X,\overline{T};S) < \frac{\lg_2(N-1)}{N} + \frac{\triangle(X,\overline{T};S)}{N},$$

where N is the number of observations in the set S, and

$$Gain(X,\overline{T};S) = Ent(S) - E(X,\overline{T};S),$$

$$\triangle(X,\overline{T};S) = \lg_2(3^q - 2) - [q.Ent(s) - q_1.Ent(S_1) - q_2Ent(S_2)],$$

and q_i is the number of class labels represented in the set S_i. Since the partitions along each branch of the recursive discretization are evaluated independently using this criteria, some areas in the continuous spaces will be partitioned very finely whereas others (which have relatively low entropy) will be partitioned coarsely.

2.4 *Discretization Algorithm Using Sub-Optimal Agglomerative Clustering (SOAC)*

In this section, we give a brief description of the discretization algorithm SOAC which is recently introduced in Yatsko *et al.* (2011).

Let consider a finite set of points A in the n-dimensional space $\mathbb{R}^n$, that is

$$A = \{a^1,...,a^m\},$$

where $a^i \in \mathbb{R}^n,\ i = 1,...,m.$ Assume the sets $A^j,\ j = 1,...,k$ be clusters, and each cluster A^j can be identified by its centroid $x^j \in \mathbb{R}^n,\ j = 1,...,k$. Algorithm SOAC proceeds as follows:

Algorithm 2.1 **(Discretization Algorithm SOAC)**

Step 1. Set $k = m$, and a small value of parameter θ, $0 < \theta < 1$. Sort values of the current feature in the ascending order. Each feature requiring discretization is treated in turn.

Step 2. Calculate the center of each cluster:

$$x^j = \sum_{a \in A^j} \frac{a}{|A^j|}, \; j = 1, ..., k,$$

and the error E_k of the cluster system approximating set A:

$$E_k = \sum_{j=1}^{k} \sum_{a \in A^j} \|x^j - a\|^2.$$

Step 3. Merge in turn each cluster with the next tentatively. Calculate the error increase $E_{k-1} - E_k$ after each merge and choose the pair of clusters giving the least increase. Merge these two clusters permanently. Set $k = k - 1$.

Step 4. If $E_k \geqslant \theta E_1$, then stop, otherwise go to Step 2.

2.5 *The Global Optimization Algorithm - AGOP*

In this subsection, we present the global optimization algorithm AGOP Mammadov *et al.* (2004, 2005) implemented in the open software library GANSO. This global optimization algorithm is designed for solving optimization problems with defined box constraints. It uses a line search mechanism where the descent direction is obtained via a dynamic systems approach. It is applicable to a wide range of optimization problems requiring only function evaluations to work. The efficiency of the algorithm has been demonstrated in solving many difficult practical problems where objective functions were discontinuous Kouhbor *et al.* (2006); Mammadov & Orsi (2005) and even piecewise constant Tilakaratne *et al.* (2009). Some variations of this method are suggested in Maroosi & Amiri (2010); Richter & Fettweis (2012).

Consider the problem of maximization the function $f(x): \mathbb{R}^n \to \mathbb{R}$ over the box $x \in \mathbb{B}$. Briefly, the global optimization procedure in AGOP is performed as follows. First a set of some initial points $\Omega = \{x(t) \in \mathbb{B}, t = 1, 2, 3, ..., T\}$ is constructed from the edge points of box $\mathbb{B}$. Suppose that $x^* \in \Omega$ is the best of the points in Ω; that is, $f(x^*) \geqslant f(x): \forall x \in \Omega$. The set Ω and the values of f at each of the points in Ω are used to generate a dynamical system. This dynamical system determines a possible search direction v at the point x^* . See for details Mammadov *et al.* (2004, 2005). In the next step, a line search performed as

$$\hat{x}(T+1) = x^* + \alpha^* v \tag{2.2}$$

where the step α^* is calculated by taking small step size $\varepsilon > 0$ as follows

$$\alpha^* = \arg\max_{\alpha} \{f(x^* + \alpha v): \; \alpha = k\varepsilon, k = 1, 2, ..., x^* + \alpha v \in \mathbb{B}\}. \tag{2.3}$$

A local search about $\hat{x}(T+1)$ is then carried out. This is done using a direct search method called local variation. This is an efficient local optimization technique that does not explicitly use derivatives and can be applied to a wide range of functions. A good survey of direct search methods can be found in Kolda *et al.* (2003). Letting $x(T+1)$ denote the optimal solution of this local search, the set Ω is augmented to include $x(T+1)$. Starting with this updated set Ω, the whole process can be repeated. The process is terminated when v is approximately 0 or a prescribed bound on the number of iterations is reached. The solution returned is the current x^*; that is, the point in Ω with the highest objective function value. If the objective function f is continuously differentiable then this solution will be local maxima.

2.6 *The Local Optimization Algorithm - CGN*

In this section, we present the recently introduced optimization algorithm, a combination of the gradient and Newton methods (CGN) Taheri & Mammadov (2012); Taheri *et al.* (2012). The CGN is a new globally convergent optimization algorithm for solving systems of nonlinear equations and unconstrained optimization problems. The idea in this algorithm is combining two directions from different local optimization methods. The first direction is the gradient direction due to its global convergence property. The second one is the Newton direction to improve the convergence rate. Two different combinations are considered in this algorithm. The first one is a novel combination in which the step length is determined only along the gradient direction. In the second one, the step length is considered along both directions.

Consider the problem of minimization the function $f(x)$ where $x \in \mathbb{R}^n$. Let $g(x) = \nabla f(x)$ and $H(x) = \nabla^2 f(x)$ be correspondingly the gradient and the Hessian matrix of the function f. We denote the gradient direction at x_k by $d_{1,k}$ and let $d_{2,k}$ stand for the Newton direction at x_k. The steps of the combined methods are presented in Algorithm 2.2.

Algorithm 2.2 **(Local optimization algorithm - CGN)**

Initialization. Select a starting point $x_0 \in \mathbb{R}^n$, and a tolerance $\varepsilon > 0$, η and δ be small positive numbers and $\vartheta > 1$, ω and L are two fixed numbers. Set k:=0.

Step 1. If $\|g(x_k)\| < \varepsilon$, then stop.

Step 2. Compute direction $d_{1,k}$ at x_k, $d_{1,k} = -g_k$.

Step 3. Compute the Newton direction $d_{2,k}$ at x_k. If direction $d_{2,k}$ at x_k is not computable, then go to Step 5.

Step 4. If $d_{2,k}^T d_{1,k} \geqslant 0$, go to Step 6.

Step 5. Use Wolfe-Powell rules (Sun & Yuan (2006)) to determine a step length $\alpha_k > 0$ along the direction $d_k = d_{1,k}$, set $s_k = \alpha_k d_k$ and go to Step 10.

Step 6. Set $j = 0,\ \eta_0 = \eta$.

Step 7. Compute ξ_k as follows:

$$\xi_k = \begin{cases} \frac{1}{1+\eta_j \|g_0\|} & \text{if } k = 0, \\ \\ \frac{1}{1+\eta_j |f_k - f_{k-1}|} & \text{if } k > 0, \end{cases} \tag{2.4}$$

and set $d(\xi_k) = (1-\xi_k)d_{1,k} + \xi_k d_{2,k}$.

Step 8. If $d(\xi_k)^T d_{1,k} < \delta\|d(\xi_k)\|\|d_{1,k}\|$, set $\eta_{j+1} = \vartheta\eta_j$ and $j = j+1$, and go to Step 7.

Step 9. Compute s_k using one of the following two approaches:

9.1. Use Wolfe-Powell rules (Sun & Yuan (2006)) to determine a step length $\alpha_k > 0$ along the direction $d_{1,k}$ and set $\bar{s}_k = \alpha_k(1-\xi_k)d_{1,k} + \xi_k d_{2,k}$. If $f(x_k+\bar{s}_k) \leqslant f(x_k) - \omega\|\bar{s}_k\|$ and $\alpha_k\|d_{1,k}\| \leqslant \varpi\|d_{2,k}\|$, set $s_k = \bar{s}_k$; otherwise set $s_k = \alpha_k d_{1,k}$.

9.2. Use Wolfe-Powell rules (Sun & Yuan (2006)) to determine a step length $\alpha_k > 0$ along the direction $d_k = d(\xi_k)$ and set $s_k = \alpha_k d_k$.

Step 10. Set $x_{k+1} = x_k + s_k,\ k := k+1$ and go to Step 1.

In this algorithm, when the slope of the function is slight, the algorithm tends to the Newton method, otherwise the contribution of the gradient method is increased. Global convergence as well as superlinear convergence rate of this algorithm are established under some conditions in Taheri & Mammadov (2012); Taheri *et al.* (2012).

3. A New Algorithm for Structure Learning in Bayesian Networks

In this section, we propose a new algorithm to learn an optimal structure of a BN. Since the learning process in BNs is based on the correlations of children and parent nodes, we propose a combinatorial optimization model to find the dependencies between features. However, some features could be independent which is considered by intruding a threshold K. We consider the following optimization model

$$\begin{gathered}\text{Maximize } \sum_{\substack{i=1\\ j\neq i}}^{n}\sum_{j=1}^{n}(K_{ij}-K)w_{ij},\\ \text{s.t. } w_{ij}\in\{0,1\},\ \ w_{ij}+w_{ji}\leqslant 1,\ \ 1\leqslant i,j\leqslant n,\ \ i<j.\end{gathered} \tag{3.1}$$

Given $1\leqslant i,j\leqslant n,\ \ i\neq j$, the value w_{ij} is the association weight (to be found), defined by

$$w_{ij} = \begin{cases} 1 & \text{if feature } X_i \text{ is the parent of feature } X_j,\\ 0 & \text{otherwise.}\end{cases} \tag{3.2}$$

and

$$K_{ij} = \sum_{v_2=1}^{|X_j|}\sum_{v_1=1}^{|X_i|}\max\{P(X_{v_2 j}|C_1,X_{v_1 i}),P(X_{v_2 j}|C_2,X_{v_1 i}),\ldots,P(X_{v_2 j}|C_q,X_{q_1 i})\}. \tag{3.3}$$

Here, $|X_j|$ and $|X_i|$ are the number of values of features X_j and X_i, respectively, and X_{vl} shows the vth value of feature $X_l,\ 1\leqslant l\leqslant n$. $C_1,C_2,\ldots,C_q$ are class labels. K is a threshold such that $K\geqslant 0$.

From formula (3.1), $w_{ij}=1$ if $K_{ij}>K_{ji}$ and $K_{ij}>K$, and therefore, $w_{ji}=0$ due to the constraint $w_{ij}+w_{ji}\leqslant 1$. It is clear that $w_{ii}=0,\ 1\leqslant i\leqslant n$. Thus problem (3.1) can be solved easily.

Let us denote the solution of the problem (3.1) by $W(K)=[w_{ij}(K)]_{n\times n}$, where

$$w_{ij}(K) = \begin{cases} 1 & \text{if } K_{ij}>K_{ji} \text{ and } K_{ij}>K,\\ 0 & \text{otherwise,}\end{cases} \tag{3.4}$$

and the set of arcs is represented by

$$A(W) = \{(i,j): \; if \; w_{ij} = 1, \; 1 \leqslant i,j \leqslant n, \; i \neq j\}. \tag{3.5}$$

Here, (i,j) shows the arc from X_i to X_j. It is clear that $A(W) \subset I$, where $I = \{(i,j), \; 1 \leqslant i,j \leqslant n\}$ is the set of all possible couples (i,j).

