LEARNING PATIENT-SPECIFIC MODELS FROM CLINICAL DATA

by

Shyam Visweswaran

MBBS, JIPMER, Pondicherry, 1987

MS (Physiology and Biophysics), University of Illinois at Urbana-Champaign, 1995

Submitted to the Graduate Faculty of

the School of Arts and Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2007

UNIVERSITY OF PITTSBURGH

SCHOOL OF ARTS AND SCIENCES

This dissertation was presented

by

Shyam Visweswaran

It was defended on

September 25, 2007

and approved by

Gregory F. Cooper, Department of Biomedical Informatics and Intelligent Systems Studies Program, University of Pittsburgh

Tom Mitchell, Machine Learning Department, Carnegie Mellon University

Marek J. Druzdzel, School of Information Sciences and Intelligent Systems Studies Program, University of Pittsburgh

Milos Hauskrecht, Department of Computer Science and Intelligent Systems Studies Program, University of Pittsburgh

Dissertation Advisor: Gregory F. Cooper, Department of Biomedical Informatics and Intelligent Systems Studies Program, University of Pittsburgh

LEARNING PATIENT-SPECIFIC MODELS FROM CLINICAL DATA

Shyam Visweswaran, MBBS, MS, PhD

University of Pittsburgh, 2007

A key purpose of building a model from clinical data is to predict the outcomes of future individual patients. This work introduces a Bayesian patient-specific predictive framework for constructing predictive models from data that are optimized to predict well for a particular patient case. The construction of such *patient-specific models* is influenced by the particular history, symptoms, laboratory results, and other features of the patient case at hand. This approach is in contrast to the commonly used *population-wide models* that are constructed to perform well on average on all future cases.

The new patient-specific method described in this research uses Bayesian network models, carries out Bayesian model averaging over a set of models to predict the outcome of interest for the patient case at hand, and employs a patient-specific heuristic to locate a set of suitable models to average over. Two versions of the method are developed that differ in the representation used for the conditional probability distributions in the Bayesian networks. One version uses a representation that captures only the so called *global structure* among the variables of a Bayesian network and the second representation captures additional *local structure* among the variables.

The patient-specific methods were experimentally evaluated on one synthetic dataset, 21 UCI datasets and three medical datasets. Their performance was measured using five different performance measures and compared to that of several commonly used methods for constructing predictive models including naïve Bayes, C4.5 decision tree, logistic regression, neural networks,

k-Nearest Neighbor and Lazy Bayesian Rules. Over all the datasets, both patient-specific methods performed better on average on all performance measures and against all the comparison algorithms. The *global structure* method that performs Bayesian model averaging in conjunction with the patient-specific search heuristic had better performance than either model selection with the patient-specific heuristic or non-patient-specific Bayesian model averaging. However, the additional learning of local structure by the *local structure* method did not lead to significant improvements over the use of global structure alone. The specific implementation limitations of the local structure method may have limited its performance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

First and foremost I express my gratitude to my advisor professor Greg Cooper for his patient guidance, invaluable advice and constant support during this research. I thank my other committee members professor Marek Drudzel, professor Milos Hauskrecht and professor Tom Mitchell for helpful discussions and critical feedback. In addition, I thank professor Roger Day for his stimulating discussions and assistance with the statistical analyses.

I thank Cleat Szczepaniak, Joseph Cummings, Rose Ann Thomas, Toni Porterfield, Kimberlee Barnhart and William Milberry for making my stay at DBMI pleasant and enjoyable.

I acknowledge the financial support of the National Library of Medicine that enabled me to pursue my interest in biomedical informatics. Without this financial support it would not have been possible for me to complete the doctoral degree.

Finally, I cannot thank enough my family for supporting my endeavor of becoming an informatics researcher. I specially appreciate my wife Rajani's unwavering support and love and my daughter Anushka's cheerful demeanor. I am extremely grateful to my parents for their support and inspiration that encouraged me to pursue my dreams.

1.0 INTRODUCTION

Making predictions, typically under uncertainty, is a common theme in clinical care activities. Critical activities in clinical care include risk assessment, diagnosis, and prognosis, all of which entail making predictions in individuals. Risk assessment implies predicting the future occurrence of disease from current exposure to risk factors; diagnosis entails predicting the possibility of disease from current symptoms, signs and tests; and, prognosis involves predicting the future course and outcome of disease both with and without therapy [1, 2]. The better these predictions can be performed, the better the decisions and the ensuing outcomes are likely to be both for the individual and for society at large. Therefore, finding ways to make better predictions is an important problem.

Typically, the clinician makes these predictions implicitly from knowledge obtained from medical training as well as experience acquired from past patient care. This is occasionally facilitated by paper-based guidelines and flowcharts derived from simple predictive models. Such use of explicit models can help in making better predictions, enhance clinical decision making, improve patient outcomes, and reduce healthcare costs. In recent years, medical artificial intelligence and machine learning techniques are being increasingly used to learn sophisticated predictive models in the biomedical domain. However, much work remains to be done in improving the performance of such models and incorporating their use in routine clinical care.

One clear trend in healthcare is the accumulation of more and more data per patient that is then available for clinical decision-making. Today's clinician not only has to assimilate large amounts of data but also has to integrate diverse types of patient data: demographic, environmental, clinical, genetic, imaging, and outcomes in the course of patient care. Predictive models can aid the clinician's decision-making in the face of this data deluge. Another trend in healthcare is the increasing use of computers in clinical care and the availability of patient data in electronic form which allows for automatic processing of such data. Thus, it is becoming feasible to use sophisticated models that require computationally intensive modeling methods that have better predictive performance over simple paper-based flowchart models. For example, it seems plausible that the future will see more computationally intensive methods that construct distinct models for each individual from patient data in clinical computer systems. The development of such methods that learn models tailored to an individual's characteristics is the focus of the research described in this dissertation.

Even modest improvements in predictive performance can have significant impact on healthcare in terms of improved patient care, better outcomes and reduced costs. For example, in [3] the authors estimate that if improved prediction of dire outcomes in pneumonia can reduce hospital admissions of pneumonia patients by just one percent this can result in 89 million dollars of savings per year in the United States without any expected decrease in clinical outcomes. Thus, finding ways to improve predictive performance of current modeling techniques is an important problem.

Two fields that have focused on the learning and application of predictive models are statistics and machine learning. Both fields use similar terminology. A *variable* (also known as attribute) is a quantity that describes an aspect of an object of the world. A *feature* is the

specification of a variable and its value. For example, *eye color* is a variable and "*eye color* = black" is a feature. Though often the term feature is used as a synonym for variable, in this document the term feature is used exclusively to refer to a variable-value pair. A *case* (also known as example, instance or record) is a single object of the world and is described by a list of features. A *dataset* (also known as database) is a collection of cases.

Predictive models can be constructed either manually or by methods that automatically extract relevant information from datasets. Classical statistical methods typically involve substantial expert input for deriving the model. In semi-automatic methods, the model structure (e.g., which variables are considered dependent) is derived from experts and the parameters of the structure (e.g., coefficients in logistic regression) are extracted from datasets. Completely automated methods learn both the structure and the parameters automatically from the dataset and are the focus in machine learning. The research described in this dissertation investigates automatic machine learning methods that learn models (tailored to an individual's characteristics) from data.

Among the machine learning methods, classification algorithms are often used for learning predictive models from clinical data. Examples of such algorithms include logistic regression, neural networks, classification trees (also know as decision trees), Bayesian networks and support vector machines. Typically, these methods induce a single model from a training set of cases, with the intent of applying it to all future patient cases. I call such a model a *population-wide* model because it is intended to be applied to an entire population of future cases. A population-wide model is optimized to predict well on average when applied to expected future cases.

Recent research in machine learning has shown that inducing models that are specific to the particular features of a given instance can improve predictive performance [4]. I call such a model an *instance-specific* model since it is constructed specifically for a particular instance (case). The structure and parameters of an instance-specific model are specialized to the particular features of an instance, so that it is optimized to predict especially well for that instance. In the context of clinical prediction models, I call such a model a *patient-specific* model, because the learning of the model is influenced by the particular history, symptoms, laboratory results, and other features of the patient case being predicted. That is, the structure and parameters of the model are influenced by the patient case being predicted. In this dissertation the term patient-specific is used, though much of the description of patient-specific models

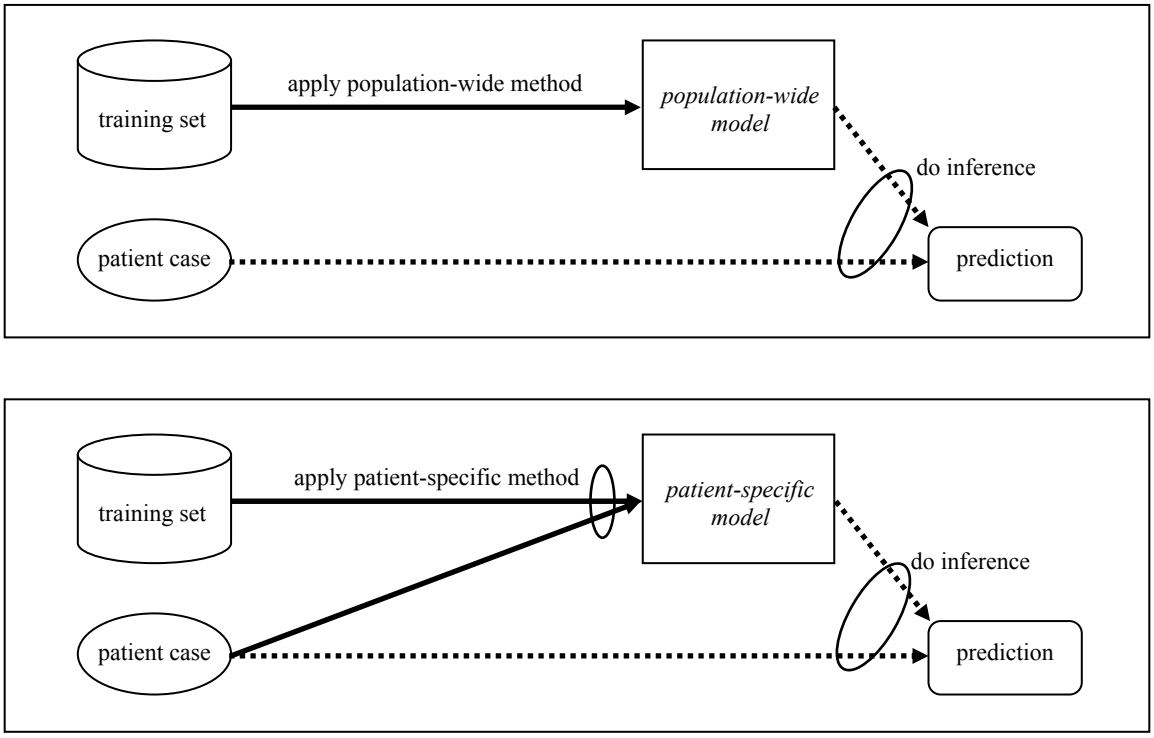


Figure 1-1: A general characterization of the induction and inference of population-wide (top panel) and patient-specific (bottom panel) models. In the bottom panel, there is an extra arc from *patient case* to *model*, because the structure and parameters of the model are influenced by the features of the patient case at hand.

applies to the more general instance-specific models. The goal of inducing a patient-specific model is to have optimal prediction *for the patient case at hand*. This is in contrast to the construction of a population-wide model where the goal is to have optimal predictive performance *on average on all future patient cases*.

Predictive modeling consists of two steps: induction of a model or models from a training set of cases and inference of the variable of interest in a patient case at hand to derive a prediction. Inference always involves the use of the features of the patient case at hand in conjunction with a model. In the biomedical literature, the adjective patient-specific in the context of predictive modeling is sometimes used more loosely to refer to model inference. However, the use of patient features for inference does not necessarily make the model or the method used for induction patient-specific. I designate only those methods as patient-specific that use the features of the patient case at hand in conjunction with a training set of cases in order to induce a model. This situation is illustrated in Figure 1-1 which shows that the patient case being predicted is used for inference in both population-wide and patient-specific methods; however, for induction, only the patient-specific method uses the patient case.

This work presents a decision-theoretic framework for induction of patient-specific models, and investigates methods for learning Bayesian network models for prediction in a patient-specific manner.

1.1 OVERVIEW OF PROPOSED PATIENT-SPECIFIC METHOD

There are several possible approaches for learning predictive models that are relevant to a single patient case. One approach is to learn a model from a subset of cases in the dataset that consist of

patients that are similar in some way to the patient case at hand. Another approach is to learn a model from a subset of variables that are pertinent in some fashion to the patient case at hand. A third approach, applicable to model averaging where a set of models is collectively used for prediction, is to locate a set of models that are relevant to prediction for the patient case at hand.

In this work, I investigate a new method for learning predictive models that uses (1) Bayesian network models, (2) carries out Bayesian model averaging over a set of models to predict the outcome of interest for the patient case at hand, and (3) employs a patient-specific heuristic to locate a set of suitable models to average over. The remainder of this section gives a brief description of each of these characteristics.

Bayesian networks (BNs) are probabilistic graphical models that provide a powerful formalism for representation, reasoning and learning under uncertainty [5-7]. These graphical models are sometimes referred to as probabilistic networks, belief networks or Bayesian belief networks. The last two decades have witnessed significant advances in the theoretical development of BNs as well as in their application to a growing number of domains. A BN combines a graphical representation with numerical information to represent a probability distribution over a set of random variables in a domain. The graphical representation constitutes the BN *structure*, and it explicitly highlights the probabilistic independencies among the domain variables. The complementary numerical information constitutes the BN *parameterization*, which quantifies the probabilistic relationships among the variables.

At the outset, BNs were constructed manually from knowledge acquired from domain experts, which proved to be both time-consuming and difficult. Subsequent advances in BN learning have culminated in numerous machine learning algorithms that learn both model

structure and model parameters automatically from data. Such algorithms underlie most current methods that use BNs for predictive modeling.

Typically, methods that learn predictive models from data, including those that learn BN models, perform *model selection*. In model selection a single good model is selected that summarizes the data well; it is then used to make future predictions. However, given finite data, there is uncertainty in choosing one model to the exclusion of all others, and this can be especially problematic when the selected model is one of several distinct models that all summarize the data more or less equally well. A coherent approach to dealing with the uncertainty in model selection is Bayesian model averaging (BMA). BMA is the standard Bayesian approach wherein the prediction is obtained from a weighted average of the predictions of a set of models, with better models influencing the prediction more than others. In practical situations, the number of models to be considered is enormous and averaging the predictions over all of them is infeasible. A pragmatic approach is to average over a few good models, termed *selective Bayesian model averaging*, which serves to approximate the prediction obtained from averaging over all models. The patient-specific method performs selective Bayesian model averaging over a set of models that have been selected in a patient-specific fashion.

The patient-specific method learns both the structure and parameters of BNs automatically from data. The patient-specific characteristic of the method is motivated by the intuition that in constructing predictive models, all the available information should be utilized *including available knowledge of the features of the current patient case*. Specifically, the patient-specific method uses the features of the patient case to inform the BN learning algorithm to select models that differ considerably in their predictions for the outcome of interest in the

patient case at hand. The differing predictions of the selected models are then combined to predict the outcome of interest.

1.2 AIMS OF THE DISSERTATION

The main aim of this dissertation is to introduce a new patient-specific method and evaluate whether it yields better predictive performance than commonly applied population-wide methods. The performance of the patient-specific method outlined in the previous section is compared to six other machine learning algorithms. Four of these comparison methods are standard machine learning algorithms that induce population-wide models. These are naïve Bayes, C4.5 decision tree, logistic regression and neural networks. Two other comparison methods, *k*-Nearest Neighbor and Lazy Bayesian Rules, are instance-based methods. In addition, the patient-specific method is compared to a *model selection* version of the method whereby a single model is selected for prediction. The methods are compared and evaluated on five performance measures: misclassification rate, area under the Receiver Operating Characteristic (ROC) curve, squared error, logarithmic loss, and a calibration score. The datasets on which the experiments are conducted include 21 publicly available datasets obtained from the UCI Machine Learning repository and three real world medical datasets.

The null (H_0) and alternate hypotheses (H_1) for the primary aim are as follows:

H_0: Patient-specific Bayesian network models do not predict better than population-wide models.

H_1: For at least some performance measures patient-specific Bayesian network models predict better than population-wide models.

1.3 OVERVIEW OF DISSERTATION

Chapter 2 provides relevant background to set the context for patient-specific modeling and surveys some of the related work in machine learning and in medical predictive modeling. Chapter 3 introduces the Bayesian network formalism, describes several representations for modeling BNs, and reviews methods for learning BNs for data. In particular, it focuses on the Bayesian approach to BN learning, learning BNs for classification, and Bayesian model averaging over BNs.

Chapter 4 describes in detail the proposed patient-specific learning method, the patient-specific search for BNs and the patient-specific score for evaluating BNs. Chapter 5 presents the results of the experimental evaluation of the patient-specific method and compares its performance to that of the algorithms listed in the previous section. Chapter 6 summarizes the contributions of this dissertation and presents some potential extensions of this work for future research.

2.0 BACKGROUND

This chapter provides the background and context for patient-specific modeling and reviews related work in machine learning and in medical predictive modeling. Section 2.1 highlights the importance of improving predictive modeling in healthcare. Section 2.2 provides illustrative examples where patient-specific modeling can improve upon population-wide models and Section 2.3 provides a decision theoretic comparison of the two paradigms. The patient-specific method described in the dissertation is characterized by the use of model averaging for prediction, induction of models in a lazy fashion, and the use of atemporal data. The following sections provide background on these characteristics of the patient-specific method. Section 2.4 compares model selection with model averaging and Section 2.5 compares lazy and eager methods for learning models. Section 2.6 presents time-varying patient-specific modeling as an extension of the patient-specific method. The final two sections summarize some of the relevant literature in machine learning and medicine respectively that is related to predictive modeling.

2.1 PREDICTION IN CLINICAL MEDICINE

A common characteristic of clinical care activities like risk assessment, diagnosis, and prognosis in individuals is making predictions [1]. *Risk assessment* is an important component of

preventive healthcare where healthcare providers judge the chances of an individual developing future medical problems based on the individual's past and current history of exposure to risk factors. For example, individuals who are judged to be at high risk for developing hypertension may undergo further medical evaluation and testing and may be advised to make changes to their lifestyle and diet. Accurate risk assessment is necessary to identify individuals at risk for developing hypertension correctly, since lifestyle and dietary changes are very difficult to initiate and maintain for most individuals.

Accurate and timely *diagnosis* in individuals with current symptoms is important in making decisions such as the need for additional testing, for choosing appropriate therapy, and the need for hospitalization. Inappropriate decisions arising from erroneous diagnoses can lead to unnecessary distress and incorrect therapy for the individual as well as needless expenditure of healthcare resources.

Prognosis entails the prediction of the course and outcome of disease and is an important component of management of an individual with a disease [2]. Given accurate diagnoses, choosing the appropriate therapy entails predicting accurately the course and outcome of the disease in the individual under various available therapies. For example, for the same disease, the optimal medication for one individual may be different from that for another individual due to differences in genetic and environmental characteristics. Thus, improvement in predictive performance is an important healthcare problem since it has the potential to improve clinical decision-making, which in turn can lead to better outcomes in patients. In addition, efficient use of healthcare resources depends on being able to determine accurately when and where a resource is likely to be useful, which in turn depends on accurately anticipating patients' healthcare-resource requirements.

A second common feature of clinical care activities that involve prediction is uncertainty. Uncertainty in medicine arises from various causes: uncertainty from incomplete medical knowledge, uncertainty from incomplete patient data, and uncertainty from noisy patient data [1]. Thus, risk assessment, diagnosis, and prognosis in individuals are associated with various degrees of uncertainty. Judging and handling uncertainty appropriately is imperative for improving predictions.

One important way to assist healthcare providers in making better predictions under uncertainty is to supplement their clinical judgments with predictions from mathematical models. Such models embedded in clinical computer systems have the potential to complement the clinician's assessment at the point-of-care. Clinical computer systems that help in clinical decision-making, called clinical decision support systems (CDSSs), have been in developed since the early 1960s. CDSSs have the potential to improve healthcare by improving patient safety, by improving quality of care, and by improving efficiency in health care delivery [8]. Though researchers have developed numerous CDSSs, few are in routine clinical use. A variety of factors have been implicated in this failure, such as, lack of deep causal knowledge in the medical domain, poor user interfaces, failure to fit into the clinical workflow, and inadequate predictive performance of the models incorporated into such systems [9].

Numerous decision-support models have been developed and employed in CDSSs. The earliest CDSSs used simple branching logic (equivalent to a flowchart). For example, Bliech and his colleagues developed a computerized flowchart using branching logic to diagnose acid-base disorders [10]. This was followed by the development of rule-based models that were incorporated into expert system CDSSs. A rule-based model represents the domain knowledge as a set of rules that is applied to patient data by a logical reasoning engine. A well known example

of such a system is MYCIN that was introduced by Shortliffe to diagnose organisms causing infections in patients [11]. Simple logical and rule-based models, however, could not handle well the ubiquitous uncertainty in clinical decision making.

One of the earliest theoretical developments of a CDSS model, outlined in a classic paper published in 1959 by Ledley and Lusted [12], stated that logic should be combined with probabilistic reasoning for automated reasoning and prediction in the medical domain. With the development of Bayesian network models (also known as probabilistic networks, belief networks or Bayesian belief networks) in the late 1980s, full-fledged probabilistic reasoning using such models was implemented in the medical domain. Examples of medical models based on Bayesian networks include the Pathfinder [13, 14], a pathology diagnostic system for diagnosing lymph-node diseases, and the probabilistic version of INTERNIST-1/QMR [15, 16], an internal medicine diagnostic system for diagnosing medical disorders. A PubMed search showed that 378 articles have been published in the biomedical literature in the last 10 years (from July 1997 to June 2007) with one of the following phrases in the title or abstract: “Bayesian network”, “Bayesian networks”, “probabilistic network”, “probabilistic networks”, “belief network”, “belief networks”, “Bayesian belief network” or “Bayesian belief networks”. Thus, Bayesian networks remain popular and their application to biomedical and clinical problems is an active area of research.

In this dissertation, the focus is on probabilistic models that make predictions in the clinical domain, and, in particular, on models that are learned from a training dataset of patient cases. Datasets in the clinical domain can be broadly categorized into two major types. Observational data are often used for assessing diagnostic conditions; experimental data (e.g.,

from randomized controlled trials) are often used for evaluating therapeutic interventions. The data collected in both these scenarios can be used for inducing predictive models.

Numerous statistical and machine learning methods have been developed for learning probabilistic models from a dataset of cases and several of them have been applied for learning predictive models from clinical and biomedical data [17]. A survey of the machine learning literature and the medical literature is provided in sections 2.8 and 2.9 respectively. Almost always, these are *population-wide methods* that learn a single *population-wide model* from a set of known patient cases, with the intent of applying it to all future patient cases. By design *population-wide models* are expected to perform well on average on all future patient cases, but can potentially perform poorly on a particular patient case. Little literature is available on methods that take into account the current patient case while learning a model. Such a *patient-specific method* constructs a *patient-specific model* for each future patient that is optimized to predict especially well for the particular patient case at hand. The following section illustrates how patient-specific models can have better performance than population-wide models.

2.2 PATIENT-SPECIFIC METHODS CAN HAVE BETTER PERFORMANCE

Figure 1-1 illustrates the key difference between population-wide and patient-specific models: the patient-specific model is constructed from data in the training set, as well as, from available data about the particular patient case to which it will be applied. In contrast, the population-wide model is constructed only from data in the training set. Thus, intuitively, the extra information available to the patient-specific method can facilitate inducing a model that provides better prediction for the patient-specific case. In patient-specific modeling, different patient cases will

potentially result in different models, because the cases contain potentially different values for the features. The patient-specific models may differ in the variables included in the model (variable selection or also known as feature selection), in the interaction among the included variables (encoded in the structure of the model), and in the strength of the interaction (encoded in the parameters of the model). Another approach is to select a subset of the training data that are similar in their feature values to those of the patient case at hand and learn the model from the subset. A generalization of this is to weight the cases in the training dataset such that cases that are more similar to the patient case are assigned greater weights than others, and then learn the model from the weighted dataset. The following are two illustrative examples where patient-specific methods may perform better than population-wide methods.

Variable selection. Many model induction methods implicitly or explicitly perform variable selection, a process by which a subset of the domain variables is selected for model construction. For example, logistic regression is often used with a stepwise variable selection process. A patient-specific version of logistic regression may select different variables for different patients being predicted, compared to the standard population-wide version that selects a single subset of variables. In the context of a healthcare scenario in the not-too-distant future, consider a gene G that has several alleles. Suppose that allele a1 is rare, and it is the only allele that predicts the development of disease D; indeed, it predicts D with high probability. For future patients, the aim is to predict $P(D \mid G)$. In a population-wide logistic regression model, G may not be included as a predictor (variable) of D, because in the vast majority of cases in the dataset $G \neq$ a1 and D is absent, and having G as a predictor would just increase the overall noise in predicting D. In contrast, if there is a patient case at hand in which $G =$ a1, then the training data may contain enough cases to indicate that D is highly likely. In this situation, G would be added

as a predictor in a patient-specific model. Thus, for a patient in whom G = a1, the typical population-wide logistic regression model would predict poorly.

This idea can be extended to examples with more than one predictor, in which some predictors are characterized by having particular values that are relatively rare but strongly predictive for the outcome. A population-wide model tends to include only those predictors that on average provide the best predictive performance. In contrast, a patient-specific model will potentially include predictors that are highly predictive for the particular patient case at hand; such predictors could be different from those included in the population-wide model.

Value-specific interactions. Variable selection is one important way in which models can be tailored to individual patient cases, as just described. Feature interaction (dependence) is another major way. Continuing with a genetic example, consider two genes E and F. When E = e1 and F = f1, disease K usually occurs; otherwise, K rarely occurs. Thus, when E = e1 and F = f1, there is an interaction between E and F in predicting K, and otherwise, there is not an interaction. Such value-specific interactions form another basis for constructing patient-specific models that take those interactions into account. Thus, patient-specific methods can construct better models by employing better variable selection and by capturing value-specific interactions among features.

2.3 DECISION THEORETIC COMPARISON OF POPULATION-WIDE AND PATIENT-SPECIFIC MODELS

This section first introduces some notation and definitions and then compares population-wide with patient-specific models in decision theoretic terms. Capital letters like X, Z, denote random

variables and corresponding lower case letters, x, z, denote specific values assigned to them. A feature is a specification of a variable and its value. Thus, $X = x$ is a feature that specifies that variable X is assigned the value x. Bold upper case letters, such as $\boldsymbol{X}$, $\boldsymbol{Z}$, represent sets of variables or random vectors and their realization is denoted by the corresponding bold lower case letters, $\boldsymbol{x}$, $\boldsymbol{z}$. A feature vector is a list of features. Thus, $\boldsymbol{X} = \boldsymbol{x}$ is a feature vector that specifies that the variables in $\boldsymbol{X}$ have the values given by $\boldsymbol{x}$. In addition, Z denotes the target variable (class variable) being predicted, $\boldsymbol{X}$ denotes the set of predictor variables, M denotes a model (and includes both structure and parameters), D denotes the training dataset, $C^i \equiv <\boldsymbol{X}^i, Z^i>$ denotes a generic training case in D and $C^t \equiv <\boldsymbol{X}^t, Z^t>$ denotes a generic test case that is not in D. A test case t is one in which the unknown value of the target variable Z^t is to be predicted from the known values of the predictors $\boldsymbol{X}^t$ and the known values of $<\boldsymbol{X}^i, Z^i>$ of a set of training cases.

A probabilistic *model* is a family of probability distributions indexed by a set of parameters. *Model selection* refers to the problem of using data to select one model from a set of models under consideration [18]. *Model averaging* refers to the process of estimating some quantity (e.g., prediction of an outcome for a patient) under each of the models under consideration and then obtaining a weighted average of their estimates [18].

Both model selection and model averaging can be done using either non-Bayesian or Bayesian approaches. Non-Bayesian methods of model selection include choosing among competing models by maximizing the likelihood, by maximizing a penalized version of the likelihood or by maximizing some measure of interest (e.g., accuracy) using cross-validation. Examples of non-Bayesian methods of model averaging include bagging and boosting. In both bagging and boosting, the data are resampled several times, a model is constructed from each sample, and the predictions of the individual models are averaged to obtain the final prediction.

In the non-Bayesian approach, the heuristics used in model selection and model averaging are typically different. In contrast, the Bayesian approach to model selection and model averaging both involve computing the posterior probability of each model under consideration.

In Bayesian model selection the single model found that has the highest posterior probability is chosen. In Bayesian model averaging the prediction is the weighted average of the individual predictions of the models with the model posterior probabilities comprising the weights.

When the goal is prediction of future data or future values of the target variable, Bayesian model averaging is preferred, since it suitably incorporates the uncertainty about the true model. However, sometimes interest is focused on a single model. For example, a single model may be useful for providing insight into the relationships among the domain variables or can be used as a computationally less expensive method for prediction. In such cases, Bayesian model selection maybe preferred to Bayesian model averaging. However, the proper Bayesian approach is to perform model averaging, and model selection, is at best, an approximation to model averaging.

Population-wide model selection and patient-specific model selection are characterized in decision theoretic terms as follows. Given training data D and a generic test case $<\boldsymbol{X}^t, Z^t>$, the *optimal population-wide model* is:

$$\arg\max_{M}\left\{\sum_{\boldsymbol{X}^t} U\left[P(Z^t \mid \boldsymbol{X}^t, D), P(Z^t \mid \boldsymbol{X}^t, M)\right] P(\boldsymbol{X}^t \mid D)\right\}, \qquad (2.1)$$

where the utility function U gives the utility of approximating the *Bayes optimal estimate* $P(Z^t \mid \boldsymbol{X}^t, D)$ with the estimate $P(Z^t \mid \boldsymbol{X}^t, M)$ obtained from model M. For a model M, Expression 2.1 considers all possible instantiations of $\boldsymbol{X}^t$ and for each instantiation computes the utility of estimating $P(Z^t \mid \boldsymbol{X}^t, D)$ with the specific model estimate $P(Z^t \mid \boldsymbol{X}^t, M)$, and weights that utility by

the posterior probability of that instantiation. The maximization is over the models M in a given model space.

The *Bayes optimal estimate* $P(Z^t \mid \boldsymbol{X}^t, D)$ in Expression 2.1 is obtained by combining the estimates of all models (in a given model space) weighted by their posterior probabilities:

$$P(Z^t \mid \boldsymbol{X}^t, D) = \int_M P(Z^t \mid \boldsymbol{X}^t, M)P(M \mid D)dM \,. \tag{2.2}$$

The term $P(\boldsymbol{X}^t \mid D)$ in Expression 2.1 is given by:

$$P(\boldsymbol{X}^t \mid D) = \int_M P(\boldsymbol{X}^t \mid M)P(M \mid D)dM \,. \tag{2.3}$$

The *optimal patient-specific model* for estimating Z^t is the one that maximizes the following:

$$\arg\max_M \left\{U\left[P(Z^t \mid \boldsymbol{x}^t, D), P(Z^t \mid \boldsymbol{x}^t, M)\right]\right\}, \tag{2.4}$$

where $\boldsymbol{x}^t$ are the values of the predictors of the test case $\boldsymbol{X}^t$ for which the target variable Z^t is to be predicted. The Bayes optimal estimate $P(Z^t \mid \boldsymbol{x}^t, D)$ is derived using Equation 2.2, for the special case in which $\boldsymbol{X}^t = \boldsymbol{x}^t$, as follows:

$$P(Z^t \mid \boldsymbol{x}^t, D) = \int_M P(Z^t \mid \boldsymbol{x}^t, M)P(M \mid D)dM \,. \tag{2.5}$$

The difference between the population-wide and the patient-specific model selection can be noted by comparing Expressions 2.1 and 2.4. Expression 2.1 for the population-wide model selects the model that on average will have the greatest utility. Expression 2.4 for the patient-specific model, however, selects the model that will have the greatest utility for the specific case $\boldsymbol{X}^t = \boldsymbol{x}^t$. For predicting Z^t given case $\boldsymbol{X}^t = \boldsymbol{x}^t$, application of the model selected using Expression 2.1 can never have an expected utility greater than the application of the model

selected using Expression 2.4. This observation provides support for developing patient-specific models.

2.4 MODEL SELECTION VERSUS MODEL AVERAGING

Equations 2.2 and 2.3 carry out Bayesian model averaging over all models in some specified model space. Expressions 2.1 and 2.4 include Equation 2.2; thus, these expressions for population-wide and patient-specific model selection, respectively, are theoretical ideals. Moreover, Equation 2.2 is the Bayes optimal prediction of Z^t. Thus, in order to do optimal model selection, the optimal prediction obtained from Bayesian model averaging must already be known.

Model selection, even if performed ideally, ignores the uncertainty inherent in choosing a single model based on limited data. Bayesian model averaging is a normative approach for dealing with the uncertainty in model selection, and has been shown to improve predictive performance as well as provide more accurate estimates of the error in prediction. Several examples of significant decrease in prediction errors with the use of Bayesian model averaging are described in [19]. Such averaging is primarily useful when no single model in the model space under consideration has a high posterior probability. However, since the number of models in practically useful model spaces is enormous, *complete Bayesian model averaging*, where the averaging is done over the entire model space, is usually not feasible. That is, it is usually not computationally feasible to solve for the exact solution given by Equation 2.2. In such cases, *selective Bayesian model averaging* is typically performed, where the averaging is done over a selected subset of models.

Methods for inducing predictive models can be classified along two axes as shown in Table 2-1. Along the horizontal axis are population-wide and patient-specific methods that differ chiefly in whether the features of the test case are utilized or not in developing the model. Along the vertical axis are model selection and model averaging.

The table focuses on a subset of supervised learning methods, namely, model-based learning methods. *Model-based methods* and *instance-based methods* (variants of which are known as instance-based learning, memory-based learning, exemplar-based learning or case-based reasoning) belong to two extremes of supervised learning methods [20]. Model-based methods learn an explicit model or models from the training cases that are then applied to the test case. In contrast, instance-based methods estimate the target variable in the test case by combining the values of the target variable in a subset of the training cases that are similar in some sense to the test case. Thus, instance-based methods are characterized by the use of a similarity (or distance) measure necessary for measuring the similarity between cases. The canonical example of an instance-based method is the *k*-Nearest Neighbor technique where the prediction of the target variable in the test case is based on the majority vote of the values of the target variable in the *k*-nearest cases (for classification) or the average over a set of *k*-nearest

Table 2-1: Categories of methods for predictive modeling.

	population-wide method	**patient-specific method (instance-specific method)**
model selection (mainly non-Bayesian)	1. Commonly used for predictive modeling; e.g., logistic regression, neural networks	3. Less commonly used for predictive modeling; e.g., Lazy Bayesian Rules [20]
model averaging (mainly Bayesian)	2. Less commonly used for predictive modeling; e.g., averaging over rule-sets [8], averaging over discrete Bayesian networks [10]	4. None described in the literature for predictive modeling; e.g., the patient-specific method described in this dissertation

training cases (for regression). It should be pointed out that instance-specific methods including patient-specific methods discussed in this dissertation are not instance-based methods but are rather model-based methods. In particular, the patient-specific methods explored here induce a model or set of models that are influenced by the values of the features of the test case; a similarity measure is not used. To keep the distinction between instance-specific methods and instance-based methods clear, I will refer to instance-based methods as *similarity-based methods*.

The typical predictive algorithms, both in machine learning and in the medical literature, perform non-Bayesian model selection to induce a population-wide model (cell 1 in Table 2-1). Examples of such methods are logistic regression, neural networks, CART-like decision trees, Bayesian networks[1] and support vector machines [17]. Less commonly, model averaging over population-wide models (cell 2 in Table 2-1) has been used for prediction; such techniques can improve predictive performance over population-wide model selection. One such example is the algorithm for classification that is described in [21]. This algorithm uses stochastic search to find multiple models of rule-sets over which to perform selective Bayesian model averaging. Instance-specific methods that perform model selection (cell 3 in Table 2-1) are also not that common. One example of such a method is Lazy Bayesian Rules that is described in detail in Section 2.8. It is a model-based, instance-specific method and not a similarity based method, and it has been shown to improve predictive performance over several population-wide model selection methods.

In this dissertation, I describe a method that belongs to the fourth category (cell 4 in Table 2-1): a patient-specific method that performs selective Bayesian model averaging. The key component of such a method is the heuristic used for searching and selecting models in the

[1] The Bayesian networks referred to here typically perform inference using non-Bayesian methods, and are hence properly classified under non-Bayesian methods.

model space over which model averaging is done. The patient-specific method employs a heuristic that searches the model space in a patient-specific manner. That is, the values of the features in the test case are used to direct the search. The algorithm is described in detail in Chapter 4.

2.5 LAZY LEARNING VERSUS EAGER LEARNING

Machine learning methods that defer model construction until a response to a test case is required are said to employ *lazy learning*. This is in contrast to methods that induce a model from the training data before ever encountering a test case, which are said to employ *eager learning* [22]. In terms of computation, lazy methods often have higher memory and time costs since they, typically, store the entire training data and construct a new model for every test case. In contrast, eager methods, usually, discard the original training cases and retain only the model that is then applied to all future cases. Table 2-2 illustrates the application of lazy versus eager

Table 2-2: Eager and lazy learning.

	population-wide method	**patient-specific method (instance-specific method)**
eager learning	1. Very common; e.g., logistic regression, neural networks, decision tree, etc., typically use eager learning.	3. In theory, it is possible to learn all possible patient-specific models in an eager fashion and then retrieve the appropriate model for a test case. However, this is feasible only if a small number of possible test cases exist.
lazy learning	2. While it is possible to construct models mentioned in cell 1. in a lazy fashion, it is uncommon to do so since the increase in memory and time requirements may not be offset by better performance.	4. Lazy learning is the usual approach used to learn patient-specific models.

learning to population-wide and patient-specific methods. Population-wide methods are usually eager, but they need not be, and patient-specific methods are usually lazy, but in theory they need not be (see cell 3 in Table 2-2). Since population-wide methods learn a single model that is applied to all future cases, it is usually more efficient in computation time to eagerly learn the model once from the training data. In addition, with eager learning, only the model needs to be retained after learning and typically the model requires lesser memory than the entire training data. Patient-specific methods are usually lazy since they require the test case for guiding model induction. The increased computation time and memory requirements of the lazy method can be offset by better predictive performance, as for example in the case of Lazy Bayesian Rules that is discussed in greater detail in Section 2.8.

2.6 TIME-VARYING PATIENT-SPECIFIC MODELS

In the clinical literature, patient-specific models sometimes refer to models that are induced from a time series of data obtained from a patient. Many of the variables included in such models have values that have been measured over multiple time points. Such *time-varying patient-specific models* are distinct from *atemporal patient-specific models* that are induced from atemporal data and/or a single (e.g., initial) time slice of data. In this work, I investigate atemporal patient-specific models, and for brevity I will refer to them as patient-specific models. This approach to patient-specific modeling is complementary to the methods that assume a time series, since the atemporal approach can construct an initial model from atemporal data (e.g., demographics of the patient) and/or data for an initial time slice (e.g., vital signs like blood pressure and temperature at the time of admission to the hospital). The initial model can then be revised based

on the time series data obtained from a specific patient (e.g., blood pressure and temperature measurements over time).

2.7 RELATED WORK

The following sections review some of the literature related to the machine learning methods introduced in the previous section. While not comprehensive in its coverage, this review provides a representative sample of related previous work in machine learning and in clinical predictive modeling that is most closely related to patient-specific modeling.

2.8 RELATED WORK IN MACHINE LEARNING

There exists a vast literature in machine learning, data mining and pattern recognition that is concerned with the problem of predictive modeling and supervised learning. This section focuses on some of the aspects of the similarity-based methods followed by a review of some recent work on instance-specific methods.

Similarity-based methods. These methods are also known as memory-based, case-based, instance-based, or exemplar-based learners. They (1) use a similarity or a distance measure, (2) defer most of the processing until a test case is encountered (i.e., they are lazy), (3) combine the training cases in some fashion to predict the target in the test case, and (4) discard the answer and any intermediate results after the prediction [23]. Typically, no explicit model is induced from the training cases at the time of prediction. The similarity measure evaluates the

similarity between the test case and the training cases and selects the appropriate training cases and their relative weights in response to the test case [24]. The selected training cases can be equally weighted or weighted according to their similarity to the test case. To predict the target variable in the test case, the values of the target variable in the selected training cases are combined in some simple fashion such as majority vote, simple numerical average or fitted with a polynomial.

The *nearest-neighbor technique* is the canonical similarity-based method. When a test case is encountered, the training case that is most similar to the test case is located and its target value is returned as the prediction [25]. A straight-forward extension to the nearest-neighbor technique is the *k-Nearest Neighbor* (*k*NN) method. For a test case, this method selects the *k* most similar training cases and either averages or takes a majority vote of their target values. Another extension is the distance-weighted *k*-Nearest Neighbor method. This weights the contribution of each of the *k* most similar training cases according to its similarity to the test case, assigning greater weights to more similar cases [26]. A further extension is locally weighted regression that selects cases similar to the test case, weights them according to their similarity, and performs regression to predict the target [27].

One drawback of the similarity-based methods is that they may perform poorly when predictors are redundant, irrelevant or noisy. To make the similarity metric more robust, variable selection and variable weighting have been employed [28]. Two generic approaches that have been used for variable selection and weighting are *filter methods* and *wrapper methods* [23]. Filter methods determine whether variables are predictive of the target variable using heuristics based on characteristics of the data. Typically, filter methods are applied as a preprocessing step and search for an optimal variable subset in the space of variable subsets independent of the

classification method to be applied subsequently. An example of a filter method is the selection of a subset of variables that are highly correlated with the target variable as measured by mutual information between each predictor variable and the target variable. Wrapper methods make use of the classification method that will ultimately be applied to the data in order to evaluate the predictive power of predictors. Typically, wrapper methods search for an optimal variable subset in the space of variable subsets using the criterion optimized by the classification method. An example of a wrapper method is the selection of a subset of variables that produces high accuracy when used by the classification method [29].

Instance-specific methods. Instance-specific methods in general and patient-specific methods in particular are model-based methods that take advantage of the features in the test case while inducing the model. Such methods are not as reliant on a similarity measure, if they use one at all, as the similarity-based methods.

Friedman et al. [30] describe one such algorithm called LazyDT that searches for the best CART-like decision tree for a test case. As implemented by the authors, LazyDT did not perform pruning and processed only nominal variables. The algorithm was compared to ID3 and C4.5 (standard population-wide methods for inducing decision trees), each with and without pruning. When evaluated on 28 datasets from the UCI Machine Learning repository, LazyDT generally out-performed both ID3 and C4.5 without pruning and performed slightly better than C4.5 with pruning.

Ting et al. [31] have developed a framework for inducing rules in a lazy fashion that are tailored to the features of the test case. Zheng et al. [32] describe an implementation of this framework called the Lazy Bayesian Rules (LBR) learner that induces a rule tailored to the features of the test case that is then used to classify it. A LBR rule consists of (1) a conjunction

of the variable-value pairs present in the test case as the antecedent and (2) a local naive Bayes classifier as the consequent. The structure of the local naive Bayes classifier consists of the target variable as the parent of all other variables that do not appear in the antecedent, and the parameters of the classifier are estimated from those training cases that satisfy the antecedent. Figure 2-1 shows an example of a LBR rule constructed using five predictor variables and a target variable from the pneumonia dataset (described in Chapter 5). The rule has two predictors in the antecedent and a naive Bayes classifier with three predictors in the consequent. A greedy step-forward search selects the optimal LBR rule for a test case to be classified. In particular, each predictor is added to the antecedent of the current best rule and evaluated for whether it reduces the overall error rate on the training set. The predictor that most reduces the overall error rate is added to the antecedent and removed from the consequent, and the search continues; if no single predictor move can decrease the current error rate, then the search halts and the current rule is applied to predict the outcome for the test case. LBR is an example of a patient-specific method that utilizes feature information available in the test case to direct the search for a suitable model in the model space.

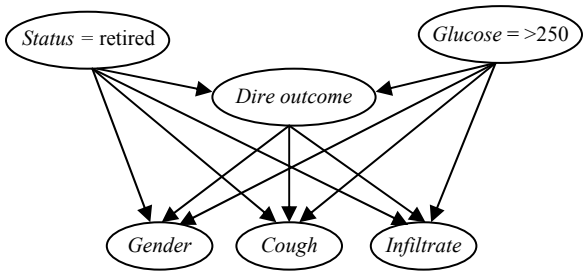


Figure 2-1: A LBR model (or rule) with five predictors and a target variable (*dire outcome*). The two nodes at the top represent predictors in the antecedent of the LBR rule that have been instantiated to their respective values in the test case. The node in the center (the target variable being predicted) and the three nodes at the bottom form the local naive Bayes classifier present in the consequent of the LBR rule.

The performance of LBR was evaluated by Zheng et al. [32] on 29 datasets from the UCI Machine Learning repository and compared to that of six algorithms: a naïve Bayes classifier (NB), a decision tree algorithm (C4.5), a Bayesian tree learning algorithm (NBTree) [33], a constructive Bayesian classifier that replaces single variables with new variables constructed from Cartesian products of existing nominal variables (BSEJ) [34], a selective naive Bayes classifier that deletes irrelevant variables using Backward Sequential Elimination (BSE) [35], and LazyDT, which is described above. Based on ten three-fold cross validation trials (for a total of 30 trials), LBR achieved the lowest average error rate across the 29 datasets. The average relative error reduction of LBR over NB, C4.5, NBTree, BSEJ, BSE and LazyDT were 9%, 10%, 2%, 3%, 5% and 16% respectively. LBR performed significantly better than all other algorithms except BSE; compared to BSE its performance was better but not statistically significantly so.

Some of the more recent algorithms have some limitations in that they can process only discrete variables – continuous variables have to be discretized. Also, they are computationally more intensive than many other learning algorithms. However, they have been shown to have better accuracy than several of the population-wide methods. These results provide empirical support for the possibility that patient-specific methods can have better predictive performance than population-wide models constructed using standard eager techniques.

2.9 RELATED WORK IN PREDICTIVE MODELING IN MEDICINE

In the medical domain, machine learning methods are been increasingly used for the induction of predictive models. In a recent study, Dreiseitl and Ohno-Machado [17] examined the number of publications indexed in Medline that used modeling and found that logistic regression, neural

networks, *k*-Nearest Neighbors (*k*NN), CART-like decision trees (classification trees) and support vector machines (SVM) were the most popular in descending order. Logistic regression is widely used in many clinical domains: for example, APACHE II is a severity of disease classification system that computes a score based upon routine physiologic measurements, age, and previous health status to provide a general measure of severity of disease [36]. Logistic regression models (without interaction terms) are easy to construct and interpret; however, they may not capture dependencies among attributes adequately. In contrast, neural networks, which generalize logistic regression models, are more flexible since they can express complex non-linear relationships among the attributes.

The *k*NN is the most commonly used similarity-based method in the medical domain. For example, it has been applied in searching for patterns in radiographic images for diagnostic purposes [37] and for diagnosis of diseases from gene expression profiles [38]. Though less commonly used than the preceding methods, CART-like decision trees are attractive since they provide a representation that lends itself to easy interpretation by humans. They can also be easily translated into a disjunction of conjunctions or the more convenient 'if-then' rules. Support vector machines have been used for several clinical problems ranging from diagnosis of breast cancer on ultrasound images [39] to classification of tumors based on gene expression profiles [40]. The basic SVM binary classification algorithm computes a maximum-margin hyperplane in a transformed predictor space. The hyperplane separates training cases of one class from the other such that the distance from the closest cases (the margin) to the hyperplane is maximized [41].

Several of the patient-specific models described in the medical literature are models that are induced from a time series of data about a patient. Such a model may be trained only on data

obtained from the patient for whom it will be used, or it may be trained from a combination of population data and data obtained from the patient. As an example, patient-specific modeling has been used to detect onset of seizures in real-time during long-term electroencephalogram (EEG) monitoring of epileptic patients. The model is trained from labeled seizure and nonseizure EEG data recorded in a patient and is then applied to ongoing EEG recording of that patient [42]. EEG patterns of seizures are very stereotypic for a given patient but vary widely among patients even for the same type of seizures. Thus, models induced from data from several patients usually perform poorly when compared to those constructed from a single patient and applied to that patient [43].

An example of patient-specific modeling that combines population data with data from a patient is the Bayesian method for forecasting drug dosage developed by Sheiner et al. [44, 45]. Here, the future drug dosage is individualized by revising the estimates of that individual's pharmacokinetic (PK) parameters. The individual's estimates of the PK parameters are obtained by combining information from population PK parameters (that describe the typical relationship between dosage and drug concentrations derived from a population of individuals) and past measurements of drug concentrations from that individual (that provide information on the relationship between dosage and drug concentrations specific to that individual). The initial prediction of a drug dose for an individual is based on just the population PK parameters, since no measurements of drug concentrations from that individual are yet available. This method has been shown to provide more accurate estimates of the individual's PK parameters than methods that use only one of the sources of information [46]. In this dissertation the focus is on developing methods that can improve initial such predictions based on data already available on the individual. Such methods aim to improve the predictive performance when repeated

measurements of the target variable are not possible (e.g., mortality) or when the initial value of a target variable that can be measured repeatedly has to be predicted (e.g., blood level after the first dose of a drug).

Though numerous predictive models in the clinical domain have been published by researchers, few are in routine clinical use. Lack of clinical credibility and lack of evidence of accuracy, generality, and effectiveness were some of the reasons identified by Wyatt et al. for the failure of acceptance of prognostic models in medicine [9]. Newer machine learning methods, such as the one described in this proposal, have the potential to improve the accuracy of predictive models. With increasing use of ever more powerful computers in clinical care and the increasing capture of patient information in clinical computer systems, computationally intensive modeling methods as part of decision support systems will become more feasible. If such methods can predict patient outcomes well and are incorporated in clinical decision support systems, they are likely to be clinically useful. In support of this point, a recent study demonstrated that physicians are quite amenable to having the recommendations of decision support systems for clinical decision making [47].

3.0 BAYESIAN NETWORKS

This chapter describes the formalism of Bayesian networks (BN) and reviews the methods for learning them from datasets. This provides the necessary background for the patient-specific algorithms that learn both structure and parameters of BNs from data. Section 3.1 introduces some notation and Section 3.2 briefly describes the BN formalism. Section 3.3 describes several representations of probability distributions in BNs. Sections 3.4 and 3.5 review some of the commonly used methods for learning the structure and parameters of BNs from data. In particular, the Bayesian approach to BN learning is reviewed in detail, including a description of the Bayesian scoring metric that is defined to be the posterior probability of the BN structure conditioned on the observed data. Section 3.6 focuses on the learning of BNs for the purpose of classification and reviews the utility of the Bayesian scoring metric for classification. Finally, Section 3.7 describes Bayesian model averaging as a coherent approach for improving predictions.

3.1 NOTATION

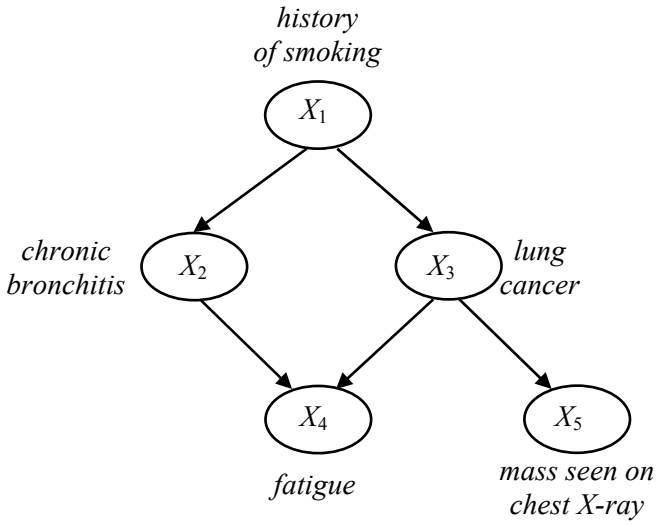
Random variables are denoted with upper case letters, such as X, Z, and their instantiation or assignment with the corresponding lower case letters x, z. Thus, $X = x$ denotes that random variable X is assigned the value (or state) x. Likewise, sets of variables or random vectors are

denoted with bold upper case letters, such as $\boldsymbol{X}$, $\boldsymbol{Z}$, and their instantiation or assignment with the corresponding bold lower case letters $\boldsymbol{x}$, $\boldsymbol{z}$. Thus, $\boldsymbol{X} = \boldsymbol{x}$ denotes that the random variables in $\boldsymbol{X}$ have the values (or states) given by $\boldsymbol{x}$. Given a domain of interest, $\boldsymbol{X} = \{X_1,...,X_n\}$ denotes the complete set of variables in the domain and $\boldsymbol{x} = \{x_1,...,x_n\}$ represents a complete instantiation of the variables in $\boldsymbol{X}$. For a discrete random variable X_i, r_i denotes its number of values and $\{x_{i1}, x_{i2}..., x_{in}\}$ denote the domain of the values.

Generally, Z denotes the target (class) variable being predicted, $\boldsymbol{X}$ denotes the set of predictor variables excluding the class variable, M denotes a model, D denotes the training dataset, $C^i \equiv <\boldsymbol{X}^i, Z^i>$ denotes a generic training case and $C^t \equiv <\boldsymbol{X}^t, Z^t>$ denotes a generic test case. The goal is to predict the value of the target variable Z^t of the test case C^t.

3.2 BAYESIAN NETWORK REPRESENTATION

A Bayesian network (BN) is a probabilistic model that combines a graphical representation (the BN structure) with quantitative information (the BN parameterization) to represent a joint probability distribution over a set of random variables [5, 6]. More specifically, a Bayesian network model M representing the set of random variables $\boldsymbol{X}$ for some domain consists of a pair $(G, \boldsymbol{\theta}_G)$. The first component G is a directed acyclic graph (DAG) that contains a node for every variable in $\boldsymbol{X}$ and an arc between a pair of nodes if the corresponding variables are directly probabilistically dependent. Conversely, the absence of an arc between a pair of nodes denotes probabilistic independence between the corresponding variables. In this document, the terms variable and node are used interchangeably in the context of random variables being modeled by



Node X_1	$P(X_1 = \text{F}) = 0.80$	$P(X_1 = \text{T}) = 0.20$
Node X_2	$P(X_2 = \text{F} \mid X_1 = \text{F}) = 0.95$	$P(X_2 = \text{T} \mid X_1 = \text{F}) = 0.05$
	$P(X_2 = \text{F} \mid X_1 = \text{T}) = 0.75$	$P(X_2 = \text{T} \mid X_1 = \text{T}) = 0.25$
Node X_3	$P(X_3 = \text{F} \mid X_1 = \text{F}) = 0.995$	$P(X_3 = \text{T} \mid X_1 = \text{F}) = 0.005$
	$P(X_3 = \text{F} \mid X_1 = \text{T}) = 0.997$	$P(X_3 = \text{T} \mid X_1 = \text{T}) = 0.003$
Node X_4	$P(X_4 = \text{F} \mid X_1 = \text{F}, X_3 = \text{F}) = 0.99995$	$P(X_4 = \text{T} \mid X_1 = \text{F}, X_3 = \text{F}) = 0.00005$
	$P(X_4 = \text{F} \mid X_1 = \text{F}, X_3 = \text{T}) = 0.50$	$P(X_4 = \text{T} \mid X_1 = \text{F}, X_3 = \text{T}) = 0.50$
	$P(X_4 = \text{F} \mid X_1 = \text{T}, X_3 = \text{F}) = 0.90$	$P(X_4 = \text{T} \mid X_1 = \text{T}, X_3 = \text{F}) = 0.10$
	$P(X_4 = \text{F} \mid X_1 = \text{T}, X_3 = \text{T}) = 0.25$	$P(X_4 = \text{T} \mid X_1 = \text{T}, X_3 = \text{T}) = 0.75$
Node X_5	$P(X_5 = \text{F} \mid X_1 = \text{F}) = 0.98$	$P(X_5 = \text{T} \mid X_1 = \text{F}) = 0.02$
	$P(X_5 = \text{F} \mid X_1 = \text{T}) = 0.40$	$P(X_5 = \text{T} \mid X_1 = \text{T}) = 0.60$

Figure 3-1: A simple hypothetical Bayesian network for a medical domain, taken from [48]. All the nodes represent binary variables, taking values in the domain {T, F} where T stands for True and F for False. The graph at the top represents the Bayesian network structure. Associated with each variable (node) is a conditional probability table representing the probability of each variable's value conditioned on its parent set. (Note: these probabilities are for illustration only; they are not intended to reflect frequency of events in any actual patient population.)

nodes in a BN. A variable X_i in the domain of interest will usually be represented by a node labeled X_i in the BN graph.

The terminology of kinship is used to denote various relationships among nodes in a graph. These kinship relations are defined along the direction of the arcs. Predecessors of a node X_i in G, both immediate and remote, are called the *ancestors* of X_i. In particular, the immediate predecessors of X_i are called the *parents* of X_i. The set of parents of X_i in G is denoted by

$\boldsymbol{Pa}(X_i, G)$ or more simply as $\boldsymbol{Pa}_i$ when the BN structure is obvious from the context. In a similar fashion, successors of X_i in G, both immediate and remote, are called the *descendants* of X_i, and the immediate successors are called the *children* of X_i. A node X_j is termed a *spouse* of X_i if X_j is a parent of a child of X_i. The set of nodes consisting of a node X_i and its parents is called the *family* of X_i. Figure 3-1 gives an illustrative example of a simple hypothetical BN taken from [48], where the top panel shows the graphical or the structural component G of the BN. In the figure, the variable *history of smoking* is a parent of the variable *lung cancer* as well as a parent of the variable *chronic bronchitis*. The variable *fatigue* is a child of the variable *lung cancer* as well as a child of the variable *chronic bronchitis*. A descendant of a node X_i is a node X_j that can be reached by a directed path from X_i to X_j. In the example, variables *lung cancer* and *mass seen on chest X-ray* are descendants of the variable *history of smoking*.

The second component $\boldsymbol{\theta}_G$ represents the parameterization of the probability distribution over the space of possible instantiations of $\boldsymbol{X}$ and is a set of local probabilistic models that encode quantitatively the nature of dependence of each variable on its parents. For each node X_i there is a local probability distribution (that may be discrete or continuous) defined on that node for each state of its parents. The set of all the local probability distributions associated with all the nodes comprises the complete parameterization of the BN. The bottom panel in Figure 3-1 gives an example of a set of parameters for G. Taken together, the top and bottom panels in Figure 3-1 provide a fully specified structural and quantitative representation for the BN.

3.2.1 Local Markov condition and factorization of the joint probability distribution

The topology of the graph G encodes compactly the set of independencies among the variables in the domain. These independence relations include both marginal and conditional independencies

and can be enumerated for a BN by the application of the local and global Markov conditions to the topology of the network. This section describes the local Markov condition and the next section describes the global Markov condition.

The *local Markov condition* identifies the independencies local to a node: A node is conditionally independent of its non-descendants given just the states its parents [5]. In Figure 3-1, one such independence relation is this: the variable *history of smoking* is independent of the variable *mass seen on chest X-ray* given the state of the variable *lung cancer*.

The local Markov condition provides a factored representation for the complete joint probability distribution over the variables in the domain, which is a crucial characteristic of a BN. This factored representation can be substantially more compact than the complete joint probability distribution, especially when the graph is sparse. The joint probability distribution is factored by applying the chain rule of probability followed by simplification of the terms based on the independencies asserted by the local Markov condition. Let the variables in $\boldsymbol{X} = \{X_1,...,X_n\}$ be topologically sorted relative to G, such that if $i < j$ then X_i is a non-descendant of X_j in G. Using the chain rule of probability the joint distribution is factored as:

$$P(X_1,...,X_n) = \prod_{i=1}^{n} P(X_i \mid X_1,...,X_{i-1}) \ . \tag{3.1}$$

The local Markov condition asserts that for all X_i in $\boldsymbol{X}$,

$$P(X_i \mid X_1,...,X_{i-1}) = P(X_i \mid \boldsymbol{Pa_i}), \tag{3.2}$$

where $\boldsymbol{Pa_i} \subseteq \{X_1,...,X_{i-1}\}$, because in the sorting of the variables all of the parents of X_i are in the set $\{X_1,...,X_{i-1}\}$, and none of the descendants of X_i are in this set. Substituting Equation 3.2 into Equation 3.1 gives the following equation, which is also known as the *chain rule for Bayesian networks*:

$$P(X_1,...,X_n) = \prod_{i=1}^{n} P(X_i \mid \boldsymbol{Pa_i}). \quad (3.3)$$

As an example, applying the local Markov condition to the BN in Figure 3-1 leads to the following factorization:

$$P(X_1, X_2, X_3, X_4, X_5) = P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_1)P(X_4 \mid X_2, X_3)P(X_5 \mid X_3) \quad (3.4)$$

The local Markov condition, thus, translates a high dimensional multivariate joint probability distribution into a product of potentially low dimensional univariate probability distributions. The BN network represents and stores univariate probability distributions which typically require fewer parameters for specification than the complete joint probability distribution. For example, the BN in Figure 3-1 requires only 11 independent probabilities to be specified (the probabilities in the right hand column are redundant), while the full joint probability distribution for the same example where a probability is to be specified for each instantiation of the five variables would require $2^5 - 1 = 31$ independent probabilities.

3.2.2 Global Markov condition and d-separation

The *global Markov condition* also identifies independencies with respect to a node: A node is conditionally independent of all other nodes in the network, given its parents, its children, and the children's parents. This set of nodes is also known as the Markov blanket of the node and is described in the next section. Figure 3-2 distinguishes graphically the local and the global Markov conditions.

The global Markov condition can be extended to identify independencies among disjoint sets of nodes in a BN. A topological procedure called *d-separation* can identify the complete set of conditional independencies in the graph implied by the global Markov condition [5]. Pearl

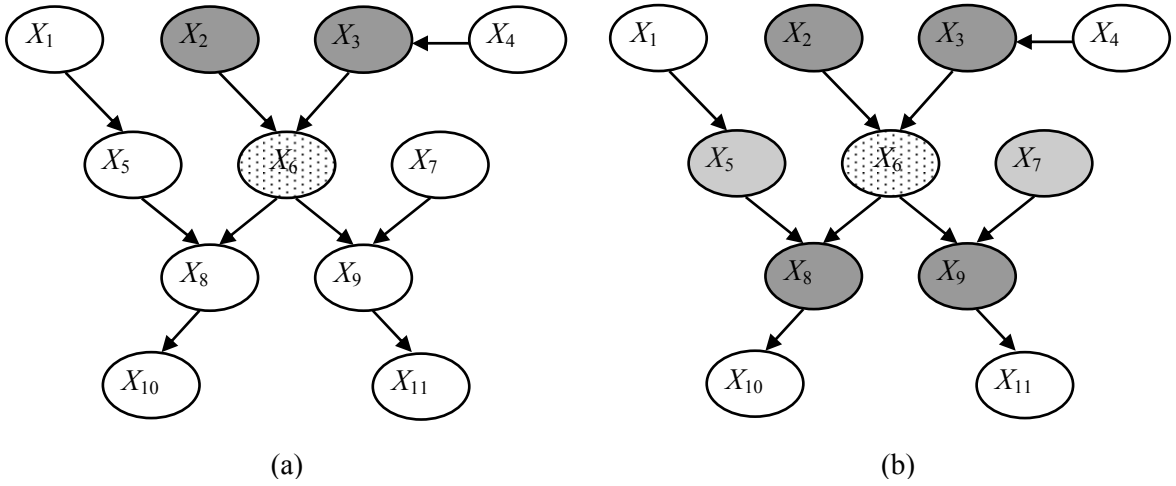


Figure 3-2: Examples of the local Markov condition and the global Markov condition. (a) Local Markov condition: The node X_6 (shown stippled) is conditionally independent of its non-descendants given its parents (shown shaded). (b) Global Markov condition: the node X_6 (shown stippled) is conditionally independent of all other nodes in the network given its Markov blanket (shown shaded).

describes the application of d-separation as follows. Consider three disjoint subsets of nodes $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{Z}$ in graph G. Whether $\boldsymbol{X}$ is independent of $\boldsymbol{Y}$ given $\boldsymbol{Z}$ is tested by testing whether the nodes in $\boldsymbol{Z}$ "block" all paths from nodes in $\boldsymbol{X}$ to nodes in $\boldsymbol{Y}$. A path refers to a sequence of consecutive arcs (of any directionality) in the graph, and "blocking" is interpreted as barring the dependency between variables that are connected by such paths. A path p is said to be d-separated or "blocked" by a set of nodes $\boldsymbol{Z}$ if and only if:

1. p contains a chain $X_i \rightarrow X_j \rightarrow X_k$ or a fork $X_i \leftarrow X_j \rightarrow X_k$ such that the middle node X_j is in $\boldsymbol{Z}$
2. p contains a collider $X_i \rightarrow X_j \leftarrow X_k$ such that the middle node X_j is not in $\boldsymbol{Z}$ and also no descendant of X_j is in $\boldsymbol{Z}$

A set $\boldsymbol{Z}$ is said to d-separate $\boldsymbol{X}$ from $\boldsymbol{Y}$ if and only if $\boldsymbol{Z}$ blocks every path from a node in $\boldsymbol{X}$ to a node in $\boldsymbol{Y}$. The global Markov condition states that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are conditionally independent given $\boldsymbol{Z}$ if and only if $\boldsymbol{Z}$ d-separates $\boldsymbol{X}$ from $\boldsymbol{Y}$ [5, 49]. Thus, d-separation identifies all the conditional independencies implied by the global Markov condition.

In a BN, the local Markov condition implies the global Markov condition and vice-versa and both conditions identify all the independencies implied by the topology of the network [49, 50].

3.2.3 Markov blanket

The *Markov blanket* (MB) of a variable Z, denoted by $MB(X_i)$, is a minimal set of variables such that X_i is conditionally independent of all other variables given $MB(X_i)$ [5]. This entails that the variables in $MB(X_i)$ are sufficient to determine the probability distribution of X_i. Since d-separation is applied to the graphical structure of a BN to identify all conditional independence relations, it can also be applied to identify the MB of a node in a BN. The MB of a node X_i consists of its parents, its children, and its children's parents and is illustrated in Figure 3-3. The parents and children of X_i are directly connected to it and are hence in its MB. In addition, the spouses are also included in the MB, because of the phenomenon of explaining away which

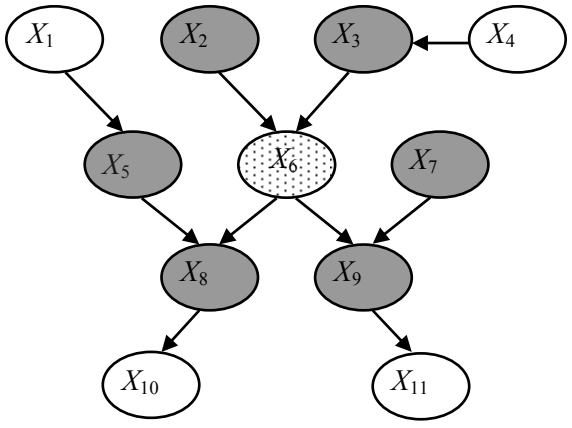


Figure 3-3: Example of a Markov blanket. The Markov blanket of the node X_6 (shown stippled) comprises the set of parents, children and spouses of the node and is indicated by the shaded nodes. The nodes in the Markov blanket include X_2 and X_3 as parents, X_8 and X_9 as children, and X_5 and X_7 as spouses of X_6. X_1, X_4, X_2 and X_{10} and X_{11} are not in the Markov blanket of X_6.

refers to the observation that when a child node is instantiated its parents in general are statistically dependent.

The MB of a node is noteworthy because it identifies all the variables that shield the node from the rest of the network. In particular, when interest centers on the distribution of a specific target node, as is the case in classification, the structure and parameters of only the MB of the target node need be learned.

3.3 REPRESENTATION OF LOCAL PROBABILITY DISTRIBUTIONS

The global structure of the BN represented by the arcs connecting the nodes implies a set of conditional independencies that allows the decomposition of a high dimensional joint probability distribution into a product of potentially low dimensional *conditional probability distributions* (CPDs). Each factor $P(X_i \mid \boldsymbol{Pa}_i)$ on the right hand side of Equation 3.3 is a set of CPDs that is associated with X_i. This section describes several representations for these CPDs, including some representations that capture additional regularities that are not implied by the global structure. The choice of representation depends on the type of the variables involved (i.e., discrete or continuous), on the nature of the relationship among the variables (i.e., deterministic or probabilistic), and on the need to represent local dependencies among parameters.

In domains with discrete random variables, the tabular representation for CPDs is simple and straight-forward. In this representation, $P(X_i \mid \boldsymbol{Pa}_i)$ is a table that contains an entry for each joint instantiation of X_i and $\boldsymbol{Pa}_i$. Each column (or row) in the table represents a single conditional probability distribution, $P(X_i \mid \boldsymbol{Pa}_i = \boldsymbol{pa}_i)$, corresponding to a particular instantiation of $\boldsymbol{Pa}_i$. Tabular CPDs are aptly called conditional probability tables (CPTs) and are almost always the

representation used in discrete BNs. For example, the CPD for node X_4 in the top panel in Figure 3-1 is represented by the hypothetical CPT shown in the bottom panel in Figure 3-1 that contains four independent parameters. CPTs are a very general representation for discrete nodes in that every possible discrete conditional probability distribution can be represented by a conditional probability table. However, the CPT representation has several disadvantages. First, in general the number of parameters of a CPT of a node grows exponentially in the number of parents of the node, and when parameters are estimated from data, this expansion of the CPT leads to poor estimates of the parameters since fewer data points contribute to the estimate of each parameter. Second, the tabular representation ignores structure and regularities within the CPDs; capturing such regularities provides additional domain knowledge about the interactions among the parents and reduces the number of parameters needed to specify the CPDs. Interactions among parents are captured by a type of independence relation called context-specific independence and is described in the next section. The subsequent sections briefly describe several representations for CPDs that explicitly capture context-specific independencies.

3.3.1 Context-specific independence

The DAG of a BN encodes statements of *variable independence*. For example, a variable X is independent of Y given variable Z if $P(x \mid y, z) = P(x \mid z)$ for all values x, y and z that the variables X, Y, Z can take.

In the standard discrete BN, the graphical structure makes explicit independence relations of the form $X \perp Y \mid Z$ which implies that $P(X \mid Y, Z) = P(X \mid Z)$ for all values of the variables X, Y and Z. However, these are not the only independencies that may be present in a domain. For instance, *value-specific independencies* that hold for only particular assignments of values to

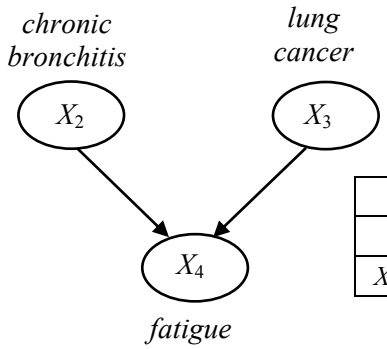
certain nodes cannot be represented by the BN graphical structure. Value-specific independencies are of the form $X \perp Y \mid Z = z$ which implies that $P(X \mid Y, Z = z) = P(X \mid Z = z)$ for all values of the variables X and Y when Z takes the particular value z. This type of independence relation is also known as *context-specific independence*; the preceding example can be interpreted as X is independent of Y in the context of Z taking the value z. In general, these independent statements imply that in some contexts, defined by an assignment of specific values to the variables in the BN, the conditional probability of a variable is independent of some of its parents [51].

In the CPT representation, *context-specific independencies* become apparent only on examining the numerical values of the parameters. Context-specific independence is present when the conditional probability distributions for two or more parent states have identical values of the parameters; such an independence relation is not explicitly represented in the CPT structure. For example, in Figure 3-4 (b), an examination of the parameters in the CPT reveals that three of the four possible parent states have the same parameter values (0.6, 0.4) implying that context-specific independencies are present. Indeed, the following two context-specific independence relations among the variables can be identified:

(1) $X_4 \perp X_2 \mid X_3 = \text{T}$ (i.e., *fatigue* $\perp$ *chronic bronchitis* | *lung cancer* = T)

(2) $X_4 \perp X_3 \mid X_4 = \text{T}$ (i.e., *fatigue* $\perp$ *lung cancer* | *chronic bronchitis* = T)

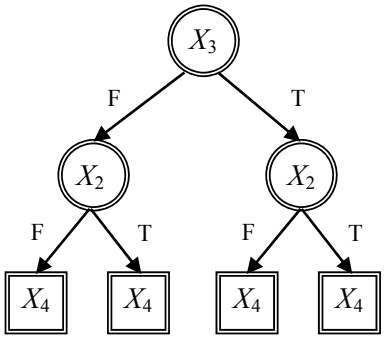
BNs that do not explicitly represent context-specific structure are referred to as *BNs with global structure*, in contrast to BNs that explicitly capture context-specific structure which are referred to as *BNs with local structure*. Several CPD representations have been developed for discrete variables that explicitly capture context-specific structure. The following sections describe two such *local structure representations*, namely, decision trees and decision graphs.



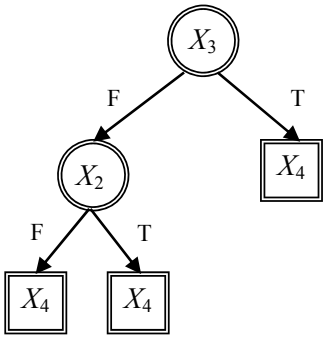
(a)

X_3 (*lung cancer*)	F		T	
X_2 (*chronic bronchitis*)	F	T	F	T
X_4 (*fatigue* = F, *fatigue* = T)	0.9, 0.1	0.6, 0.4	0.6, 0.4	0.6, 0.4

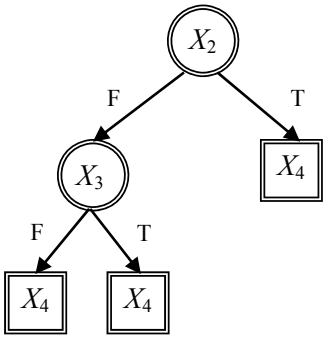
(b)



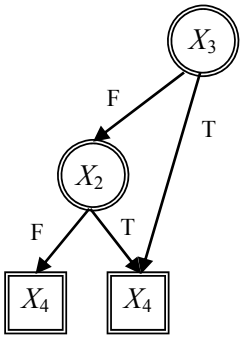
(c)



(d)



(e)



(f)

Figure 3-4: Examples of CPD representations. Several CPD representations for the BN node X_4 (*fatigue*) in panel (a) are shown in subsequent panels. Panel (b) shows a CPT for the node X_4 with four parameters. The CPT can be equivalently represented by a complete decision tree as shown in panel (c). Panels (d) and (e) show alternate decision trees where each one captures one of the two context specific independence relations that is present but not both (see text for details). Panel (f) shows a decision graph that captures both the context specific independence relations (see text for details). Nodes of a BN are shown as ellipses with single lines while nodes of decision trees and decision graphs are shown as either circles with double lines (interior nodes) or as rectangles with double lines (leaf nodes).