The best value for K will be found based on the maximum training accuracy for $W(K)$, defined by (3.4), that is a solution to (3.1). K is in the interval $[0, K^{max}]$ where

$$K^{max} = max\{K_{ij}, \; 1 \leqslant i,j \leqslant n, \; i \neq j\}. \tag{3.6}$$

We will consider different values $K = K_r \doteq K^{max} - \varepsilon r, \; r = 0,1,...$ until $K_r < 0$. Given K_r, let $W(K_r) = [w_{ij}(K_r)]_{n \times n}$ be the matrix defined by (3.4). With the matrix $W(K_r)$, the set of arcs $A(W(K_r))$ and, therefore, a network will be learnt.

Since the network structure is constrained to be acyclic, we should not have any cycle in the network obtained by $A(W(K_r))$. Suppose we have m cycles in the network found by $A(W(K_r))$, and consider

$$\mathscr{C}(W(K_r)) = \cup_{l=1}^{m} \mathscr{C}_l(W(K_r))$$

where $\mathscr{C}_l(W(K_r))$ denotes the set of arcs which makes l-th cycle, $l = 1,2,...,m$. Clearly $\mathscr{C}_l(W(K_r)) \subset A(W(K_r))$ for all $l = 1,...,m$, and if $(i,j) \in \mathscr{C}_l(W(K_r))$ then $w_{ij} = 1$.

We define $\overline{\mathscr{C}}(W(K_r)) = I \setminus \cup_{l=1}^{m} \mathscr{C}_l(W(K_r))$. Let us denote by

$$V(K_r) = \{v_{r_1}, v_{r_2}, ..., v_{r_{\overline{m}}}\}, \tag{3.7}$$

all arcs that are at least in one cycle in the network obtained by $A(W(K_r))$, where each r related to an arc (i_r, j_r) and $v_r = w_{i_r j_r} = 1$.

The aim is to delete a minimal number of arcs to have an acyclic structure. Deleting existing arcs in (3.7) means setting 0 to some v_{r_l}, $l = 1,...,\overline{m}$. We apply an optimization procedure to existing arcs in cycles (3.7). We utilize two different methods.

1. The first one is a simple case that can be used if the number $\overline{m}$ is small. In this case, we can consider all the possible combinations of deleting arcs in the existing cycles. Let us denote by

$$\mathbb{V} = \{V_s(K_r), \; s = 1,2,...,\rho\}, \tag{3.8}$$

the set of all possible combinations of $\overline{m}$ dimensional vectors $V_s(K_r)$ with values 0 and 1. Clearly $\rho = 2^{\overline{m}}$. Then we chose a vector $V^* = (v^*_{r_1}, v^*_{r_2}, ..., v^*_{r_{\overline{m}}})$ that has a maximal norm $\|V^*\|$ provided that the corresponding network is acyclic.

2. We consider continuous variables $(\mathfrak{v}_{\mathfrak{r}_1}, \mathfrak{v}_{\mathfrak{r}_2}, ..., \mathfrak{v}_{\mathfrak{r}_{\overline{\mathfrak{m}}}})$ with $\mathfrak{v}_{\mathfrak{r}_i} \in [0,1]$, $i = 1,...,\overline{m}$, to formulate an optimization problem. Let denote by B a binary transformation $B(\mathfrak{v}_{\mathfrak{r}_1}, \mathfrak{v}_{\mathfrak{r}_2}, ..., \mathfrak{v}_{\mathfrak{r}_{\overline{\mathfrak{m}}}}) = (\widetilde{\mathfrak{v}}_{\mathfrak{r}_1}, \widetilde{\mathfrak{v}}_{\mathfrak{r}_2}, ..., \widetilde{\mathfrak{v}}_{\mathfrak{r}_{\overline{\mathfrak{m}}}})$, where for $i = 1,...,\overline{m}$

$$\widetilde{\mathfrak{v}}_{\mathfrak{r}_i} = \begin{cases} 0 & \text{if } \mathfrak{v}_{\mathfrak{r}_i} \leqslant \frac{1}{2}, \\ \\ 1 & \text{if } \mathfrak{v}_{\mathfrak{r}_i} > \frac{1}{2}. \end{cases} \tag{3.9}$$

Let $\gamma(B(\mathfrak{v}_{\mathfrak{r}_1}, \mathfrak{v}_{\mathfrak{r}_2}, ..., \mathfrak{v}_{\mathfrak{r}_{\overline{m}}}))$ denote the number of cycles in the corresponding structure. For a large number $\overline{m}$, ρ will grow exponentially, and searching an acyclic network by considering all the possible combinations with the maximal norm $\|V\|$ will be impossible. In this case, we generate an optimization problem involving variables $\mathfrak{v}_{\mathfrak{r}_i}$, $i = 1, ..., \overline{m}$, as follows:

$$\text{Maximize } \sum_{i=1}^{\overline{m}} \left((\mathfrak{v}_{\mathfrak{r}_i} - \frac{1}{2})^2 + \zeta \mathfrak{v}_{\mathfrak{r}_i}\right) - \mu\gamma(B(\mathfrak{v}_{\mathfrak{r}_1}, \mathfrak{v}_{\mathfrak{r}_2}, ..., \mathfrak{v}_{\mathfrak{r}_{\overline{m}}})), \quad \text{s.t. } \mathfrak{v}_{\mathfrak{r}_i} \in [0,1], \forall\, i. \tag{3.10}$$

Here $\zeta \in (0, \frac{1}{2})$ and μ is a penalty parameter assigned to the number of cycles.

Problem (3.10) attempts to find an acyclic network with the largest number of arcs. We apply algorithm AGOP and CGN to solve problem (3.10). Let $(\mathfrak{v}^*_{\mathfrak{r}_1}, \mathfrak{v}^*_{\mathfrak{r}_2}, ..., \mathfrak{v}^*_{\mathfrak{r}_{\overline{m}}})$ be a global optimal solution to (3.10). The proposition below shows that it is a binary vector. Therefore, we can set $V^* = (v^*_{r_1}, v^*_{r_2}, ..., v^*_{r_{\overline{m}}}) = (\mathfrak{v}^*_{\mathfrak{r}_1}, \mathfrak{v}^*_{\mathfrak{r}_2}, ..., \mathfrak{v}^*_{\mathfrak{r}_{\overline{m}}})$.

Proposition: Let $(\mathfrak{v}^*_{\mathfrak{r}_1}, \mathfrak{v}^*_{\mathfrak{r}_2}, ..., \mathfrak{v}^*_{\mathfrak{r}_{\overline{m}}})$ is a global optimal solution to problem (3.10). Then it is a binary vector; that is, $\mathfrak{v}^*_{\mathfrak{r}_i} \in \{0,1\}, \forall\, i$; and the corresponding structure is acyclic: $\gamma(B(\mathfrak{v}^*_{\mathfrak{r}_1}, \mathfrak{v}^*_{\mathfrak{r}_2}, ..., \mathfrak{v}^*_{\mathfrak{r}_{\overline{m}}})) = 0$.

Proof: The fact $\gamma(B(\mathfrak{v}^*_{\mathfrak{r}_1}, \mathfrak{v}^*_{\mathfrak{r}_2}, ..., \mathfrak{v}^*_{\mathfrak{r}_{\overline{m}}})) = 0$ is a direct result of applying a large penalty parameter μ; thus, the corresponding structure is acyclic. Now we show that the vector $(\mathfrak{v}^*_{\mathfrak{r}_1}, \mathfrak{v}^*_{\mathfrak{r}_2}, ..., \mathfrak{v}^*_{\mathfrak{r}_{\overline{m}}})$ is binary.

Take any $1 \leqslant i \leqslant \overline{m}$, and denote $x = \mathfrak{v}^*_{\mathfrak{r}_i}$. For the sake of simplicity, let $i = 1$. After fixing all other elements $\mathfrak{v}^*_{\mathfrak{r}_j}$, $j \neq i$, we obtain

$$\psi(x) = \varphi(x) + \lambda, \tag{3.11}$$

where

$$\varphi(x) = (x - \frac{1}{2})^2 + \zeta x, \tag{3.12}$$

and

$$\lambda = \sum_{j \neq i} \left((\mathfrak{v}_{\mathfrak{r}_j} - \frac{1}{2})^2 + \zeta \mathfrak{v}_{\mathfrak{r}_j}\right) - \mu\gamma(B(x, \mathfrak{v}_{\mathfrak{r}_2}, ..., \mathfrak{v}_{\mathfrak{r}_{\overline{m}}})). \tag{3.13}$$

By assumption $x = \mathfrak{v}^*_{\mathfrak{r}_1}$ is a global maximum of $\psi(x)$. On the contrary, assume that $\mathfrak{v}^*_{\mathfrak{r}_1}$ is not binary.

It is clear that the function $\varphi(x)$ has one minimum at $\overline{x} = \frac{1-\zeta}{2}$. Moreover, $\varphi(0) = \frac{1}{4}$ and since $0 < \zeta < \frac{1}{2}$, we have

$$\varphi(\frac{1}{2}) = \frac{1}{2}\zeta < \frac{1}{4}.$$

Therefore, $\varphi(x)$ has one global maximum $x^* = 0$ on the interval $[0, \frac{1}{2}]$; and has one global maximum $x^{**} = 1$ on the interval $[0,1]$.

Now, if $\mathfrak{v}^*_{\mathfrak{r}_1} \in (\frac{1}{2}, 1)$, then taking $x = 1$, from (3.9), we have $B(1, \mathfrak{v}^*_{\mathfrak{r}_2}, ..., \mathfrak{v}^*_{\mathfrak{r}_{\overline{m}}}) = B(\mathfrak{v}^*_{\mathfrak{r}_1}, \mathfrak{v}^*_{\mathfrak{r}_2}, ..., \mathfrak{v}^*_{\mathfrak{r}_{\overline{m}}})$ and, therefore, $\gamma(B(1, \mathfrak{v}^*_{\mathfrak{r}_2}, ..., \mathfrak{v}^*_{\mathfrak{r}_{\overline{m}}})) = 0$. Then $\psi(1) > \psi(\mathfrak{v}^*_{\mathfrak{r}_1})$ which is a contradiction.

If $\mathfrak{v}^*_{\mathfrak{r}_1} \in (0, \frac{1}{2}]$, then taking $x = 0$, from (3.9), we have $B(0, \mathfrak{v}^*_{\mathfrak{r}_2}, ..., \mathfrak{v}^*_{\mathfrak{r}_{\overline{\mathfrak{m}}}}) = B(\mathfrak{v}^*_{\mathfrak{r}_1}, \mathfrak{v}^*_{\mathfrak{r}_2}, ..., \mathfrak{v}^*_{\mathfrak{r}_{\overline{\mathfrak{m}}}})$ and, therefore, $\gamma(B(0, \mathfrak{v}^*_{\mathfrak{r}_2}, ..., \mathfrak{v}^*_{\mathfrak{r}_{\overline{\mathfrak{m}}}})) = 0$. Then $\psi(0) > \psi(\mathfrak{v}^*_{\mathfrak{r}_1})$ that is again a contradiction. □

Once the acyclic network structure (ANS) is found, the training accuracy for the ANS, *accuracy*(ANS), is calculated for each *r*. Based on the highest training accuracy, the corresponding value for *r* and, therefore, the best value for K_r is chosen.

According to explanations above, we present the following algorithm for learning an optimal structure of a BN, and we call it Algorithm OpBN.