3.3.2 Decision tree CPDs

Friedman and Goldszmidt describe a *decision tree* representation for learning BNs with local structure from data [51]. In this representation, a decision tree is used to represent the local structure for a BN node X_i. Such a representation is called a *decision tree CPD* or a *tree CPD* for short. A decision tree is a graph where the root node has no parents, and all other nodes have a single parent. Nodes that have children and appear in the interior of the tree are called *interior nodes* and terminal nodes are called *leaf nodes*. Each leaf node in the tree contains a conditional distribution over X_i, and the path to the leaf from the root provides the context in which the distribution is valid. Each interior node is annotated with the name of one of the parent variables $X_j \in \boldsymbol{Pa}_i$ and out-going arcs from that interior node are annotated with mutually exclusive and collectively exhaustive sets of values for the variable X_j. In other words, values of parent nodes of X_i appear along the path and determine the parent states for which the distribution in the corresponding leaf node is applicable. Each leaf node contains a set of k parameters – where k is the number of states of X_i – that defines a single conditional probability distribution $P(X_i \mid \boldsymbol{Pa}_i = \boldsymbol{pa}_i)$ corresponding to a particular instantiation of $\boldsymbol{Pa}_i$.

As an example, Figure 3-4 (d) shows a decision tree CPD that represents the local structure of the node X_4 in Figure 3-4 (a). The decision tree representation is more compact than the CPT representation in that the decision tree CPD contains one less CPD and hence requires one fewer set of parameters than the CPT. This is achieved by capturing the context-specific independence relation $X_4 \perp X_2 \mid X_3 = \text{T}$ (*fatigue* $\perp$ *chronic bronchitis* | *lung cancer* = T), which is seen in Figure 3-4 (d) by noticing that the path along the right from the root node X_3 (*lung cancer*) does not contain a node for the variable *chronic bronchitis*. An alternate decision

tree, shown in Figure 3-4 (e), that has as the root node the variable X_2 (*chronic bronchitis*) can capture the other context-specific independence relation $X_4 \perp X_3 \mid X_4 = \text{T}$ (*fatigue* $\perp$ *lung cancer* | *chronic bronchitis* = T). However, no decision tree is unable to capture both context-specific independence relations given in the example.

3.3.3 Decision graph CPDs

Chickering et al. generalized the decision tree representation to *decision graphs*, which can represent a richer set of context-specific independence relations [52]. A decision tree is a graph where the root node has no parents, and all other nodes have one or more parents. Nodes that have children and appear in the interior of the tree are called *interior nodes* and terminal nodes are called *leaf nodes*. A decision graph differs from a decision tree in that an interior node may have multiple parents, rather than just one parent. A decision graph, thus, allows two or more distinct paths from the root node to terminate in the same leaf node. Such a representation is called a *decision graph CPD* or a *graph CPD* for short. For a BN node X_i that is represented by a decision graph, the leaf nodes contain conditional distributions over X_i similar to those in a decision tree, and interior nodes and outgoing arcs in a decision graph are annotated in a similar fashion as in a decision tree. All paths that lead to the same leaf node represent distinct parent states for which X_i has the same conditional distribution. The decision graph representation is more general than the decision tree representation, in that, any local structure that can be represented compactly as a tree can be represented as a graph, but the converse is not true.

As an example, Figure 3-4 (f) shows a decision graph CPD representing the local structure of the node X_4 in Figure 3-4 (a). In this example, the decision graph CPD is more compact than either the CPT or the decision tree representation since it requires one less set of

parameters than the decision tree CPD and two fewer sets of parameters than the CPT. The decision graph is able to capture both context-specific independence relations given in the example, demonstrating that it is a more general representation than the decision tree.

3.3.4 Summary of CPD representations

From the preceding discussion, it is evident that the CPDs in BNs can be represented with varying degrees of parsimony. Representations of CPDs that do not attempt to capture context specific independencies are the least parsimonious; such representations explicitly capture only the *global structure*. The classical representation for a CPD is the complete table (commonly known as the CPT); each node X_i stores $q_i \times r_i$ parameters in a large table where q_i is the number of parent states for X_i and r_i the number of states of X_i. An equivalent representation to the complete table is the complete decision tree which can represent all of the parameters in a complete table. A complete decision tree for a node X_i is a tree of depth q_i such that every interior node at level l is annotated by the lth parent $X_l \in \boldsymbol{Pa}_i$ and has exactly as many children as there are states in X_l. It follows from this definition that a *complete decision tree* has the same number of leaf nodes as the number of columns in an equivalent complete table. For example, the CPT in Figure 3-4 (b) is equivalently represented by the complete decision tree in

Table 3-1: Labels for CPDs, BNs and MBs based on the CPD representation.

structure	CPD representation	CPD label	BN label	MB label
global	complete table (CPT)	tabular CPD	tabular CPD BN	tabular CPD MB
	complete decision tree	complete CPD	complete CPD BN	complete CPD MB
local	decision tree	decision tree CPD	decision tree CPD BN	decision tree CPD MB
	decision graph	decision graph CPD	decision graph CPD BN	decision graph CPD MB

Figure 3-4 (c). I will refer to a BN with tabular CPDs as a *tabular CPD BN*, and a BN with complete decision tree CPDs as a *complete CPD BN*.

Representations for CPDs that explicitly represent context-specific independence relations include decision trees and decision graphs. I will refer to a BN with decision tree CPDs as a *tree CPD BN*, and a BN with decision graph CPDs as a *graph CPD BN*. Both tree CPD BNs and graph CPD BNs capture the *local structure*. Of note, the local structure implies the global structure. This can be seen from the observation that the parents of a BN node X_i are those nodes that appear in the decision tree or the decision graph associated with X_i. Table 3-1 summarizes the nomenclature for BNs based on the CPD representations.

3.4 LEARNING BAYESIAN NETWORKS FROM DATA

Learning a BN from a dataset of cases consists of learning its two components: the graphical structure of conditional dependencies (*structure learning*) and the conditional probability distributions (*parameter estimation*). Given a fixed network structure, parameter estimation is the easier problem and frequentist or Bayesian statistical approaches can be applied to the problem. The learning of the graphical structure that best fits the data is a more challenging task.

3.4.1 Parameter estimation

This section focuses on the estimation of the parameters of the conditional probability distributions $P(X_i \mid \boldsymbol{Pa}_i)$ under the assumptions that the BN structure is known, all the variables are discrete and the data has no missing values for any of the variables. There are two main

approaches for parameter estimation: the *maximum likelihood* approach that depends only on the data and the *Bayesian* approach that combines prior probabilities for the parameters with the data. In both approaches, the task is to find a good value or a set of good values for each parameter in the BN. This can be formulated as a learning task that consists of a *hypothesis space* which defines the set of all possible values being considered and a *scoring function* that scores different hypotheses in the space relative to the given data. The BN structure reduces the parameter estimation problem to a set of unrelated simpler parameter estimation problems involving only a node and its parents. In particular, for discrete variables in which conditional probability distributions are multinomial distributions whose parameters are stored in tables, decision trees or decision graphs, the parameters for each multinomial distribution can be estimated independently.

3.4.1.1 Maximum likelihood estimation

The scoring function in maximum likelihood estimation is the standard likelihood. The likelihood function computes the probability of the data as a function of the parameter values. Parameter values with higher likelihood are more likely to generate the data; thus the likelihood function measures how well different parameter values predict the data. In the maximum likelihood method, the parameter values that maximize the likelihood are selected, and the estimator is called the *maximum likelihood estimator* (MLE). For many parametric distributions, including the multinomial distribution, the likelihood function is maximized easily in closed form.

In a discrete BN with n nodes, the parameterization $\boldsymbol{\theta}$ over the entire network can be decomposed as $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1,...,\boldsymbol{\theta}_i,...,\boldsymbol{\theta}_n\}$ where each $\boldsymbol{\theta}_i$ represents the set of parameters defining the

conditional distributions $P(X_i \mid \boldsymbol{Pa}_i)$ associated with node X_i. It is typically assumed that the $\boldsymbol{\theta}_i$ are mutually independent; an assumption that is referred to as the *global parameter independence*. Further, each $\boldsymbol{\theta}_i$ is decomposed as $\boldsymbol{\theta}_i = \{\boldsymbol{\theta}_{i1},...,\boldsymbol{\theta}_{ij},...,\boldsymbol{\theta}_{iq_i}\}$ where q_i is the number of possible instantiations of $\boldsymbol{Pa}_i$. Each $\boldsymbol{\theta}_{ij}$ represents the parameters defining the single conditional distribution $P(X_i \mid \boldsymbol{Pa}_i = j)$. It is typically assumed that the $\boldsymbol{\theta}_{ij}$ are mutually independent; an assumption that is referred to as the *local parameter independence*. For a discrete variable each $\boldsymbol{\theta}_{ij}$ parameterizes a multinomial distribution. Thus, for each instantiation $\boldsymbol{Pa}_i = j$,

$$P(X_i \mid \boldsymbol{Pa}_i = j) = multinomial(\boldsymbol{\theta}_{ij}), \tag{3.5}$$

where, $\boldsymbol{\theta}_{ij} = \{\boldsymbol{\theta}_{ij1},...,\boldsymbol{\theta}_{ijk},...,\boldsymbol{\theta}_{ijn_i}\}$ and $\boldsymbol{\theta}_{ijk} = P(X_i = k \mid \boldsymbol{Pa}_i = j)$. Similarly, the multinomial likelihood function for a discrete BN decomposes into a product of local likelihood functions over the nodes, and the local likelihood function at each node further decomposes into a product of simple likelihood functions over the instantiations of the parent nodes. Each simple likelihood function is a multinomial likelihood function that is easily maximized to obtain the following MLE parameters:

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}, \tag{3.6}$$

where, N_{ijk} is the number of cases in the training data such that node i has value k and the parents of i have the state denoted by j, and $N_{ij} = \sum_k N_{ijk}$.

3.4.1.2 Bayesian parameter estimation

The MLE approach attempts to find a single set of parameter values $\hat{\theta}$ that explain the data well. The Bayesian approach, in contrast, does not attempt to find a single set of best parameter values. Rather, it provides a distribution over the possible parameter values that quantifies the

uncertainty of each of the values. Thus, it combines prior knowledge of the parameterization $\boldsymbol{\theta}$ with the data to arrive at a posterior distribution over $\boldsymbol{\theta}$. The prior knowledge of $\boldsymbol{\theta}$ is encoded with a probability distribution; this distribution represents the *a priori* knowledge and beliefs about the different values of the parameters. Then, a joint distribution over $\boldsymbol{\theta}$ and the data D captures all the necessary information:

$$P(D,\boldsymbol{\theta}) = P(D \mid \boldsymbol{\theta})P(\boldsymbol{\theta}) \tag{3.7}$$

The first term on the right hand side is the likelihood function, which is the same function that is used for obtaining the MLE estimates. The second term is the prior distribution over the parameter values. Once the likelihood function and the prior have been specified, Bayes rule is applied to obtain the posterior distribution over the parameters:

$$P(\boldsymbol{\theta} \mid D) = \frac{P(D \mid \boldsymbol{\theta})P(\boldsymbol{\theta})}{P(D)}. \tag{3.8}$$

The term $P(D)$ in the denominator is the *marginal likelihood* of the data obtained by integrating the likelihood over all possible parameter values:

$$P(D) = \int_{\theta} P(D \mid \boldsymbol{\theta})P(\boldsymbol{\theta})d\boldsymbol{\theta}. \tag{3.9}$$

This term represents the *a priori* likelihood of observing the obtained data given the prior beliefs.

In a discrete BN, when the variables are multinomial, typically the prior over the parameters of the multinomial distributions are represented by Dirichlet distributions. For a node X_i with the distribution

$$P(X_i \mid \boldsymbol{Pa_i} = j) = multinomial(\boldsymbol{\theta}_{ij}) \equiv multinomial(\theta_{ij1,} ...,\theta_{ijk},...,\theta_{ijr_i}), \tag{3.10}$$

the Dirichlet prior is specified as:

$$P(\boldsymbol{\theta}_{ij}) = Dirichlet(\boldsymbol{\alpha}_{ij}) \equiv Dirichlet(\alpha_{ij1},...,\alpha_{ijk},...,\alpha_{ijr_i}), \tag{3.11}$$

where the parameters α_{ijk} of the Dirichlet distribution are called hyperparameters. Applying Bayes rule to a Dirichlet prior and a multinomial likelihood produces a posterior that is also a Dirichlet distribution:

$$P(\boldsymbol{\theta}_{ij} \mid D) = Dirichlet(\alpha_{ij1} + N_{ij1},...,\alpha_{ijk} + N_{ijk},...,\alpha_{ijr_i} + N_{ijr_i}), \qquad (3.12)$$

where, N_{ijk} is the number of occurrences of $P(X_i = k \mid \boldsymbol{Pa}_i = j)$ in the data.

The posterior summarizes all the information available about the different values of the parameters. A common use of a BN, which has been parameterized from a training dataset D of N cases $\{\boldsymbol{X}^1, \boldsymbol{X}^2,\ldots, \boldsymbol{X}^N\}$, is to predict the probability of a new case $\boldsymbol{X}^{N+1}$ taking the value $\boldsymbol{x}$, that is, compute the predictive probability distribution $P(\boldsymbol{X}^{N+1} = \boldsymbol{x} \mid D)$. This prediction is obtained by averaging over all the parameters weighted by the posterior as follows:

$$\begin{aligned} P(\boldsymbol{X}^{N+1} = \boldsymbol{x} \mid D) &= \int P(\boldsymbol{X}^{N+1} = \boldsymbol{x} \mid D, \boldsymbol{\theta}) P(\boldsymbol{\theta} \mid D) d\boldsymbol{\theta} \\ &= \int P(\boldsymbol{X}^{N+1} = \boldsymbol{x} \mid \boldsymbol{\theta}) P(\boldsymbol{\theta} \mid D) d\boldsymbol{\theta} \\ &= \boldsymbol{E}_{P(\boldsymbol{\theta}|D)} P(\boldsymbol{X}^{N+1} = \boldsymbol{x} \mid \boldsymbol{\theta}) \end{aligned} . \qquad (3.13)$$

When the prior and corresponding posterior are Dirichlet, under the assumptions of global and local parameter independence, the prediction for a future example can be decomposed as follows:

$$\begin{aligned} P(\boldsymbol{X}^{N+1} = \boldsymbol{x} \mid D) &= \int P(\boldsymbol{X}^{N+1} = \boldsymbol{x} \mid \boldsymbol{\theta}) P(\boldsymbol{\theta} \mid D) d\boldsymbol{\theta} \\ &= \prod_{i=1}^{n} \int P(X_i \mid \boldsymbol{Pa}_i, \boldsymbol{\theta}_i) P(\boldsymbol{\theta}_i \mid D) d\boldsymbol{\theta}_i \end{aligned} . \qquad (3.14)$$

Assuming that in the future example $\boldsymbol{X} = \boldsymbol{x}$, $X_i = k$ and $\boldsymbol{Pa}_i = j$, the computation yields:

$$\begin{aligned} \boldsymbol{E}[\theta_{ijk}] &\equiv \int P(X_i = k \mid \boldsymbol{Pa}_i = j, \boldsymbol{\theta}_{ij}) P(\boldsymbol{\theta}_{ij} \mid D) d\boldsymbol{\theta}_{ij} \\ &= \frac{\alpha_{ijk} + N_{ijk}}{\alpha_{ij} + N_{ij}} \end{aligned} . \qquad (3.15)$$

From Equations 3.6 and 3.15, it can be seen that the maximum likelihood and the Bayesian parameter estimators have similar form. Both rely on sufficient statistics of the data that are counts of the form $count(X_i = k \text{ and } \boldsymbol{Pa}_i = j)$ that can all be collected simultaneously in a single pass through the data. Both estimators are asymptotically correct in that as the number of cases increases, both methods converge to the true parameter values if the data is actually generated from the given network structure. However, in the finite sample setting the maximum likelihood estimator may overfit. In the Bayesian method such overfitting is counteracted by the use of suitable parameter priors as described in Section 3.4.4.

3.4.2 Structure learning

Learning the structure of a BN is harder than estimating the parameters of a known structure. Typically, structure learning methods learn the structure as well as the parameters of the structure to generate a fully specified BN. Structure learning is influenced by the goal of the learning task as well as the representation used for the conditional probability distributions.

There are two main motivations for learning a BN from data. The first is for knowledge acquisition – to learn important dependencies and independencies among the domain variables. A BN structure not only discovers the independencies but also distinguishes between direct and indirect dependencies both of which lead to correlations in the data. Since the data available for learning is finite and noisily sampled from the actual but unknown joint probability distribution P^*, it is not possible with complete reliability to detect all the independencies in the underlying distribution. Thus, the learned structure may contain fewer arcs and miss true dependencies or may learn more arcs some of which are spurious dependencies. Additionally, several different BN structures can all represent the same distribution; such structures are said to be members of

the *Markov equivalence* class of G^*. Members of G^* cannot be distinguished based on observational data alone. Observational data is data that is passively observed in contrast to experimental data in which one or more variables are manipulated and the effects on other variables are measured [48]. In causal learning, for example, it is important to try to distinguish between members of a Markov equivalence class since different member graphs represent different causal knowledge.

The second reason to learn a BN structure is for density estimation. Here it is less important to capture the actual dependencies and independencies in G^* than it is to approximate the underlying P^* well. That is, the aim is to estimate well a statistical model of the underlying joint probability distribution. Typically, the goal is to learn a statistical model from a training set of data that can be applied to future cases. For example, in classification, the goal is to be able to correctly predict the true state of a target variable using the BN structure learned from the training set.

At first glance it appears that as G^* captures the true dependencies and independencies in the domain, the best generalization will be obtained by recovering the structure G^*. Moreover it appears that it is better to err on having too many arcs in the learned structure than too few, since a more complex structure can still represent the data-generating distribution P^*. However, it turns out that because data available for learning is limited, complex structures can lead to less reliable parameter estimates. For example, adding more parents to a variable Z leads to a larger CPT for Z with fewer data available for estimating each cell of the CPT. Thus, it is often better to prefer a sparser structure even if this structure cannot accurately represent the underlying P^*.

There are two major approaches for learning BNs: (1) *constraint-based* methods that employ independence tests among the domain variables, and (2) *search-and-score* methods that

employ a scoring metric to evaluate the goodness of the statistical model represented by a BN structure. More recently, methods that combine these two traditional approaches have emerged.

Constraint-based methods. These methods view a BN as a representation of independence relations among the domain variables. They attempt to discover a set of conditional dependence and independence relations in the data and use them to determine the presence or absence of arcs in the network. The final induced BN structure is then hopefully one that best captures the independencies in the domain. A key component of these methods is the use of statistical tests that are applied repeatedly to the data for testing conditional independence relations. The main disadvantage of these methods is that with limited data the statistical tests can sometimes fail, and a few errors made by the testing procedure can significantly mislead the network construction procedure.

Search-and-score methods. These methods view a BN as a representation of a statistical model of the domain variables. The scoring function measures the goodness of a BN in terms of how well the corresponding statistical model fits the observed data. The methods then search a hypothesis space of possible network structures to find a single structure or a set of high scoring structures. However, the space of BN structures suffers from combinatorial explosion; it contains a superexponential number of structures – $O(2^{n^2})$, where n is the number of nodes in the network. In general, finding the highest-scoring network has been shown to be NP-hard for a variety of scores, and various heuristic search techniques are used to locate good networks [53].

Search-and-score methods consider the whole structure at once, and are therefore less sensitive to the absence or presence of a few erroneous arcs. In general, finding the optimal BN is intractable and heuristic search is typically used.

The subsequent sections give details of the search-and-score methods, since the patient-specific methods explored in this dissertation employ these methods to learn BNs from data.

3.4.3 Structure scores

Learning a fully specified BN from data using the search-and-score method consists of three components: (1) a scoring function that measures the quality of a network structure with respect to the data; (2) a heuristic search method for exploring the space of network structures; and (3) an estimator for learning the parameters of the conditional probability distributions associated with a specified network structure.

Several scoring metrics have been described. The non-Bayesian scores are discussed first followed by a detailed discussion of the Bayesian scores.

Likelihood score. The simplest scoring function is the likelihood function. This is the same function as the one used for maximum likelihood parameter estimation. Maximizing the likelihood of a BN entails finding both a graph structure and parameters for that structure that maximize the likelihood. For a given structure the likelihood is maximized by simply choosing the maximum likelihood parameters as noted in the parameter estimation section. Extending this, in a given set of BN structures the maximum likelihood structure G is the one which has the highest likelihood when parameterized with MLE estimates:

$$\begin{aligned} &\max_{G,\theta_G} L(B;D) = \max_{G,\theta_G} L(\langle G,\theta_G\rangle;D) \\ &= \max_{G}\left[\max_{\theta_G} L(\langle G,\theta_G\rangle;D)\right] \\ &= \max_{G} L(\langle G,\hat{\theta}_G\rangle;D) \end{aligned} \quad . \qquad (3.16)$$

Therefore, the likelihood structure score is defined as:

$$score_L(G;D) \equiv LL(B;D) = \log L(\langle G, \hat{\theta}_G \rangle; D), \tag{3.17}$$

where, $\hat{\theta}_G$ are the maximum likelihood parameters for *G*. The logarithmic form of the likelihood function is usually used as it makes mathematical manipulations easier. For a discrete BN, the likelihood structure score is:

$$score_L(G;D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}}. \tag{3.18}$$

The likelihood score is a good measure of the fit of a BN to the training data. Typically, the likelihood score overfits the training data and the maximum likelihood BN generalizes poorly. The likelihood score learns a model that precisely fits the specifics of the empirical distribution in the training data, and hence the model captures both true dependencies present in *P** and spurious dependencies that are artifacts of the specific set of cases in the training data. In BN structure learning with the likelihood score, on average the addition of an arc never decreases the score and the highest scoring structure is the completely connected network. The completely connected network will obviously generalize poorly since it captures no independencies present in *P**.

The likelihood score is still useful when additional constraints are present that disallow more complex structures. For example, limiting the maximum number of parents for a node can overcome the likelihood score's tendency to overfit. Another alternative, that is widely used, is to penalize the likelihood score; typically with a term that in some manner measures the complexity of the structure.

Description Length score. The structure learning method based on the Minimum Description Length (MDL) principle searches for a BN that minimizes the combined length of the encoding of the model and the data. The model score is the length of the encoding and is

called the Description Length which consists of two components: (1) the length of the encoding of the model (for a BN the model consists of the domain variables, the structure and the parameters), and (2) the length of the encoding of the observed data. The first component penalizes model complexity, while the second component rewards the model's fitness to the data. Thus, the MDL criterion optimizes the trade-off between the complexity of the selected structure and its fit to the training data. For a discrete BN, the Description Length structure score is defined as:

$$score_{DL}(G;D) = \frac{Dim[G]}{2}\log N - LL(B;D), \qquad (3.19)$$

where $Dim[G]$ is the number of independent parameters in the BN and N is the cardinality of the data D.

3.4.4 Bayesian score

In the Bayesian approach, the scoring function is based on the posterior probability $P(G \mid D)$ of the BN structure G given data D. The Bayesian approach treats both the structure and parameters as random uncertain quantities and incorporates prior distributions for both. The specification of the structure prior $P(G)$ assigns prior probabilities for the different graph structures, and the specification of the parameter prior $P(\boldsymbol{\theta}_G \mid G)$ assigns prior probabilities for the possible parameter values for a specified structure. Application of Bayes rule gives:

$$P(G \mid D) = \frac{P(D \mid G)P(G)}{P(D)}. \qquad (3.20)$$

Since the denominator $P(D)$ does not vary with the structure, it simply acts as a normalizing factor that does not distinguish between different structures. Dropping the denominator gives the Bayesian score which is defined as:

$$score_B(G;D) = \log P(D \mid G) + \log P(G) . \quad (3.21)$$

The second term on the right is the prior over structures, while the first term is the marginal likelihood (also know as the integrated likelihood or evidence) which measures the goodness of fit of the given structure to the data. The marginal likelihood is computed as follows:

$$P(D \mid G) = \int_{\boldsymbol{\theta}_G} P(D \mid \boldsymbol{\theta}_G, G) P(\boldsymbol{\theta}_G \mid G) d\boldsymbol{\theta}_G , \quad (3.22)$$

where $P(D \mid \boldsymbol{\theta}_G, G)$ is the likelihood of the data given the BN (G, $\boldsymbol{\theta}_G$) and $P(\boldsymbol{\theta}_G \mid G)$ is the specified prior distribution over the possible parameter values for the network structure G. Intuitively, the marginal likelihood measures the goodness of fit of the structure over all possible values of its parameters. It is to be noted that the marginal likelihood is distinct from the maximum likelihood, though both are computed from the same function: the likelihood of the data given the structure. The maximum likelihood is the maximum value of this function while the marginal likelihood is the integrated (or the average) value of this function with the integration being carried out with respect to the prior $P(\boldsymbol{\theta}_G \mid G)$.

Marginal likelihood for discrete Bayesian networks. Equation 3.22 can be evaluated analytically when the following assumptions hold: (1) the variables are discrete and the data D is a multinomial random sample with no missing values; (2) global parameter independence, that is, the parameters associated with each variable are independent [54]; (3) local parameter independence, that is, the parameters associated with each parent state of a variable are independent [54]; and (4) the parameters' prior distribution is Dirichlet. Under the above assumptions, the closed form for $P(D \mid G)$ is given by [54-57]:

$$P(D \mid G) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}, \tag{3.23}$$

where $\Gamma(\bullet)$ is the Gamma function, and $\alpha_{ij} = \sum_k \alpha_{ijk}$. Also, as previously described, N_{ijk} is the number of cases in the data where node *i* has value *k* and the parents of *i* have the state denoted by *j,* and $N_{ij} = \sum_k N_{ijk}$. The Bayesian score based on the marginal likelihood is called the *Bayesian-Dirichlet metric* because of the assumption that the parameters are distributed according to a Dirichlet distribution [56].

Priors. The Bayesian score in Equation 3.22 incorporates both structure and parameter priors. The term $P(G)$ is called the structure prior and is the prior probability assigned to the BN structure *G*. In many situations, a uniform prior over all structures is used, in which case the Bayesian score reduces to the marginal likelihood. Heckerman et al. suggest the following structure prior with reference to a prior structure specified by an expert [56]:

$$P(G) = c\kappa^{\delta}, \tag{3.24}$$

where *c* is a normalization constant, $\kappa \in (0,1]$ is a factor that penalizes the structure for each unmatched arc with the prior structure, and δ is the so-called symmetric difference between *G* and the prior structure. If the prior structure is set to the empty network, the Bayesian scoring metric based on this prior gives preference to simpler structures. In the case of BN structures that represent local structure in the form of decision trees or decision graphs, the structure prior will incorporate terms for priors on the global structure as well as terms for priors on the local structures. An example of such a prior is described in the next chapter.

The parameter priors are incorporated in the marginal likelihood as is obvious from Equation 3.23. In the case of the Bayesian-Dirichlet metric several parameter priors have been described. Cooper and Herkovits introduced the K2 metric where all hyperparameters α_{ijk} in

Equation 3.24 are set to 1 [55, 58]. Heckerman et al. showed that if for all nodes *i* the sum $\alpha_0 = \sum_{jk} \alpha_{ijk}$ is constant, then Equation 3.23 yields the same score for all Markov equivalent structures given *D*. Due to this property of likelihood equivalence, this scoring metric is known as the Bayesian Dirichlet likelihood-equivalent (BDe) metric. A special case of the BDe metric is the BDeu (Bayesian Dirichlet likelihood-equivalent and uniform) metric that uses uniform priors such that $\alpha_{ijk} = \alpha_0 / q_i r_i$ where q_i is the number of parent states of X_i and r_i is the number of values of X_i [59].

Bayesian score avoids overfitting. The difference between the marginal likelihood and the maximum likelihood provides one view of why the Bayesian score avoids overfitting. The maximum likelihood overfits because it evaluates the likelihood function at the best parameter values for the training data. These parameter values need not be the optimal values for the data in general because of noise in the training data. The Bayesian approach concurs that the MLE parameter values are the most likely given the training data; however, it emphasizes that there are other parameter values which though less likely are still plausible and should be taken into consideration. By integrating $P(D \mid \boldsymbol{\theta}_G, G)$ rather than maximizing it, the Bayesian approach measures the expected likelihood averaged over different choices of $\boldsymbol{\theta}_G$, which typically leads to a more conservative estimate of the goodness of fit of the model.

3.4.5 Search methods

Given a scoring function, a training dataset and a space of possible network structures, the goal of a search-and-score method is to find a network structure or a set of network structures that maximize the score. Efficient algorithms have been developed for learning network structures

under certain restrictions. For example, in the restricted space of networks that are trees the optimal tree can be learned efficiently in polynomial time [56, 60]. Also, given an ordering on the domain variables finding the network with the highest score consistent with the ordering is not NP-hard [55, 59]. Unfortunately, the task of finding a network structure that optimizes the score is a combinatorial optimization problem, and is known to be NP-hard [53, 61], even if each node is restricted to having at most two parents. Since it is unlikely that there is an efficient algorithm for finding the highest scoring structure, the search-and-score methods employ heuristic search that attempts to find the best network but is not guaranteed to do so. In practice several heuristic search methods like greedy hill-climbing search work well.

Several properties of the scoring function make heuristic search computationally efficient. A key property that aids the search algorithm is the *decomposability* of the score, that is, the score can be expressed as a sum of sub-scores where each sub-score is a function of only one node and its parents (termed *FamScore* below):

$$score(G;D) = \sum_{i=1}^{n} FamScore(X_i \mid \boldsymbol{Pa}_i; G) . \qquad (3.25)$$

To appreciate the advantage of a decomposable score, consider two network structures that differ only in the presence or absence of an arc into a node X_i. To compare the scores of the two networks, it suffices to compute the sub-score $FamScore(X_i \mid \boldsymbol{Pa}_i; G)$ for both structures; the remaining sub-scores are the same for both structures and need not be recomputed. Since the cost of computing the scores of structures usually consumes most of the running time of the algorithm, score decomposability provides a large reduction in running time.

A second property of a scoring function that is useful is *score equivalence*, that is, if two structures belong to the same *Markov equivalence class* they are assigned the same score. In the standard representation of BN that uses DAGs, several distinct DAGs may represent the same

statistical model because they encode the same set of conditional independencies. All DAGs that encode the same set of conditional independencies are said to belong to a single *Markov equivalence class*. Searching in the space of DAGs to find high scoring statistical models can be wasteful since the search procedure is likely to score multiple structures in the same equivalence class. A *score equivalent* scoring function can enable the search algorithm to search in the space of equivalence classes which is smaller than the corresponding space of DAGs. Typically, this property is less crucial than score decomposability for the search algorithm.

Heuristic search for BN structures encompasses several aspects. The major components of heuristic search include the *search space* together with the *operators* for navigating this space, the *scoring function* for evaluating candidate network structures, and the *search procedure*.

The search space is a graph where each vertex represents a candidate network structure and each arc connects two vertices where a network structure represented by one vertex can be converted to the network structure represented by the neighboring vertex by the single application of a valid operator. Each vertex in the search space is associated with a score (that is computed by the scoring function) and has a set of neighboring vertices. The search procedure begins at one vertex and explores the search space via the neighboring vertices to find a high scoring vertex.

One of the earliest search-and-score methods that was developed is the K2 algorithm [55]. This algorithm assumes a topological ordering on the nodes and constrains the number of parents that a node can have. For each node X_i, the search procedure iteratively adds as a parent the node from the set of predecessors of X_i (given in the topological ordering) that most increases the K2 score. The search for the parents of X_i terminates when none of the remaining predecessor nodes when added to the parent set increases the score, or the number of parents exceeds a

predetermined constant. Since an ordering of the nodes may not be easily obtainable in many domains, attempts have been made to relax this requirement. Cooper and Herskovits describe a modification of the K2 algorithm that is based on many random node orderings, and thus does not require a pre-specified node ordering [55]. Another method that overcomes the necessity for a node ordering performs heuristic search over the space of node orderings rather than the space of network structures [62].

The search-and-score methods for leaning BN structures traverse the space of structures by making small modifications to the structure at each step, typically a single arc change, and evaluating the merit of each change. The K2 algorithm, for example, at each iteration of the search selects a new BN structure that has one more arc more than the current structure. The K2 algorithm uses a single operator of adding an arc between two unlinked nodes to generate candidate BN structures. More typically, search-and-score methods employ several single arc change operators [57]. The commonly used operators to make single arc changes are:

- add an arc between two nodes if there is no arc connecting them,
- remove an existing arc, and
- reverse an existing arc.

In the application of these operators only those operations are considered that result in a legal network: the network should be acyclic and should satisfy other additional constraints that may have been specified like a maximum indegree. These algorithms are typically coupled with a decomposable score that consists of a sum of sub-scores, one for each node. The application of the above operators to a structure results in structures whose scores differ from that of the previous structure either in one sub-score (in the case of addition or deletion of an arc) or in two sub-scores (in the case of arc reversal).

These algorithms are said to employ *local search procedures* (hence they are known *local search* algorithms) where the search procedure moves from one candidate structure to a neighboring structure that differs from the previous structure in a single arc resulting from the application of a single operator. The simplest, and often used, local search procedure is the *greedy hill-climbing* procedure. At each iteration, the search procedure selects the neighboring structure with the largest improvement in the score to replace the current structure. The search terminates when no neighboring structure can be found that improves on the score of the current structure. Greedy hill-climbing often works well in practice, although it has the propensity of terminating in a local maximum or a plateau.

Several strategies are employed to escape from local maxima or plateaus and improve on the performance of greedy hill-climbing. One effective strategy is TABU search [63]. TABU search keeps a list of the recently applied operators and at each iteration those operators that result in the reversal of the effect of the recently applied operators are not considered. This forces the search to explore new directions in the search space and escape local maxima.

Another strategy is *random restart* search. When greedy hill-climbing search terminates, the resulting best network is perturbed by the application of several randomly chosen operators, and the greedy search is restarted from the new network. A third strategy is *simulated annealing* that interleaves regular hill-climbing moves with random moves that may temporarily decrease the score in the hopes of leading the search to eventually find models with higher scores.

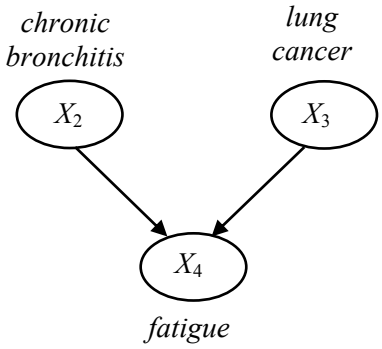
Finally, more exhaustive search methods like *best-first search* and *beam search* can improve on the performance of local search methods. Recent advances in structure learning have led to efficient methods of discovering the optimal structure in small to medium sized domains using exhaustive search. Koivisto describes an algorithm that is only $O(n2^n)$ in time complexity

and $O(n2^n)$ in space complexity where n is the number of domain variables, and demonstrates that the algorithm is practically feasible for domains containing up to 25 variables [64, 65]. This algorithm obtains computational time savings of the order of n^2 over standard structure learning methods.

3.5 LEARNING BAYESIAN NETWORKS WITH LOCAL STRUCTURE

The discussion on structure learning has so far focused on *tabular CPD BNs* and the same discussion is also applicable to *complete CPD BNs* which use the alternate representation of complete decision trees for CPDs. When additional local structure is captured by the use of decision tree CPDs or decision graph CPDs, the standard scoring metrics and search methods described previously can be used with minor modifications. This section describes the modifications to the standard search-and-score method for learning *decision graph CPD BNs*, and in particular, focuses on the Bayesian score and greedy hill-climbing search.

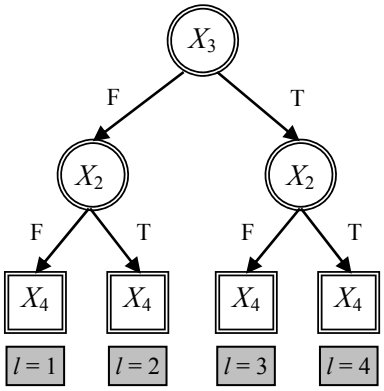
Learning *decision tree CPD BNs* where the local CPDs are represented by decision trees is discussed in detail by Friedman and Goldszmidt [51]. Chickering et al. describe in detail the learning of *decision graph CPD BNs* where the local CPDs are represented by decision graphs [52]. As noted previously, decision graph CPDs are a generalization of decision tree CPDs in that they can represent a richer set of equality constraints among local parameters than either decision tree CPDs or CPTs.



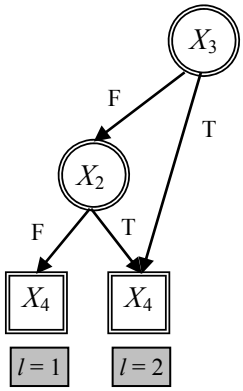
(a)

X_3 (*lung cancer*)	F		T	
X_2 (*chronic bronchitis*)	F	T	F	T
parent state index j	$j = 1$	$j = 2$	$j = 3$	$j = 4$
X_4 (*fatigue* = F, *fatigue* = T)	0.9, 0.1	0.6, 0.4	0.6, 0.4	0.6, 0.4

(b)



(c)



(d)

Figure 3-5: Examples of indexing of parent states in CPDs. Several CPD representations for the BN node X_4 (*fatigue*) in panel (a) are shown in subsequent panels. Panel (b) shows a CPT associated with node X_4 in which the parent states of X_4 are indexed by j as indicated in the shaded row. The CPT can be equivalently represented by a full decision tree as shown in panel (c) in which the parent states of X_4 are indexed by l as indicated in the shaded boxes. Panel (d) shows a decision graph CPD for the node X_4 in which the parent states are indexed by l as indicated in the shaded boxes. In both decision tree and decision graph CPDs the indexing of parent states is carried over the set of leaf nodes.

3.5.1 Bayesian score

A *tabular CPD BN* or a *complete CPD BN* is represented by the pair $(G, \boldsymbol{\theta}_G)$ where G, the global network structure, specifies the set of parents for each node X_i. The local conditional probability distributions $P(X_i \mid \boldsymbol{Pa}_i)$ for each X_i in a *tabular CPD BN* are represented by a unique CPT, the size of which is determined by the number of parent states and the number of states of X_i. Equivalently, in a *complete CPD BN* the local conditional probability distributions for each X_i are represented by a complete decision tree. For both these representations, the Bayesian score of the BN structure is given by Equation 3.21 and the Bayesian parameter estimates for the CPDs at each node X_i are given by Equation 3.15.

For a *decision graph CPD BN*, the specification of the structure S consists of the global network structure G that specifies the set of parents for each node X_i, and, additionally, a local decision graph structure DG_i for each X_i. Thus, the fully specified *decision graph CPD BN* is given by $\langle S \equiv \{G, DG_1, ..., DG_n\}, \boldsymbol{\theta}_S \rangle$ where each DG_i represents the local conditional probability distributions $P(X_i \mid \boldsymbol{Pa}_i)$ for the corresponding node X_i. In both *complete CPD BNs* and *decision graph CPD BNs*, the BN nodes are indexed by the variable i and the states of a BN node X_i are indexed by the variable k. However, the two representations will potentially differ in the number of parent states for a BN node X_i. An illustrative example is given in Figure 3-5. For the BN node X_4 in Figure 3-5 (a), the CPT representation is given in Figure 3-5 (b) where each column corresponds to a parent state and the columns are indexed by the variable j. For the same node, the decision graph representation is given in Figure 3-5 (d) where each leaf node corresponds to a set of parent states that have the same CPD and the leaf nodes are indexed by the variable l. Both representations have the same number of parent states and differ only in the

name of the indexing variable: CPT columns are indexed by variable *j* and decision graph leaf nodes are indexed by the variable *l*. Of note, the CPT in Figure 3-5 (b) can also be represented by a complete decision tree as shown in 3-5 (c).

The Bayesian score and the Bayesian parameter estimates for the *decision graph CPD BN* are now derived. Analogous to the *tabular CPD BN* Bayesian score given by Equation 3.21, the Bayesian score for the *decision graph CPD BN* is:

$$score_B(S;D) = \log P(D \mid S) + \log P(S). \tag{3.26}$$

The marginal likelihood is derived in an analogous fashion to the marginal likelihood of the *tabular CPD BN* given by Equation 3.23:

$$P(D \mid S) = \prod_{i=1}^{n} \prod_{l=1}^{|L_i|} \frac{\Gamma(\alpha_{il})}{\Gamma(\alpha_{il} + N_{il})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ilk} + N_{ilk})}{\Gamma(\alpha_{ilk})}, \tag{3.27}$$

where $|L_i|$ is the cardinality of the set of leaves in the decision graph DG_i of X_i, N_{ilk} is the number of cases in the dataset D that have $X_i = k$ and have parent states of X_i that correspond to one of the paths in the decision graph leading to the leaf node l, and $N_{il} = \sum_k N_{ilk}$. The key difference between Equation 3.23 and 3.27 is in the middle product, which in Equation 3.23 runs over all the columns in the CPT, while in Equation 3.27 it runs over all the leaf nodes of the decision graph of X_i.

The structure prior $P(S)$ in Equation 3.26 can be decomposed into a prior over the global structure G and a prior for each decision graph structure DG_i:

$$\begin{aligned} P(S) &\equiv P(G, DG_1, DG_2, ..., DG_n) \\ &= P(G)P(DG_1, DG_2, ..., DG_n \mid G) \\ &= P(G)\prod_{i=1}^{n} P(DG_i \mid G), \end{aligned} \tag{3.28}$$

where, the decomposition in the second line is obtained by the application of the chain rule of probability and the product in the third line is based on the assumption that the priors for the local structure at each node are specified independently of each other. Substituting Equation 3.28 into Equation 3.26 and expanding *S* gives:

$$score_B(G, DG_1, ..., DG_n; D) = \log P(D \mid G, DG_1, ..., DG_n) + \log P(G) + \sum_{i=1}^{n} \log P(DG_i \mid G) \tag{3.29}$$

A complete specification of a decomposable prior over both the global BN structure and the local decision graph structures is given in the next chapter in conjunction with the description of the patient-specific algorithms.

The Bayesian parameter estimates for the decision graph CPDs of X_i are derived in a similar fashion to the parameter estimates for the CPT of X_i (which is given in Equation 3.15):

$$\hat{\theta}_{ilk} = \frac{\alpha_{ilk} + N_{ilk}}{\alpha_{il} + N_{il}}, \tag{3.30}$$

where, N_{ilk} is the number of cases in the dataset D that have $X_i = k$ and have parent states of X_i that correspond to one of the paths in the decision graph leading to the leaf node l, and $N_{il} = \sum_k N_{ilk}$.

3.5.2 Search methods

The search space for learning *decision graph CPD BN*s is richer than the corresponding space for learning *tabular CPD BN*s (or equivalently the *complete CPD BNs*). The tabular representation provides a single CPT associated with a node X_i, while the decision graph representation provides several possible decision graphs for the node X_i, where each decision graph asserts a

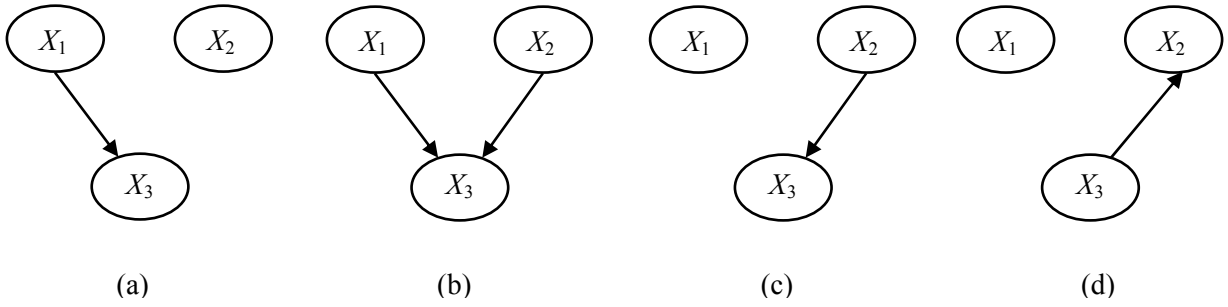


Figure 3-6: Bayesian network global operators: (a) the original BN with three BN nodes where X_1 is a parent of X_3, (b) the result of applying the *add* operator, which adds an arc from X_2 to X_3, (c) the result of applying the *remove* operator, which removes the existing arc between X_1 to X_3, and (d) the result of applying the *reverse* operator, which reverses the existing arc between X_2 and X_3. The *add* and *remove* operators modify the parent set of a single BN node while the *reverse* operator modifies the parent set of a pair BN nodes.

distinct set of equality constraints among the local parameters for X_i. Consequently, the search in the space of *decision graph CPD BN*s has to include a search over the local structure of the nodes. There are two approaches to performing this search.

Encapsulated search spaces. This approach uses a two-tier search procedure that consists of an outer search procedure and an inner search procedure. The outer search procedure generates candidate global structures while the inner search procedure refines a given global structure by generating and evaluating candidate local structures. The operators used in the outer search procedure will be referred to as *global operators* to distinguish them from those that are used in the inner search procedure which will be referred to as *local operators*. Also, two types of nodes will be distinguished. The nodes in the DAG structure of the BN will be termed as *BN nodes* while the nodes in the decision graph structure will be termed as *DG nodes*.

The global operators modify the DAG structure of the BN by adding, removing or reversing an arc between the BN nodes (see Figure 3-6). Application of a global operator results in a new global structure that has one more arc, one less arc or an arc that is reversed which implies that the parent sets of at most two BN nodes have changed.

Following the generation of a new global structure by the outer search procedure, the inner search procedure searches for an optimal decision graph for only those BN nodes whose parent sets have changed by the application of the global operator. A decision graph for a BN node X_i is a graph that contains two types of DG nodes: internal DG nodes and leaf DG nodes. An internal DG node represents a parent of X_i and a leaf DG node represents the parameters of a CPD of X_i. To traverse the space of decision graph structures associated with X_i, a set of three local operators was defined by Chickering [52]. The three local operators are the *complete split*, the *binary split*, and the *merge*. Each local operator modifies the current set of leaf DG nodes in the decision graph structure as follows (see Figure 3-7):

- The *complete split* local operator replaces a leaf DG node in the graph with an internal DG node corresponding to a variable from the parent set. New leaf DG nodes are added as children to the new internal DG node, with one leaf DG node for each distinct state of the variable.
- The *binary split* local operator also replaces a leaf DG node in the graph with an internal DG node corresponding to a variable from the parent set. However, only two new leaf DG nodes are added as children to the new internal DG node, with one leaf DG node for a distinct value of the variable and the remaining leaf DG node for all other states of the variable.
- The *merge* local operator merges two distinct leaf DG nodes into a single leaf DG node that inherits all incoming arcs from both the original leaf DG nodes.

Examples of the application of these local operators are shown in Figure 3-7. These operators are sufficient for moving from a decision graph structure to any other one in the search space. For example, starting with a decision graph containing a single DG leaf node, a complete decision tree can be generated by repeatedly applying the complete split operator. By repeatedly applying the merge operator to the leaves of the complete decision tree, a graph that represents any parameter set equalities can be generated. Though the complete split operator is not essential since it can replaced by a series of applications of the binary split operator, it is included to enable the search procedure to move more efficiently in the search space.

Typically, greedy hill-climbing search is used to locate a high-scoring decision graph structure. The search starts with a decision graph containing a single DG leaf node and applies the local operators to generate candidate local structures. An example showing search in an

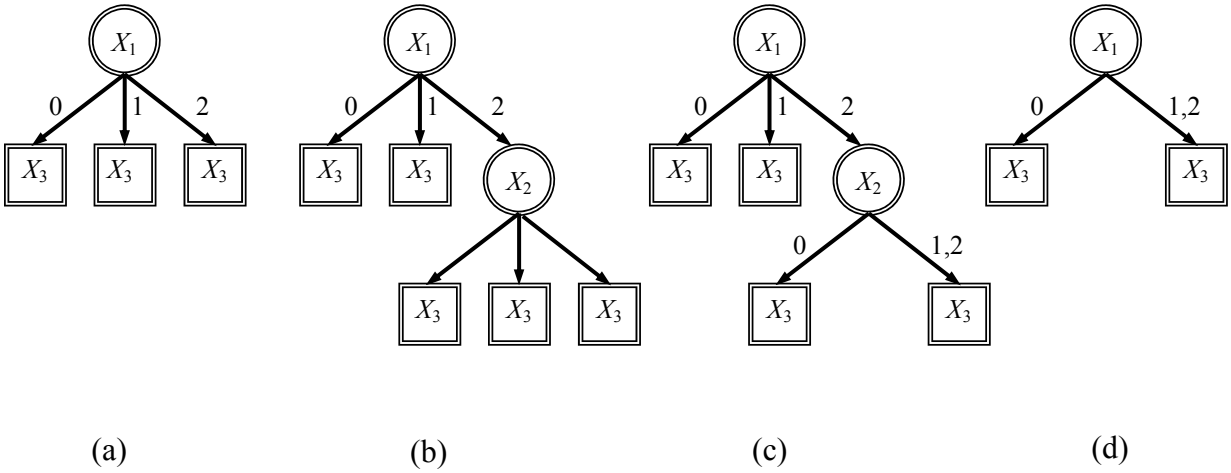


Figure 3-7: Bayesian network local operators: (a) the original decision graph for the BN node X_3, showing one parent X_1 that is in X_3's decision graph, (b) the result of applying the *complete split* operator, which splits based on all values of X_1, (c) the result of applying the *binary split* operator, which splits one state of X_2 from all other states, and (d) the result of applying the *merge* operator, which merges two values of X_1 into a single value. In this figure, all the variables have three states, but in general each variable can have an arbitrary number of discrete states. Figure modified from [52].

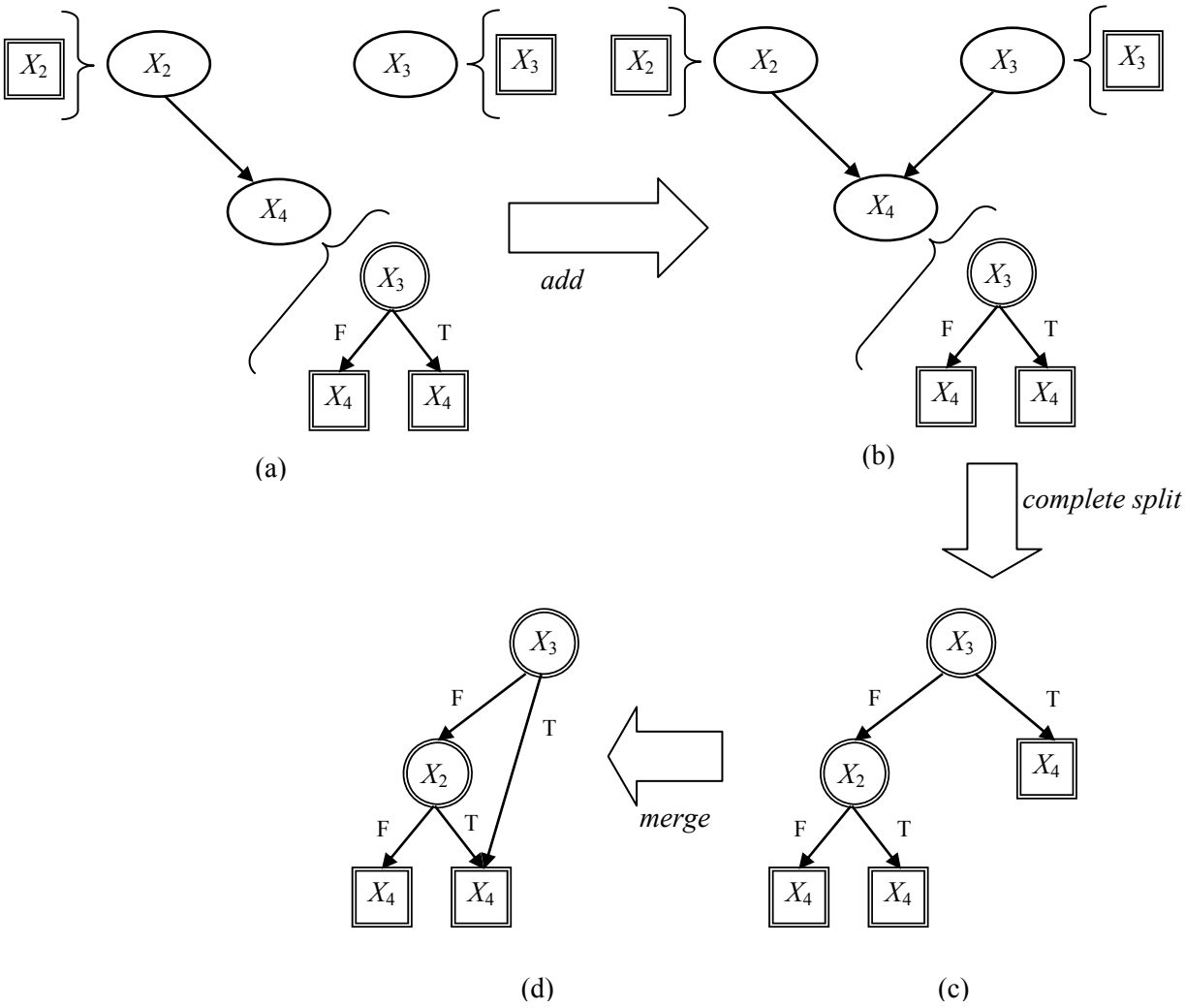


Figure 3-8: Example of encapsulated search demonstrating the application of a global operator followed by the application of two local operators. BN nodes are shown as ellipses with a single border and DG nodes are shown as circular or rectangular nodes with double borders. Application of the global operator *add* that adds an arc from node X_3 to X_4 to the structure in (a) results in the structure in (b). Application of the local operator *complete split* to the left hand leaf node of the decision graph of X_4 in (b) results in the decision graph in (c). Application of the local operator *merge* to the two right hand leaf nodes of the decision graph of X_4 in (c) results in the decision graph in (d).

encapsulated search space is given in Figure 3-8.

Unified search spaces. An alternative approach for learning the local structure of a BN node employs a unified search space. Instead of two sets of operators, one for modifying the global DAG and another for modifying the local decision graph, a single set of operators modify the joint representation of the global network and the local structures in a single search space.

This is feasible since the local structure associated with the BN node X_i identifies the set of BN nodes that are the parents of X_i; the set of all local structures thus identifies the parents of every BN node which is sufficient to uniquely identify the DAG of the BN.

In the case of *decision graph CPD BNs*, each member in the unified search space consists of a collection of n decision graphs $\langle DG_1,...,DG_n \rangle$ that represent the local structures associated

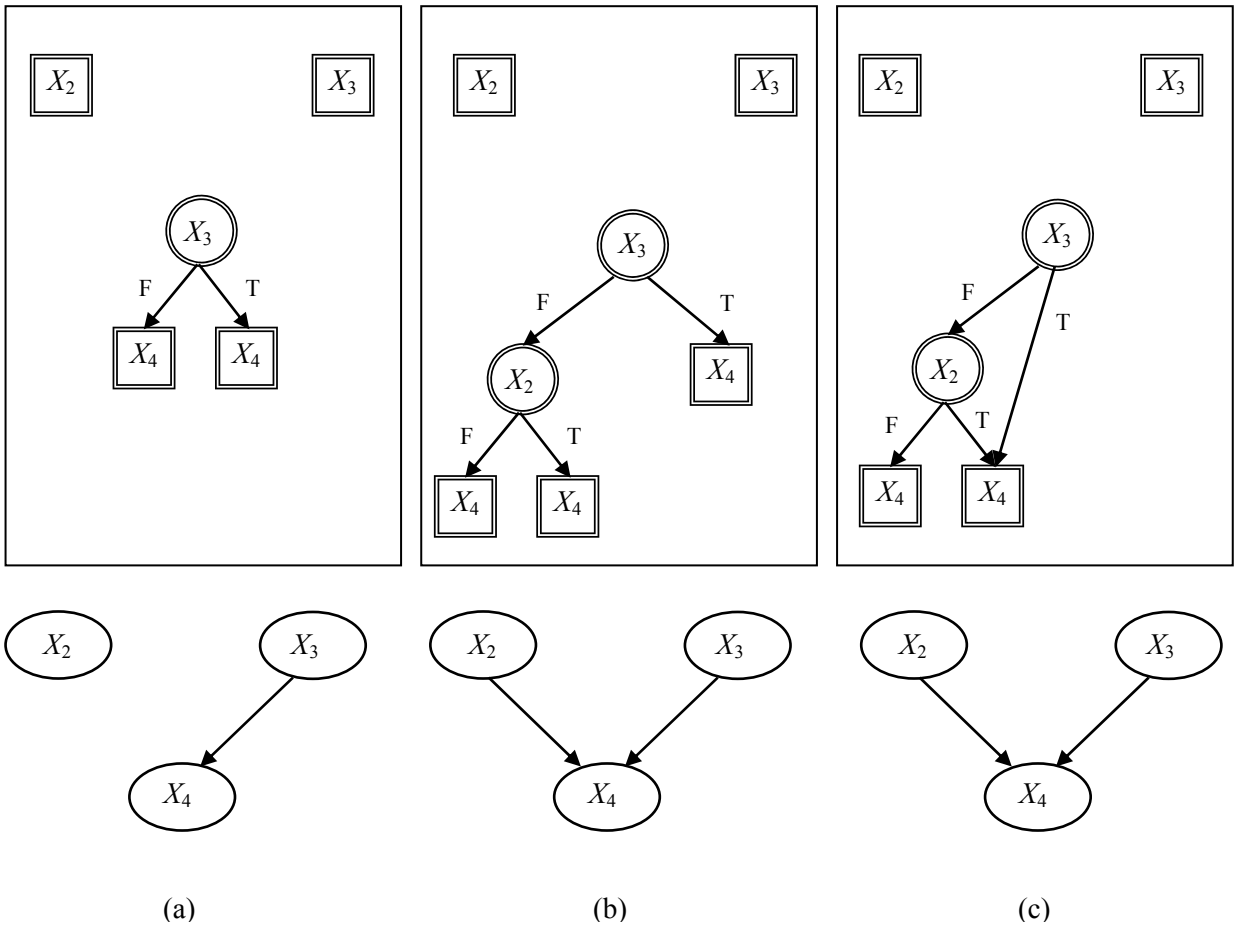


Figure 3-9: Example of unified search demonstrating the application of two operators. BN nodes are shown as ellipses with a single border and DG nodes are shown as circular or rectangular nodes with double borders. In each panel, the set of local structures is shown at the top enclosed in a box and the implied global structure is shown at the bottom. Application of the operator *complete split* to the left hand leaf node of the lower decision graph in (a) results in the decision graph in (b). Application of the operator *merge* to the two right hand leaf nodes of the lower decision graph in (b) results in the decision graph in (c).

with the *n* domain variables $X_1,...,X_n$. A collection of local structures induces a unique network structure among the domain variables: an arc $X_j \rightarrow X_i$ is present in the network structure if there is a node for X_j in the decision graph DG_i. Only those collections of decision graphs that induce an acyclic network among the domain variables are legal members of the search space. The operators for moving in this search space are typically the same ones that modify the local structure in the encapsulated search space, namely, the complete split, binary split, and merge local operators described above. The global operators of adding, deleting and reversing arcs are not needed in this search space.

Typically, greedy hill-climbing search is used to locate a high-scoring structure. The search starts with an empty network with each node's decision graph initialized to a single root *DG* node. Each node X_i is considered in turn as follows. All non-descendants that are not already parents of X_i are added to the parent set of X_i and the highest scoring decision graph is learned in a greedy fashion. This decision graph DG_i becomes the local structure for X_i. Any parent variable that is not present in DG_i is removed from the parent set of X_i and the corresponding arc $X_j \rightarrow X_i$ is deleted from the global structure. The search terminates after all the nodes have been considered. An example showing search in a unified search space is given in Figure 3-9.

3.6 LEARNING BAYESIAN NETWORK CLASSIFIERS FROM DATA

Classification is a central problem in machine learning that involves inducing a classifier from a set of classified training cases that can be applied to unclassified cases. The goal in classification is to predict the value of a discrete class variable *Z* from the known values of a set of predictor

variables $\boldsymbol{X} = \{X_1, X_2, ..., X_n\}$. Given a set of classified training cases $D = \{(\boldsymbol{x}^1, z^1), (\boldsymbol{x}^2, z^2), ..., (\boldsymbol{x}^m, z^m)\}$ a classification algorithm induces a classifier or a classification rule $c(\boldsymbol{x})$ that is capable of predicting the likely value of Z for future cases where only the values of the predictor variables $\boldsymbol{X}$ are known.

Probabilistic classifiers specify a probability distribution over the class variable conditioned on the predictor variables. This distribution is then used for deciding the predicted class for a test case where the values of the predictor variables are known. For example, if the class variable is binary, a threshold is selected and the test case is assigned to one class value if the predicted probability is above the threshold or to the other class value if below the threshold. A test case where the predicted probability is equal to the threshold may be assigned to either class value. For a class variable with more than two classes, the test case is typically assigned to the class value with the highest probability.

In the decision-theoretic framework, the distribution over the class variable specified by the probabilistic classifier is combined with a *loss function* (or *cost function*) to formally specify a *decision rule* which is termed a *classification rule* in the context of classification. A classification rule $c(\boldsymbol{x})$ is a function that maps every possible combination of variable values $\boldsymbol{x}$, to a class value z. A loss function $L(z_i, z_j)$ specifies the loss or cost that is incurred by predicting the class value z_i when the true value is z_j, for all values z_i and z_j. The *expected loss* or *expected misclassification cost*, incurred in predicting the class value z_i on observing the variable values $\boldsymbol{x}$ is:

$$EL(z_i \mid \boldsymbol{x}) = \sum_j L(z_i, z_j)\, p(z_j \mid \boldsymbol{x}). \qquad (3.31)$$

This is the weighted average of the losses incurred by predicting the particular class value z_i as the true class value ranges over all possible values z_j, with each loss corresponding to the true class value z_j being weighted by the predicted probability $p(z_j \mid \boldsymbol{x})$ for that class. According to the *Bayes decision procedure*, the optimal class value is the one that minimizes the expected loss or the expected classification cost. Thus, the classification rule is:

$$c(x) = \arg\min_{z_i}\left[EL(z_i \mid \boldsymbol{x})\right]. \qquad (3.32)$$

To summarize, probabilistic classifiers such as BN classifiers (described later), operate by estimating the probability distribution $p(z \mid \boldsymbol{x})$ for a test case at hand, which is then used in computing the expected misclassification costs for each of the class values using equation 3.31. Finally, the classification rule in equation 3.32 selects the class value with the minimum expected misclassification cost. Thus, the predicted class value depends on both the probabilities estimated by the classifier and the specified loss function.

3.6.1 Minimum error rate classification

A commonly used criterion for evaluating the performance of classifiers is the *misclassification error rate* or simply the *error rate*. If the true class value is z_i, then predicting any class value other than z_i results in a misclassification error. If the error rate is to be low then it is natural to seek a classification rule that minimizes the probability of error. The loss function used for minimizing the error rate is the *zero-one* loss function, which assigns zero loss to a correct classification and unit loss to any misclassification thus penalizing all errors equally:

$$L(z_i, z_j) = 1 \quad if\ i \neq j\ and\ 0\ otherwise. \qquad (3.33)$$

The expected loss for predicting the class value z_j is:

$$
\begin{aligned}
EL(z_j \mid \boldsymbol{x}) &= \sum_j L(z_i, z_j)\, p(z_i \mid \boldsymbol{x}) \\
&= \sum_{j \neq i} p(z_i \mid \boldsymbol{x}) \\
&= 1 - p(z_j \mid \boldsymbol{x})
\end{aligned}
\quad , \qquad (3.34)
$$

where the last term $1 - p(z_j \mid \boldsymbol{x})$ is the average probability of error, since $p(z_j \mid \boldsymbol{x})$ is the conditional probability that the class value z_j is correct given $\boldsymbol{x}$ is observed. The classification rule according to the Bayes decision procedure is

$$
\begin{aligned}
c(x) &= \arg\min_{z_j} \left[EL(z_j \mid \boldsymbol{x}) \right] \\
&= \arg\min_{z_j} \left[1 - p(z_j \mid \boldsymbol{x}) \right], \\
&= \arg\max_{z_j} \left[p(z_j \mid \boldsymbol{x}) \right]
\end{aligned}
\qquad (3.35)
$$

which states that the average probability of error is minimized when the class value z_j with the highest predicted probability is selected. Thus, the optimal minimum error rate classifier simply chooses the class value with the highest probability.

3.6.2 Calibration

Another criterion that is used for evaluating the performance of a classifier is *calibration*. Calibration is the extent to which the classifier's probability estimate agrees with the true probability. More precisely, the predicted probability p of a class z_i is well calibrated when the cases assigned a probability p of belonging to class z_i actually belong to that class a fraction p of the time.

It is possible for a classifier to have a small error rate yet have poor calibration. For example, consider a classifier that consistently produces excessively high probabilities for the true class. The classifier remains accurate if it produces any probability that is higher than the

calibrated probability p for the true class. However, its predicted probability for the true class is far too high leading to poor calibration. Naïve Bayes classifiers have been shown to produce probabilities that are arbitrarily close to 1 or arbitrarily close to 0 that are more extreme than warranted [66]. This behavior typically does not increase the error rate but leads to poor calibration.

Quite commonly, a classifier learns a classification rule by directly optimizing the error rate or the zero-one loss function. However, this may be inappropriate when the correct loss function is unknown or the zero-one loss function is unsuitable. For example, in predicting clinical outcomes with classifiers, the misclassification cost may be significantly influenced by patient utilities in which case the misclassification loss function varies from case to case. Therefore minimizing the error rate, which corresponds to minimizing a fixed misclassification cost, is inappropriate. If the estimated probabilities are well calibrated, optimal predictions will be obtained for any future misclassification costs that may need to be applied.

3.6.3 Bayesian network classifiers

Several probabilistic classification algorithms use Bayesian network models for classification. The naïve Bayes classifier is the simplest BN classifier that is learned very efficiently from data. Among its early applications was in the context of a medical diagnostic system [67]. In spite of its simplicity it often has excellent misclassification error rates and it is widely used as a benchmark against which to compare new classifiers. The naïve Bayes classifier makes the strong assumption that the predictor variables are mutually independent conditioned on the class variable, which implies that all predictors are considered equally important for classification and

that the predictors do not interact. This assumption allows for a very parsimonious representation of the joint probability distribution over the domain variables:

$$p(Z, \boldsymbol{X}) = p(Z)\prod_{i=1}^{n} p(X_i \mid Z) . \qquad (3.36)$$

Learning the naïve Bayes classifier from data, without variable selection, is simple since no structure learning is necessary. The learning of the parameters of the model requires estimating the class probability distribution $p(Z)$ and the conditional probabilities $p(X_i \mid Z)$ for each predictor variable X_i. This is done easily and efficiently from data using either the Bayesian or the non-Bayesian methods described in Section 3.4.1.

The Tree Augmented Naïve Bayes (TAN) classifier extends the naïve Bayes classifier by relaxing the naïve Bayes structure to allow one extra parent per predictor variable in addition to the class variable. This enables the modeling of interactions among predictor variables not captured by the class variable and thus overcomes the main weakness of the naïve Bayes classifier. The TAN classifier has been shown to improve on the accuracy of the naïve Bayes classifier while maintaining its computational simplicity of learning [68].

However, extending the TAN classifier to the general Bayesian network classifier does not necessarily improve on the classification performance over simpler BN structures. Intuitively, learning general BNs corresponds to solving the more general problem of learning the joint probability distribution over all the variables in the domain, whereas learning a classifier corresponds to solving the simpler problem of learning the conditional distribution of the class variable given the predictor variables. The standard scores used for learning BNs are proportional to the joint likelihood of all the variables. The Bayesian score, for example, is proportional to the log marginal likelihood, $\log p(D \mid M)$, which can be decomposed as

$$\log p(D \mid M) = \log p(D^Z \mid D^X, M) + \log p(D^X \mid M), \qquad (3.37)$$

where, D^Z is the segment of the training dataset limited to the values of the target variable Z and D^X is the remaining portion of the dataset limited to the values of the predictor variables $\boldsymbol{X}$. The first term on the right hand side of Equation 3.37 measures how well M estimates the probability of the target variable Z given the predictor variables $\boldsymbol{X}$, and this term directly relates to the performance of M as a classifier. Note, however, that as the number of predictors increases the second term dominates the overall score. Thus, it is possible for a high scoring BN structure to have a high misclassification error rate if the contribution of the second term to the score is relatively larger than that of the first term to the overall score.

This first term, known as the conditional likelihood, can form the basis for a better score for learning BN classifiers. However, the conditional likelihood does not decompose into separate terms for each variable and hence is not node-decomposable as is the case for the marginal likelihood. There is, also, no known closed form for computing the parameters that maximize the conditional likelihood, as is the case for the marginal likelihood; hence there is no known tractable algorithm that learns both the structure and parameters of a general Bayesian network that maximizes the conditional likelihood. Recently, efforts have been made to induce Bayesian network classifiers that approximately maximize the conditional likelihood. One approach uses a numerical method to estimate parameter values that maximize the conditional likelihood of a given BN structure [69]. Another approach computes the conditional likelihood of a BN structure using the easily estimated maximum likelihood parameter values rather than the maximum conditional likelihood parameter values [70]. These approaches have been shown to improve on the performance of naïve Bayes and TAN classifiers.

3.7 BAYESIAN MODEL AVERAGING

Choosing a single BN structure that has a large Bayesian score as described so far is a form of *model selection* that can sometimes serve as an approximation to the complete Bayesian approach of *model averaging*. In Bayesian model averaging (BMA) with BNs, in addition to modeling the uncertainty in parameters, the uncertainty in the BN structure is modeled as well.

The general procedure for BMA is as follows [19]. If h is a quantity of interest, such as an effect size, a future observable, or the utility of a course of action, BMA computes the probability distribution of h given the data D by averaging over possible structures and their parameters:

$$p(h \mid D) = \sum_m p(h \mid m, D) p(m \mid D), \tag{3.38}$$

where

$$p(h \mid m, D) = \int p(h \mid \boldsymbol{\theta}_m, m) p(\boldsymbol{\theta}_m \mid m, D) d\boldsymbol{\theta}_m . \tag{3.39}$$

Equation 3.39 emphasizes that the probability distribution of h is an average of the probability distributions of h under each model m weighted by the posterior probability of that model given the data. Equation 3.38 shows that for each model m, the probability distribution of h is obtained by integrating over all parameters.

Given data, the posterior distributions for each m and $\boldsymbol{\theta}_m$ are obtained by applying Bayes' rule:

$$p(m \mid D) = \frac{p(D \mid m) p(m)}{\sum_{m'} p(D \mid m') p(m')}, \tag{3.40}$$

$$p(\boldsymbol{\theta}_m \mid m, D) = \frac{p(D \mid \boldsymbol{\theta}_m, m) p(\boldsymbol{\theta}_m \mid m)}{p(D \mid m)}, \tag{3.41}$$

where

$$p(D \mid m) = \int p(D \mid \boldsymbol{\theta}_m, m)\, p(\boldsymbol{\theta}_m \mid m) d\boldsymbol{\theta}_m \,. \tag{3.42}$$

The term $p(D \mid m)$ is called the marginal likelihood or the integrated likelihood. Here the uncertainty about m is encoded in the prior distribution $p(m)$. In addition, for each model structure m, the uncertainty about $\boldsymbol{\theta}_m$ is encoded in the prior distribution $p(\boldsymbol{\theta}_m \mid m)$.

When BNs structures are used for classification, the quantity of interest h is the value of the class variable Z^t for a future case t with features $\boldsymbol{X}^t = \boldsymbol{x}^t$. According to BMA, the posterior distribution $P(Z^t \mid \boldsymbol{x}^t, D)$ is obtained by averaging over all BN structures G:

$$P(Z^t \mid \boldsymbol{x}^t, D) = \sum_G P(Z^t \mid \boldsymbol{x}^t, G, D) P(G \mid D) \,. \tag{3.43}$$

Averaging over all the models in this fashion provides better predictive ability, as measured by logarithmic loss than using any single model [71, 72].

4.0 METHODOLOGY

This chapter describes the patient-specific approach to learning Bayesian networks from data. After a summary of the main ideas of this approach that have been covered in the previous chapters, a detailed description of two versions of the patient-specific algorithm is given.

The goal of the patient-specific algorithm is to predict well a discrete target variable of interest, such as a patient outcome. In particular, the focus is on the prediction of low-dimensional, atemporal outcomes (e.g., binary outcomes such as death versus survival). In machine learning terminology, the patient-specific algorithm is an example of an instance-specific classification algorithm.

Relative to some model space, *Bayesian model averaging* is the optimal method for making predictions in the sense that it achieves the lowest expected error rate in predicting the outcomes of future cases. Such Bayes optimal predictions involve averaging over all models in the model space which is usually computationally intractable. One approach, termed *selective model averaging*, has been to approximate the Bayes optimal prediction by averaging over a subset of the possible models and has been shown to improve predictive performance [19, 71, 72]. The patient-specific algorithm performs selective model averaging and uses a novel heuristic search to select the models over which averaging is done. The patient-specific characteristic of the algorithm arises from the observation that the search heuristic is sensitive to the features of the particular case at hand.

The model space employed by the patient-specific algorithm is the space of Bayesian networks over the domain variables. In particular, the algorithm considers only Markov blankets of the target node, since this is sufficient for predicting the target variable. Two versions of the patient-specific algorithm are considered that differ in the representation employed for the conditional probability distributions. The *patient-specific Markov blanket global* structure (PSMBg) algorithm learns MBs that allow for explicit representation of only global independencies among nodes of the MB. The *patient-specific Markov blanket local* structure (PSMBl) algorithm learns MBs that allow for explicit representation of value-specific independencies in the conditional distributions associated with a node. This implies that the PSMBl algorithm employs a richer space of models than the PSMBg algorithm.