Algorithm 3.1 **Algorithm OpBN**

Step 1. Compute $\{K_{ij},\ 1 \leqslant i, j \leqslant n,\ i \neq j\}$ using (3.3).

Step 2. Determine K^{max} using (3.6), and set $r = 0$.

Step 3. While $K_r = K^{max} - \varepsilon r \geqslant 0$:
3.1. Compute $\{w_{ij}(K_r),\ 1 \leqslant i, j \leqslant n,\ i \neq j\}$, using (3.4). Set $w_{ij}(K_r) = 0$ for $i = j$, and let $W(K_r) = [w_{ij}(K_r)]_{n\times n}$.

3.2. Find the set of arcs $A(W(K_r))$ using (3.5).

3.3. Apply the topological traversal algorithm (see Section 2.2) to detect possible cycles in the network obtained by $A(W(K_r))$. If there is no cycle, then calculate the training accuracy, *accuracy*$(A(W(K_r)))$; set $r = r + 1$ and go back to Step 3.

3.4. Apply the DFS algorithm (see Section 2.2) to determine a vector $V(K_r)$ in (3.7).

3.5. Find $\mathbb{V}$, using (3.8), and determine ρ. If $\rho > \rho_0$ go to 3.10.

3.6. For $s = 1, 2, ..., \rho$, check the network obtained by $A(\overline{W}_s(K_r))$ for any possible cycle, using the topological traversal algorithm, where $\overline{W}_s(K_r) = [\overline{w}_{ij}(K_r)]_{n\times n}$, and

$$\overline{w}_{ij}(K_r) = \begin{cases} w_{ij}(K_r) & \text{if } (i, j) \in \overline{\mathscr{C}}(W(K_r)), \\ \\ v_{r_\tau} & \text{if } (i_{r_\tau}, j_{r_\tau}) \in \mathscr{C}(W(K_r)). \end{cases}$$

3.7. Set $\widetilde{\mathbb{V}} = \{\widetilde{V_{\tilde{s}}}(K_r),\ \tilde{s} = 1, 2, ..., \widetilde{\rho}\}$ including those vectors from the set $\mathbb{V}$ that are acyclic, and $\widetilde{\rho} \leqslant \rho$.

3.8. Let $\widehat{\mathbb{V}} = \{\widehat{V_{\hat{s}}}(K_r),\ \hat{s} = 1, 2, ..., \widehat{\rho}\}$ combines all vectors in the set $\widetilde{\mathbb{V}}$ having maximum norm. Clearly $\widehat{\rho} \leqslant \widetilde{\rho}$ and often there are several vectors with the same maximum norm; that is $\widehat{\rho}$ might be greater than 1.

3.9. Find the maximum training accuracy, *accuracy*$(A(\widehat{W^*}(K_r)))$ between the network structures obtained by $\widehat{W_{\hat{s}}}(K_r),\ \hat{s} = 1, 2, ..., \widehat{\rho}$ corresponding to $\widehat{V_{\hat{s}}}(K_r)$ and set $W(K_r) = \widehat{W^*}(K_r)$; set $r = r + 1$ and go back to Step 3.

3.10. Solve the optimization problem (3.10) by applying algorithm AGOP; denote the solution found by $V'(K_r)$. Then apply algorithm CGN starting from this solution to obtain a vector $V^*(K_r)$. After this optimization procedure we create corresponding matrix $W^*(K_r)$. Set $W(K_r) = W^*(K_r)$, and find the new acyclic network structure by a set of arcs $A(W(K_r))$ using (3.5).

3.11. Compute the training accuracy, $accuracy(A(W(K_r))$; set $r = r + 1$ and go back to Step 3.

Step 4. Find the best K_r^* where $accuracy(A(W(K_r^*)))$ has the maximum value among the training accuracies, $accuracy(A(W(K_r))),\ r = 1, 2, \ldots.$

Step 5. Return an optimal acyclic structure using a set of arcs $A(W(K_r^*))$.

4. Numerical Experiments

This paper studies 22 benchmark data sets taken from the UCI machine learning repository Asuncion & Newman (2007) and the tools page of the LIBSVM Chang & Lin (2001). These data sets have been considered quite frequently in the literature. A brief description of the data sets is given in Table 1.

We use three different methods to discretize the continuous features. In the first one, we apply a mean value of each feature to discretize values to binary, $\{0, 1\}$. In the second one, we apply the Fayyad and Irani's discretization method Fayyad & Irani (1993). The third one is Algorithm SOAC Yatsko *et al.* (2011); the parameter θ in this algorithm is chosen as 0.2. This parameter has not been fitted by preliminary experimentation, and is similar to the one used for other problems in Yatsko *et al.* (2011).

We conduct experiments to compare the proposed algorithm (OpBN) with the Naive Bayes (NB), the Tree Augmented Naive Bayes (TAN), the k-Dependency Bayesian Networks (k-DBN), $k = 2$, the unrestricted dependency BNs algorithm (UDBN), the SVM and the C4.5 in terms of the test set accuracy. In all cases we use 10-fold cross validation. Runs with the various classifiers were carried out on the same training sets and evaluated on the same test sets. In particular, the cross validation folds are the same for all experiments on each data set.

In calculations, we take $\eta = 10^{-3}$, $\vartheta = 1.1$, $\delta = 10^{-3}$, $\omega = 10^{-10}$, $\varpi = 10^{10}$ for the CGN and we set $\mu = 10^3$, $\varepsilon = 0.1$, $\rho_0 = 2^{10}$ for the proposed algorithm.

4.1 *Results*

In this section, we present accuracies obtained with the proposed algorithm OpBN. We compare the OpBN by means of the predictive accuracies obtained with some well-known classifiers such as the NB, the TAN, the k-DBN, the UDBN, the SVM, and the C4.5. The predictive accuracy of each method is the percentage of test sets for which it predicts the class correctly. The predictive accuracies, for each classifier in each data set, are summarized in Tables 2 to 4, where continuous features are discretized by using mean values, the Fayyad and Irani's method and discretization algorithm SOAC, respectively. Since the UDBN is an algorithm proposed for binary classification, we do not have the accuracy results for multi class data sets.

Figure 1 shows the scatter plots comparing the proposed algorithm with the NB, the TAN, the k-DBN, the UDBN, the SVM, and the C4.5 on different data sets using the Fayyad and Irani's method discretization method. In these plots, each point represents a data set, where the x coordinate of a point is the percentage of miss classifications according to the proposed algorithm, and the y coordinate is the percentage of miss classification according to the chosen classifier for comparison. Therefore, points above the diagonal line correspond to data sets where the proposed algorithm performs better, and points below the diagonal line correspond to data sets where the chosen classifier performs better.

The results presented indicate that the proposed algorithm has produced more accurate results: it has the highest average accuracies for all discretizing methods. Now, we summarize the highlights as follows.

The test set accuracies of the proposed algorithm (OpBN), using mean values for discretization, are significantly higher than the NB, the TAN and the *k*-DBN in all data sets. The OpBN has also better accuracies than the UDBN in the majority of data sets. In 21 cases out of 22, the OpBN shows higher accuracies than the UDBN. It has the same accuracy as the UDBN in the data set Svmguide3, which is 85.41 percent. The proposed algorithm also has much better accuracies than the SVM in the most of data sets. In 2 cases out of 22, accuracies of this algorithm almost ties with those obtained by the SVM. Observe from the results, the OpBN has greater accuracies than the C4.5 in 20 cases out of 22.

The proposed algorithm, where continuous features are discretized by applying the Fayyad and Irani's method (FaI), also performs significantly better than the NB, the TAN and the *k*-DBN in all data sets. Compared to the UDBN, it has better accuracies in 20 cases out of 22 data sets. The accuracies of data sets Spambase and Svmguide3 are equal in both algorithms. These results also show that the OpBN has higher accuracies than the SVM in 21 data sets out of 22, where as the later method has slightly higher accuracy than the former method in the data set Waveform. It is also notable that the proposed algorithm has greater accuracies than the C4.5 in 20 data sets. In the data set Letter Recognition, it has the same accuracy as the C4.5, with the value of 87.68 percent, and it almost ties with the later one in the data set Image Segmentation.

The accuracies obtained by the proposed algorithm, using the discretization algorithm SOAC, in all data sets are higher than those obtained by the NB, the TAN and the *k*-DBN. The accuracies of this algorithm are better than those of the UDBN in the most of data sets. In 21 cases out of 22, the OpBN has greater accuracies than the UDBN. They have the same value of accuracy for the data set Svmguide3, which is 82.92 percent. The results also demonstrate that the OpBN has significantly higher accuracies than the SVM in 21 data sets out of 22, and the accuracy for these methods almost ties in the data set Lymphography. The proposed algorithm has higher accuracies in 19 data sets when compared to the C4.5.

4.2 *Dynamic structures generated by OpBN*

As mentioned above the main advantage of the proposed method is that it does not set the number of parents a priory. This number comes from the optimization procedure; it might be different for different folds on the same data. To demonstrate this we consider one example.

Table 5 shows the structure of the Diabetes data set with 8 features (see Table 1) obtained by Algorithm OpBN. Four different structures obtained by the proposed algorithm when applying 10-folds cross validation. For instance parents of feature X_7 are: features X_3, X_4, X_5 for folds 1 and 5-10; features X_4, X_5 for folds 2 and 3. This feature (X_7) does not have any parent in the structure obtained for fold 4.

5. Conclusion

In this paper, we present a new algorithm to learn an optimal structure of a Bayesian Network from data. The proposed algorithm is based on a new combinatorial optimization formula. We utilize this formula to find the most probable networks. Then, we apply optimization techniques to have the best acyclic structure in a Bayesian Network. When there is a small number of cycles, we search the network for different combinations of deleting some arcs in the existing cycles. In the second case, when there is a large number of cycles, we apply optimization methods, AGOP and CGN.

Benchmark tests are performed to evaluate the effectiveness of the proposed algorithm and compare its performance with other commonly used classifiers. The data set are chosen from the UCI machine learning repository and the LIBSVM, and the continuous features are discretized by applying three different methods. The results from the tests indicate that the new algorithm outperforms the other mentioned algorithms for accuracy. An interesting aspect of the presented algorithm and its learning method is that it discovers unrestricted edges between nodes (dependencies between features) which is common in real life data sets.

REFERENCES

ASUNCION, A. & NEWMAN, D. (2007), UCI machine learning repository. *School of Information and Computer Science, University of California http://www.ics.uci.edu/mlearn/MLRepository.html.*

BENDER, M. A. & FINEMAN, J. T. GILBERT, S. (2009) A New Approach to Incremental Topological Ordering *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms Society for Industrial and Applied Mathematics Philadelphia, PA, USA.*

CAMPOS, L. & FERNANDEZ-LUNA, M. & GAMEZ, A. & PUERTA, M. *(2002) Ant colony optimization for learning Bayesian networks,* Ant colony optimization for learning Bayesian networks. International Journal of Approximate Reasoning, ***31**, 291-311.*

CASTILLO, E. & GUTIERREZ, J.M & HADI, A.S. *(1997) Expert Systems and Probabilistic Network Models,* Springer Verlag, New York.

CHANG, C. & LIN, C. *(2001) LIBSVM: A library for support vector machines.* www.csie.ntu.edu.tw/cjlin/libsvm.