The remainder of the chapter describes the patient-specific algorithms in terms of the (1) model space, (2) representations of the models, (3) scoring metrics including parameter and structure priors, and (4) the search procedure for exploring the space of models. The current versions of the algorithms handle only discrete variables and do not handle missing values.

4.1 MODEL SPACE

The primary goal is to compute the predictive distribution of the target variable. In a BN, the nodes that effect the distribution of the target node are those contained in the Markov blanket (MB) of the target, and include the parents, the children and the parents of the children (spouses) of the target node. Provided the MB nodes of the target are observable, nodes of the BN that are not part of the MB are not required for determining the distribution of the target and are hence

not needed for inference as asserted by the global Markov condition. Therefore, the patient-specific algorithms learn MBs of the target variable rather than entire BNs over all the variables.

Typically, BN structure learning algorithms that learn from data, such as the search-and-score and the constraint-based methods described in the previous chapter, induce a BN structure over all the variables in the domain. The MB of the target variable can be extracted from the learned BN structure by ignoring those nodes and their relations that are not members of the MB. However, it is practically somewhat more efficient to modify the typical BN structure learning algorithm to learn only MBs of the target node of interest, by using a set of global operators that have been modified to generate only the MB structures of the target variable.

The patient-specific Markov blanket algorithms are search-and-score methods that search in the space of possible MB structures. The Bayesian network structure learning algorithms search in the space of possible BN structures which is exponential in the number of domain variables. The number of BN structures with n variables is given by the following recurrence formula where $BN(n)$ is the number of DAGs that can be constructed from n nodes [73, 74]:

$$\begin{aligned} BN(n) &= \sum_{k=1}^{n} (-1)^{k-1} C(n,k) 2^{k(n-k)} BN(n-k) \quad for\ n > 0 \\ BN(0) &= 1 \end{aligned}, \qquad (4.1)$$

where, $C(n, k)$ is the count of the number ways to choose k objects from n distinct objects. Hence, exhaustive search in the space of BN structures is infeasible for domains containing more than a few variables and heuristic search is appropriate.

The number of MB structures with respect to a single target variable in a domain with m variables (where m excludes the target variable) is given by the following equation:

$$MB(m) = \sum_{m_P=0}^{m} \sum_{m_c=0}^{m-m_p} \left(\frac{m!}{m_p!\, m_c!\, m_o!} \right) 2^{m_c . m_p} 2^{m_c . m_o} BN(m_c) \quad for\ m > 0 \qquad (4.2)$$
$$MB(0) = 1$$

where, m_p is number of parent nodes, m_c is the number of child nodes, m_o is the number of other nodes, and is computed as $m_o = m - m_p - m_o$. $BN(m_c)$ is the number of DAGs that can be constructed from n_c nodes and is computed from Equation 4.1. The derivation of the recurrence formula in Equation 4.2 is given in Appendix A.

Table 4-1 gives the number of BN and MB structures for domains having up to 12 variables. It can be seen from the table that while there are fewer MB structures of the target variable than there are BN structures, the number of MB structures is nevertheless exponential in the number of variables. Thus, exhaustive search in the space of MB structures is also infeasible for domains containing more than a few variables and heuristic search is appropriate.

Table 4-1: Number of Bayesian network structures *BN*(*n*) and Markov blanket structures *MB*(*n-1*) as a function of number of nodes *n*. The number of Markov blanket structures is with respect to a single node and is not a count of all Markov blanket structures for all nodes. The last column gives the ratio of the two types of structures. Both *BN*(*n*) and *MB*(*n*-1) are exponential in *n*.

n	*BN*(*n*)	*MB*(*n-1*)	*BN*(*n*) / *MB*(*n-1*)
0	1	-	-
1	1	1	1.0
2	3	3	1.0
3	25	15	1.67
4	543	153	3.55
5	29281	3567	8.21
6	3781503	196833	19.21
7	1138779265	25604415	44.48
8	783702329343	7727833473	101.41
9	1213442454842881	5321887813887	228.01
10	4175098976430598143	8241841773665793	506.57
11	31603459396418917607425	28359559029362676735	1114.38
12	521939651343829405020504063	214672167825864945784833	2431.33

4.2 MARKOV BLANKET LOCAL STRUCTURE

The PSMBg algorithm learns *complete CPD MBs* in which the CPDs are represented with complete decision trees. Complete decision trees capture only the *global structure*, that is,

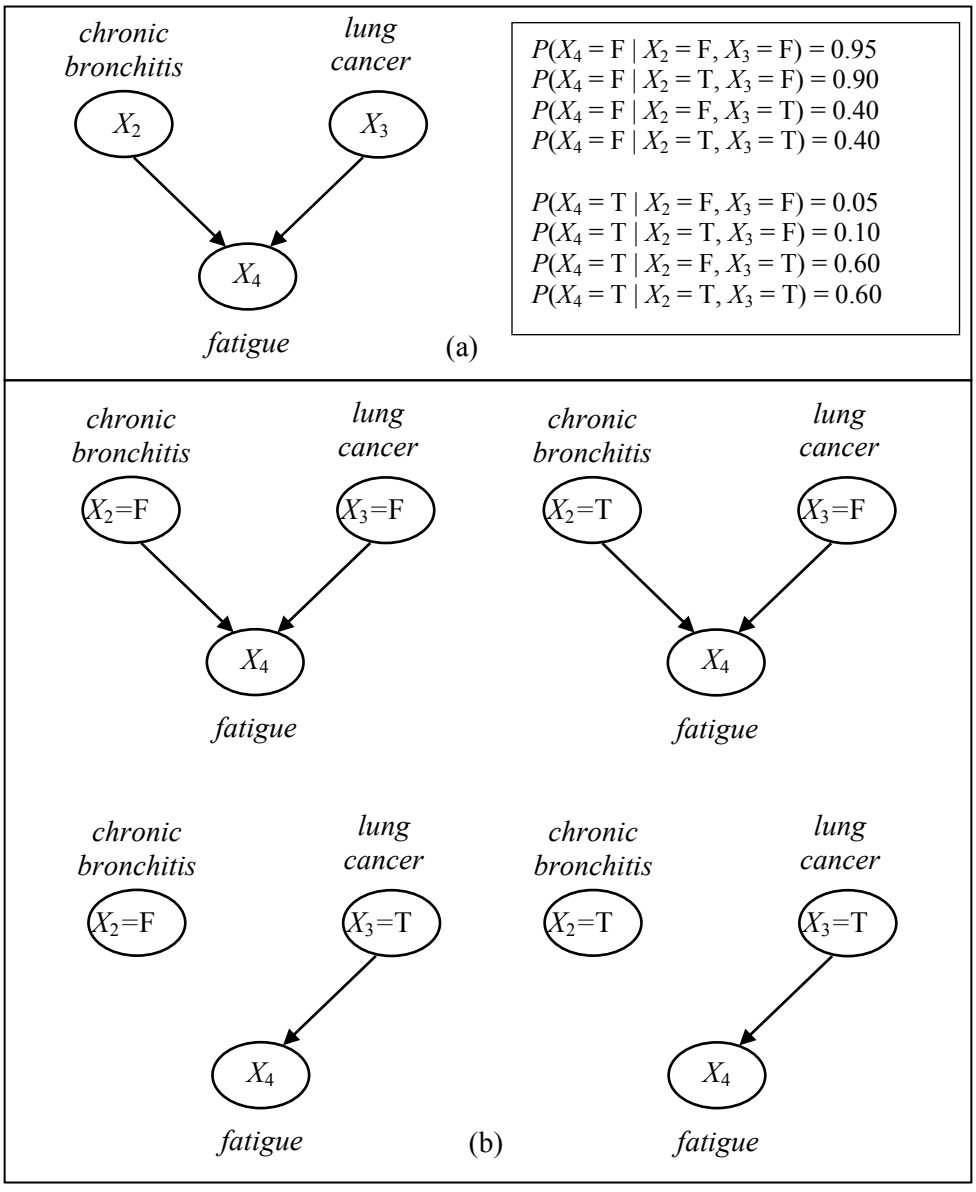
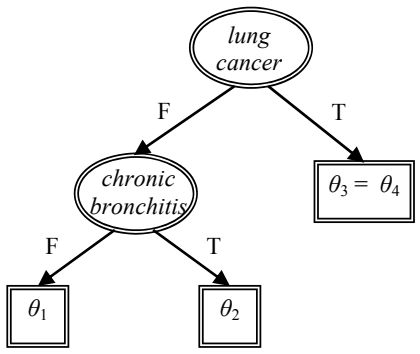


Figure 4-1: An example in which the local Markov blanket structure depends on the value of lung cancer. Panel (a) shows the global structure and the associated parameters for the node X_4. Panel (b) illustrates four structures that explicitly demonstrate how the structure varies for different values of *lung cancer*. The values T and F stand for True and False respectively.

independence relations among variables that hold for all values of the variables. The PSMBl algorithm learns *graph CPD MBs* in which CPDs are represented with decision graphs. Decision graphs capture the *local structure*, that is, value-specific independencies among the variables. Value-specific independencies are those that hold only for particular assignments of values to certain nodes and cannot be explicitly represented by the global structure.

Bayesian networks that can represent local structure (i.e., local constraints) that hold among the parameters of a node have been shown to capture additional independences with fewer parameters [75]. Corresponding to the parents of a node in a global Bayesian network, in a local Bayesian network, a node X_i has a set of global parents $\boldsymbol{Pa_i}$. However, in general, for a particular instantiation of the variables in $\boldsymbol{Pa_i}$, only some of those variables will be correlated



lung cancer	F	F	T	T
chronic bronchitis	F	T	F	T
fatigue = F	θ_1	θ_2	θ_3	θ_4
fatigue = T	1- θ_1	$1-\theta_2$	$1-\theta_3$	$1-\theta_4$

Figure 4-2: A decision tree representation of the local structure for the variable *fatigue* that captures the four structures shown in Figure 4-1 (b). The parameters at the leaves are explained in the table.

with X_i. For example, in Figure 4-1 (a), suppose when the node *lung cancer* has the value True, the node *chronic bronchitis* has no influence on the node *fatigue*. Indeed the probabilities in Figure 4-1 (a) reflect this situation. Figure 4-1 (b) illustrates the local Bayesian network structure that represents this example.

One representation for the local structure is the decision tree; this representation is described in detail in Section 3.3.2. Briefly, in this representation, a decision tree is used to represent the local structure between a node and its parents in a Bayesian network [76]. The decision tree used here is not a decision analytic decision tree, but a CART-like probability tree in which branches denote variable values. Each leaf node in the tree contains the probability distribution of the variable being predicted given the values of the predictor variables that appear along the path that goes from the root node to that leaf node. Figure 4-2 shows a decision tree representation of the local structure for the variable *fatigue* given in Figure 4-1. The parameters used for the CPDs at the leaves of the decision tree are shown in the associated table in the figure. Root-to-leaf paths in the tree correspond to value-specific parents of the variable *fatigue*.

Another representation for the local structure is the decision graph which is a generalization of the decision tree; this representation is described in detail in Section 3.3.3. Briefly, in a decision graph non-root nodes can have multiple parents, rather than a single parent as in a decision tree [52]. A decision graph, thus, allows two or more branches of a decision tree to share the same leaf node, which expresses the following equality constraint: conditioned on the variable values in any one of the shared branches (the parents), the conditional probability distribution of the leaf node (the child) is the same. All equality constraints represented by decision trees can be represented by decision graphs, but not vice-versa. Figure 4-3 illustrates a decision graph representation for the decision tree in Figure 4-2. Figure 4-4 shows an example of

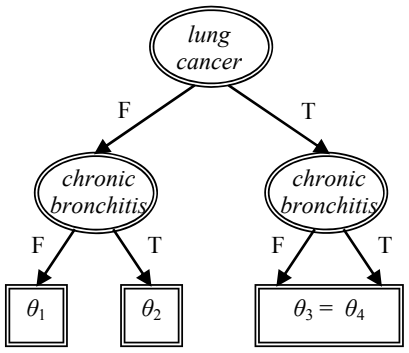


Figure 4-3: A decision graph representation of the local structure represented by the decision tree in Figure 4-2.

a decision graph where the shared leaf node expresses that $P(\textit{fatigue} = \text{T} \mid \textit{lung cancer} = \text{F}, \textit{chronic bronchitis} = \text{T}) = P(\textit{fatigue} = \text{T} \mid \textit{lung cancer} = \text{T}, \textit{chronic bronchitis} = \text{F})$. This is an example of a local structure that cannot be represented with a decision tree.

The PSMBl algorithm uses decision graphs to represent and explicitly capture value-specific independences among the CPDs of a node, while the PSMBg algorithm represents the CPDs of a node with a complete decision tree which is equivalent to a conditional probability table. Figure 4-5 shows an example of a complete decision tree that would be used by the

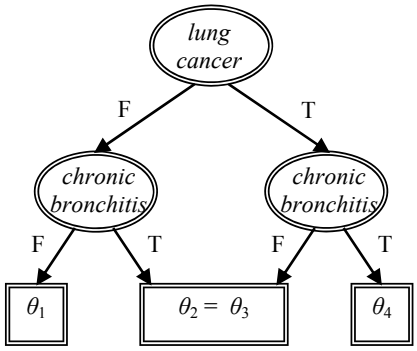


Figure 4-4: A decision graph representation of a local structure that cannot be represented by a decision tree.

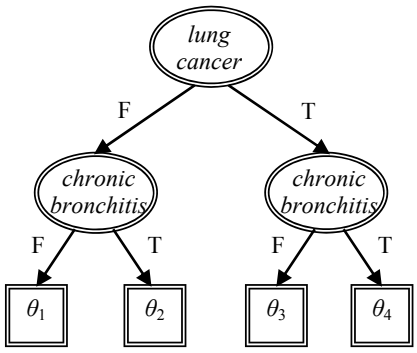


Figure 4-5: An example of a complete decision tree representation used by the PSMBg algorithm to represent the probability distributions associated with the node *fatigue*.

PSMBg algorithm. Note that the complete decision tree is a type of decision graph, in which, every possible parent state is accorded a distinct leaf node. In effect, the complete decision tree encodes the same parameters as the equivalent CPT; each leaf node contains a table that corresponds to a column in the CPT that encodes the parameters of a conditional probability distribution.

Given a set of parents, a MB node has a single global structure while it can have several possible local structures. The single global structure can be represented by a complete table (CPT) or a complete decision tree. The number of possible local structures is the number of all possible decision graphs that can be constructed for a given set of parent states. A formula for counting all possible decision graphs is derived in Section 4.3.2. From this it follows that the PSMBl algorithm's model space is richer than the PSMBg algorithm's model space; in fact the latter is a subset of former.

4.3 PATIENT-SPECIFIC BAYESIAN MODEL AVERAGING

The objective of the patient-specific algorithms is to derive the posterior distribution $P(Z^t \mid \boldsymbol{x}^t, D)$ for the target variable Z^t in the case at hand, given the values of the other variables $\boldsymbol{X}^t = \boldsymbol{x}^t$ and the training data D. The Bayes decision rule is then applied to select the target value with the highest posterior probability. The ideal computation of the posterior distribution $P(Z^t \mid \boldsymbol{x}^t, D)$ by Bayesian model averaging is as follows:

$$P(Z^t \mid \boldsymbol{x}^t, D) = \sum_{G \in M} P(Z^t \mid \boldsymbol{x}^t, G, D) P(G \mid D) \ , \qquad (4.3)$$

where the sum is taken over *all* MB structures G in the model space M. The first term on the right hand side, $P(Z^t \mid \boldsymbol{x}^t, G, D)$, is the probability $P(Z^t \mid \boldsymbol{x}^t)$ computed with a MB that has structure G and parameters $\boldsymbol{\theta}_G$ that are given by Equation 4.4 below. This parameterization of G produces predictions equivalent to those obtained by integrating over all the possible parameterizations for G. The second term, $P(G \mid D)$, is the posterior probability of the MB structure G given the data D. In essence, Equation 4.3 states that a conditional probability of interest $P(Z^t \mid \boldsymbol{x}^t)$ is derived by taking a weighted average of that probability over all MB structures, where the weight associated with a MB structure is the probability of that MB structure given the data. In general, $P(Z^t \mid \boldsymbol{x}^t)$ will have different values for the different sets of MB structures over which the averaging is carried out.

4.3.1 Inference in Markov blankets

Computing $P(Z^t \mid \boldsymbol{x}^t, G, D)$ in Equation 4.3 involves doing inference in the MB with a specified global structure G:

$$P(Z^t \mid \boldsymbol{x}^t, G, D) = \int_{\boldsymbol{\theta}_G} P(Z^t \mid \boldsymbol{x}^t, \boldsymbol{\theta}_G, G) P(\boldsymbol{\theta}_G \mid G, D) d\boldsymbol{\theta}_G . \quad (4.4)$$

The parameters for the MB structure G are estimated using Bayesian parameter estimation that is described in Section 3.4.1.2. In summary, under the assumptions of (1) the variables are discrete and the dataset D consists of independent and identically distributed (i.i.d.) cases without missing values; (2) parameterization $\boldsymbol{\theta}_G$ over the entire MB of n nodes can be decomposed as $\boldsymbol{\theta}_G = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_i, ..., \boldsymbol{\theta}_{n_i}\}$ where the $\boldsymbol{\theta}_i$, the parameters associated with node X_i, are mutually independent (global parameter independence); (3) parameterization $\boldsymbol{\theta}_i$ of a node with q_i parent states is decomposed as $\boldsymbol{\theta}_i = \{\boldsymbol{\theta}_{i1}, \boldsymbol{\theta}_{i2}, ..., \boldsymbol{\theta}_{ij}, ..., \boldsymbol{\theta}_{iq_i}\}$ where the $\boldsymbol{\theta}_{ij}$, the parameters of a single conditional distribution of X_i, are mutually independent (local parameter independence); and (4) the parameter prior distribution is Dirichlet; the parameter estimates are given by the following expression:

$$P(X_i = k \mid Pa_i = j) \equiv \theta_{ijk} = \frac{\alpha_{ijk} + N_{ijk}}{\alpha_{ij} + N_{ij}}, \quad (4.5)$$

where (1) N_{ijk} is the number of cases in dataset D in which $X_i = k$ and the parents of X_i have the state denoted by j, (2) $N_{ij} = \sum_k N_{ijk}$, (3) α_{ijk} is a parameter prior that can be interpreted as belief equivalent to having previously (i.e., prior to D) seen α_{ijk} cases in which $X_i = k$ and the parents of X_i have the state denoted by j, and (4) $\alpha_{ij} = \sum_k \alpha_{ijk}$. For the patient-specific algorithms α_{ijk} is set to 1 for all i, j, and k, as a simple non-informative parameter prior [55].

For decision graph representations of CPDs, the interpretation of Equation 4.5 has to be modified slightly. Specifically, the parent state index j is replaced with a new index l that ranges over the leaf nodes in the decision graph for node X_i. Thus, Equation 4.5 now becomes:

$$P(X_i = k \mid Pa_i = l) \equiv \theta_{ilk} = \frac{\alpha_{ilk} + N_{ilk}}{\alpha_{il} + N_{il}}, \quad (4.6)$$

where (1) N_{ilk} is the number of cases in dataset D in which $X_i = k$ and the parents of X_i have a configuration corresponding to any of the paths that converge to the leaf node l in the decision graph. For example, in Figure 4-4, l ranges over the three leaves in the decision graph and N_{431} represents the number of cases in D where *fatigue* = F (i.e., $i = 4$ and $X_4 = 1$) with the parent configuration index $l = 3$. This index corresponds to a single parent configuration represented by the path leading to θ_4, namely, *lung cancer* (T) → *chronic bronchitis* (T) → θ_4 (the rightmost path in the figure). As another example, consider N_{421}, which represents the number of cases in D where *fatigue* = F (i.e., $i = 4$ and $X_4 = 1$) with the parent configuration index $l = 2$. This index corresponds to two parent configurations represented by the two paths converging on $\theta_2 = \theta_3$, namely, (1) *lung cancer* (F) → *chronic bronchitis* (T) → θ_2 and (2) *lung cancer* (T) → *chronic bronchitis* (F) → θ_3 (the two paths along the middle in the figure). The same interpretation is also applicable to a decision tree representation of CPDs (e.g., the decision trees shown in Figure 4-2 and Figure 4-5), since a decision tree is a type of a decision graph. Equation 4.6 is applicable to both the PSMBl algorithm (which uses decision graph CPDS) and the PSMBg algorithm (which uses complete decision tree CPDs).

4.3.2 Bayesian score of Markov blankets

The second term on the right hand side in Equation 4.3, $P(G \mid D)$, is the posterior probability of the MB structure G given the data and is computed by applying Bayes rule as follows:

$$P(G \mid D) = \frac{P(D \mid G)P(G)}{P(D)}, \qquad (4.7)$$

where $P(D)$ is a constant for all G and need not be computed. Thus, the Bayesian score for G, as previously shown in Equation 3.21, is:

$$score_B(G;D) = \log P(D \mid G) + \log P(G) . \quad (4.8)$$

The computation of the marginal likelihood, $P(D \mid G)$, for the decision graph representation is derived in Section 3.5.1. In summary, under the assumptions of (1) the variables are discrete and the data D is a multinomial random sample with no missing values; (2) global parameter independence, that is, the parameters associated with each variable are independent; (3) local parameter independence, that is, the parameters associated with each parent state of a variable are independent; and (4) the parameter prior distribution is Dirichlet; the marginal likelihood is:

$$P(D \mid G) = \prod_{i=1}^{n} \prod_{l=1}^{|L_i|} \frac{\Gamma(\alpha_{il})}{\Gamma(\alpha_{il} + N_{il})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ilk} + N_{ilk})}{\Gamma(\alpha_{ilk})}, \quad (4.9)$$

where $|L_i|$ is the cardinality of the set of leaves in the decision graph DG_i of X_i, N_{ilk} is the number of cases in the dataset D that have $X_i = k$ and have parent states of X_i that correspond to one of the paths in the decision graph leading to the leaf node l, and $N_{il} = \sum_k N_{ilk}$. Both the PSMBg and PSMBl algorithms use the logarithmic form of Equation 4.9 in computing the Bayesian score as follows:

$$\log P(D \mid G) = \sum_{i=1}^{n} \sum_{l=1}^{|L_i|} \frac{\Gamma(\alpha_{il})}{\Gamma(\alpha_{il} + N_{il})} \sum_{k=1}^{r_i} \frac{\Gamma(\alpha_{ilk} + N_{ilk})}{\Gamma(\alpha_{ilk})} . \quad (4.10)$$

The term P(G) in Equation 4.8 is the structure prior which represents the prior belief that the data was generated by some distribution that can be modeled with the MB structure G. For the PSMBg algorithm, a uniform prior belief over all G is assumed which reduces the term $P(G)$ to a constant. Thus, $P(G \mid D)$ is equal to $P(D \mid G)$ up to a proportionality constant and the Bayesian score for G is defined simply as the logarithmic marginal likelihood as follows:

$$score_B^{PSMBg}(G;D) = \log P(D \mid G) . \quad (4.11)$$

Equation 4.11 implies that the PSMBg algorithm evaluates MB structures only on the basis of the marginal likelihood and does not apply a penalty for structure complexity.

For the PSMBl algorithm, a two-level hierarchical structure prior is derived corresponding to the *global* and the *local* structures. In the PSMBg algorithm, specification of the global DAG structure uniquely specifies each MB node's complete decision tree. In contrast, in the PSMBl algorithm, a specified global structure corresponds to a family of local structures consisting of all possible decision graphs for each of the MB nodes, and the complete structure specification is given by $\{G, DG_1, \ldots, DG_i, \ldots, DG_n\}$ where G is the global DAG structure as before and DG_i is the local decision graph structure for node X_i. Including both the global and local structure priors, the Bayesian score in Equation 4.8 is rewritten as (see Section 3.5.1 for the derivation):

$$score_B(G, DG_1, ..., DG_n; D) = \log P(D \mid G, DG_1, ..., DG_n) + \log P(G) + \sum_{i=1}^{n} \log P(DG_i \mid G). \tag{4.12}$$

As in the case of the PSMBg algorithm, a uniform prior belief over all G is assumed, and therefore the term $P(G)$ reduces to a constant. Given G, a uniform prior belief over all possible decision graph structures for each MB node is assumed. The number of possible decision graph structures is the same as the number of ways in which the parent configurations can be partitioned into nonempty sets. The number of ways in which n elements can be partitioned into nonempty subsets is called a *Bell number* and is denoted by $B(n)$ [77]. The Bell number is typically computed as the following sum:

$$B(n) = \sum_{k=1}^{n} S(n,k), \tag{4.13}$$

where, $S(n, k)$ is a *Stirling number of the second kind* which is the number of ways in which n elements can be partitioned into k nonempty sets [78]. The Stirling number of the second kind is computed as:

$$S(n,k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^{k-i}C(k,i)i^n \ , \qquad (4.14)$$

where $C(k, i)$ is the binomial coefficient. The prior for the local structure associated with node X_i is then given by $1 / B(|\boldsymbol{Pa_i}|)$, where $|\boldsymbol{Pa_i}|$ is the number of parent states of X_i. The Bayesian score used by the PSMBl algorithm is now defined as:

$$score_B^{PSMBl}(G, DG_1,..., DG_n; D) = \log P(D \mid G, DG_1,..., DG_n) - \sum_{i=1}^{n}\log B(\left|\boldsymbol{Pa}_i\right|) . \qquad (4.15)$$

The derivation of the two terms needed for the computation of Equation 4.3 is now complete.

The structure prior just described biases the PSMBl algorithm to prefer simpler local structures over more complex ones. On average, the algorithm will prefer decision graphs with a smaller number of nodes to ones with a larger number of nodes. By incorporating a hierarchical structure prior, the Bayesian score given by Equation 4.15 penalizes complex structures. This is in contrast to the Bayesian score used by the PSMBg algorithm (given by Equation 4.11) which has no analogous structure penalty.

4.3.3 Selective Bayesian Model Averaging

Since Equation 4.3 sums over a very large number of MB structures, it is not feasible to solve it exactly. Hence, complete model averaging given by Equation 4.3 is approximated with selective model averaging, and heuristic search (described in the next section) is used to sample the model

space. For a set R of MB structures that have been chosen from the model space by heuristic search, selective model averaging estimates $P(Z^t \mid \boldsymbol{x}^t, D)$ as:

$$P(Z^t \mid \boldsymbol{x}^t, D) \cong \frac{\sum_{G \in R} P(Z^t \mid \boldsymbol{x}^t, G, D) P(G \mid D)}{\sum_{G \in R} P(G \mid D)}. \qquad (4.16)$$

The computations for the inference of the target variable Z^t and the posterior probability for the MB structure G are described in the preceding Sections 4.31 and 4.32 respectively.

4.4 PATIENT-SPECIFIC SEARCH

The patient-specific algorithms that I developed and applied use a two-phase search to sample the space of MB structures. A high level description of the two-phase search is now given. The first phase ignores the current patient case evidence $\boldsymbol{x}^t$ at hand, while searching for MB structures that best fit the training data. The second phase continues to add to the set of MB structures obtained from the first phase, but now searches for MB structures that have the greatest impact on the prediction of Z^t for the current patient case.

The first phase uses *greedy hill-climbing search* and accumulates the best model discovered at each iteration of the search in a set R. At each iteration of the search, successor models are generated from the current best model; the best of the successor models is added to R *only if* this model is better than current best model; and the remaining successor models are discarded. Since, no backtracking is possible, the first phase search terminates in a local maximum.

The second phase uses *best-first search* and adds the best model discovered at each iteration of the search to the set *R*. Unlike greedy hill-climbing search, best-first search holds models that have not been expanded (i.e., whose successors have not be generated) in a *priority queue Q*. At each iteration of the search, successor models are generated from the current best model and added to *Q*; the best model from *Q* is added to *R even if* this model is not better than the current best model. The second phase search terminates when a user set criterion is satisfied. Since, the number of successor models that are generated can be quite large, the priority queue *Q* is limited to a capacity of at most *w* models. Thus, if *Q* already contains *w* models, addition of a new model to it leads to removal of the worst model from it. The queue allows the algorithm to keep in memory up to *w* good scoring models, and facilitates limited backtracking to escape local maxima.

The next section describes in detail the search used by the PSMBg algorithm and the section after that describes in detail the search used by the PSMBl algorithm.

4.4.1 PSMBg search and operators

The PSMBg algorithm searches in the space of *global* MB structures in which the CPDs are represented as complete decision trees, and the operators used in traversing this space are the same as those used in standard BN structure learning with minor modifications. As mentioned in the previous chapter, the standard BN structure learning operators are (1) add an arc between two nodes if one does not exist, (2) delete an existing arc, and (3) reverse an existing arc, with the constraint that an operation is considered valid only if it generates a legal BN structure. This constraint simply implies that the graph of the generated BN structure be a DAG. A similar constraint is applicable to the generation of global MB structures, namely, that an operation is

X \ Y	T	P	C	S	O
T				✓	✓
P			✓		
C			✓*		
S	✓		✓		
O	✓		✓		

(a) Add arc X → Y

X \ Y	T	P	C	S	O
T			✓		
P	✓		✓		
C			✓		
S			✓		
O					

(b) Delete arc X → Y

X \ Y	T	P	C	S	O
T			✓		
P	✓				
C			✓*		
S					
O					

(c) Reverse arc X → Y

Figure 4-6: Constraints on the Markov blanket global operators. The nodes are categorized into five groups: T = target, P = parent, C = child, S = spouse, and O = other (not in the Markov blanket of T). The cells with check marks indicate valid operations and are the only ones that need to be considered in generating candidate structures. The cells with an asterisk indicate that the operation is valid only if the resulting graph is acyclic.

considered valid if it produces a legal MB structure of the target node. This constraint entails that some of the operations be deemed invalid, as illustrated in the following examples. With respect to a MB, the nodes can be categorized into five groups: (1) the target node, (2) parent nodes of the target, (3) child nodes of the target, (4) spousal nodes, which are parents of the children, and (5) other nodes, which are not part of the current MB. Incoming arcs into parents or spouses are not part of the MB structure and, hence operations that add such arcs are deemed invalid. Arcs between nodes not in the MB are not part of the MB structure and, hence operations that add such arcs are also deemed invalid. Figure 4-6 gives exhaustively the validity of the MB global operators. Furthermore, the application of the delete arc or the reverse arc operators may lead to additional removal of arcs to produce a valid MB structure (see Figure 4-7 for an example).

The search for MB structures proceeds in two sequential phases: phase 1 uses greedy hill-climbing search and phase 2 uses best-first search with a priority queue of capacity *w*. In phase 1 the candidate MB structures are scored with the phase 1 score which is the Bayesian score shown in Equation 4.11. This phase of the search begins with the empty graph and terminates in a local

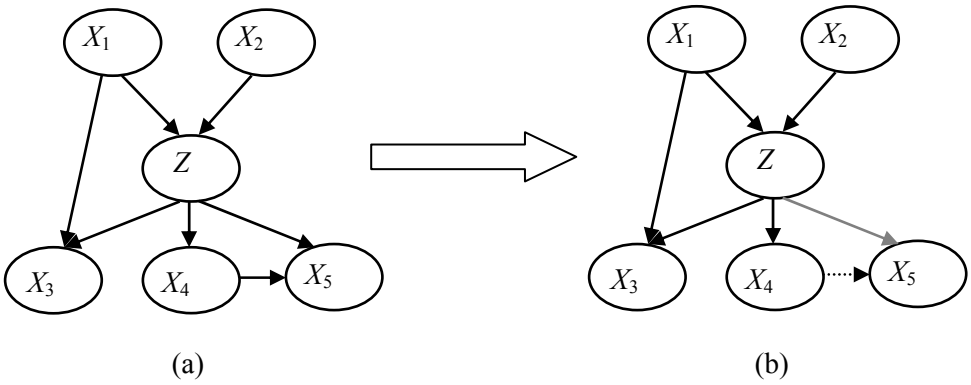


Figure 4-7: An example where the application of an operator leads to additional removal of arcs to produce a valid Markov blanket structure. Deletion of arc $Z \rightarrow X_5$ leads to removal of the arc $X_4 \rightarrow X_5$ since X_5 is no longer a part of the Markov blanket. Reversal of the same arc also leads to removal of the arc $X_4 \rightarrow X_5$ since X_5 is now a parent and is precluded from having incoming arcs. Also, unless $X_4 \rightarrow X_5$ is removed there will be a cycle.

maximum when none of the successor MB structures generated from the current best MB structure has a score higher than that of the current best MB structure. The highest scoring MB structure from each iteration of the search is accumulated in a set R. The purpose of this phase is to identify a set of MB structures that are highly probable, given data D.

Phase 2 searches for MB structures that change the current model-averaged estimate of $P(Z^t \mid \boldsymbol{x}^t, D)$ the most. The intuition here is to find viable competing MB structures for making this posterior probability prediction. When no competitive MB structures can be found, the prediction is assumed to be stable. Phase 2 differs from phase 1 in two aspects: it uses best-first search and it employs a different scoring function for evaluating candidate MB structures.

At the beginning of phase 2, R contains MB structures that were generated in phase 1. Successors to the MB structures in R are generated, scored with the phase 2 score (described in detail below) and added to the priority queue Q. At each iteration of the search, the highest scoring MB structure in Q is removed from Q and added to R; all global operations leading to legal MB structures are applied to it; the successor structures are scored with the phase 2 score;

and the scored structures are added to Q. Phase 2 search terminates when no MB structure in Q has a score higher than some small value ε or when a period of time t has elapsed, where ε and t are user specified.

The model score for phase 2 is computed as follows. Each successor MB structure G^* to be added to Q is scored based on how much it changes the current estimate of $P(Z^t \mid \boldsymbol{x}^t, D)$; this is obtained by model averaging over the MB structures in R. More change is better. The phase 2 score of a candidate MB structure G^* is computed as the distance between two distributions for Z^t as follows:

$$f(R, G^*) = \left| \frac{\sum_{G \in R} P(Z^t \mid \boldsymbol{x}^t, G, D) P(D \mid G)}{\sum_{G \in R} P(D \mid G)} - \frac{\sum_{G \in R \cup \{G^*\}} P(Z^t \mid \boldsymbol{x}^t, G, D) P(D \mid G)}{\sum_{G \in R \cup \{G^*\}} P(D \mid G)} \right|. \quad (4.17)$$

Specifically, the Kullback-Leibler divergence (KL divergence) is used as the distance metric for the experiments described in Chapter 5. The KL divergence, or relative entropy, is a quantity which measures the difference between two probability distributions [79]. That is, the phase 2 score for G^* is the KL divergence between the two estimates of $P(Z^t \mid \boldsymbol{x}^t, D)$, one estimate computed without and another computed with G^* in the set of models over which the model averaging is carried out. Thus, the phase 2 score for a candidate MB structure G^* is:

$$score(R, G^*) = KL(p \,\|\, q) \equiv \sum_x p(x) \log \frac{p(x)}{q(x)}, \quad (4.18)$$

where

$$p(x) = \frac{\sum_{G \in R} P(Z^t \mid \boldsymbol{x}^t, G, D) P(D \mid G)}{\sum_{G \in R} P(D \mid G)}, \text{ and}$$

$$q(x) = \frac{\sum_{G \in R \cup \{G^*\}} P(Z^t \mid \boldsymbol{x}^t, G, D) P(D \mid G)}{\sum_{G \in R \cup \{G^*\}} P(D \mid G)}.$$

The pseudocode for the two-phase search procedure used by the PSMBg algorithm is given in Figure 4-8.

4.4.2 PSMBl search and operators

The PSMBl algorithm searches in the space of *local* MB structures in which the CPDs are represented as decision graphs, and uses the two-tier search procedure described in Section 3.5.2. The PSMBl algorithm employs the same global operators for the outer search procedure as those used in the PSMBg algorithm and a set of local operators for the inner search procedure. The PSMBl algorithm may be considered an extension of the PSMBg algorithm, in that, it supplements the search procedure used in the PSMBg algorithm with an inner search procedure that is invoked at every iteration of the outer search procedure.

The local operators used by the PSMBl algorithm are those described in Section 3.5.2 that are used for traversing the space of local decision graph structures. Briefly, they are (1) the *complete split* operator that replaces a leaf node with an internal node and a set of leaf nodes corresponding to the states of the parent variable which is used for the split, (2) the *binary split* operator that is similar to the complete split operator, except that only two new leaf nodes are introduced, and (3) the *merge* operator that merges two leaf nodes into a single leaf node. An example of the application of these operators is illustrated in Figure 3-7.

The outer search procedure generates global MB structures; a global MB structure is specified by a DAG among the domain variables. In essence the DAG specifies the parent set of

nodes for each node. The nodes of the DAG are referred to as *MB nodes* to distinguish them from the nodes of the local decision graph structures which are referred to as *DG nodes*. DG nodes are either called internal nodes if the nodes have children or leaf nodes if the nodes are terminal.

The search for MB structures is very similar to that used by the PSMBg algorithm and proceeds in two sequential phases. Additionally, for every application of a global operator, a secondary search is performed to find an optimal local decision graph structure for the nodes affected by the application of the global operator. Given a MB node X_i and its parent MB nodes, the secondary search starts with a decision graph containing a single DG node that represents the parameters of the CPD of X_i assuming that X_i has no parents. Application of the local operators introduces the parents of X_i as internal DG nodes in the decision graph structure. For reasons of efficiency, the secondary search always uses the phase 1 score to score the decision graph structure irrespective of which phase in the primary search procedure invokes it. The pseudocode for the two-phase and two-tier search procedure used by the PSMBl algorithm is given Figure 4 9.

4.5 SPACE AND TIME COMPLEXITY

This section provides an analysis of the space and time complexity for the PSMBg and the PSMBl algorithms.

4.5.1 PSMBg algorithm

The PSMBg algorithm runs in $O((b + d)Nn)$ time and uses $O((w + d)Nn)$ space, where N is the number of cases in the training dataset D, n is the number of domain variables, d is the sum of the number of iterations of the search for phase 1 and phase 2, b – the branching factor – is the maximum number of successors generated from a MB structure, and w is the capacity of the priority queue Q.

Space complexity of PSMBg. The PSMBg algorithm searches in the space of MB structures using greedy hill-climbing search for phase 1 and best-first search with a priority queue of capacity w for phase 2. For d iterations of the search, the maximum number of MB structures that is stored is $O(w + d)$. The space required for each MB structure is linear in the number of its parameters. For a given MB node, the number of parameters (using a CPT or a complete decision tree) is exponential in the number of its parent nodes. However, the number of distinct parameters cannot be greater than the number of cases N in the training data D; the remaining parameters have a single default value. Thus, the space required for the parameters of a BN node is $O(N)$. In a domain with n variables, a MB structure can have up to n nodes and thus requires space of $O(Nn)$. In total, the space required by the PSMBg algorithm that runs for d iterations of the search is $O((w + d)Nn)$.

Time complexity of PSMBg. At each iteration of the search, a maximum of b successor MB structures are generated. For d iterations of the search, the number of MB structures generated and scored with the phase 1 score is $O(bd)$. Note that both phase 1 and phase 2 require successor MB structures to be scored with the phase 1 score.

Since phase 1 score decomposes over the MB nodes, to compute it for a newly generated MB structure only those MB nodes whose parent nodes have changed need be evaluated. The

number of MB nodes that need to be evaluated is either one (when the *add* or *remove* global operator is applied) or two (when the *reverse* global operator is applied). Computing the phase1 score for a MB node entails estimating the parameters for that node and calculating the marginal likelihood from those parameters. Estimating the parameters requires one pass over D and takes $O(Nn)$ time. Calculating the marginal likelihood requires retrieving every parameter of the CPDs associated with the MB node and takes $O(N)$ time. Thus, scoring a MB structure with phase 1 score takes $O((N + 1)n)$ time.

Phase 2 score computes the effect of a candidate MB structure on the model averaged estimate of the distribution of the target variable. This requires doing inference for the target node in a MB structure which takes $O(n)$ since at most n nodes influence the target distribution and hence at most n sets of parameters need be retrieved. Computing both phase 1 and phase 2 scores for a MB structure takes $O((N + 1)(n + 1))$ time which is approximately $O(nN)$ time. In the worst case, every candidate structure generated in all d iterations of the search will have to be evaluated with the phase 2 score. In total, the time required by the PSMBg algorithm that runs for d iterations of the search and generates b MB structures at each iteration is $O((b + d)Nn)$.

4.5.2 PSMBl algorithm

The PSMBl algorithm runs in $O((b + d)Nn2^n)$ time and uses $O((w + d)Nn)$ space. It has the same space complexity as the PSMBg algorithm but has exponential time complexity.

Space complexity of PSMBl. The PSMBl algorithm employs a two-phase search (namely, phase 1 and phase 2) and each phase uses a two-tier search procedure (namely, outer search procedure and inner search procedure). The outer search procedure is the same as the search procedure used by the PSMBg algorithm. At each iteration of the outer search procedure,

the inner search procedure is invoked on those MB nodes whose parent nodes have been modified. The inner search procedure when invoked on a MB node, locates a local structure represented by a decision graph using greedy hill-climbing search in the space of decision graphs. The maximum space required by a decision graph is the same as that required by a CPT or a complete decision tree for that MB node. Hence, the PSMBl algorithm has the same space complexity as the PSMBg algorithm, that is, of $O((w + d)Nn)$.

Time complexity of PSMBl. Estimating the parameters and computing both the phase 1 and phase 2 scores of a MB node represented by a CPT or a complete decision tree takes $O(Nn)$ time for the PSMBg algorithm as described above. The time requirement is the same for a MB node represented by a decision graph. For each structure generated by the outer search procedure, the inner search procedure can potentially generate all possible decision graphs for one or two MB nodes whose parent nodes have changed. The number of possible decision graphs for a MB node is given by Equation 4.2 and is exponential in n, that is, of $O(2^n)$. In total, the time required by the PSMBl algorithm that runs for d iterations of the outer search procedure and generates b MB structures at each iteration where each MB structure can have $O(2^n)$ local structures, is of $O((b + d)Nn2^n)$.

```
ProcedureGlobalSearchForPSMBg
        // phase 1 greedy hill-climbing search
1       R ← empty set
2       BestModel ← empty graph
3       Score BestModel with phase 1 score
4       BestScore ← phase 1 score of BestModel
5       Add BestModel to R

6       Do
7               For every possible global operator O that can be applied to BestModel
8                       Apply O to BestModel to derive Model
9                       Score Model with phase 1 score
10                      ModelScore ← phase 1 score of Model
11                      If ModelScore > BestScore
12                              BestModel ← Model
13                              BestScore ← ModelScore
14                              FoundBetterModel ← True
15                      End if
16              End for

17              If FoundBetterModel is True
18                      Add BestModel to R
19              Else
20                      Terminate do
21              End if
22      End do

        // phase 2 best-first search
23      Q ← empty priority queue with maximum capacity w
24      Generate all successors for the MB structures in R and add them to Q
25      Score all MB structures in Q with phase 2 score

26      Do while elapsed time < t
27              BestModel ← remove MB structure with highest phase 2 score from Q
28              BestScore ← phase 2 score of BestModel
29              For every possible global operator O that can be applied to BestModel
30                      Apply O to BestModel to derive Model
31                      Score Model with phase 2 score
32                      Add Model to Q
33              End for

34              If BestScore > ε
35                      Add BestModel to R
36              Else
37                      Terminate do
38              End if
39      End do

40      Return R
```

Figure 4-8: Pseudocode for the two-phase (phase 1 and phase 2) search procedure used by the PSMBg algorithm. Phase 1 uses greedy hill-climbing search while phase 2 uses best-first search.

```
ProcedureGlobalSearchForPSMBl
        // phase 1 greedy hill-climbing search
1       R ← empty set
2       BestMB ← empty graph
3       Score BestMB with phase 1 score
4       BestScore ← phase 1 score of BestMB
5       Add BestMB to R

6       Do
7               For every possible global operator O that can be applied to BestMB
8                       Apply O to BestMB to derive MB
9                       For every MBNode in MB whose parent nodes have been modified by O do
                                ProcedureLocalSearchForPSMBl(MBNode, MBNodeParents)
10                      Score MB with phase 1 score
11                      MBScore ← phase 1 score of MB
12                      If MBScore > BestScore
13                              BestMB ← MB
14                              BestScore ← MBScore
15                              FoundBetterMB ← True
16                      End if
17              End for

18              If FoundBetterMB is True
19                      Add BestMB to R
20              Else
21                      Terminate do
22              End if
23      End do

        // phase 2 best-first search
24      Q ← empty priority queue with maximum capacity w
25      Generate all successors for the MB structures in R and add them to Q
26      Score all MB structures in Q with phase 2 score

27      Do while elapsed time < t
28              BestMB ← remove MB structure with highest phase 2 score from Q
29              BestScore ← phase 2 score of BestMB
30              For every possible global operator O that can be applied to BestMB
31                      Apply O to BestMB to derive MB
32                      For every MBNode in MB whose parent nodes have been modified by O do
                                ProcedureLocalSearchForPSMBl(MBNode, MBNodeParents)
33                      Score MB with phase 2 score
34                      Add MB to Q
35              End for

36              If BestScore > ε
37                      Add BestMB to R
38              Else
39                      Terminate do
40              End if
41      End do

42      Return R                                                    (continues on next page)
```

Figure 4-9: Pseudocode for the two-phase (phase 1 and phase 2) and two-tier (outer and inner) search procedure used by the PSMBl algorithm. Figure continues on next page.

```
ProcedureLocalSearchForPSMBl(MBNode, MBNodeParents, MB)
        // inner loop greedy hill-climbing search
        Inputs:  MBNode: MB node for which a decision graph (DG) is to be located
                 MBNodeParents: set of parent MB nodes of MBNode
                 MB: current MB structure

43      BestDG ← decision graph for MBNode with a single leaf DGNode
44      Score BestDG with phase 1 score
45      BestScore ← phase 1 score of BestDG

46      Do
47              For every possible local operator O that can be applied to BestDG
48                      Apply O to BestDG to derive DG
49                      Score DG with phase 1 score
50                      DGScore ← phase 1 score of DG
51                      If DGScore > BestScore
52                              BestDG ← DG
53                              BestScore ← DGScore
54                              FoundBetterDG ← True
55                      End if
56              End for

57              If FoundBetterDG is not True
58                      Terminate do
59              End if
60      End do

61      For each MBNodePa in MBNodeParents
62              If MBNodePa is not represented as a DGNode in BestDG
63                      Remove MBNodePa from MBNodeParents
64                      Remove arc from MBNodePa to MBNode in MB
65              End if
66      End for

67      Return MB
```

Figure 4-9: Continued from previous page. Pseudocode for the two-phase (phase 1 and phase 2) and two-tier (outer and inner) search procedure used by the PSMBl algorithm. The procedure *ProcedureGlobalSearchForPSMBl* is similar to the one used by the PSMBg algorithm shown in Figure 4-8. The main difference is in the extra lines 9 and 32 which invoke an additional procedure *ProcedureLocalSearchPSMBl* for the inner search. As in the PSMBg algorithm, the PSMBl algorithm uses greedy hill-climbing in phase 1 and best-first search in phase 2. The inner search procedure uses greedy hill-climbing. *MBNode* represents a node in the Markov blanket (MB) while *DGNode* represents a node in a decision graph (DG).

5.0 EXPERIMENTAL EVALUATION

The hypothesis put forth in Section 1.2 is that for at least some performance measures patient-specific Bayesian network models predict better than population-wide models. This chapter evaluates the merits of the patient-specific algorithms on several datasets including a synthetic dataset, 21 datasets from the UCI Machine Learning repository (UCI datasets) and three medical datasets.

Section 5.1 describes the UCI datasets and the medical datasets in detail. Section 5.2 gives details of preprocessing that includes descriptions of the handling of missing values and the discretization of continuous variables. The algorithms are evaluated on five performance measures that are described in Section 5.3. Several versions of the patient-specific algorithms are used in the experiments and these are described in Section 5.4 along with the algorithms used for comparison.

Sections 5.5 through 5.7 provide the experimental results. Section 5.5 describes the experimental evaluation of the patient-specific algorithms on a small synthetic dataset. Section 5.6 gives results obtained from the PSMBg algorithm which searches in the model space of MB structures that capture only *global structure*, and Section 5.7 gives results obtained from the PSMBl algorithm which searches in the model space of MB structures that capture additional *local structure*. The final section provides a summary of the results.

5.1 DATASETS

The publicly available UCI Machine Learning Repository [80] has more than 100 datasets and machine learning algorithms are often trained and validated on a subset of these datasets. Twenty-one UCI datasets were selected; these datasets have between four and 60 predictor variables and a single target variable that has between two and seven classes. The size of the

Table 5-1: Description of the UCI datasets used in the experiments described in this chapter. In the column on predictors, the number of continuous (cnt) and discrete (dsc) predictors as well as the total number of predictor variables (excluding the target variable) are given. In the column on cases, the numbers of cases used in the experiments are given; this may be less than the total number of cases in the original UCI dataset since cases with missing values were removed. In the column on test method, "10-fold CV * 2" is short for 10-fold stratified cross-validation done twice.

Dataset	# Predictors (cnt + dsc = total)	# Classes	# Cases	Test Method
australian	6 + 8 = 14	2	690	10-fold CV * 2
breast-cancer	9 + 0 = 9	2	683	10-fold CV * 2
cleveland	6 + 9 = 13	2	296	10-fold CV * 2
corral	0 + 6 = 6	2	128	10-fold CV * 2
crx	6 + 9 = 15	2	653	10-fold CV * 2
diabetes	8 + 0 = 8	2	768	10-fold CV * 2
flare	0 + 10 = 10	2	1066	10-fold CV * 2
german	7 + 13 = 20	2	1000	10-fold CV * 2
glass2	9 + 0 = 9	2	163	10-fold CV * 2
glass	9 + 0 = 9	7	214	10-fold CV * 2
heart	13 + 0 = 13	2	270	10-fold CV * 2
hepatitis	6 + 13 = 19	2	80	10-fold CV * 2
iris	4 + 0 = 4	3	150	10-fold CV * 2
lymphography	0 + 18 = 18	4	148	10-fold CV * 2
pima	8 + 0 = 8	2	768	10-fold CV * 2
postoperative	1 + 7 = 8	3	87	10-fold CV * 2
sonar	60 + 0 = 60	2	208	10-fold CV * 2
vehicle	18 + 0 = 18	4	846	10-fold CV * 2
vote	0 + 16 = 16	2	435	10-fold CV * 2
wine	13 + 0 = 13	3	178	10-fold CV * 2
zoo	0 + 16 = 16	7	101	10-fold CV * 2

datasets, the number and type of predictor variables, and the number of classes (states) taken by the target variable are given in Table 5-1. In addition to the UCI datasets, three medical datasets with five target variables were used in the experiments (see Table 5-2). Each target variable in a medical dataset represents a clinically relevant patient outcome like mortality or the occurrence of a clinical condition. In the following sections, the medical datasets are described in detail.

5.1.1 Pneumonia

Community acquired pneumonia (CAP) is an important clinical condition, both from a resource-utilization and a patient-outcome point of view. Pneumonia affects over three million people annually in the U.S. [81] and is the sixth leading cause of death [82]. Accurate evaluation of CAP patients in the Emergency Department followed by appropriate treatment (including the decision whether to admit to the hospital or not) is an important clinical problem.

Pneumonia Dataset. The pneumonia data comes from the Pneumonia Patient Outcomes Research Team (PORT) project and was collected during October 1991 to March 1994 from five

Table 5-2: Description of the medical datasets used in the experiments described in this chapter. In the column on predictors, the number of continuous (cnt) and discrete (dsc) predictors as well as the total number of predictor variables (excluding the target variable) are given. All outcome variables that were studied are binary. The last column gives the number of cases in the training set and the test set respectively.

Dataset	# Predictors (cnt + dsc = total)	Outcome variable	# Classes	# Cases	# Train + # Test
pneumonia	120 + 36 = 156	dire outcome	2	2287	1601 + 686
sepsis-d	7 + 14 = 21	death	2	1673	1115 + 558
sepsis-s	7 + 14 = 21	severe sepsis	2	1673	1115 + 558
heart failure-d	11 + 10 = 21	death	2	11178	7453 + 3725
heart failure-c	11 + 10 = 21	complication incld. death	2	11178	7453 + 3725

hospitals in three geographical regions: Pittsburgh, Boston, and Halifax, Nova Scotia. The dataset has 2287 CAP hospitalized and ambulatory care patients many of whom where seen in the Emergency Department. One purpose of the project was to develop accurate criteria for prognosis of patients with CAP that could help physicians assess their risks and provide guidance on which patients should be hospitalized. Fine et al. [83] developed a scoring system called the Pneumonia Severity Index (PSI) for stratifying patients with CAP with respect to 30-day mortality. The PSI was derived from 20 demographic and clinical variables that were selected using regression analysis. This dataset has been extensively analyzed with machine learning algorithms for predicting various outcomes [84].

Pneumonia Predictor Variables. The PORT project collected data on more than 150 patient variables including demographic data and findings on admission like co-existing diseases, symptoms, signs, initial laboratory tests, and initial medications.

For the experiments, 156 patient variables were selected as predictors; these variables are typically available in the Emergency Department at the time the decision whether to admit or not is made. The variables include demographic information, history and physical examination information, laboratory results, and chest X-ray findings. Of the 156 variables, 120 are discrete and the remaining 36 are continuous. A majority of the discrete variables are binary and the rest have between three to seven values. The 36 continuous variables are derived mainly from laboratory tests and were discretized based on thresholds provided by clinical experts on the PORT project.

Pneumonia Outcome Variables. Several patient outcomes on both inpatients and outpatients were measured at 30 days in the PORT project. The binary outcome variable selected for prediction is called *dire outcome*. A patient was considered to have experienced a dire

outcome if any of the following occurred: (1) death within 30 days of presentation, (2) an initial intensive care unit admission for respiratory failure, respiratory or cardiac arrest, or shock, or (3) the presence of one or more specific, severe complications. Based on these criteria, 261 patients (11.4%) had a dire outcome.

5.1.2 Sepsis

Sepsis is a syndrome of systemic inflammation in response to infection that leads to complex physiologic and metabolic changes and can result in multi-system organ dysfunction [85]. Sepsis occurs in more than 450,000 individuals annually in the U.S. and is associated with a 30% mortality rate [86]. The incidence of sepsis is rising in the U.S. and hospital care of sepsis is a significant cost to the healthcare system [87]. Thus, considerable research is underway towards a fuller understanding of the complex pathophysiology of human sepsis, including the identification of markers predictive of response to specific therapies and subsequent outcomes [88].

Sepsis Dataset. The sepsis data comes from the GenIMS (Genetic and Inflammatory Markers of Sepsis) project that has collected data on 2320 patients with community acquired pneumonia who presented to the emergency departments of 30 hospitals in southwestern Pennsylvania, Connecticut, Michigan and Tennessee. These patients were followed during their hospitalization for the development of sepsis. The aims of the GenIMS project include investigating the relationships among genetic polymorphisms, inflammatory mediator response, and clinical course and outcome.

Sepsis Predictor Variables. The GenIMS project collected data on 108 clinical variables at the time of patient enrollment in the study and 99 clinical variables on a daily basis during the

period of hospitalization. In addition, the project collected data on five inflammatory mediators and on 29 genes that are believed to be involved in sepsis. The genetic data include information on single nucleotide polymorphisms (SNPs), short tandem repeat (STR) polymorphisms, and variable-number-tandem-repeat (VNTR) polymorphisms within the regulatory and coding regions of these genes.

The experimental datasets consist of 21 variables as predictors that include three demographic variables, six clinical variables, one inflammatory marker and 10 genetic variables. These variables were selected by the GenIMS project investigators for analysis of a subset of the dataset to investigate the role of the macrophage migration inhibitory factor (MIF) gene in the susceptibility, severity, and outcome of community-acquired pneumonia.

Sepsis Outcome Variables. Several outcomes including include death, severe sepsis, interventions such as mechanical ventilation, and hospital length of stay were measured in the project. Two binary outcome variables were selected for prediction: (1) death within 90 days of inclusion in the study, and (2) the development of severe sepsis during the study.

5.1.3 Heart Failure

Heart failure is an acute and chronic condition that affects 5 million people in the U.S. [89] leading to one million hospital admissions each year with a primary discharge diagnosis of heart failure and another two million with a secondary discharge diagnosis of this condition [90]. Hospital care for heart failure accounts for a significant portion of annual healthcare expenditure in the U.S. Accurate evaluation of heart failure patients in the Emergency Department followed by appropriate treatment (including the decision whether to admit to the hospital or not) is an important clinical problem. However, existing heart failure predictive models and guidelines

have limited utility in this setting because they are based on narrowly defined patient subgroups rather than the broad spectrum of heart failure patients treated in the Emergency Department, or they rely on clinical data unavailable in this setting [91].

Heart Failure Dataset. The Heart Failure dataset includes 33,533 heart failure patients who were hospitalized from the Emergency Departments of 180 general acute care hospitals in Pennsylvania for the year 1999. Overall, 1498 (4.5%) patients died during hospitalization. Among survivors at hospital discharge, 2269 (6.8%) experienced a serious medical complication. This dataset has been analyzed by the original investigators to construct a prediction rule to identify patients who are at low risk of death and serious complications [92].

Heart Failure Predictor Variables. The Heart Failure dataset contains data on numerous variables that were collected the day of admission or the day before admission if the patient was already in the Emergency Department at that time. Such information includes demographic data, historical and physical examination findings, and electrocardiographic, routine laboratory tests and radiographic findings at the time of admission.

The experimental datasets consist of 21 variables as predictors that include demographic, clinical, laboratory, electrocardiographic and radiographic findings. These variables had been identified as good predictors during the construction of a prediction rule by the original investigators [92].

Heart Failure Outcome Variables. Outcome variables that were recorded in the study included death from any cause and several serious medical complications that occurred during the hospitalization. A patient was counted as having a serious medical complication if he or she experienced a life-threatening clinical condition or received a life-saving inpatient treatment. Life-threatening clinical conditions included were acute myocardial infarction, ventricular

fibrillation, cardiogenic shock, and cardiac arrest. Life-saving inpatient treatments were: (1) resuscitation defined as intubation or mechanical ventilation not initiated during surgery, cardiac compression, resuscitation, defibrillation, and (2) reperfusion therapy defined as coronary artery bypass graft surgery, percutaneous transluminal coronary angioplasty, or intravenous thrombolytics.

Two binary outcome variables were selected for prediction: (1) the occurrence of death from any cause during the hospitalization, and (2) the development of a serious medical complication (including death) during the hospitalization.

5.2 PREPROCESSING

This section describes the several preprocessing steps that were carried out on the datasets. Since, the patient-specific algorithms do not currently handle continuous variables or missing values, the continuous variables were discretized and missing values were either imputed or eliminated.

Training and test sets. The UCI datasets were evaluated with two stratified 10-fold cross-validation. Hence, each UCI dataset was split twice into 10 stratified training and test folds to create a total of 20 training and test folds. All experiments were carried out on the same set of 20 training and test folds. The medical datasets were each evaluated with a single training and test set. For each medical dataset associated with a specific outcome, the training set was created by randomly sampling from the entire dataset such that both the training and the test datasets had approximately the same proportion of cases with the positive outcome. The training set included approximately 70% of the dataset and the test set the remaining 30%. The numbers of cases in

```
Impute(dataset, Dist(i, j))
Inputs: dataset with N cases and F features
        Dist(i, j): distance metric defined on cases i and j that have no missing values

1       Repeat until convergence (i.e., no change in the estimates of the unknown values)
        or until some fixed number of iterations has been reached:
2       For c ∈ {1,..., N}
3               For f ∈ {1,..., F}
4                       If value[c, f] = unknown
5                               Re-impute value[c, f] using 1-Nearest Neighbor:
6                               Produce a new estimate of value[c, f] by setting it to
                                value[n, f] where n is the nearest neighbor case to c. The
                                nearest neighbor case n is the case where value[n, f] ≠
                                missing and Dist(c, n) is the least.
7                       End if
8               End for
9       End for


Dist(i, j)
Input: cases i and j that have no missing values and have F features

10      distance ← 0
11      For f ∈ {1,..., F}
12              If f is a continuous variable
13                      distance ← square(value[i, f] – value[j, f]) + distance
14              End if
15              If f is a nominal variable
16                      distance ← 1 + distance if value[i, f] = value[j, f]
17                      distance ← 0 + distance if value[i, f] ≠ value[j, f]
18              End if
19      End for
20      Return distance
```

Figure 5-1: Pseudocode for non-parametric imputation of missing values using 1-Nearest Neighbor (modified from [93]). In the pseudocode, values that are missing in the original dataset are called "*unknown*" (as opposed to "*known*" values that are never missing) and values that have not yet been filled-in by the algorithm are called "*missing*".

the training and test sets for the medical datasets are given in Table 5-2. The original Heart Failure dataset contains 33,533 cases. However, for the experiments described in this chapter only one-third of the cases that were randomly selected from the original dataset were used. This was done to reduce the running times of the patient-specific algorithms to several days from the several weeks that would be needed if the complete dataset was used.

Missing values. For the UCI datasets, any case that had one or more missing values was removed from the dataset, as is done in [68]. Sixteen of the 21 UCI datasets have no missing values and no cases were removed. In the remaining five datasets, removal of missing values resulted in a decrease in the size of the dataset of less than 10%.

The medical datasets have a large proportion of missing values and eliminating cases with missing values would have led to substantial reduction in the size of the datasets. Instead, the missing values were imputed using an iterative non-parametric imputation algorithm described by Caruana [93]. The pseudocode for this algorithm is given in Figure 5-1. This method had previously been applied to fill in missing predictor values for the pneumonia dataset with good results [93].

Discretization. All target variables in all the datasets are discrete. However, some of the predictor variables are continuous (numerical) as indicated in Tables 5-1 and 5-2. All continuous variables were discretized using the method described by Fayyad and Irani [94]. This is an entropy based method that analyzes the values of a continuous variable and creates thresholds such that the resulting intervals have high information gain. The discretization thresholds were determined only from the training sets and then applied to both the training and test sets.

5.3 PERFORMANCE MEASURES

Many methods and measures are available to measure the performance of classifiers and predictive algorithms [95, 96]. The performance of the algorithms was evaluated on two measures of discrimination and three probability measures. The discrimination measures used are the misclassification error (ERR) and the area under the ROC curve (AUC). The discrimination

Table 5-3: Brief description of the performance measures used in evaluation of the performance of the algorithms. For the AUC, scores closer to 1 indicate better performance. For the remaining measures, scores closer to 0 indicate better performance.

Performance measure	Abbreviation	Range	Best score
misclassification error	ERR	[0, 1]	0
area under the ROC curve	AUC	[0, 1]	1
mean squared error / Brier score	MSE	[0, 2]	0
mean logarithmic loss / mean cross entropy	MXE	[0, ∞)	0
calibration score	CAL	[0, 1]	0

measures evaluate how well the algorithm differentiates among the various classes (or values of the target variable). The probability measures considered are the logarithmic loss or cross entropy (MXE), Brier score or squared error (MSE), and calibration (CAL). The probability measures are uniquely minimized (in expectation) when the predicted value for the target of each case coincides with the true probability of that case taking that target value. A brief description of the measures is given in Table 5-3.

5.3.1 Misclassification error (ERR)

Misclassification error (or its complement classification accuracy) is probably the most widely used performance measure for evaluating classifiers and prediction algorithms. It is defined as the proportion of incorrect class predictions the algorithm makes relative to the size of dataset. If an algorithm produces a continuous output, as in the case of a probabilistic classifier, then the class with the highest value of the output is declared to be the predicted class. Misclassification error is sometimes a poor criterion for assessing performance since it makes the implicit assumption that costs of the different kinds of misclassification are equal; an assumption that often does not hold in practice. For example, falsely diagnosing a deadly disease in a healthy

person is typically considered a less costly error than not diagnosing the disease when it is actually present.

Misclassification error is computed as the number of misclassifications divided by the total number of cases in the dataset. It varies from 0 to 1 with 0 representing perfect classification.

5.3.2 Area under the ROC curve (AUC)

The Receiver Operating Characteristic (ROC) curve is defined only for a binary target and is a plot of the sensitivity versus (1 – specificity) for all possible thresholds. Given two classes, namely, class 0 and class 1 that the target variable can take, sensitivity is defined as the probability of predicting correctly a case that belongs to class 1 and specificity is defined as the probability of predicting correctly a case that belongs to class 0. The area under the ROC curve (AUC) is typically used as a summary statistic of discrimination. The AUC is equivalent to the probability that a randomly chosen case from class 0 will have a smaller predicted probability of belonging to class 1 than a randomly chosen case from class 1. Based on this interpretation, the binary class AUC can be estimated as follows:

$$AUC_{binary} \equiv \hat{A}(0 \mid 1) = \frac{S_0 - n_0(n_0 + 1)/2}{n_0 n_1}, \qquad (5.1)$$

where S_0 is the sum of the ranks of the cases that belong to class 0, after the cases have been ranked in ascending order of the predicted probability of belonging to class 0; n_0 is the number of cases that belong to class 0 and n_1 is the number of cases that belong to class 1 [97, 98]. The AUC is generally considered superior to misclassification error since it is independent of costs, priors and any classification threshold.

Several extensions of the AUC from binary to multiple classes have been described, which include the volume under the ROC surface [99] and the mean of the AUCs obtained by aggregation over all pairs of classes [98, 100]. For multiple classes, say *c* classes, the method described by Hand and Till [98] is used for computing the AUC, as follows:

$$AUC_{multiclass} = \frac{2}{c(c-1)} \sum_{i<j} \hat{A}(i,j), \tag{5.2}$$

where

$$\hat{A}(i,j) = \frac{\hat{A}(i \mid j) + \hat{A}(j \mid i)}{2}, \tag{5.3}$$

such that $\hat{A}(i \mid j)$ represents the binary class AUC when only classes *i* and *j* are considered and is computed using Equation 5.1.

The AUC varies from 0 to 1 with 1 representing perfect discrimination.

5.3.3 Brier score or mean squared error (MSE)

Let a binary target variable Y^i take on values in {0, 1}, and let y^i denote an indicator variable such that $y^i = 1$ if $Y^i = 1$ and $y^i = 0$ if $Y^i = 0$. Let p^i denote the predicted probability that Y^i for case *i* takes the value 1. The mean squared error or the Brier score for a dataset of *n* cases is defined as [101]:

$$MSE_{binary} = \frac{1}{n} \sum_{i=1}^{n} (y^i - p^i)^2 . \tag{5.4}$$

If the predicted probabilities p^i are constrained to be equal to 0 or 1, the Brier score is equal to the misclassification error. The Brier score is a probability measure and depends on the predicted

values and not merely on the ranking of the cases based on the values (as in the AUC) or how the values fall relative to a threshold (as in the misclassification error).

The MSE or the generalized Brier score for the multiclass case is a natural extension of the binary case [102]. Let a target variable Y^i take on values in {0, 1, …, K-1}, and let y^{ik} denote an indicator variable such that $y^{ik} = 1$ if $Y^i = k$ and $Y^i = 0$ otherwise, where k = {0, 1, …, K-1}. Let p^{ik} denote the predicted probability of class k for case i. The generalized Brier score is defined as:

$$MSE_{multiclass} = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=0}^{K-1}(y^{ik} - p^{ik})^2 \,. \tag{5.5}$$

The Brier score ranges from 0 to 2 with a score of 0 indicating perfect predictive performance.

5.3.4 Mean logarithmic loss or mean cross-entropy (MXE)

Mean logarithmic loss or mean cross entropy is another probability measure. Let a binary target variable Y^i take on values in {0, 1}, and let y^i denote an indicator variable such that $y^i = 1$ if $Y^i = 1$ and $y^i = 0$ if $Y^i = 0$. Let p^i denote the predicted probability that Y^i for case i takes the value 1. The mean logarithmic loss for a dataset of n cases is defined as [103]:

$$MXE_{binary} = \frac{1}{n}\sum_{i=1}^{n} - y^i \log(p^i) - (1 - y^i)\log(1 - p^i) \,. \tag{5.6}$$

For the multiclass case, the mean logarithmic loss is defined in a similar fashion to the generalized Brier score [104]:

$$MXE_{multiclass} = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=0}^{K-1} - y^{ik} \log(p^{ik}) \,, \tag{5.7}$$

where y^{ik} and p^{ik} are defined as in the preceding section.

The logarithmic loss ranges from 0 to ∞ with a score of 0 indicating perfect predictive performance, meaning that a probability of 1 was assigned to the correct state of Y^i in very case.

5.3.5 Calibration score (CAL)

The calibration score, CAL, was developed by Caraua [96] and is based on reliability diagrams [105]. Let a binary target variable Y^i take on values in {0, 1}, and let y^i denote an indicator variable such that $y^i = 1$ if $Y^i = 1$ and $y^i = 0$ if $Y^i = 0$. Let p^i denote the predicted probability that Y^i for case i takes the value 1. The calibration score is calculated as follows. Order all cases by p^i, and put cases 1 to c in the same bin, where c is a suitable number that is smaller than the total number of test cases. Calculate the proportion of these cases where $Y^i = 1$; this proportion approximates the true probability that in these cases Y^i takes the value 1. Then calculate the mean prediction for these cases. The absolute value of the difference between the observed proportion and the mean prediction is the calibration error for this bin. Similarly, compute the calibration errors for the bins containing cases 2 to (c + 1), 3 to (c + 1), and so on. Then, CAL is the grand mean of the calibration errors of all the bins. For the experiments reported in this chapter, c was set to 50. This value of c provided an adequate number of bins since the number of test cases was not less than 50 for any dataset.

For the multiclass case, as before, let p^{ik} denote the predicted probability of class k for case i. The CAL score is computed separately for each class k over all the cases. The final CAL score is the mean of the k CAL scores.

The CAL score ranges from 0 to 1 with a score of 0 indicating perfect calibration.

5.4 MACHINE LEARNING ALGORITHMS

Several versions of the patient-specific algorithm were evaluated on the UCI and the medical datasets and their performance compared with six machine leaning methods that are commonly used for developing clinical prediction models.

5.4.1 Patient-specific algorithms

Six versions of the patient-specific algorithms are used in the experiments described later in this chapter. They are listed in Table 5-4. The PSMBg, PSMBg-MS and the NPSMBg algorithms all use MB structures with the complete decision tree representation for CPDs that captures only the global MB structure (hence the suffix "g" in the acronym). The PSMBg algorithm is described in detail Section 4.4.1 and it performs selective model averaging to estimate the distribution of the target variable of the case at hand (i.e., the test case). The MB structures used for the model averaging are selected through a two-phase search, where phase 1 of the search is non patient-specific while phase 2 of the search is patient-specific. Phase 1 uses greedy hill-climbing search that terminates at a local maximum and phase 2 uses best-first search and terminates when no candidate MB structure has a score higher than a small value ε or when a period of time t has elapsed, where ε and t are user specified. The PSMBg-MS algorithm is a *model selection* version of the PSMBg algorithm. It chooses the MB structure that has the highest posterior probability from those selected by the PSMBg algorithm in the two-phase search, and uses that single model to estimate the distribution of the target variable of the case at hand. Comparing the PSMBg algorithm to the PSMBg-MS algorithm measures the effect of approximating the selective model averaging by model selection. When the training dataset is large the performance of the PSMBg

algorithm and the PSMBg-MS algorithm may be similar if a single model with a relatively large posterior probability overwhelms the contributions of the remaining models during model averaging.

The NPSMBg algorithm is the *non-patient-specific* (i.e., population-wide) version of the PSMBg algorithm. Phase 1 of the NPSMBg algorithm is identical to that of the PSMBg algorithm. In phase 2, the NPSMBg algorithm accumulates the same number of models as the PSMBg algorithm except that the MB structures are identified on the basis of the non-patient-specific phase 1 score. Thus, the NPSMBg algorithm averages over the same number of models as the PSMBg algorithm. Comparing the PSMBg algorithm to the NPSMBg algorithm measures the additional effect of the patient-specific heuristic on the performance of model averaging

Table 5-4: Six versions of the patient-specific algorithm with a brief description of each one.

Acronym	Algorithm	Phase1	Phase 2	Prediction
PSMBg	Patient-specific Markov blanket (global)	Is non-patient-specific Uses greedy hill-climbing Uses phase 1 score	Is patient-specific Uses best-first Uses phase 2 score	By model averaging over models selected in phase 1 and phase 2
PSMBg-MS	Patient-specific Markov blanket (global) – Model Selection	Same as PSMBg	Same as PSMBg	Based on the highest scoring model from models selected by PSMBg
NPSMBg	Non-patient-specific Markov blanket (global)	Is non-patient-specific Uses greedy hill-climbing Uses phase 1 score	Is non-patient-specific Uses best-first Uses phase 1 score	By model averaging; number of selected models is the same as in PSMBg
PSMBl	Patient-specific Markov blanket (local)	Is non-patient-specific Uses greedy hill-climbing Uses phase 1 score	Is patient-specific Uses best-first Uses phase 2 score	By model averaging over models selected in phase 1 and phase 2
PSMBl-MS	Patient-specific Markov blanket (local) – Model Selection	Same as PSMBl	Same as PSMBl	Based on the highest scoring model from models selected by PSMBl
NPSMBl	Non-patient-specific Markov blanket (local)	Is non-patient-specific Uses greedy hill-climbing Uses phase 1 score	Is non-patient-specific Uses best-first Uses phase 1 score	By model averaging; number of selected models is the same as in PSMBl

realized by a non-patient-specific (i.e., population-wide) method.

The PSMBl, PSMBl-MS and the NPSMBl algorithms are very similar to the PSMBg, PSMBg-MS and the NPSMBg algorithms respectively, and differ from them in two respects. First, they use MB structures with the decision graph representation for CPDs that captures both global and local MB structure (hence the suffix "l" in the acronym). Second, the two-phase search procedure of the global algorithms is supplemented with an inner search procedure. Specifically, for each phase of the search, the local algorithms use a nested search procedure: an outer search procedure (which is the same as in the PSMBg algorithm) that applies the global operators to generate global MB structures and in inner search procedure (which is new to the PSMBl algorithm) that applies local operators to generate local structures. Note that the inner and outer search procedures are distinct from the two phases of the search. In summary, the PSMBl algorithm employs a two-phase two-tier search as follows. The outer search procedure of phase 1 uses greedy hill-climbing and the outer search procedure for phase 2 uses best-first search just as in the PSMBg algorithm. The inner search procedures for both phase 1 and phase 2 use greedy hill-climbing. The PSMBl algorithm is described in detail Section 4.4.2.

5.4.2 Comparison algorithms

The performance of the patient-specific algorithms is compared to the following methods: (1) naïve Bayes (NB), (2) C4.5 decision tree (DT), (3) logistic regression (LR), (4) neural networks (NN), (5) *k*-Nearest Neighbor (*k*NN), and (6) Lazy Bayesian Rules (LBR). The first five methods are among the commonest multivariate techniques currently applied in developing clinical prediction models [17]. The first four are representative population-wide methods, and the next two are examples of patient-specific methods. *k*NN is a similarity-based patient-specific method.