CHICKERING, D. M. *(1996) Learning Bayesian Networks is NP-complete,* Artificial Intelligence and statistics V, Springer, *121-130.*

DALY, R., & SHEN, Q. *(2009) Learning Bayesian Network Equivalence Classes with Ant Colony Optimization.*Journal of Artificial Intelligence Research, Elsevier, *391-447.*

DOMINGOS, P. & PAZZANI, M. *(1997) On the optimality of the simple Bayesian classifier under zero-one loss,* Machine Learning, ***29**, 103-130.*

FAYYAD, U. M & IRANI, K. B. *(1993) On the Handling of Continuous-Valued Attributes in Decision Tree Generation,* Machine Learning, ***8**, 87-102.*

FRIEDMAN, N. & GEIGER, D. & GOLDSZMIDT, M. *(1997) Bayesian network classifiers,* Machine Learning, ***29**, 131-163.*

HAEUPLER, B. & KAVITHA, T. & MATHEW, R. & SEN, S. & TARJAN, R. E. *(2008) Incremental Cycle Detection, Topological Ordering, and Strong Component Maintenance,* 35th International Colloquium on Automata, Languages, and Programming (ICALP), Reykjavik, Iceland.

HECKERMAN, D. & GEIGER, D. & CHICKERING, D.M. *(1995) Learning Bayesian Networks: the Combination of Knowledge and Statistical Data,* Machine Learning, ***20**, 197-243.*

HECKERMAN, D. & CHICKERING, D. & MEEK, C. *(2004) Large-Sample Learning of Bayesian Networks is NP-Hard.* Machine Learning*, 1287-1330.*

JANZURA, M.& NIELSEN, J. *(2006) A Simulated Annealing-Based Method for Learning Bayesian Networks from Statistical Data,* International Journal of Intelligent Systems, ***21**, 335-348.*

JENSEN1, F. *(1996) An Introduction to Bayesian Networks,* Springer, New York

JI, Z. & ZHONG, H. & HU. R. & LIU, C. *(2009) Bayesian Network Learning Algorithm Based on Independence Test and Ant Colony Optimization,* Acta Automatica Sinica*, **35**.*

KABLI, R. & HERRMANN, F. & MCCALL, J. *(2007) A Chain-Model Genetic Algorithm for Bayesian Network Structure Learning,* Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM New York, NY, USA.

KOLDA, G. & LEWIS, M. & TORCZON, V. *(2003) Optimization by direct search: New perspectives on some classical and modern methods,* SIAM Review ***45**, 385-482.*

KOUHBOR, S. & UGON, J. & RUBINOV, A. & KRUGER, A. & MAMMADOV, M. *(2006) Coverage in WLAN with minimum number of access points,* Vehicular Technology Conference, VTC Spring, ***63**, 1166-1170.*

LANGLEY, P. & IBA, W. & THOMPSON, K. *(1992) An Analysis of Bayesian Classifiers,* In 10th International Conference Artificial Intelligence, AAAI Press and MIT Presspp*, 223-228.*

LARRANAGA, P. & MURGA, R. & POZA, M. & KUIJPERS, C. *(1996) Structure Learning of Bayesian Networks* *by Hybrid Genetic Algorithms,* Spriger-Verlag, *165-174.*

LARRANAGA, P. & POZA, M. & YURRAMENDI, Y. & MURGA, H. & KUIJPERS, C. *(1996) Structure Learning of Bayesian Networks by Genetic Algorithms: A Performance Analysis of Control Parameters,* IEEE Transactions on Pattern Analysis and Machine Intelligence archive, ***18***.

LARRANAGA, P. & SIERRA, B. & GALLEGO, J. & MICHELENA, J. & PICAZA. M, *(1997) Learning Bayesian Networks by Genetic Algorithms: A case study in the prediction of survival in malignant skin melanoma. Artificial Intelligence in Medicine,* Lecture Notes in Computer Science*, 261-272.*

MAMMADOV, M.A. & RUBINOV A.M & SNIEDOVICH, M. *(2004) A New Global Optimization Algorithm Based on Dynamical Systems Approach,* 6th International Conference on Optimization: Techniques and Applications, Ballarat, Australia.

MAMMADOV, M.A. & RUBINOV A.M & YEARWOOD, J. *(2005) Dynamical Systems Described by Relational Elasticities with Applications to Global Optimization,* 6th In Continuous Optimisation: Current Trends and Modern Applications, V.Jeyakumar and A. Rubinov, Eds. Springer*, 365-385.*

MAMMADOV, M.A. & ORSI, R. *(2005) H-infinity via a nonsmooth, nonconvex optimization approach,* Pacific Journal of Optimization*, 405-420.*

MARINESCU, R. & DECHTER, R. *(2009) AND/OR Branch-and-Bound search for combinatorial optimization in graphical models,* Artificial Intelligence, Elsevier, *1457-1491.*

MAROOSI, A. & AMIRI, B. *(2010) A new clustering algorithm based on hybrid global optimization based on a dynamical systems approach algorithm,* Expert Systems with Applications*, **37**, 5645-5652.*

PARK, H. & CHO, S. *(2006) An Efficient Attribute Ordering Optimization in Bayesian Networks for Prognostic Modeling of the Metabolic Syndrome,* Springer-Verlag Berlin Heidelberg*, 381-391.*

PEARL, J. *(1988) Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference,* Morgan Kaufmann.

RICHTER, F. & FETTWEIS, G. *(2012) Base Station Placement Based on Force Fields,* IEEE VTC-Spring*, Yokohama, Japan.*

ROBINSON, R. W. (1997) Counting unlabeled acyclic diagraphs, *Springer-Verlag, New York,* 28-43.

SAHAMI, M. (1996) Learning Limited Dependence Bayesian Classifiers, *In the 2nd International Conference. Knowledge Discovery and Data mining (KKD)*, 335-338.

SAHIN, F. & YAVUZ, M. & ARNAVUT, Z. & ULUYOL, O. (2007) Fault diagnosis for airplane engines using Bayesian networks and distributed particle swarm optimization, *Parallel Computing, Elsevier*, 124-143.

SCHLEIP, C. & RAIS, A. & MENZEL, A. (2009) Bayesian analysis of temperature sensitivity of plant phenology in Germany, *Agricultural and Forest Meteorology, Elsevier*, 1699-1708.

SEDGEWICK, R. (1983) Graph algorithms, *Addison-Wesley, ISBN 0-201-06672-6.*

SHAFER, G. & PEARL, J. (1990) Readings in Uncertain Reasoning, *Morgan Kaufmann, San Mateo, CA.*

SUN, W. & YUAN, Y.X. (2006) Optimization theory and methods, nonlinear programming, *Springer*.

TAHERI, S. & MAMMADOV, M. (2012) Solving Systems of Nonlinear Equations using a Globally Convergent Optimization Algorithm, *Global Journal of Technology and Optimization,* **3**, 132-138.

TAHERI, S., MAMMADOV, M. SEIFOLLAHI, S. (2012) Globally Convergent Optimization Methods for Unconstrained Problems, *Optimization*, 124-143.

TAHERI, S. & MAMMADOV, M. (2012) Structure Learning of Bayesian Networks using a New Unrestricted Dependency Algorithm, *Second International Conference on Social Eco-Informatics, Venice, Italy*.

TILAKARATNE, C.D. & MAMMADOV, M. & MORRIS, S.A. (2009) Modified neural network algorithms for predicting trading signals of stock market indices, *J. Appl. Mathe. Decis. Sci,* **22**.

TUCKER, A. (2006) Covering Circuits and Graph Coloring, *Applied Combinatorics (5th ed). Hoboken: John Wiley and sons* **49**.

YATSKO, A. & BAGIROV, A. & STRANIERI, A. (2011) On the Discretization of Continuous Features for Classification, *In the proceedings of Ninth Australasian Data Mining Conference (AusDM 2011), Ballarat, Australia,* **125**.

ZHAO, J. & SUN, J. & XU, W. & ZHOU, D. (2009) Structure Learning of Bayesian Networks Based on Discrete Binary Quantum-behaved Particle Swarm Optimization Algorithm, *Proceedings of the Fifth International Conference on Natural Computation, IEEE Computer Society Washington, DC, USA*, **6**.

TABLE 1 *A brief description of data sets.*

Data sets	# Observations	# Features	# Classes
Breast Cancer	699	10	2
Congres Vote	435	16	2
Credit Approval	690	15	2
Diabetes	768	8	2
German.numer	1000	24	2
Glass Identification	214	10	7
Haberman Survival	306	3	2
Heart Disease	303	14	2
Hepatitis	155	19	2
Image Segmentation	2310	19	7
Ionosphere	351	34	2
Iris	150	4	3
Letter Recognition	20000	16	26
Liver Disorders	345	6	2
Lymphography	148	18	4
Sonar	208	60	2
Soybean-Large	307	35	19
Spambase	4601	57	2
Svmguide1	7089	4	2
Svmguide3	1284	21	2
Vehicle Silhouettes	946	18	4
Waveform-21	5000	21	3

TABLE 2 *Average predictive accuracy over 10 fold cross validation for 22 data sets using mean value for discretization.*

Data sets	Bayesian Network Classifiers					Benchmark Classifiers	
	NB	TAN	*k*-DBN	UDBN	OpBN	SVM	C4.5
Breast Cancer	96.10	95.71	97.31	97.66	**97.89**	95.15	91.06
Congress Vote	90.31	91.42	94.62	95.48	**96.93**	96.02	95.51
Credit	84.95	82.88	86.87	87.46	**88.14**	85.31	87.47
Diabetes	75.90	76.48	75.03	75.98	**77.81**	76.72	75.98
German	75.41	74.13	76.35	76.27	**78.30**	76.11	72.43
Glass	69.40	68.95	69.64	—	**73.74**	71.14	69.35
Haberman	75.01	73.85	76.43	77.86	**78.92**	73.34	71.60
Heart Disease	81.12	84.12	84.27	84.71	**84.85**	80.14	81.53
Hepatitis	83.61	83.14	84.12	85.25	**86.08**	83.61	82.97
Image Seg	90.65	85.01	91.08	—	93.10	89.35	**93.13**
Ionosphere	82.90	84.02	88.35	89.98	**90.21**	85.94	86.20
Iris	95.66	95.66	95.66	—	**96.11**	95.66	95.66
Letter	64.65	73.01	73.91	—	86.98	82.10	**87.41**
Liver	61.86	61.89	62.22	64.17	**65.73**	60.16	60.79
Lymphography	79.71	66.89	71.43	—	86.24	**86.31**	77.12
Sonar	75.18	75.44	75.61	76.89	**78.92**	76.98	76.65
Soybean	91.08	92.02	92.27	—	**94.28**	93.52	91.85
Spambase	90.03	89.69	89.27	92.37	**94.12**	90.17	91.89
Svmguide1	92.57	91.99	92.98	94.17	**97.09**	93.24	93.62
Svmguide3	81.51	83.04	83.64	85.41	**85.41**	80.16	81.24
Vehicle	59.15	68.70	68.91	—	**76.18**	73.98	69.99
Waveform-21	76.98	75.12	75.86	—	85.51	**85.59**	74.51

TABLE 3 *Average predictive accuracy over 10 fold cross validation for* 22 *data sets the discretization method FaI.*

Data sets	Bayesian Network Classifiers					Benchmark Classifiers	
	NB	TAN	*k*-DBN	UDBN	OpBN	SVM	C4.5
Breast Cancer	97.18	96.52	96.92	97.72	**97.98**	96.52	94.11
Congress Vote	90.11	93.21	94.73	95.12	**96.71**	95.04	95.32
Credit	86.10	84.78	86.44	87.21	**88.93**	85.03	84.87
Diabetes	74.56	75.14	75.12	75.85	**77.84**	75.51	73.83
German	74.50	73.13	76.32	76.27	**79.82**	76.41	71.92
Glass	69.63	69.15	69.84	—	**74.30**	71.50	69.58
Haberman	75.09	74.41	76.89	77.91	**79.18**	73.20	71.24
Heart Disease	82.93	81.23	83.45	85.14	**85.31**	81.67	82.85
Hepatitis	84.56	83.91	83.90	85.17	**86.87**	85.16	83.87
Image	91.15	85.31	91.18	—	93.58	89.52	**93.62**
Ionosphere	88.62	89.77	89.83	91.10	**92.62**	89.67	89.98
Iris	95.87	95.87	95.87	—	**96.11**	95.87	95.87
Letter	64.93	73.41	73.86	—	**87.68**	82.22	**87.68**
Liver	63.26	63.18	64.17	65.91	**66.86**	62.03	62.15
Lymphography	79.70	66.85	76.34	—	**87.94**	86.48	77.01
Sonar	76.32	76.47	76.49	76.74	**79.35**	77.96	77.31
Soybean	91.19	92.10	92.52	—	**94.12**	93.85	91.97
Spambase	90.41	89.78	89.39	93.18	**93.18**	90.43	92.97
Svmguide1	92.39	91.61	92.76	94.45	**97.22**	94.31	95.99
Svmguide3	81.23	82.47	83.23	84.42	**84.42**	80.37	81.38
Vehicle	58.27	67.85	67.88	—	**77.32**	74.34	72.45
Waveform-21	77.87	75.35	76.71	—	86.50	**86.68**	74.68

TABLE 4 *Average predictive accuracy over 10 fold cross validation for* 22 *data sets using the discretization Algorithm SOAC.*

Data sets	Bayesian Network Classifiers					Benchmark Classifiers	
	NB	TAN	*k*-DBN	UDBN	OpBN	SVM	C4.5
Breast Cancer	96.12	95.60	96.76	97.65	**97.94**	95.31	91.16
Congress Vote	90.11	91.42	92.61	94.16	**96.97**	96.75	95.12
Credit	85.85	84.98	86.53	87.17	**89.11**	86.11	87.54
Diabetes	75.78	75.90	75.82	76.22	**78.02**	76.68	75.63
German	74.61	74.01	75.31	76.15	**79.14**	76.35	72.21
Glass	69.52	69.02	69.76	—	**73.84**	71.62	69.46
Haberman	74.66	76.08	75.64	77.31	**79.24**	73.36	72.15
Heart Disease	78.62	77.37	79.54	81.69	**83.46**	77.96	79.17
Hepatitis	82.93	81.54	84.21	85.93	**86.12**	84.24	82.34
Image	91.37	85.51	91.24	—	93.41	89.47	**93.72**
Ionosphere	85.92	86.18	85.94	88.62	**90.23**	86.15	86.71
Iris	93.43	93.42	94.11	—	**95.36**	94.18	94.18
Letter	64.80	73.71	73.98	—	87.34	82.41	**87.71**
Liver	65.82	65.73	65.95	65.97	**66.81**	63.69	64.98
Lymphography	79.76	66.95	71.81	—	86.34	**86.73**	77.11
Sonar	75.09	75.76	75.85	76.91	**79.31**	77.74	76.41
Soybean	91.21	92.15	92.31	—	**94.79**	93.81	91.99
Spambase	89.30	89.04	90.69	92.54	93.26	91.56	**93.73**
Svmguide1	95.81	94.91	96.32	97.54	**97.91**	95.94	96.91
Svmguide3	77.25	79.99	80.75	82.92	**82.92**	78.32	78.49
Vehicle	62.23	69.97	69.78	—	**75.24**	73.81	72.88
Waveform-21	76.98	74.58	75.64	—	**88.78**	86.12	74.06

TABLE 5 *Parents of each features X_i of the data set 'Diabetes' obtained by Algorithm 'OpBN'.*

Features	Parents			
	Folds: 1, 9, 10	Folds: 2, 3	Fold: 4	Folds: 5, 6, 7, 8
X_1	$X_3 - X_5, X_7$	$X_3 - X_5, X_7$	$X_3 - X_5, X_7$	$X_3 - X_5, X_7$
X_2	$X_1, X_3 - X_8$	$X_1, X_3 - X_8$	$X_1, X_3 - X_8$	$X_1, X_3 - X_8$
X_3	X_4, X_5	X_4, X_5, X_7	X_4, X_5, X_7	X_4, X_5
X_4	*no parent*	*no parent*	X_7	*no parent*
X_5	X_4	X_4	X_4, X_7	X_4
X_6	$X_1, X_3 - X_5, X_7, X_8$	$X_1, X_3 - X_5, X_7, X_8$	$X_1, X_3 - X_5, X_7$	$X_1, X_3 - X_5, X_7$
X_7	$X_3 - X_5$	X_4, X_5	$-$	$X_3 - X_5$
X_8	$X_1, X_3 - X_5, X_7$	$X_1, X_3 - X_5, X_7$	$X_1, X_3 - X_7$	$X_1, X_3 - X_7$

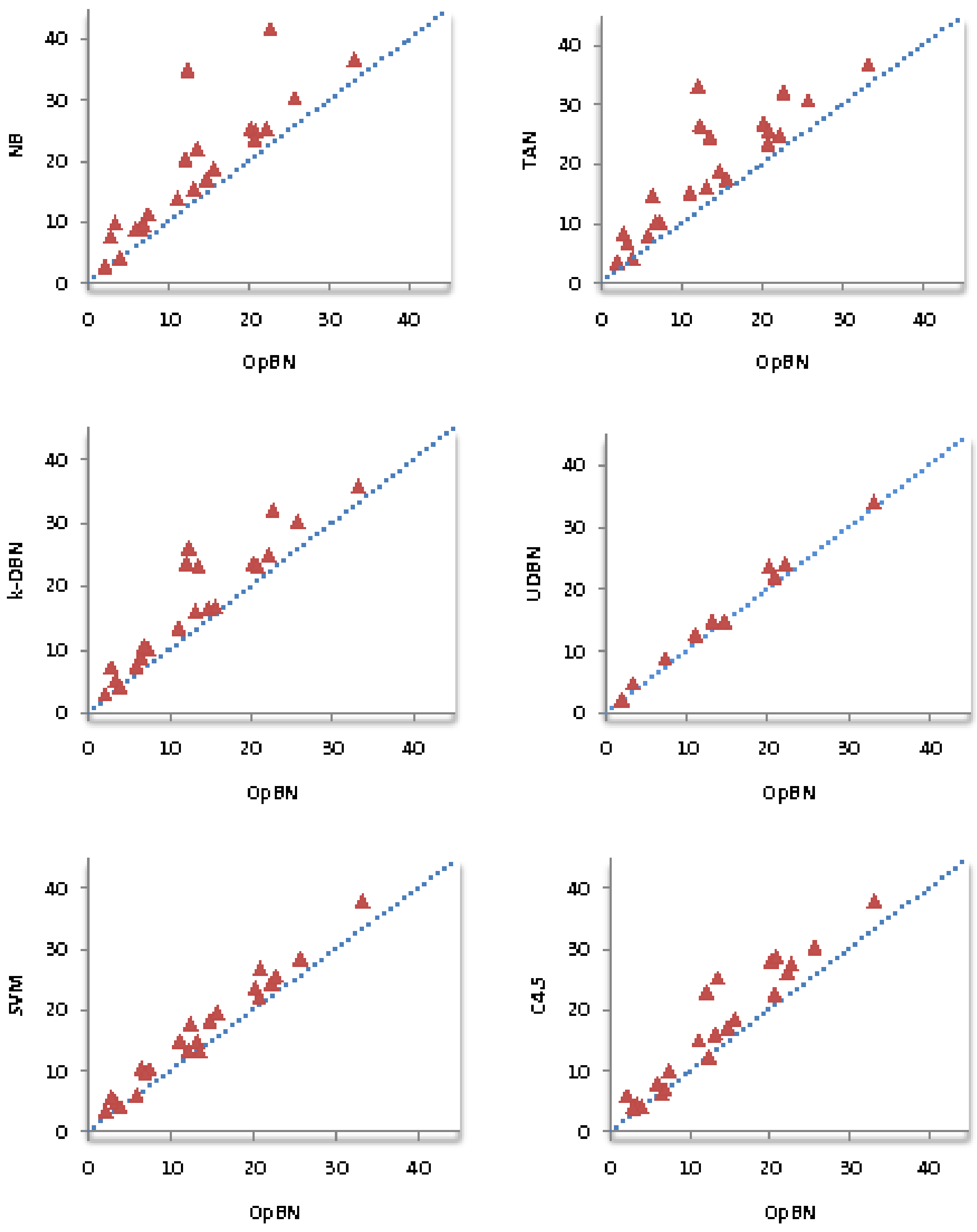


FIG. 1. Scatter plots comparing miss classifications of the proposed algorithm (*x* coordinate) with competing methods (*y* coordinate); using the discretization method FaI

Chapter 6

Conclusions and Future Work

This chapter includes the key contributions that this research has made to knowledge. The suggestions for further research are also presented.

6.1 Summary of Contributions

The learned structure of Bayesian Networks can be used for guiding future action and understanding the causal mechanisms of a system if structure learning algorithms are able to learn accurate structure and certain assumptions are met. Once an optimal structure has been specified, then the network is trained by learning parameters. It consists of computing probabilities and conditional probabilities.

The goal of this thesis is to develop new algorithms to learn structure and parameters of Bayesian Networks. The proposed algorithms apply different optimization problems. To solve these problems, novel optimization algorithms are introduced. Therefore, this thesis makes the following contributions towards improving the process of learning Bayesian Networks:

1. Development of new algorithms to learn an optimal structure in a Bayesian Network. We introduce three different algorithms for structure learning. The first one is improving the Naive Bayes' structure by alleviating the feature independence assumption. We use conditional probabilities to find dependencies between features. In the proposed algorithm,

each feature has the class and at most one other feature as parents. The second one is an unrestricted dependency algorithm in which some features could have a large number of parents, whereas others just have a few. We present a combinatorial optimization problem to find the dependency between features. The number of parents of each feature is found by the proposed algorithm internally. We also propose a new algorithm to find an optimal structure in Bayesian Networks using global optimization method.

Another alternative to improve the performance of the Naive Bayes without violating the feature independence assumption is using feature (attribute) weights. We present a novel attribute weighted Naive Bayes by assigning weights to conditional probabilities. An objective function has been constructed based on the Naive Bayes' structure and the attribute weights. These weights are considered in the form of powers to conditional attribute class probabilities. The weights, then, are found by using a local optimization method.

2. Development of new models to learn parameters of Bayesian Networks. We introduce three different optimization models to learn parameters in the Naive Bayes. The objective functions are constructed by considering some unknown variables corresponding to class probabilities and conditional feature class probabilities. To optimize these functions to find optimal values of variables, we apply newly developed local optimization method. We consider the similar strategy to the Tree Augmented Naive Bayes by introducing a different objective function.

3. Development of novel optimization algorithms for optimizing the proposed models in 1 and 2 efficiently:

- Local methods: We propose new local optimization algorithms based on the combination of the gradient method and Newton based methods. The gradient method is globally convergent, but it suffers from the slow convergence rate as a stationary point is approached. The Newton method has a high convergence rate, but it depends on initial point. We develop the combined algorithms with the global and superlinear convergence properties.