The LBR algorithm induces a rule tailored to the features of the test case that is then used to classify it and is described in detail in Section 2.8. It is an example of a model-based patient-specific method that performs model selection. For all the six methods listed above, the implementations in the Weka software package (version 3.4.3) were used. The Weka software package is available freely from the Weka machine learning project at the University of Waikato, New Zealand [103].

5.5 EVALUATION ON SYNTHETIC DATA

This section describes the evaluation of the PSMBg and PSMBl algorithms on a small synthetic dataset. The synthetic domain consists of five binary variables A, B, C, D, Z where Z is a deterministic function of the other variables:

$$Z = A \vee (B \wedge C \wedge D).$$

This function implies the value-specific independence relation $(Z \perp B, C, D \mid A = T)$ that can be represented explicitly by a MB with local structure (i.e., with a decision graph) but not by a MB with global structure (i.e., with a CPT or a complete decision tree). The training and the test sets used in the experiments are shown in Figure 5-2. The training set simulates a low occurrence of A = T (only five out of 69 cases have A = T), and the test set consists of three cases of A = T which are not present in the training set.

The following algorithms were used in the experiments: (1) a complete model averaged version of the PSMBg algorithm where model averaging is carried over all 3567 possible MB structures, (2) the PSMBg algorithm, (3) the NPSMBg algorithm, (4) the PSMBl algorithm, and (5) the NPSMBl algorithm.

The settings used for the PSMBg and PSMBl algorithms are as follows:

- Phase 1: The model score for phase 1 is the Bayesian score computed using Equation 4.11 for the PSMBg algorithm and Equation 4.15 for the PSMBl algorithm, with a Dirichlet parameter prior with hyperparameters $\alpha_{ijk} = 1$ for all *i*, *j*, *k*. Phase 1 uses greedy hill-climbing search that terminates at a local maximum.
- Phase 2: The model score for phase 2 is computed using Equation 4.18 that is based on KL-divergence. Phase 2 uses best-first search with a priority queue Q whose maximum capacity w is set to 1000. Phase 2 search terminates when no MB structure in Q has a phase 2 score higher than ε = 0.001 for 10 consecutive iterations of the search. The maximum period of running time t for phase 2 was not specified since the algorithm terminated in a reasonable period of time with the specified value for ε.

Training set		**Test set**
A, B, C, D, Z		***A, B, C, D, Z***
T, F, F, F, T		T, F, F, T, T
T, F, T, F, T		T, T, F, F, T
T, T, F, T, T		T, F, T, T, T
T, T, T, F, T		
T, T, T, T, T		
F, F, F, F, F	**Repeated 8 times**	
F, F, F, T, F		
F, F, T, F, F		
F, F, T, T, F		
F, T, F, F, F		
F, T, F, T, F		
F, T, T, F, F		
F, T, T, T, T		

Figure 5-2: Training and test datasets derived from the deterministic function $Z = A \vee (B \wedge C \wedge D)$. The training set contains a total of 69 cases and the test set a total of three cases as shown; the test cases are not present in the training set. The training set simulates low prevalence of A = T since only five of the 69 cases have this variable value combination.

- The predicted distribution for the target variable of the test case is computed using Equation 4.16; for each MB structure the parameters are estimated using Equation 4.6.

5.5.1 Results

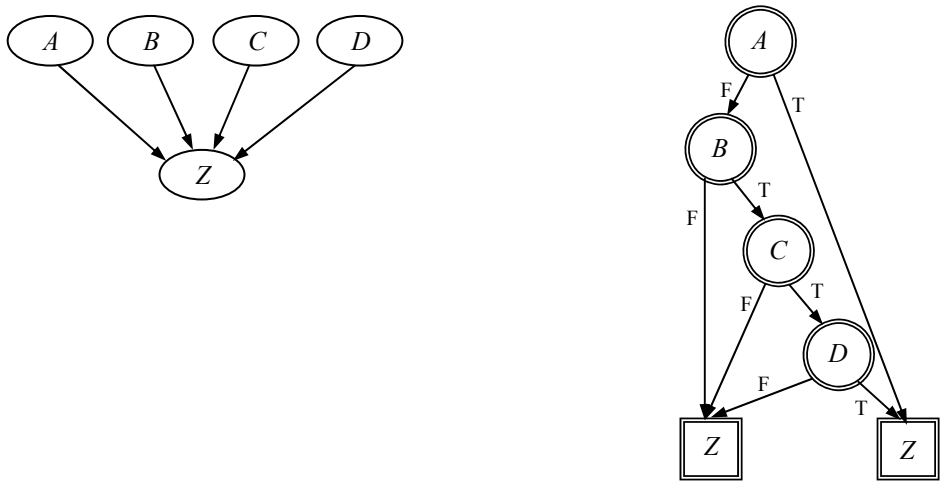
The results are given in Table 5-5. All performance measures except the AUC were computed for the test set of three cases. The AUC could not be computed since all the cases in the test set are from the same class, Z = T. The results from complete model averaging represent the best achievable performance given the training set and the class of MB models with global structure. The PSMBg and the NPSMBg algorithms that average over a subset of all models had poorer performance than complete model averaging. However, the PSMBg algorithm improved over the performance of the NPSMBg algorithm. Though both methods average over the same number of models, the PSMBg algorithm uses the patient-specific phase 2 score to choose phase 2 models

Table 5-5: Results obtained from the training and test sets that are shown in Figure 5-2. The AUC could not be computed since the test set has cases from a single class. Results in the first column are obtained by model averaging over all 3567 MBs with global structure. Similar complete model averaging over all MBs with local structure is not given since it was not tractable.

Performance measure	PSMBg complete model averaged	PSMBg	NPSMBg	PSMBl	NPSMBl
Misclassification error	0.0000	0.0000	0.3333	0.0000	0.0000
AUC	-	-	-	-	-
Logarithmic loss	0.0684	0.0783	0.0862	0.0184	0.0183
Squared error	0.0406	0.0505	0.0585	0.0042	0.0042
CAL score	0.3720	0.4092	0.4284	0.1106	0.1103

while the NPSMBg algorithm uses the non-patient-specific phase 1 score to choose both phase1 and phase 2 models. The phase 2 models chosen by the PSMBg algorithm are potentially different for each test case in contrast to the NPSMBg algorithm which selects the same models irrespective of the test case. This result provides support that the patient-specific search for models is able to choose models that better approximate the distribution of the target variable of the case at hand.

Figure 5-4 plots the estimate of $P(Z = \mathrm{T})$ for each test case as it varies with each addition of a model to the set of models being averaged over. A second curve plots the model score as the logarithmic posterior probability of the model given the data; this score measures the relative contribution of the model to the final estimate of $P(Z = \mathrm{T})$. Each row in the figure contains a pair of plots for a single test case, the plot on the left is obtained from the PSMBg algorithm and the corresponding plot on the right is obtained from the NPSMBg algorithm. The plot for the estimate of $P(Z = \mathrm{T})$ is shown in black while the plot for the model score is shown in gray. In



(a) (b)

Figure 5-3: Markov blanket model with the best score discovered by the PSMBl algorithm for the dataset given in Table 5-2. The global structure is given in (a) and the corresponding local structure for Z is given by the decision graph in (b). All other nodes from A through D have local structures consisting of a single DG node.

each plot, on going from left to right, the estimate of $P(Z = \mathrm{T})$ initially fluctuates widely and then settles to a stable estimate as the number of models providing the estimate increases. In the first two test cases the final estimates of $P(Z = \mathrm{T})$ obtained from the patient-specific and non-patient-specific model averaging respectively are very close; both the PSMBg and the NPSMBg algorithms predicted the value of Z correctly as T. In the third test case, the final estimates of $P(Z = \mathrm{T})$ are quite different; the PSMBg algorithm predicted the value of Z correctly as T while the NPSMBg algorithm predicted the value of Z incorrectly as F.

Table 5-5 also gives the results obtained from the PSMBl and NPSMBl algorithms. Complete model averaging over the space of MB with local structures could not be carried out since the number of models in this space is too large to be tractable. Both the PSMBl and NPSMBl algorithms have similar performance on the three test cases and show considerable improvement in logarithmic loss, squared error and the CAL score over the PSMBg algorithms. Their performance is also better than that of complete model averaging over MBs with *global* structure. This is due the fact that the generating model can be represented exactly by a MB with local structure. One such structure is shown in Figure 5-3; this structure was discovered by both the PSMBl and the NPSMBl algorithms as the best scoring structure. For the three test cases the PSMBl algorithm produced estimates of $P(Z = \mathrm{T})$ that are only marginally different from those produced by the NPSMBl algorithm, and the two algorithms are nearly indistinguishable on all the performance measures.

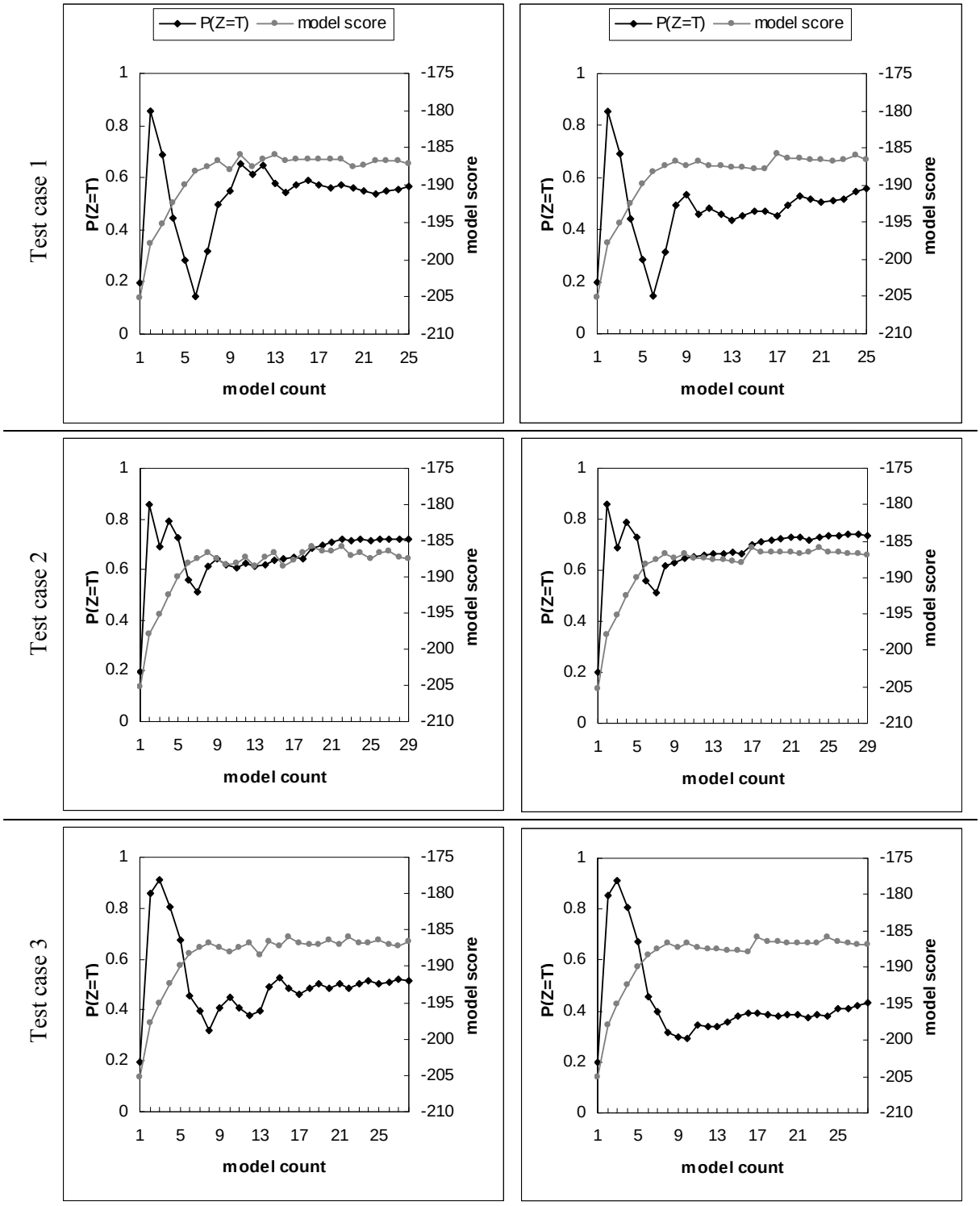


Figure 5-4: Plots of model averaged estimate of $P(Z = \mathrm{T})$ and the model score obtained by the PSMBg and the NPSMBg algorithms on the three test cases given in Figure 5-2. Each row represents a single test case with the plot on the left obtained from the PSMBg algorithm and the plot on the right obtained from the NPSMBg algorithm. The value of the final averaged estimate of $P(Z = \mathrm{T})$ is the point where the black curve meets the Y-axis on the right.

The best scoring structure shown in Figure 5-3 was the last structure discovered in phase 1 by both the PSMBl and the NPSMBl algorithms. This structure has a posterior probability about 200 times larger than the next best structure, and thus its estimate of Z contributes to a great extent to the final model averaged estimate of *Z*. This explains the lack of improvement in performance of the PSMBl algorithm over the NPSMBl algorithm. In contrast, several structures with similar posterior probabilities were found by the PSMBg and the NPSMBg algorithms and the final model averaged estimate of *Z* was not dominated by the estimate obtained from a single structure.

5.6 EVALUATION OF THE PSMBG ALGORITHM

This section describes the evaluation of the PSMBg algorithm on 21 UCI datasets and three medical datasets. The performance of the PSMBg algorithm is compared to that of the PSMBg-MS and the NPSMBg algorithms which are described in Section 5.4.1, and also to that of the six comparison machine learning methods which are described in Section 5.4.2. To recall, the PSMBg algorithm selects MB structures for model averaging using a two-phase search where phase 2 is patient-specific, the PSMBg-MS algorithm is a *model selection* version of the PSMBg algorithm that measures the effect of approximating the model averaging by model selection, and the NPSMBg algorithm is a *non-patient-specific* (i.e., population-wide) version of the PSMBg algorithm that measures the additional effect of the patient-specific heuristic on the performance of model averaging that can be achieved by a non-patient-specific method.

5.6.1 Experimental design

The experimental design is as follows:

- For each dataset, a total of 10 machine learning algorithms were run: PSMBg, PSMBg-MS, NPSMBg, NB, DT, LR, NN, *k*NN, LBR and ZR.
- The datasets used in the experiments are the 21 UCI datasets (listed in Table 5-1) and the three medical datasets with five target variables (listed in Table 5-2).
- Summary statistics were measured using 10-fold stratified cross-validation done twice (for a total of 20 training-test pairs) for the UCI datasets and a single training-test pair for the medical datasets. The summary statistics were computed for misclassification error, the AUC, logarithmic loss, squared error and the CAL score.
- The statistical tests performed were (1) significance testing with the Wilcoxon paired-samples signed ranks test, and (2) effect size testing with paired-samples *t* test.

The settings for the PSMBg algorithm are as follows:

- Phase 1: The model score for phase 1 is the Bayesian score computed using Equation 4.11, with a Dirichlet parameter prior with hyperparameters $\alpha_{ijk} = 1$ for all *i*, *j*, *k*. Phase 1 uses greedy hill-climbing search that terminates at a local maximum.
- Phase 2: The model score for phase 2 is computed using Equation 4.18 that is based on KL-divergence. Phase 2 uses best-first search with a priority queue Q whose maximum capacity is set to 1000. Phase 2 search terminates when no MB

structure in Q has a phase 2 score higher than $\varepsilon = 0.001$ for 10 consecutive iterations of the search. The maximum period of running time t for phase 2 was not specified since the algorithm terminated in a reasonable period of time on all the datasets with the specified value for ε.

- The predicted distribution for the target variable of the test case is computed using Equation 4.16; for each MB structure the parameters are estimated using Equation 4.6.

5.6.2 Results

Tables 5.6 to 5.10 report the means of the misclassification error, the AUC, logarithmic loss, squared error and the CAL score respectively for the PSMBg algorithm, its variants and the comparison algorithms. In each table, a row represents a dataset and a column represents an algorithm. The last three rows in each table give for each algorithm the overall mean of the specified performance measure across the UCI datasets, the medical datasets and the combined UCI and medical datasets respectively. The results are also plotted in Figures 5-5 to 5-9 along with the standard errors of the means. From the tables, it is seen that on all five performance measures, the PSMBg algorithm achieved a better overall average score than all other algorithms.

Tables 5.11 and 5.12 report results from pair-wise comparisons of the performance of the algorithms on the combined UCI and medical datasets that is aimed at assessing the statistical significance and the magnitude of the observed differences in the measures. Table 5-11 reports results from the Wilcoxon paired-samples signed ranks test. This test is a non-parametric procedure used to test whether there is sufficient evidence that the median of two probability

distributions differ in location. In evaluating algorithms, it can be used to test whether two algorithms differ significantly in performance on a specified measure. As it takes into account the magnitude and the direction of the difference between paired samples, this test is more powerful than the sign test [106]. Being a non-parametric test, it does not make any assumptions about the form of the underlying probability distribution of the sampled population. The test is carried out by sorting the absolute values of the paired differences from smallest to largest, assigning ranks to the absolute values (rank 1 to the smallest, rank 2 to the next smallest, and so on) and then finding the sum of the ranks of the positive differences. If the null hypothesis is true, the sum of the ranks of the positive differences should be about the same as the sum of the ranks of the negative differences.

Table 5-12 reports results from the paired-samples *t* test. This test is a parametric procedure used to determine whether there is a significant difference between the average values of the same performance measure for two different algorithms. The test assumes that the paired differences are independent and identically normally distributed. Though the measurements themselves may not be normally distributed, the pair-wise differences are often normally distributed.

The results are encouraging in that they show that the PSMBg algorithm never underperformed on any performance measure when compared to the other learning methods including the variants of the PSMBg algorithm that do model selection and non-patient-specific model averaging. This can be seen in the mean differences shown in Table 5-12. For misclassification error, logarithmic loss, squared error and the CAL score, the mean difference is always negative which denotes that the PSMBg algorithm always has a lower score on these measures. For the AUC, the difference is always positive which means that the PSMBg

algorithm always has a higher AUC. However, all mean differences are not statistically significant at the 0.05 level as can be seen by the p-values in Table 5-12. The best performance is seen in logarithmic loss where the PSMBg algorithm significantly outperforms all other methods, followed by squared error and CAL score where the PSMBg algorithm significantly outperforms many of the methods. On misclassification error and the AUC, the PSMBg algorithm has smaller performance gains. Similar results are seen in Table 5.11 that gives the Z statistics from the Wilcoxon paired-samples signed ranks test..

Table 5-13 gives the average number of models selected by the PSMBg and the NPSMBg algorithms in each of the phases for each dataset. The average number of models varies from 17.99 for the iris dataset (with four predictor variables) to 589.52 for the pneumonia dataset (with 152 predictor variables). The average number of phase 1 models in the pneumonia dataset was unusually high. This was due to the fact that this dataset has the largest number of variables and the phase 1 hill-climbing search terminates at a local maximum after a large number of iterations.

The average running time of the PSMBg algorithm for a single test case was approximately 1 hour and 30 minutes (see Table 5-21). This includes time spent in both phase 1 and 2 of the search. Typically, 70% - 90% of the running time was spent in phase 2.

5.6.3 Discussion

Overall, the PSMBg algorithm significantly improved on the probabilities of the predictions while maintaining or slightly improving on discrimination over all other algorithms used in the experiments. The non-patient-specific NPSMBg algorithm had inferior performance on logarithmic loss and squared error but similar performance on the other measures when

compared to the PSMBg algorithm. Both the PSMBg and the NPSMBg algorithms average over the same number of models and both select the same models in phase 1 of the search. In phase 2 of the search, while the number of selected models is the same, the two methods identify potentially different models. This provides evidence that the models selected in phase 2 by the PSMBg algorithm, using patient-specific search, are able to improve the performance of the PSMBg algorithm over the already good performance obtained by the NPSMBg algorithm.

Comparison of the PSMBg algorithm with the PSMBg-MS algorithm shows that model averaging outperformed the single best model on all the performance measures (Tables 5.11 and 5.12).

Table 5-6: Mean misclassification errors of different algorithms based on 10-fold cross-validation done twice on the UCI datasets and a single train-test validation on the medical datasets. To avoid cluttering only the mean for each dataset is shown. The bottom three rows give the average misclassification errors for the UCI datasets, the medical datasets and all the datasets respectively. Best results are in underlined.

Dataset	PSMBg	PSMBg-MS	NPSMBg	NB	DT	LR	NN	*k*NN	LBR
australian	0.1457	0.1457	0.1435	0.1449	0.1333	0.1486	0.1848	0.1457	0.1471
breast-cancer	0.0256	0.0271	0.0256	0.0256	0.0403	0.0337	0.0373	0.0286	0.0256
cleveland	0.1740	0.1791	0.1740	0.1655	0.2095	0.1655	0.1993	0.1791	0.1655
corral	0.0000	0.0156	0.0000	0.1328	0.0508	0.1289	0.0000	0.0977	0.1250
crx	0.1547	0.1577	0.1485	0.1348	0.1317	0.1424	0.1692	0.1485	0.1340
diabetes	0.2116	0.2129	0.2142	0.2201	0.2194	0.2135	0.2272	0.2201	0.2207
flare	0.1806	0.1834	0.1825	0.2012	0.1735	0.1721	0.2054	0.1806	0.1750
german	0.2580	0.2585	0.2580	0.2445	0.2845	0.2425	0.2980	0.2695	0.2475
glass2	0.1503	0.1564	0.1472	0.1595	0.1933	0.1442	0.1442	0.1411	0.1503
glass	0.2150	0.2220	0.2196	0.2687	0.2500	0.2547	0.2220	0.2173	0.2500
heart	0.1778	0.1778	0.1778	0.1630	0.1870	0.1630	0.1963	0.1741	0.1630
hepatitis	0.0938	0.1000	0.1000	0.1375	0.1250	0.1375	0.1688	0.0688	0.1375
iris	0.0567	0.0600	0.0633	0.0533	0.0600	0.0567	0.0633	0.0633	0.0533
lymphography	0.1622	0.1486	0.1622	0.1486	0.2365	0.2365	0.1622	0.1622	0.1520
pima	0.2155	0.2135	0.2142	0.2214	0.2259	0.2148	0.2389	0.2246	0.2227
postoperative	0.3391	0.3851	0.3391	0.3103	0.2989	0.3736	0.4138	0.3333	0.3103
sonar	0.1635	0.1659	0.1731	0.1490	0.1659	0.1442	0.1611	0.1707	0.1490
vehicle	0.2600	0.2577	0.2612	0.3712	0.2843	0.2914	0.2825	0.2766	0.2784
vote	0.0453	0.0582	0.0453	0.0927	0.0388	0.0733	0.0711	0.0819	0.0927
wine	0.0084	0.0084	0.0056	0.0112	0.0702	0.0253	0.0169	0.0281	0.0112
zoo	0.0347	0.0396	0.0347	0.0644	0.0792	0.0594	0.0495	0.0347	0.0644
pneumonia	0.1531	0.1516	0.1531	0.2274	0.1443	0.1472	0.1356	0.1137	0.1210
sepsis-d	0.0986	0.0968	0.0968	0.1649	0.0896	0.0914	0.1272	0.0896	0.1057
sepsis-s	0.2294	0.2276	0.2276	0.2563	0.2240	0.2312	0.2491	0.2348	0.2204
heart failure-d	0.0462	0.0464	0.0475	0.0561	0.0448	0.0448	0.0623	0.0464	0.0472
heart failure-c	0.1246	0.1246	0.1272	0.1426	0.1106	0.1111	0.1420	0.1128	0.1272
UCI average	0.1463	0.1511	0.1471	0.1629	0.1647	0.1629	0.1672	0.1546	0.1463
medical average	0.1304	0.1294	0.1304	0.1695	0.1227	0.1251	0.1432	0.1195	0.1304
overall average	0.1432	0.1469	0.1439	0.1641	0.1566	0.1557	0.1626	0.1478	0.1499

Table 5-7: Mean AUCs of different algorithms based on 10-fold cross-validation done twice on the UCI datasets and a single train-test validation on the medical datasets. To avoid cluttering only the mean for each dataset is shown. The bottom three rows give the average AUCs for the UCI datasets, the medical datasets and all the datasets respectively. Best results are in underlined.

Dataset	PSMBg	PSMBg-MS	NPSMBg	NB	DT	LR	NN	*k*NN	LBR
australian	0.9315	0.9303	0.9313	0.9200	0.9032	0.9187	0.8937	0.9092	0.9186
breast-cancer	0.9926	0.9922	0.9925	0.9933	0.9613	0.9879	0.9818	0.9930	0.9933
cleveland	0.9098	0.9079	0.9084	0.9141	0.7952	0.9089	0.8781	0.8995	0.9141
corral	1.0000	0.9997	1.0000	0.9252	0.9916	0.9459	1.0000	0.9827	0.9373
crx	0.9303	0.9280	0.9302	0.9301	0.9087	0.9138	0.9002	0.9057	0.9302
diabetes	0.8468	0.8468	0.8466	0.8438	0.7991	0.8439	0.8311	0.8148	0.8423
flare	0.7289	0.7288	0.7261	0.7557	0.4916	0.7451	0.6445	0.6797	0.7520
german	0.7662	0.7633	0.7641	0.7903	0.6736	0.7839	0.7340	0.7442	0.7891
glass2	0.8703	0.8653	0.8700	0.8769	0.7982	0.8845	0.8483	0.8384	0.8826
glass	0.9364	0.9361	0.9361	0.9408	0.8834	0.9101	0.9241	0.9112	0.9434
heart	0.9055	0.9049	0.9073	0.9106	0.8239	0.9032	0.8649	0.8791	0.9106
hepatitis	0.9225	0.9262	0.9237	0.9013	0.8203	0.7784	0.8436	0.8792	0.8970
iris	0.9890	0.9900	0.9905	0.9938	0.9629	0.9846	0.9785	0.9886	0.9938
lymphography	0.9139	0.9156	0.9173	0.9193	0.7741	0.8571	0.9192	0.9087	0.9175
pima	0.8431	0.8424	0.8424	0.8450	0.7977	0.8456	0.8237	0.8134	0.8449
postoperative	0.5026	0.4943	0.4538	0.5035	0.4228	0.4515	0.4113	0.3665	0.5035
sonar	0.9203	0.9204	0.9217	0.9343	0.8521	0.9275	0.9331	0.9132	0.9345
vehicle	0.9234	0.9228	0.9235	0.8655	0.8761	0.9016	0.8931	0.9032	0.9109
vote	0.9875	0.9850	0.9854	0.9684	0.9578	0.9582	0.9871	0.9735	0.9660
wine	0.9994	0.9994	0.9994	1.0000	0.9660	0.9967	0.9994	0.9981	1.0000
zoo	0.9994	0.9992	0.9992	0.9989	0.9565	0.9967	0.9916	0.9995	0.9989
pneumonia	0.8236	0.8262	0.8261	0.8585	0.5591	0.7414	0.7740	0.7874	0.8306
sepsis-d	0.8619	0.8618	0.8575	0.8698	0.7894	0.8482	0.8093	0.8517	0.8522
sepsis-s	0.7689	0.7697	0.7677	0.7748	0.6870	0.7558	0.7401	0.7702	0.7814
heart failure-d	0.7457	0.7424	0.7393	0.7725	0.1769	0.7607	0.7073	0.7455	0.7576
heart failure-c	0.7709	0.7698	0.7712	0.7879	0.4573	0.7898	0.6419	0.7465	0.7805
UCI average	0.8962	0.8952	0.8938	0.8919	0.8293	0.8783	0.8705	0.8715	0.8943
medical average	0.7942	0.7940	0.7924	0.8127	0.7059	0.7792	0.7345	0.7803	0.8005
overall average	0.8786	0.8757	0.8743	0.8767	0.8056	0.8592	0.8444	0.8539	0.8763

Table 5-8: Mean logarithmic losses of different algorithms based on 10-fold cross-validation done twice on the UCI datasets and a single train-test validation on the medical datasets. To avoid cluttering only the mean for each dataset is shown. The bottom three rows give the average logarithmic losses for the UCI datasets, the medical datasets and all the datasets respectively. Best results are in underlined.

Dataset	PSMBg	PSMBg-MS	NPSMBg	NB	DT	LR	NN	*k*NN	LBR
australian	0.3390	0.3456	0.3417	0.4476	0.4091	0.7136	0.4263	0.8627	0.4482
breast-cancer	0.1068	0.1138	0.1083	0.2497	0.2955	0.1485	0.1205	0.2138	0.2497
cleveland	0.3925	0.4067	0.4021	0.4491	1.3001	0.6500	0.4584	0.8625	0.4491
corral	0.1018	0.1101	0.0989	0.3326	0.1475	0.2753	0.1542	0.0175	0.3130
crx	0.3451	0.3564	0.3525	0.4113	0.3783	0.9377	0.4678	0.8747	0.4018
diabetes	0.4601	0.4606	0.4604	0.4809	0.5497	0.4588	0.6039	0.5028	0.4826
flare	0.4282	0.4294	0.4314	0.5904	0.4879	0.4042	0.5333	0.5858	0.5182
german	0.5331	0.5413	0.5377	0.5213	1.4604	0.5229	0.5801	1.5415	0.5221
glass2	0.4238	0.4302	0.4246	0.4532	0.8498	0.4154	0.8853	0.4562	0.4447
glass	0.7112	0.7239	0.7113	0.7697	2.3005	4.0749	1.3612	0.8685	0.7264
heart	0.3996	0.4069	0.3973	0.4560	0.6920	0.3907	0.6109	0.8483	0.4560
hepatitis	0.2396	0.2517	0.2583	0.4247	0.6122	17.7871	0.3562	0.6253	0.4272
iris	0.1560	0.1909	0.1620	0.1621	0.5287	0.7579	0.5770	0.2240	0.1621
lymphography	0.4100	0.4289	0.4430	0.4282	2.9112	21.6371	0.5765	0.7272	0.4409
pima	0.4647	0.4657	0.4657	0.4793	0.5268	0.4572	0.5873	0.5114	0.4774
postoperative	0.7381	0.7776	0.7287	0.7953	1.1395	2.8236	1.3339	1.9418	0.7953
sonar	0.3573	0.3726	0.3743	0.4573	1.2814	0.5762	0.4170	0.5728	0.4554
vehicle	0.5863	0.5900	0.5866	1.8645	2.3842	3.9997	1.0134	1.2590	0.7815
vote	0.1393	0.1635	0.1588	0.6804	0.3028	5.5427	0.3171	0.2782	0.5629
wine	0.0418	0.0402	0.0367	0.0303	0.8270	0.9593	0.1032	0.0409	0.0303
zoo	0.1297	0.1202	0.1268	0.1474	1.1102	0.5325	0.0596	0.1595	0.1474
pneumonia	0.5728	0.5713	0.5733	1.8092	1.6659	0.7102	0.5795	0.8787	0.6483
sepsis-d	0.2525	0.2520	0.2528	0.5183	0.3700	0.3492	0.2569	0.6711	0.3299
sepsis-s	0.4726	0.4751	0.4748	0.7639	0.5990	0.6016	0.8871	1.5491	0.6199
heart failure-d	0.3174	0.3179	0.3182	0.3491	0.5374	0.2962	0.3269	1.2575	0.3212
heart failure-c	0.1690	0.1700	0.1707	0.1797	0.1825	0.1626	0.1740	0.7067	0.1686
UCI average	0.3573	0.3679	0.3622	0.5063	0.9759	3.0507	0.5497	0.6654	0.4425
medical average	0.3569	0.3573	0.3580	0.7240	0.6710	0.4240	0.4449	1.0126	0.4176
overall average	0.3572	0.3659	0.3614	0.5481	0.9173	2.5456	0.5295	0.7322	0.4377

Table 5-9: Mean squared errors of different algorithms based on 10-fold cross-validation done twice on the UCI datasets and a single train-test validation on the medical datasets. To avoid cluttering only the mean for each dataset is shown. The bottom three rows give the average squared errors for the UCI datasets, the medical datasets and all the datasets respectively. Best results are in underlined.

Dataset	PSMBg	PSMBg-MS	NPSMBg	NB	DT	LR	NN	*k*NN	LBR
australian	0.2054	0.2082	0.2060	0.2234	0.2066	0.2116	0.3062	0.2287	0.2287
breast-cancer	0.0440	0.0449	0.0441	0.0474	0.0731	0.0542	0.0689	0.0484	0.0474
cleveland	0.2433	0.2499	0.2462	0.2553	0.3516	0.2339	0.3364	0.2526	0.2553
corral	0.0352	0.0463	0.0354	0.2056	0.0887	0.1836	0.0038	0.1051	0.1951
crx	0.2081	0.2146	0.2087	0.2092	0.1965	0.2121	0.2948	0.2363	0.2078
diabetes	0.2978	0.2981	0.2979	0.3073	0.3219	0.2978	0.3156	0.3315	0.3086
flare	0.2619	0.2626	0.2652	0.3145	0.2846	0.2513	0.3203	0.2843	0.2700
german	0.3526	0.3570	0.3555	0.3419	0.4196	0.3368	0.5104	0.3591	0.3433
glass2	0.2469	0.2513	0.2468	0.2450	0.3116	0.2409	0.2572	0.2603	0.2393
glass	0.3609	0.3635	0.3605	0.3823	0.4186	0.4363	0.4075	0.3880	0.3673
heart	0.2444	0.2486	0.2420	0.2570	0.3113	0.2394	0.3273	0.2611	0.2570
hepatitis	0.1410	0.1495	0.1534	0.2079	0.2170	0.2750	0.2579	0.1481	0.2090
iris	0.0727	0.0828	0.0753	0.0751	0.1122	0.0942	0.1032	0.1086	0.0751
lymphography	0.2391	0.2353	0.2433	0.2344	0.4162	0.4545	0.2687	0.2650	0.2406
pima	0.3009	0.3011	0.3011	0.3065	0.3264	0.2968	0.3248	0.3332	0.3060
postoperative	0.4772	0.5044	0.4748	0.4894	0.4525	0.6011	0.7221	0.6168	0.4894
sonar	0.2349	0.2391	0.2369	0.2411	0.2887	0.2228	0.2764	0.2402	0.2405
vehicle	0.3471	0.3481	0.3470	0.5805	0.4171	0.4109	0.4672	0.3934	0.4059
vote	0.0788	0.0903	0.0810	0.1681	0.0703	0.1461	0.1172	0.1293	0.1529
wine	0.0183	0.0158	0.0142	0.0191	0.1268	0.0503	0.0213	0.0407	0.0191
zoo	0.0612	0.0652	0.0630	0.0860	0.1415	0.0991	0.0568	0.0406	0.0860
pneumonia	0.2442	0.2435	0.2433	0.4149	0.2647	0.2546	0.2453	0.1952	0.2051
sepsis-d	0.1501	0.1500	0.1516	0.2473	0.1505	0.1513	0.2226	0.1458	0.1754
sepsis-s	0.3120	0.3123	0.3139	0.4156	0.3428	0.3155	0.4431	0.3499	0.3348
heart failure-d	0.0839	0.0842	0.0844	0.0951	0.0853	0.0810	0.1161	0.0852	0.0838
heart failure-c	0.1883	0.1886	0.1892	0.2137	0.1925	0.1726	0.2677	0.1910	0.1913
UCI average	0.2129	0.2179	0.2142	0.2475	0.2644	0.2547	0.2745	0.2415	0.2354
medical average	0.1957	0.1957	0.1965	0.2773	0.2072	0.1950	0.2590	0.1934	0.1981
overall average	0.2096	0.2137	0.2108	0.2532	0.2534	0.2432	0.2715	0.2323	0.2283

Table 5-10: Mean CAL scores of different algorithms based on 10-fold cross-validation done twice on the UCI datasets and a single train-test validation on the medical datasets. To avoid cluttering only the mean for each dataset is shown. The bottom three rows give the average CAL scores for the UCI datasets, the medical datasets and all the datasets respectively. Best results are in underlined.