- Global methods: We apply the Algorithm AGOP introduced by Mammadov et al. in conjunction with the newly suggested local optimization methods.

4. Application of the developed methods to the real world problems. To validate the proposed methods for Bayesian Networks, we conduct experiments across a number of real world data sets from the UCI machine learning repository and the LIBSVM. We also verify the efficiency of the optimization algorithms when utilize to several well known test problems and Bayesian Network models including real world data sets.

6.2 Looking Beyond

According to author's opinion, a possible avenue for further research in this direction could be:

We have verified the efficiency of proposed algorithms for learning structure of BNs in several well known data classification problems. Exploring the robustness of these algorithms under the variety of feature selection and the clustering problems would be the future work.

We have introduced new algorithms to learn an optimal structure in a BN and presented different optimization models to learn parameters in the NB and the TAN, when applied to binary classification problems. The proposed algorithms will be extended to multi class and multi label data sets.

We have discretized the values of continues features in data sets using the existing algorithms, Fayyad and Irani's discretization method and the Algorithm SOAC, sub-optimal agglomerative clustering method. Although these methods are robust and efficient, they were initially developed in learning contexts other than BNs learning. Because it is likely to find more dependencies when discretizing the data to only few intervals than when using many intervals, we require a method to guarantee that each of the intervals has a reasonable amount of observations. Developing an appropriate discretization method for BNs that takes into account the interactions of each feature variable with the other feature variables would be an interesting research.

We have applied the Algorithm AGOP as a global method to solve the optimization problem corresponding to learning an optimal structure in a BN. Developing new strategies for generating initial points for this global search method would be another research topic.

Appendix1: Status of Manuscripts

[1] Sona Taheri, Musa Mammadov and Adil Bagirov, Improving Naive Bayes Classifier using Conditional Probabilities. Published in Ninth Australasian Data Mining Conference, Ballarat, Australia, Volume 121, pp. 63-68, 2011.

[2] Sona Taheri and Musa Mammadov, Structure Learning of Bayesian Networks using a New Unrestricted Dependency Algorithm. Published in Second International Conference on Social Eco-Informatics, Venice, Italy, 2012.

[3] Musa Mammadov and Sona Taheri, A Globally Convergent Optimization Algorithm for Systems of Nonlinear Equations. Published in Third Global Conference on Power Control and Optimization, Gold Coast, Australia, 2010.

[4] Sona Taheri and Musa Mammadov, Solving Systems of Nonlinear Equations using a Globally Convergent Optimization Algorithm. Published in Global Journal of Technology and Optimization, Volume 3, pp. 132-138, 2012.

[5] Sona Taheri, Musa Mammadov, and Sattar Seifollahi, Globally Convergent Optimization Methods for Unconstrained Problems. Published in Optimization Journal, 2012.

[6] Sona Taheri and Musa Mammadov, Learning Naive Bayes Classifier with Optimization Models. Accepted in International Journal of Applied Mathematics and Computer Sciences, 2012.

[7] Sona Taheri, John Yearwood, Musa Mammadov, and Sattar Seifollahi, Attribute Weighted Naive Bayes Classifier using a Local Optimization. Submitted to Neural Computing and Applications Journal, February 2012.

[8] Sona Taheri and Musa Mammadov, Tree Augmented Naive Bayes Classifier based on Optimization. Published in Forty second Annual Iranian Mathematics Conference, Rafsanjan, Iran, pp. 594-597, 2011.

[9] Sona Taheri and Musa Mammadov, Structure Learning of Bayesian Networks using Global Optimization. Submitted to IMA Journal of Applied Mathematics Journal, October 2012.

Appendix 2: Contribution of Candidate

Acknowledgement of contribution to the research work and authorship

The theme of the thesis is Bayesian Network Models based on Global Optimization. This thesis includes nine original published/submitted papers in conferences and journals.

The main ideas and writing of all the papers in the thesis were the principal responsibility of the candidate, working within the School of Science, Information Technology and Engineering under the supervision of Dr. Musa Mammadov and Assoc Prof. Adil Bagirov.

The list of co-authors reflects the fact that the work came from active collaboration between the candidate and supervisors.

Paper 1:

Authors Contribution

Title: Improving Naive Bayes Classifier using Conditional Probabilities

Chapter in thesis: 3

Name of Publication: Ninth Australasian Data Mining Conference

First co-author: Dr. Musa Mammadov

The contribution is related to the following:

Research framework

Mathematics

Development of an algorithm

Interpretation of results

Editing of manuscript

Percentage contribution: 25

Signature:− − − − − − − −−

Second co-author: Assoc Prof. Adil Bagirov

The contribution is related to the following:

Research framework

Mathematics

Development of an algorithm

Programming codes

Interpretation of results

Editing of manuscript

Percentage contribution: 25

Signature:− − − − − − − −−

Paper 2:

Authors Contribution

Title: Structure Learning of Bayesian Networks using a New Unrestricted Dependency Algorithm

Chapter in thesis: 3

Name of Publication: Second International Conference on Social Eco-Informatics

Co-author: Dr. Musa Mammadov

The contribution is related to the following:

Research framework

Mathematics

Development of an algorithm

Interpretation of results

Editing of manuscript

Percentage contribution: 50

Signature:− − − − − − − −−

Paper 3:

Authors Contribution

Title: A Globally Convergent Optimization Algorithm for Systems of Nonlinear Equations

Chapter in thesis: 4

Name of Publication: Third Global Conference on Power Control and Optimization

Co-author: Dr. Musa Mammadov

The contribution is related to the following:

Research framework

Mathematics

Development of an algorithm

Interpretation of results

Editing of manuscript

Percentage contribution: 50

Signature:− − − − − − − −−

Paper 4:

Authors Contribution

Title: Solving Systems of Nonlinear Equations using a Globally Convergent Optimization Algorithm

Chapter in thesis: 4

Name of Publication: Global Journal of Technology and Optimization

Co-author: Dr. Musa Mammadov

The contribution is related to the following:

Research framework

Mathematics

Development of an algorithm

Interpretation of results

Editing of manuscript

Percentage contribution: 50

Signature:− − − − − − − −−

Paper 5:

Authors Contribution

Title: Globally Convergent Optimization Methods for Unconstrained Problems

Chapter in thesis: 4

Name of Publication: Optimization Journal

First co-author: Dr. Musa Mammadov

The contribution is related to the following:

Research framework

Mathematics

Development of an algorithm

Interpretation of results

Editing of manuscript

Percentage contribution: 25

Signature:− − − − − − − −−

Second co-author: Dr. Sattar Seifllohi

The contribution is related to the following:

Research framework

Mathematics

Development of an algorithm

Programming codes

Interpretation of results

Editing of manuscript

Percentage contribution: 25

Signature:− − − − − − − −

Paper 6:

Authors Contribution

Title: Learning Naive Bayes Classifier with Optimization Models

Chapter in thesis: 5

Name of Publication: International Journal of Applied Mathematics and Computer Sciences

Co-author: Dr. Musa Mammadov

The contribution is related to the following:

Research framework

Mathematics

Construction of manuscript

Interpretation of results

Editing of manuscript

Percentage contribution: 50

Signature:− − − − − − − −

Paper 7:

Authors Contribution

Title: Attribute Weighted Naive Bayes Classifier using a Local Optimization

Chapter in thesis: 5

Name of Publication: Neural Computing and Applications Journal

First co-author: Prof. John Yearwood

The contribution is related to the following:

Research framework

Mathematics

Construction of manuscript

Interpretation of results

Editing of manuscript

Percentage contribution: 15

Signature:− − − − − − − −−

Second co-author: Dr. Musa Mammadov

The contribution is related to the following:

Research framework

Mathematics

Construction of manuscript

Interpretation of results

Editing of manuscript

Percentage contribution: 15

Signature:− − − − − − − −−

Third co-author: Dr. Sattar Seifllohi

The contribution is related to the following:

Research framework

Mathematics

Construction of manuscript

Programming codes

Interpretation of results

Editing of manuscript

Percentage contribution: 15

Signature:− − − − − − − −−

Paper 8:

Authors Contribution

Title: Tree Augmented Naive Bayes Classifier based on Optimization

Chapter in thesis: 5

Name of Publication: 42^{nd} Annual Iranian Mathematics Conference

Co-author: Dr. Musa Mammadov

The contribution is related to the following:

Research framework

Mathematics

Construction of manuscript

Interpretation of results

Editing of manuscript

Percentage contribution: 50

Signature:− − − − − − − −−

Paper 9:

Authors Contribution

Title: Structure Learning of Bayesian Networks using Global Optimization

Chapter in thesis: 5

Name of Publication: IMA Journal of Applied Mathematics Journal

Co-author: Dr. Musa Mammadov

The contribution is related to the following:

Research framework

Mathematics

Development of an algorithm

Interpretation of results

Editing of manuscript

Percentage contribution: 50

Signature:− − − − − − − −−

Bibliography

[1] Akaike, H. (1978). Analysis of Cross Classified Data by AIC, Ann. Inst. Statist, 185-197.

[2] Allwein, E.L., Schapire, R.E., and Singer, Y. (2001). Reducing multiclass to binary: A unifying approach for margin classifiers. The Journal of Machine Learning Research, Vol 1, 113–141.

[3] Andreassen, S., Woldbye, M., Falk, B., and Anderson, S. K. (1987). A Causal Probabilistic Network for interpretation of electromyographic findings. In proceeding of Tenth International Joint Conference in Artificial Intelligence, Milan, Italy, 366-372.

[4] [11] Antao, P., Guedes Soares, C., Grande, O., and Trucco, P. (2009). Analysis of maritime accident data with BBN models. Safety, reliability and risk analysis theory, methods and applications, 65–73.

[5] Antal, P., Gezsi, A., Hullm, G., and Millinghoffer, A. (2006). Learning Complex Bayesian Network Features for Classification. Probabilistic Graphical Models, 9-16.

[6] Antal, P., Fannes, G., Timmerman, D., Moreau, Y., and Moor, B. (2003). Bayesian applications of belief networks and multilayer perceptrons for ovarian tumor classification with rejection. Artificial Intelligence in Medicine 29, 39-60.

[7] Bang, J., and Gil, D. (2002). Using Bayesian Networks with Hidden Nodes to Recognise Neural Cell Morphology. Springer-Verlag Berlin Heidelberg, 385-394.

[8] Bazaraa, M., Sherali, D., and Shetty, C. (1929). Nonlinear Programming; Theory and Algorithms. Wiley-Interscience.

[9] Beygelzimer, A., Langford, J., and Zadrozny, B. (2005). Weighted one-against-all. Proceedings Of The National Conference On Artificial Intelligence, Vol 20, 720–725.

[10] Bilmes, J., and Pernkopf, F. (2010). Efficient Heuristics for Discriminative Structure Learning of Bayesian Network Classifiers. Journal of Machine Learning Research, 2323-2360.

[11] Binder, J., Koller, D., Russell, S., and Kanazawa, K. (1997). Adaptive Probabilistic Networks with Hidden Variables. Machine Learning, 29, 213-244.

[12] Borglet C., and Kruse K. (2002). Graphical Models - Methods for Data Analysis and Mining, John Wiley and Sons, Chichester, UK.

[13] Boyd, S., and Vandenberghe, L. (2004). Convex Optimization, Cambridge University Press.

[14] Buntine, W. (1991). Theory Refinement on Bayesian Networks. Proceedings of the seventh conference on UAI, 52-60.

[15] Buntine, W. (1994). Operations for Learning with Graphical Models. Journal of Artificial Intelligence Research 2, 159-225.

[16] Burge, J., and Lane, T. (2005). Learning Class Discriminative Dynamic Bayesian Netwoks. International Conference on Machine Learning, Bonn, Germany.

[17] Burge, J. (2007). Learning Bayesian Networks from Hierarchically Related Data with a Neuroimaging Application. PhD. Dissertation. Computer Science. University of New Mexico.

[18] Burge, J., and Lane, T. (2007). Shrinkage Estimator for Bayesian Network Parameters, Springer-Verlag Berline Heidelberg, 67-78.

[19] Burnell, L., and Horvitz, E. (1995). Structure and Chance: Melding Logic and Probability for Software Debugging. Communications of the ACM, 38:3, 31-41.

[20] Campos, L., Fernandez-Luna, M., Gamez, J., and Puerta, M. (2002). Ant colony optimization for learning Bayesian networks. International Journal of Approximate Reasoning 31, 291–311.

[21] Campos, L. (2006). A Scoring Function for Learning Bayesian Networks based on Mutual Information and Conditional Independence Tests. Journal of Machine Learning Research 7, 2149-2187.

[22] Campos, L., and Zeng, Z. (2009). Structure Learning of Bayesian Network Using Constraints. International conference on machine Learning, Montreal, Canada.

[23] Cano, A., Gomez, M., Moral, S., and Abellan, J. (2007). Hill-climbing and branch-and-bound algorithms for exact and approximate inference in credal networks. International Journal of Approximate Reasoning, Elsevier, 261–280.

[24] Castillo, E., Gutierrez, J.M and Hadi, A.S. (1997). Expert Systems and Probabilistic Network Models. Springer Verlag, New York.

[25] Charniak, E. (1991). Bayesian Networks Without Tears. AI Magazine, 12 (4).

[26] Cheng, J., Bell, D., and Liu, W. (2002). Learning Belief Networks from Data. An Information Theory Based Approach. Artificial Intelligence, 137, 43-90.

[27] Chickering, D.M. (1996). Learning Bayesian Networks is NP-complete. Artificial Intelligence and statistics V, Springer, 121-130.

[28] Cooper, G.F., and Herskovits, E. A. (1990). Bayesian Method for Constructing Bayesian Belief Networks from Databases. Conference on Uncertainty in AI, 86-94.

[29] Crammer, K. and Singer, Y. (2002). On the algorithmic implementation of multiclass kernel-based vector machines, The Journal of Machine Learning Research, Vol 2, 265–292.

[30] Daly, R., and Shen, Q. (2009). Learning Bayesian Network Equivalence Classes with Ant Colony Optimization. Journal of Artificial Intelligence Research, Elsevier, 391–447.

[31] Darwiche, A. (2009). Modeling and reasoning with Bayesian networks, Cambridge University.

[32] Dietterich, T.G. and Bakiri, G. (1995). Solving Multiclass Learning Problems via Error-Correcting Output Codes, Journal of Artificial Intelligence Research, Vol 2, 263–286.

[33] Domingos, P. and Pazzani, M. (1996). Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. Thirteenth International Conference on Machine Learning, 105-112.

[34] Dougherty, J., Kohavi, R., and Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In Proceedings of the 12th International Conference on Machine Learning, 194–202.

[35] Drugan, M., and Wiering, M. (2010). Feature selection for Bayesian network classifiers using the MDL-FS score. International Journal of Approximate Reasoning, Elsevier, 695–717.

[36] Duda, R.O., and Hart, P.E. (1973). Pattern Classifcation and Scene Analysis. John Wiley and Sons, New York.

[37] Farmani, R., Henriksen, H., and Savic, D. (2009). An evolutionary Bayesian belief network methodology for optimum management of groundwater contamination. Environmental Modelling and Software, Elsevier, 303–310.

[38] Fayyad, U.M., and Irani, K.B. (1993). On the Handling of Continuous-Valued Attributes in Decision Tree Generation, Machine Learning 8, 87-102.

[39] Fayyad, U. M., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery: An overview. In Advances in Knowledge Discovery in Data Mining. AAAI Press, Menlo Park, CA, 1-34.

[40] Fenton, N., and Neil, M. (2010). Comparing risks of alternative medical diagnosis using Bayesian arguments, Journal of Biomedical Informatics, 485–495.

[41] Freund, Y., and Schapire, R. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting, Computational learning theory, 23–37.

[42] Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian Network Classifiers. Machine Learning 29 (2), 131–163.

[43] Friedman, N. (1997). Learning belief networks in the presence of missing values and hidden variables, Proceedings of Fourteenth International Conference on Machine Learning, 125–133.

[44] Fung, R., and Favero, B. D. (1995). Applying Bayesian networks to information retrieval. Communications of the ACM, 38(2), 42–57.

[45] Gavai, A. (2009). Bayesian Networks for Omics Data Analysis. PhD thesis, Wageningen University.

[46] Gilks W. R., Richardson S., and Spiegelhalter D.J. (1996). Markov Chain Monte Carlo in Practice, Chapman and Hall, London, UK.

[47] Greiner, R., Su, X., Shen, S., and Zhou, W. (2005). Structural Extention to Logistic Regression: Discriminative Parameter Learning of Belief Network Classifiers. Machin Learning 59, 297–322.

[48] Grossman, D., and Domingos, P. (2004). Learning Bayesian Classifiers by Maximizing Conditional Likelihood. International Conference on Machine Learning, Banff, Canada.

[49] Gu, Y., Peiris, D., Crowford, J., NcNicol, J., Marshall, B., and Jefferies, R. (1994). An Applications of Belief Networks to future crop production. In proceeding of Tenth International Conference in Artificial Intelligence for applications, San Antonio, 366–372.

[50] Hastie, T. and Tibshirani, R. (1998). Classification by pairwise coupling. Journal of Annals of Statistics, Vol 26, 451–471.

[51] Hekerman, D., Horvitz, E., and Natwani, B. (1992). Toward normative expert systems, Methods of information in Medicine, 90-105.

[52] Heckerman, D., Mamdani, A and Michael, W. (1995). Real-World Applications of Bayesian Networks. Communications of the ACM, 38-68.

[53] Heckerman, D., Geiger, D., Chickering, D.M. (1995). Learning Bayesian Networks: the Combination of Knowledge and Statistical Data. Machine Learning, 20, 197-243.

[54] Heckerman, D. (1997). Bayesian Networks for Data Mining. Data Mining and Knowledge Discovery, Kluwer Academic Publishers, Manufactured in the Netherlands, 79-119.

[55] Heckerman, D., Chickering, D.M., and Meek, C. (2004). Large-Sample Learning of Bayesian Networks is NP-Hard. Journal of Machine Learning Research, 1287-1330.

[56] Herskovits, E., Cooper. G. (1991). An Entropy-Driven System for Construction of Probabilistic Expert Systems from Databases. 6th Internatial Conference on Uncertainty in Artificial Intelligence (UAI90), Cambridge, MA, USA, Elsevier Science, New York, 54-62.

[57] Hinsbergen, V., Lint, V., and Zuylen, V. (2009). Bayesian committee of neural networks to predict travel times with confidence intervals. Transportation Research Part, Elsevier, 498-509.

[58] Holmes, D. (2008). Toward a Generalized Bayesian Network. Springer-Verlag Berlin Heidelberg, 281-288.

[59] Hsu, C., Wanga, K., and Chang, S. (2010). Bayesian decision theory for support vector machines: Imbalance measurement and feature optimization. Expert Systems with Applications, Elsevier.

[60] Janzura, M., and Nielsen, J. (2006). A Simulated Annealing-Based Method for Learning Bayesian Networks from Statistical Data. International Journal of Intelligent Systems, VOL 21, 335-348.

[61] Jensen, F. (1996). An Introduction to Bayesian Networks. Springer, New York.

[62] Jensen, F. V. (2001). Bayesian Networks and Decision Graphs. Springer-Verlag, New York.

[63] Jing, Y., Pavlovic, V., and Rehg, J. (2005). Efficient discriminative learning of Bayesian network classifier via Boosted Augmented Naive Bayes. The 22 nd International Conference on Machine Learning, Bonn, Germany.

[64] John, G. H. and Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. In Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence, 338-345.

[65] [13] Jones, B., Jenkinson, I., Yang, Z., and Wang, J. (2010). The use of Bayesian network modelling for maintenance planning in a manufacturing industry. Reliability Engineering and System Safety 95, 267-277.

[66] Jonsson, A. (2007). and Barto, A. Active Learning of Dynamic Bayesian Networks in Markov Decision Processes. Springer-Verlag Berlin Heidelberg, 273-284.

[67] Kabli, R., Herrmann, F., and McCall, J. (2007). A Chain-Model Genetic Algorithm for Bayesian Network Structure Learning. GECCO'07. Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM New York, NY, USA.

[68] Kafai, M., and Bhanu, B. (2012). Dynamic Bayesian Networks for Vehicle Classification in Video , 100 IEEE Transactions on industrial information, VOL 8, NO. 1.

[69] Keogh, E., and Pazzani, M. (1999). Learning Augmented Bayesian Classifiers. A Comparison of Distributionbased and Classification-Based Approaches. In 7th Intl. Workshop Artificial Intelligence and Statistics, 225-230.

[70] Kitakoshi, D., Shioya, H., and Nakano, R. (2010). Empirical analysis of an on-line adaptive system using a mixture of Bayesian networks. Information Sciences, Elsevier, 2856-2874.

[71] Kollat, J., Reed, P., and Kasprzyk, J. (2008). A new epsilon-dominance hierarchical Bayesian optimization algorithm for large multiobjective monitoring network design problems. Advances in Water Resources, Elsevier, 828-845.

[72] Kontkanen, P., Wettig, H., and Myllymki, P. (2008). NML Computation Algorithms for Tree-Structured Multinomial Bayesian Networks, EURASIP Journal on Bioinformatics and Systems Biology.

[73] Kontkanen, P., Silander, T., Roos, T., and Myllymki, P. (2008). Factorized NML Criterion for Learning Bayesian Network Structures. 4th European Workshop on Probabilistic Graphical Models.

[74] Kontkanen, P., Silander, T., Roos, T., and Myllymki, P. (2008). Bayesian Network

Structure Learning Using Factorized NML Universal Models. Information Theory and Applications Workshop (ITA-08), IEEE Press.

[75] Langley, P., Iba, W., and Thompson, K. (1992). An Analysis of Bayesian Classifiers. In 10th International Conference Artificial Intelligence, AAAI Press and MIT Press, 223-228.

[76] Langley, P., and Sage, S. (1994). Induction of Selective Bayesian Classifiers. 10th International Conference Uncertainty in Artificial Intelligence, Morgan Kaufmann, 399-406.

[77] Larranaga, P., Murga, R., Poza, M., and Kuijpers, C. (1996). Structure Learning of Bayesian Networks by Hybrid Genetic Algorithms. In Learning from Data: AI and Statistics V, Lecture Notes in Statistics 112. D. Fisher, H.-J. Lenz (eds.), New York, NY: Spriger-Verlag, 165-174.

[78] Larranaga, P., Poza, M., Yurramendi, Y., Murga, H., and Kuijpers, H. (1996). Structure Learning of Bayesian Networks by Genetic Algorithms: A Performance Analysis of Control Parameters. IEEE Transactions on Pattern Analysis and Machine Intelligence archive Vol 18 Issue 9.

[79] Larranaga, P., Sierra, B., Gallego, J., Michelena, J., and Picaza. M. (1997). Learning Bayesian Networks by Genetic Algorithms: A case study in the prediction of survival in malignant skin melanoma. Artificial Intelligence, Elsevier, Vol 1211/1997, 261-272.

[80] Zhao, L. (2012). Learning from Noisy Data: Robust Data Classification, PhD Thesis.

[81] Levvit, T., Agosta, J., and Binford, T. (1990). Model Based Influence diagrams for machine vision, Uncertainty in Artificial Intelligence 5, 371-388.

[82] Li, J., Zhang, Ch., Wang, T., and Zhang, Y. (2007). Generalized Additive Bayesian Network Classifiers, 20th International Joint Conference on Artifical Intelligence Table of Contents Hyderabad, India, 913-918.

[83] Liaoa, W., and Ji, Q. (2009). Learning Bayesian Network Parameters under Incomplete Data with Domain Knowledge. Pattern Recognition, Elsevier, 3046–3056.

[84] Ji, Z., Zhong, H., Hu. R., and Liu, C. (2009). A Bayesian Network Learning Algorithm Based on Independence Test and Ant Colony Optimization. Acta Automatica Sinica, Vol 35, No 3.

[85] Marinescu, R., and Dechter, R. (2009). AND/OR Branch-and-Bound search for combinatorial optimization in graphical models. Artificial Intelligence, Elsevier, 1457-1491.

[86] Mammadov, M.A., Rubinov A.M, and Sniedovich, M. (2004). A New Global Optimization Algorithm Based on Dynamical Systems Approach. 6th International Conference on Optimization: Techniques and Applications, Ballarat, Australia.

[87] Mammadov, M.A., Rubinov A.M. and Yearwood, J. (2005). Dynamical Systems Described by Relational Elasticities with Applications to Global Optimization. In Continuous Optimisation: Current Trends and Modern Applications, V.Jeyakumar and A. Rubinov, Eds. Springer, 365-385.

[88] Mammadov, M., and Taheri, S. (2010). A Globally Convergent Optimization Algorithm for Systems of Nonlinear Equations. Third Global Conference on Power Control and Optimization, Gold Coast, Australia.

[89] Mitchell, T.M. (1997). Machine Learning. McGraw-Hill Companies, 2-4.

[90] Mitchell, T.M, Niculescu, R., and Rao, R. (2006). Bayesian Network Learning with Parameter Constraints. Journal of Machine Learning Research, 1357-1383.

[91] Myers, J.W., Laskey, K.B., DeJong, K.A. (1999). Learning Bayesian networks from incomplete data using evolutionary algorithms, in Proceedings of the International Conference on Genetic Algorithms (GECCO-99).

[92] Myllymaki, P. (2005). Advantages of Bayesian networks in data mining and knowledge discovery, http://www.bayesit.com/docs/advantages.html.

[93] Najafi, M., Auslander, M., Bartlett, P., Haves, P. (2008). Application of Machine Learning in Fault Diagnostics of Mechanical Systems. Proceedings of the World Congress on Engineering and Computer Science.

[94] Neapolitan, R. (2003). Learning Bayesian Networks. Prentice-Hall, Inc.

[95] Neapolitan, R. (2009). Probabilistic Methods for Bioinformatics with an Introduction to Bayesian Networks. Morgan Kaufmann, APR.

[96] Niculescu, R. (2005). Exploiting Parameter Domain Knowledge for Learning in Bayesian Networks. PhD thesis, School of Computer Science Carnegie Mellon University.

[97] Nocedal, J., and Wright, S.J. (1999). Numerical Optimization, Springer Series in Operations Research.

[98] Nordgard, D.E., and Sand, K. (2010). Application of Bayesian networks for risk analysis of MV air insulated switch operation, Reliability Engineering and System Safety 95, 1358-1366.

[99] Park, H., and Cho, S. (2006). An Efficient Attribute Ordering Optimization in Bayesian Networks for Prognostic Modeling of the Metabolic Syndrome. Springer-Verlag Berlin Heidelberg, 381 -391.

[100] Pearl, J. (1988). Probabilistic Reasoning in Intelligent Systems. Networks of Plausible Inference, Morgan Kaufmann.

[101] Pearl, J. (2000). Causality: Models, Reasonings and Inference, Cambridge University Press.

[102] Pernkopf, F., and Wohlmayr, M. (2009). On Discriminative Parameter Learning of Bayesian Network Classifiers. Springer-Verlag Berlin Heidelberg, 221-237.

[103] Pitiot, P., Coudert, T., Geneste, L., and Baron. C. (2010). Hybridation of Bayesian networks and evolutionarya lgorithms for multi-objective optimization in an integrated product design and project management context. Engineering Applications of Artificial Intelligence, Elsevier, 830-843.

[104] Quinlan, J.R. (1993). C4.5: programs for machine learning, Morgan Kaufmann.

[105] Retzer, J., Soofi, E., and Soyer, R. (2009). Information importance of predictors: Concept, measures, Bayesian inference, and applications. Computational Statistics and Data Analysis, Elsevier, 2363-2377.

[106] Rifkin, R. and Klautau, A. (2004). In defense of one-vs-all classification, The Journal of Machine Learning Research, Vol 5, 101–141.

[107] Russell, S., Binder, J., and Koller, D. (1994). Adaptive probabilistic networks, Technical Report UCB/CSD-94-824.

[108] Sahami, M. (1996). Learning Limited Dependence Bayesian Classifiers. In the 2nd International Conference. Knowledge Discovery and Data mining (KKD), 335-338.

[109] Sahin, F., Yavuz, M., Arnavut, Z., and Uluyol, O. (2007). Fault diagnosis for airplane engines using Bayesian networks and distributed particle swarm optimization. Parallel Computing, Elsevier, 124-143.

[110] Sebastiani, P., and Ramoni, M. (2000). Bayesian inference with missing data using bound and collapse. Journal of Computational and Graphical Statistics 9, 779-800.

[111] Schleip, C., Rais, A., and Menzel, A. (2009). Bayesian analysis of temperature sensitivity of plant phenology in Germany. Agricultural and Forest Meteorology, Elsevier, 1699-1708.

[112] Schwarz, G. (1978). Estimating the Dimension of a Model. Annals of Stastics, 461-464.

[113] Shafer, G., and Pearl, J. (1990). Readings in Uncertain Reasoning. Morgan Kaufmann, San Mateo, CA.

[114] Silander, T., Roos, T., and Myllymaki, P. (2010). Learning locally minimax optimal Bayesian networks. International Journal of Approximate Reasoning, Elsevier, 544-557.

[115] Smaili, C., Najjar, M.E., and Charpillet, F. (2007). Multi-Sensor Fusion Method using Dynamic Bayesian Network for Precise Vehicle Localization and Road Matching, Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence, Vol 01, 146-151.

[116] Spiegelhalter, D.J. and Lauritzen, S. (1990). Sequential Updating of Conditional Probabilities on Directed Graphical Structure Networks, 20, 579-605.

[117] Spiegelhalter, D.J., Dawid, P., Lauritzen, S.L., and Cowell, R. (1993). Bayesian analysis in expert systems, Statistical Science 8, 219-282.

[118] Spirtes, P., Glymour, C., and Sheines, R. (1993). Causation, Prediction, and Search.

[119] Stajduhar, I., Dalbelo-Basic, B., and Bogunovic, N. (2009). Impact of censoring on learning Bayesian networks in survival modelling. Artificial Intelligence in Medicine, Elsevier, 199-217.

[120] Sun, W., and Yuan, Y.X. (2006). Optimization Theory and Methods. Nonlinear Programming, Springer.

[121] Taheri, S., and Mammadov, M. (2012). Solving Systems of Nonlinear Equations using a Globally Convergent Optimization Algorithm. Global Journal of Technology and Optimization, Vol 3, 132-138.

[122] Taheri, S., Mammadov, M., and Seifollahi, S. (2012). Globally Convergent Optimization Methods for Unconstrained Problems. Optimization Journal.

[123] Takekawa, T., and Fukai, T. (2009). A novel view of the variational Bayesian clustering. Neuro computing, Elsevier, 3366-3369.

[124] Tan, P.N. and Steinbach, M. and Kumar, V. (2006). Introduction to data mining. Pearson Addison Wesley Boston.

[125] Thiesson, B. (1994). Score and Information for Recursive Exponential Modeles with Incomplete Data. 13th Conference on Uncertainety in Artificial Intelligence.

[126] Tian, F., Li, H., Wang, Z., and Yu, J. (2007). Learning Bayesian Networks based on a Mutual Information Scoring Function and EMI Method. Springer-Verlag Berlin Heidelberg, 414-423.

[127] Trucco, P., De Ambroggi, M., and Grande, O. (2009). Quantitative analysis of the anatomy and effectiveness of occupational safety culture. In Martorell, Guedes Soares, Barnett, editors. Safety, reliability and risk analysi, Vol 2. Leiden: CRC, 1431-1438.

[128] Tsoumakas, G. and Katakis, I. (2007). Multi-label classification: An overview, International Journal of Data Warehousing and Mining, Vol 3, 1–13.

[129] Vapnik, V.N. (1998). Statistical learning theory, Wiley-Interscience.

[130] Verrona, S., Li, J., and Tiplica. T. (2010). Fault detection and isolation of faults in a multivariate process with Bayesian network, Journal of Process Control, 902-911.

[131] Weber, P., Medina-Oliva, G., Simon, C., and Iung, B. (2010). Overview on Bayesian networks applications for dependability, riskanalysis and maintenance areas. Engineering Applications of Artificial Intelligence, 671-682.

[132] Weston, J. and Watkins, C. (1998). Multi-class support vector machines. Pattern Recognition.

[133] Wilson, A., Fern, A., and Tadepalli, F. (2010). Incorporating Domain Models into Bayesian Optimization for RL. Springer-Verlag Berlin Heidelberg, 467-482.

[134] Wittig, F., and Jameson, A. (2000). Exploiting qualitative knowledge in the learning of conditional probabilities of Bayesian networks. Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, 644-652.

[135] Wolbrecht, E., DAMBROSIO, B., Paasch, R., and Kibry, D. (2000). Monitoring and diagnosis of a multistage manufacturing process using Bayesian networks. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 53-67.

[136] Yahya, A., Mahmod, R., and Ramli, R. (2010). Dynamic Bayesian networks and variable length genetic algorithm for designing cue-based model for dialogue act recognition. Computer Speech and Language, Elsevier, 190-218.

[137] Yan, J., and Cercone, N. (2010). Bayesian network modeling for evolutionary genetic structures. Computers and Mathematics with Applications, Elsevier, 2541-2551.

[138] Yang, Y. (2003). Discretization for Naive-Bayes Learning, PhD Thesis.

[139] Yatsko, A., Bagirov, A., and Stranieri, A. (2011). On the Discretization of Continuous Features for Classification. In the proceedings of Ninth Australasian Data Mining Conference (AusDM 2011), Ballarat, Australia, Vol 125.

[140] Zhao, J., Sun, J., Xu, W., and Zhou, D. (2009). Structure Learning of Bayesian Networks Based on Discrete Binary Quantum-behaved Particle Swarm Optimization Algorithm. ICNC 09 Proceedings of the Fifth International Conference on Natural Computation, Vol 6, IEEE Computer Society Washington, DC, USA.