Dataset	PSMBg	PSMBg-MS	NPSMBg	NB	DT	LR	NN	*k*NN	LBR
australian	0.0470	0.0459	0.0454	0.0775	0.0463	0.0440	0.0526	0.1423	0.0817
breast-cancer	0.0146	0.0146	0.0144	0.0200	0.0261	0.0155	0.0114	0.0299	0.0200
cleveland	0.0497	0.0630	0.0569	0.0930	0.0690	0.0295	0.0432	0.1543	0.0930
corral	0.0583	0.0656	0.0561	0.0470	0.0505	0.0473	0.0162	0.0115	0.0368
crx	0.0452	0.0518	0.0503	0.0711	0.0440	0.0394	0.0722	0.1354	0.0689
diabetes	0.0403	0.0401	0.0411	0.0618	0.0633	0.0433	0.0813	0.0662	0.0590
flare	0.0551	0.0546	0.0562	0.1260	0.0467	0.0414	0.0762	0.1000	0.0707
german	0.0684	0.0696	0.0699	0.0625	0.1038	0.0504	0.0547	0.2363	0.0645
glass2	0.0359	0.0395	0.0373	0.0644	0.0386	0.0322	0.0482	0.0561	0.0569
glass	0.0188	0.0189	0.0186	0.0282	0.0223	0.0262	0.0258	0.0246	0.0241
heart	0.0498	0.0585	0.0513	0.0913	0.0641	0.0321	0.0624	0.1385	0.0913
hepatitis	0.0422	0.0294	0.0381	0.0488	0.0306	0.0462	0.0197	0.0492	0.0466
iris	0.0110	0.0115	0.0114	0.0132	0.0188	0.0142	0.0219	0.0205	0.0132
lymphography	0.0226	0.0259	0.0256	0.0326	0.0279	0.0863	0.0272	0.0512	0.0359
pima	0.0532	0.0539	0.0539	0.0596	0.0660	0.0444	0.0960	0.0805	0.0586
postoperative	0.0404	0.0358	0.0438	0.0436	0.0450	0.0707	0.0844	0.1175	0.0436
sonar	0.0437	0.0656	0.0643	0.1042	0.0591	0.0814	0.0503	0.1336	0.1045
vehicle	0.0479	0.0481	0.0480	0.1272	0.0654	0.0632	0.0567	0.0984	0.0690
vote	0.0247	0.0285	0.0306	0.0722	0.0227	0.0520	0.0603	0.0346	0.0658
wine	0.0062	0.0043	0.0054	0.0083	0.0247	0.0154	0.0256	0.0133	0.0083
zoo	0.0065	0.0067	0.0069	0.0078	0.0094	0.0055	0.0029	0.0075	0.0078
pneumonia	0.0998	0.0991	0.0985	0.2078	0.0696	0.1001	0.0730	0.1051	0.0905
sepsis-d	0.0353	0.0466	0.0310	0.1255	0.0373	0.0314	0.0161	0.0921	0.0585
sepsis-s	0.0627	0.0710	0.0690	0.1923	0.0857	0.0578	0.1077	0.2066	0.1329
heart failure-d	0.0263	0.0288	0.0255	0.0448	0.0667	0.0280	0.0266	0.0490	0.0269
heart failure-c	0.0533	0.0523	0.0584	0.0854	0.1505	0.0372	0.0586	0.1271	0.0577
UCI average	0.0372	0.0396	0.0393	0.0600	0.0450	0.0419	0.0471	0.0810	0.0533
medical average	0.0555	0.0596	0.0565	0.1312	0.0820	0.0509	0.0564	0.1160	0.0733
overall average	0.0407	0.0434	0.0426	0.0737	0.0521	0.0437	0.0489	0.0877	0.0572

Table 5-11: Wilcoxon paired-samples signed ranks test comparing the performance of PSMBg with other algorithms. For each performance measure the number on top is the Z statistic and the number at the bottom is the corresponding p-value. The Z statistic is negative when PSMBg has a lower score on a performance measure than the competing algorithm. On all measures except the AUC, a negative Z statistic indicates better performance by PSMBg; on the AUC a positive Z statistic indicates better performance by PSMBg. Underlined results indicate p-values of 0.05 or smaller.

Performance measure	PSMBg-MS	NPSMBg	NB	DT	LR	NN	*k*NN	LBR
Misclassification error	-2.338	-0.776	-2.121	-2.070	-1.181	-3.861	-0.825	-0.368
	0.019	0.438	0.034	0.038	0.238	0.000	0.409	0.713
AUC	2.085	1.257	1.511	4.457	2.197	4.029	4.203	0.927
	0.037	0.209	0.131	0.000	0.028	0.000	0.000	0.354
Logarithmic loss	-3.595	-2.426	-4.280	-4.457	-3.340	-4.254	-4.026	-4.051
	0.000	0.015	0.000	0.000	0.001	0.000	0.000	0.000
Squared error	-3.608	-2.313	-3.975	-3.24	-2.121	-4.127	-3.518	-3.213
	0.000	0.021	0.000	0.000	0.034	0.000	0.000	0.001
CAL score	-2.032	-1.867	-4.026	-2.806	-0.063	-4.076	-1.892	-3.543
	0.042	0.062	0.000	0.005	0.949	0.000	0.058	0.000

Table 5-12: Paired-samples *t* test comparing the performance of PSMBg with other algorithms. For each performance measure the number on top is the mean difference between PSMBg and the indicated algorithm and the number at the bottom is the corresponding p-value. The mean difference is negative when PSMBg has a lower score on a performance measure than the competing algorithm. On all measures except the AUC, a negative mean difference indicates better performance by PSMBg; on the AUC a positive mean difference indicates better performance by PSMBg. Underlined results indicate p-values of 0.05 or smaller.

Performance measure	PSMBg-MS	NPSMBg	NB	DT	LR	NN	*k*NN	LBR
Misclassification error	-0.004	-0.001	-0.021	-0.013	-0.012	-0.019	-0.005	-0.007
	0.077	0.312	0.014	0.021	0.065	0.000	0.334	0.289
AUC	0.001	0.002	0.000	0.104	0.017	0.032	0.023	0.000
	0.077	0.242	0.975	0.000	0.022	0.000	0.000	0.932
Logarithmic loss	-0.009	-0.004	-0.163	-0.211	-0.215	-0.306	-0.140	-0.071
	0.001	0.026	0.005	0.000	0.006	0.000	0.000	0.000
Squared error	-0.004	-0.001	-0.044	-0.044	-0.034	-0.062	-0.023	-0.019
	0.003	0.054	0.002	0.000	0.009	0.002	0.000	0.017
CAL score	-0.003	-0.002	-0.033	-0.011	-0.003	-0.047	-0.008	-0.016
	0.044	0.058	0.000	0.018	0.441	0.000	0.079	0.001

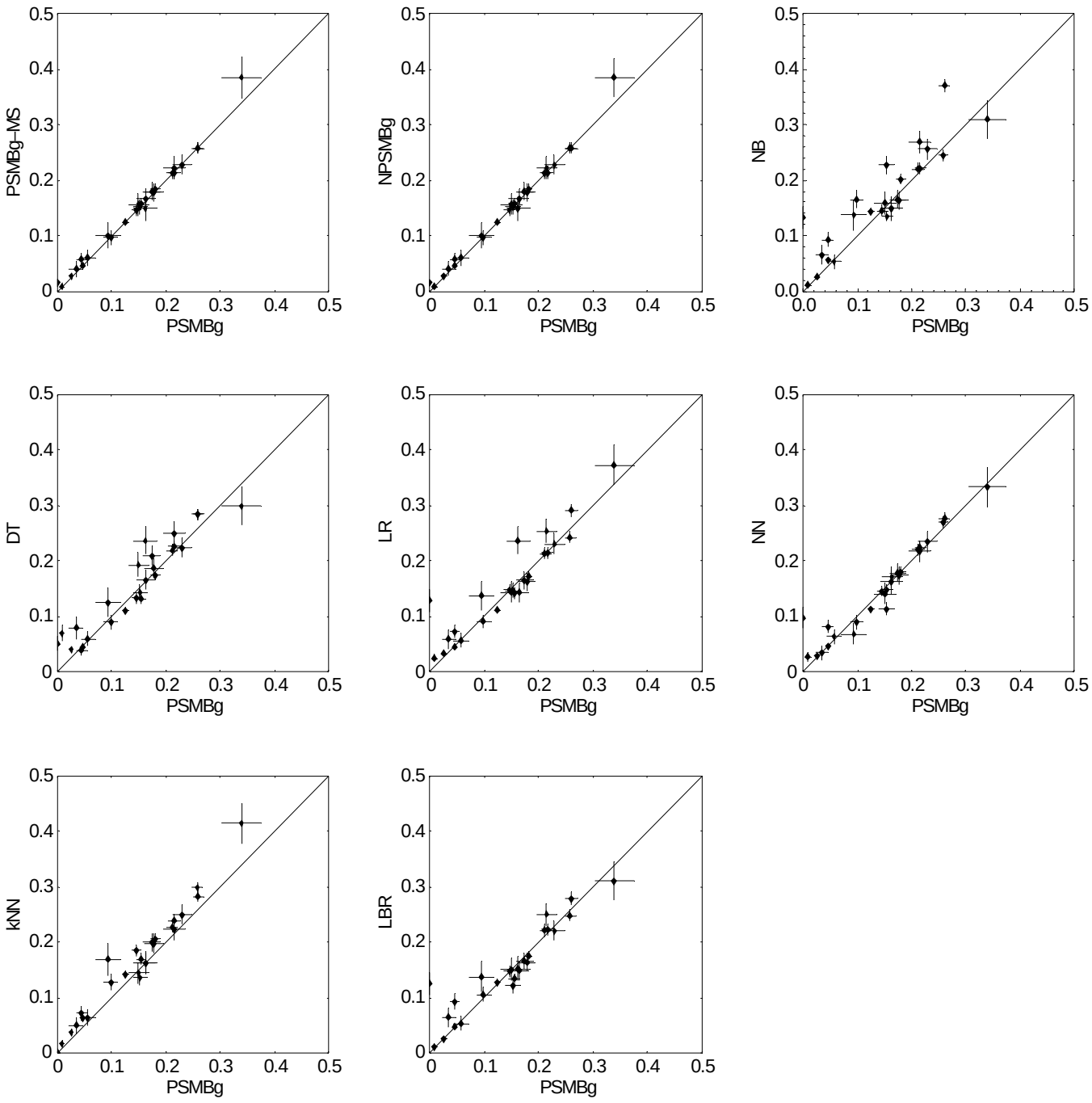


Figure 5-5: Pairwise plots of the mean misclassification errors of PSMBg vs. competing algorithms. Each point represents the mean misclassification errors of PSMBG and a competing algorithm on a single dataset, and the crosshairs represent one standard deviation on either side of the mean misclassification errors. Points above the diagonal line represent better performance by PSMBg than the competing algorithm.

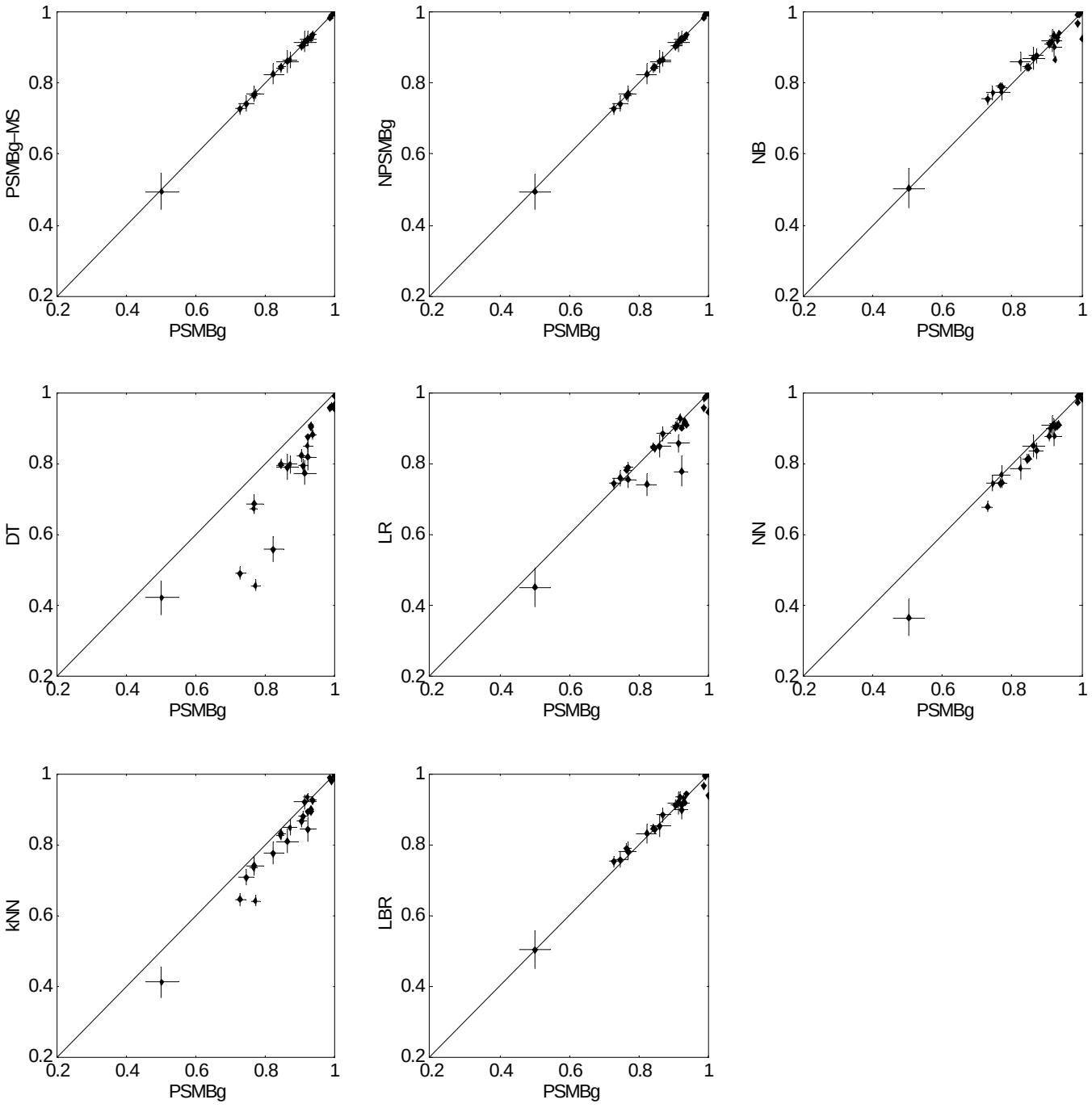


Figure 5-6: Pairwise plots of the mean AUCs of PSMBg vs. competing algorithms. Each point represents the mean AUCs of PSMBG and a competing algorithm on a single dataset, and the crosshairs represent one standard deviation on either side of the mean AUCs. Points above the diagonal line represent better performance by PSMBg than the competing algorithm.

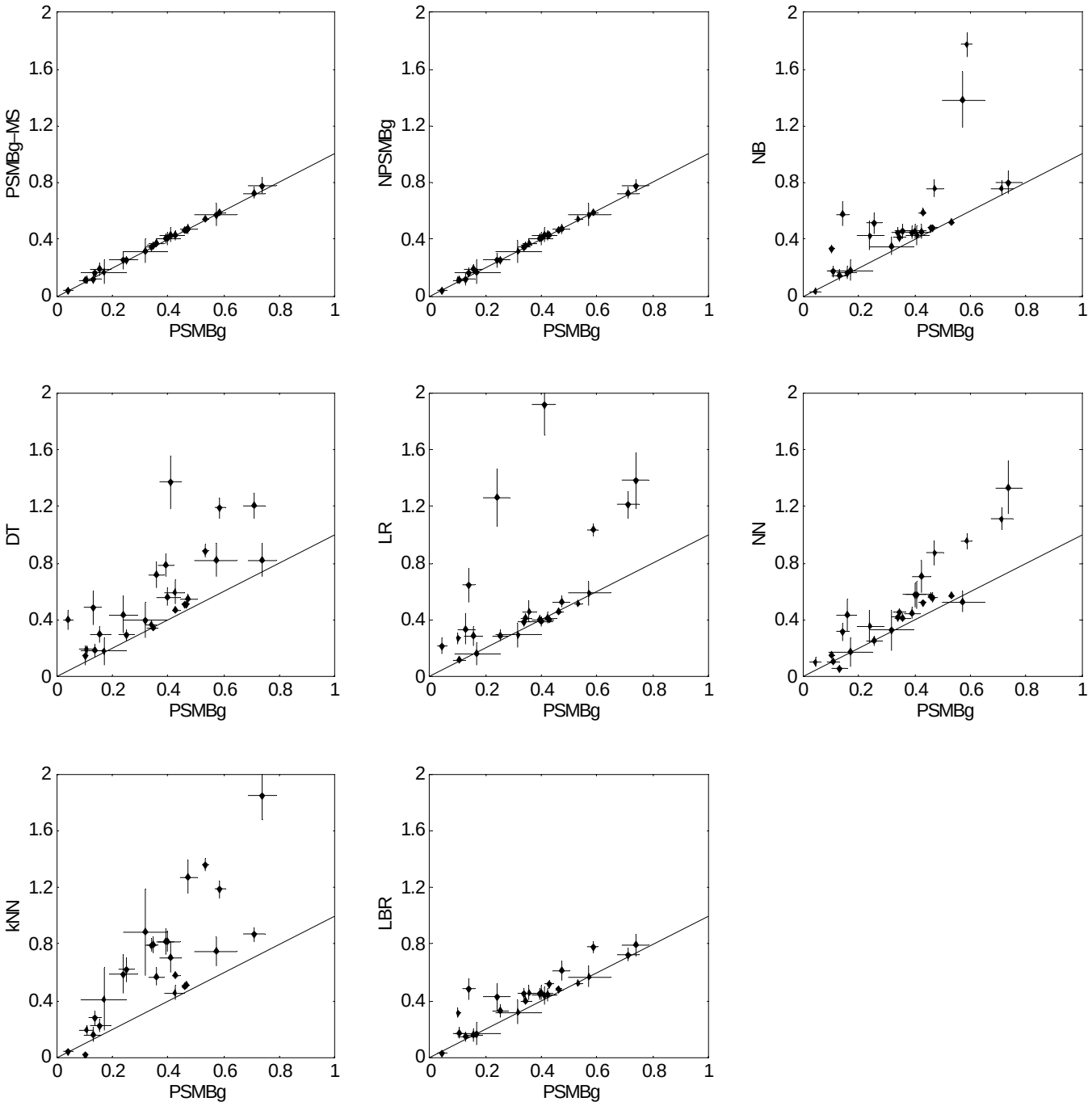


Figure 5-7: Pairwise plots of the mean logarithmic losses of PSMBg vs. competing algorithms. Each point represents the mean logarithmic losses of PSMBG and a competing algorithm on a single dataset, and the crosshairs represent one standard deviation on either side of the mean logarithmic losses. Points above the diagonal line represent better performance by PSMBg than the competing algorithm.

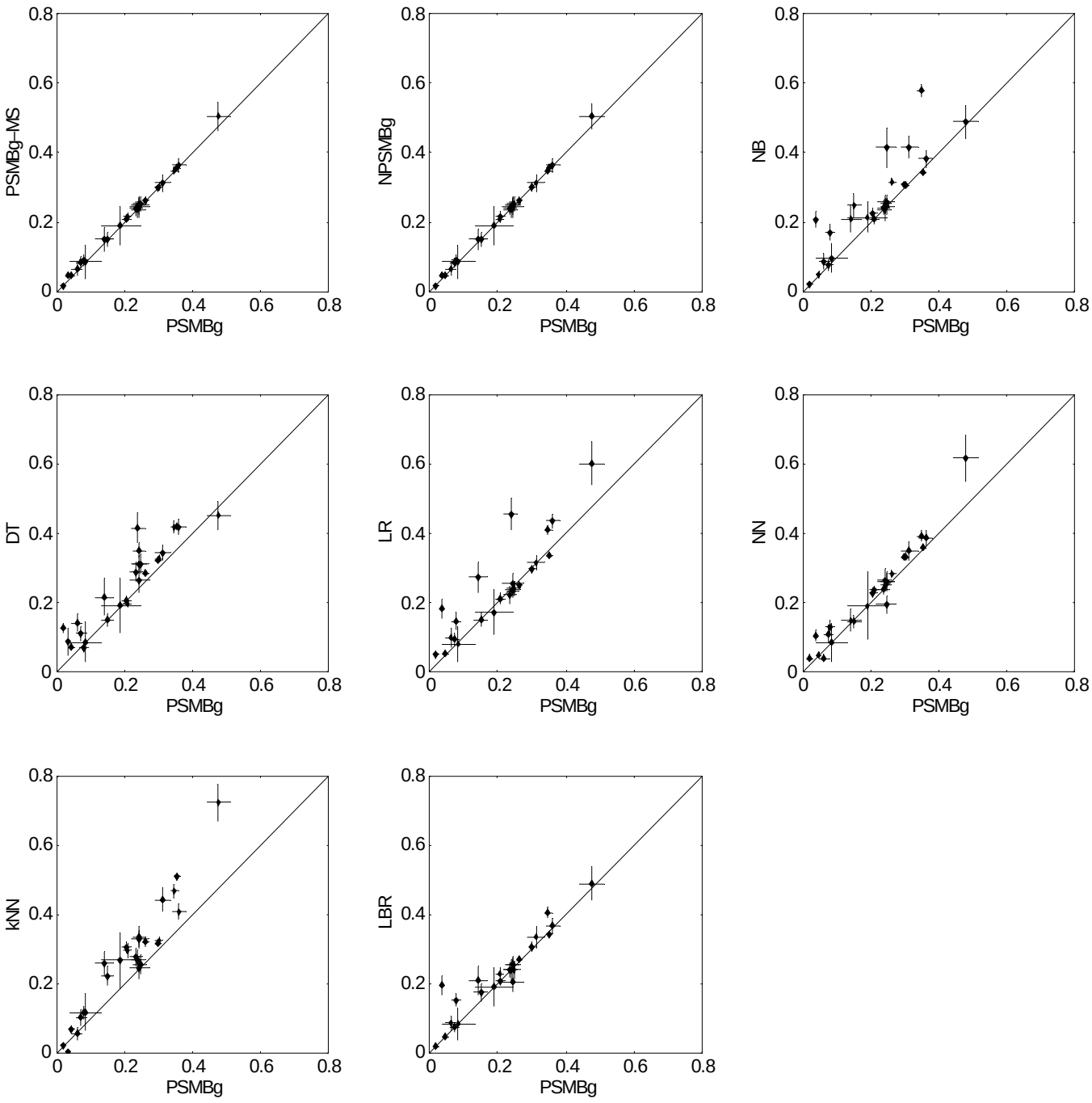


Figure 5-8: Pairwise plots of the mean squared errors of PSMBg vs. competing algorithms. Each point represents the mean squared errors of PSMBG and a competing algorithm on a single dataset, and the crosshairs represent one standard deviation on either side of the mean squared errors. Points above the diagonal line represent better performance by PSMBg than the competing algorithm.

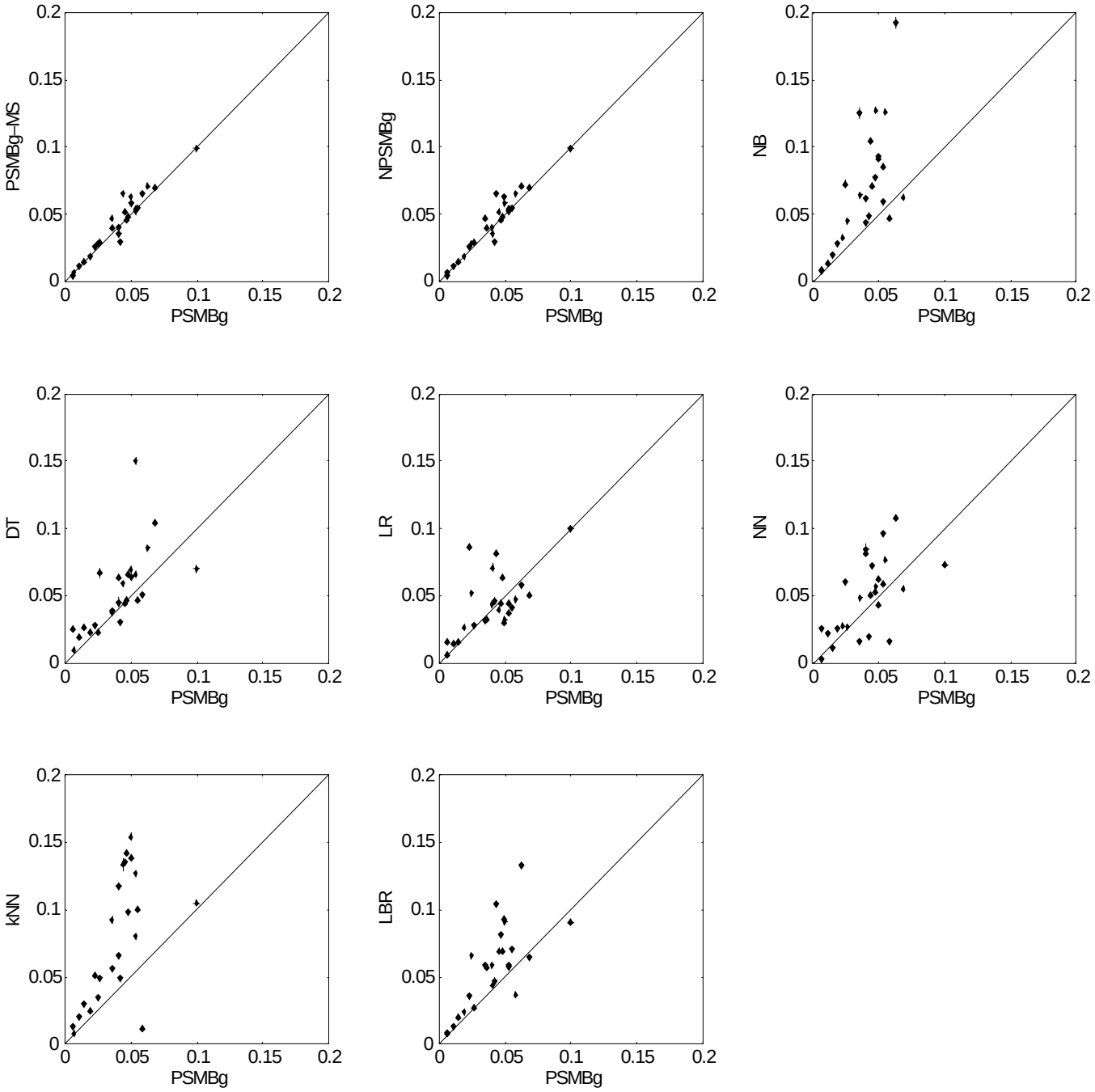


Figure 5-9: Pairwise plots of the mean CAL scores of PSMBg vs. competing algorithms. Each point represents the mean CAL scores of PSMBG and a competing algorithm on a single dataset, and the crosshairs represent one standard deviation on either side of the mean CAL scores. Points above the diagonal line represent better performance by PSMBg than the competing algorithm.

Table 5-13: Average number of models in phases 1 and 2 over which averaging is carried out by the PSMBg and NPSMBg algorithms. Both algorithms average over the same number of models in each phase. Both algorithms select the same models in phase 1 but potentially different models in phase 2. The number of models in phases 1 and 2 is the sum of the models selected in the two phases.

Dataset	# models phase 1	# models phase 2	# models phases 1 and 2
australian	28.55	11.00	39.55
breast-cancer	18.85	10.15	29.00
cleveland	20.45	11.99	32.44
corral	10.65	15.03	25.68
crx	32.10	13.42	45.52
diabetes	11.65	10.03	21.68
flare	20.75	11.44	32.19
german	22.45	19.23	41.68
glass2	12.05	13.26	25.31
glass	15.80	10.73	26.53
heart	18.50	11.32	29.82
hepatitis	27.45	26.63	54.08
iris	7.25	10.74	17.99
lymphography	51.55	37.83	89.38
pima	40.40	16.97	57.37
postoperative	12.00	10.02	22.02
sonar	11.65	10.09	21.74
vehicle	1.15	21.09	22.24
vote	59.80	18.44	78.24
wine	39.30	10.73	50.03
zoo	45.55	13.53	59.08
pneumonia	577.00	12.52	589.52
sepsis-d	23.80	11.45	35.25
sepsis-s	27.00	13.63	40.63
heart failure-d	47.00	19.07	66.07
heart failure-c	22.00	20.92	42.92

5.7 EVALUATION OF THE PSMBL ALGORITHM

This section describes the evaluation of the performance of the PSMBl algorithm and compares its performance to that of the PSMBg algorithm. Analogous to the PSMBg algorithm, the PSMBl algorithm selects MB structures for model averaging using a two-phase search where phase 2 is patient-specific. The difference between the two is in the model space: the PSMBl algorithm searches in the richer space of MB structures that explicitly represent local structure by using decision graphs for CPDs. The performance of the PSMBl algorithm is compared to: (1) the PSMBl-MS algorithm which is a *model selection* version of the PSMBl algorithm, (2) the NPSMBl algorithm which is a *non-patient-specific* (i.e., population-wide) version of the PSMBl algorithm, and (3) the PSMBg algorithm. The first two algorithms are analogous to the PSMBg-MS and the NPSMBg algorithms. The PSMBl-MS algorithm chooses the model with the highest posterior probability from the set of models identified by the PSMBl algorithm, and uses that single model for predicting the target variable of the case at hand. The NPSMBl algorithm averages over the same number of models as the PSMBl algorithm except that the patient-specific model score of phase 2 is not used to score models; instead all models are scored with the non-patient-specific model score of phase 1.

5.7.1 Experimental design

The experimental design for the evaluation the PSMBl algorithm is the same as that for the PSMBg algorithm and is described in section 5.6.1.

5.7.2 Results

Tables 5-15 to 5-19 report the means of the misclassification error, the AUC, logarithmic loss, squared error and the CAL score respectively. The last three rows in each table give for each algorithm the overall mean of the specified performance measure across the UCI datasets, the medical datasets and combined UCI and medical datasets respectively. The results are also plotted in Figures 5-8 and 5-9 along with the standard errors of means. Table 5-18 reports results from the paired-samples *t* test and Table 5-19 reports results from the Wilcoxon paired-samples signed ranks test for the combined UCI and medical datasets. The PSMBl algorithm performed significantly better than the PSMBl-MS algorithm on all measures except the CAL score, which is similar to the results obtained from the PSMBg algorithm. However, the PSMBl algorithm showed no improvement in performance over the NPSMBl algorithm. When compared to the PSMBg algorithm, the PSMBl algorithm performed at a similar level on all the measures except the AUC on which it performed slightly worse.

The average running time of the PSMBl algorithm for a single test case was approximately 5 hours and 30 minutes (see Table 5-21). This includes time spent in both phase 1 and 2 of the search. Typically, 80% - 90% of the running time was spent in phase 2.

5.7.3 Discussion

While on the synthetic dataset the PSMBl algorithm performed considerably better than the PSMBg algorithm, on most of the UCI datasets and on all the medical datasets it showed no improvement over the PSMBg algorithm. One exception is the corral dataset which is a synthetic

dataset with seven variables of which four deterministically determine the target variable Z as follows:

$$Z = (A \wedge B) \vee (C \wedge D).$$

Of the remaining two variables, one is correlated with Z and the other is irrelevant for predicting Z. On this dataset, the PSMBl algorithm significantly improved on logarithmic loss, squared error and the CAL score over the PSMBg algorithm while both algorithms had perfect performance on misclassification error and the AUC. The superior performance of the PSMBl algorithm stems from the fact that the deterministic function can be represented exactly by a MB with local structure. In fact, the PSMBl-MS algorithm which chooses the best scoring model performed even better since the chosen model is the generating model.

Two possible reasons may explain the inability of the PSMBl algorithm to improve significantly over the performance of the PSMBg algorithm in the UCI and medical datasets. First, there may not be many value-specific independencies present in the datasets and the PSMBl algorithm may be capturing spurious value-specific independencies. Second, the patient-specific phase 2 search as implemented in the PSMBl algorithm is not optimal. Ideally, in phase 2 every candidate MB structure should be evaluated with the patient-specific phase 2 score. However, for reasons of computational efficiency, phase 2 search in the PSMBl algorithm is implemented as follows. At each iteration of the outer search procedure, all possible global operators are applied to the current best MB structure to generate successor MB structures. Then, for each successor MB structure, the inner search procedure is invoked on those MB nodes whose parent sets have been modified by the application of the global operator. For each MB node on which it is invoked, the inner search procedure performs greedy hill-climbing search to identify a decision graph with phase 1 score, rather than a decision graph with high phase 2 score

(see the pseudocode for *ProcedureLocalSearchForPSMBl* in Figure 4-9). This is because the phase 1 score is computed efficiently for the MB node in question, while computation of the phase 2 score requires doing inference in the MB and is less efficient in the current implementation of the PSMBl algorithm. After the inner search procedure returns decision graphs with high phase 1 scores, the phase 2 score is computed for the MB. A more efficient implementation of the PSMBl algorithm can allow the inner search procedure to evaluate candidate local structures with the phase 2 score, which can potentially improve its performance.

5.8 SUMMARY

The two patient-specific algorithms, namely, the PSMBg and the PSMBl algorithms, were evaluated on one synthetic, 21 UCI and three medical datasets. Their performances on five measures were compared to that of non-patient-specific and model selection versions as well six commonly used predictive algorithms.

The PSMBg algorithm improved the prediction of the target variable on average over all the comparison algorithms. The greatest improvements occurred in logarithmic loss and squared error, followed by good improvement in calibration and smaller improvements in misclassification error and the AUC. In addition, the PSMBg algorithm that performs Bayesian model averaging in conjunction with the patient-specific heuristic had better performance than either model selection with the patient-specific heuristic or non-patient-specific Bayesian model averaging.

The PSMBl algorithm did not improve significantly over the PSMBg algorithm on any measure. In addition, the PSMBl algorithm that performs Bayesian model averaging in

conjunction with the patient-specific heuristic had better performance than model selection with the patient-specific heuristic but performed on par with non-patient-specific Bayesian model averaging. The use of local structure did not lead to significant improvements over the use of global structure alone. Possible reasons for this lack of improvement were given in the previous section.

Table 5-14: Mean misclassification errors of different algorithms based on 10-fold cross-validation done twice on the UCI datasets and a single train-test validation on the medical datasets. To avoid cluttering only the mean for each dataset is shown. The bottom three rows give the average misclassification errors for the UCI datasets, for the medical datasets and for all the datasets respectively. Best results are underlined.

Dataset	PSMBl	PSMBl-MS	NPSMBl	PSMBg
australian	0.1428	0.1457	0.1428	0.1457
breast-cancer	0.0264	0.0271	0.0271	0.0256
cleveland	0.1723	0.1774	0.1807	0.1740
corral	0.0000	0.0000	0.0000	0.0000
crx	0.1455	0.1462	0.1386	0.1547
diabetes	0.2188	0.2194	0.2168	0.2116
flare	0.1820	0.1815	0.1801	0.1806
german	0.2480	0.2460	0.2475	0.2580
glass2	0.1718	0.1718	0.1840	0.1503
glass	0.2734	0.2850	0.2734	0.2150
heart	0.1704	0.1741	0.1759	0.1778
hepatitis	0.1438	0.1375	0.1500	0.0938
iris	0.0533	0.0533	0.0533	0.0567
lymphography	0.1486	0.1588	0.1520	0.1622
pima	0.2155	0.2148	0.2129	0.2155
postoperative	0.2989	0.2989	0.2989	0.3391
sonar	0.1731	0.1755	0.1731	0.1635
vehicle	0.2902	0.2908	0.2931	0.2600
vote	0.0603	0.0625	0.0647	0.0453
wine	0.0140	0.0140	0.0140	0.0084
zoo	0.0396	0.0396	0.0396	0.0347
pneumonia	0.1530	0.1545	0.1532	0.1531
sepsis-d	0.1022	0.1022	0.1022	0.0986
sepsis-s	0.2079	0.2133	0.2061	0.2294
heart failure-d	0.0467	0.0479	0.0466	0.0462
heart failure-c	0.1241	0.1255	0.1244	0.1246
UCI average	0.1518	0.1533	0.1533	0.1463
medical average	0.1268	0.1287	0.1265	0.1304
overall average	0.1521	0.1537	0.1533	0.1478

Table 5-15: Mean AUCs of different algorithms based on 10-fold cross-validation done twice on the UCI datasets and a single train-test validation on the medical datasets. To avoid cluttering only the mean for each dataset is shown. The bottom three rows give the average AUCs for the UCI datasets, for the medical datasets and for all the datasets respectively. Best results are underlined.

Dataset	PSMBl	PSMBl-MS	NPSMBl	PSMBg
australian	0.9209	0.9189	0.9208	0.9315
breast-cancer	0.9910	0.9907	0.9910	0.9926
cleveland	0.9049	0.9035	0.9036	0.9098
corral	1.0000	1.0000	1.0000	1.0000
crx	0.9242	0.9231	0.9262	0.9303
diabetes	0.8436	0.8425	0.8451	0.8468
flare	0.7140	0.7138	0.7185	0.7289
german	0.7662	0.7659	0.7667	0.7662
glass2	0.8763	0.8752	0.8757	0.8703
glass	0.9242	0.9214	0.9244	0.9364
heart	0.9076	0.9076	0.9081	0.9055
hepatitis	0.8312	0.8243	0.8203	0.9225
iris	0.9930	0.9938	0.9930	0.9890
lymphography	0.9124	0.9195	0.9195	0.9139
pima	0.8444	0.8444	0.8449	0.8431
postoperative	0.4363	0.4346	0.4324	0.5026
sonar	0.9292	0.9253	0.9332	0.9203
vehicle	0.9135	0.9134	0.9131	0.9234
vote	0.9822	0.9813	0.9808	0.9875
wine	1.0000	1.0000	1.0000	0.9994
zoo	0.9879	0.9871	0.9865	0.9994
pneumonia	0.8232	0.8241	0.8238	0.8236
sepsis-d	0.8597	0.8583	0.8549	0.8619
sepsis-s	0.7594	0.7617	0.7609	0.7689
heart failure-d	0.7465	0.7472	0.7464	0.7457
heart failure-c	0.7702	0.7717	0.7711	0.7709
UCI average	0.8859	0.8851	0.8859	0.8962
medical average	0.7918	0.7926	0.7914	0.7942
overall average	0.8792	0.8785	0.8791	0.8786

Table 5-17: Mean logarithmic losses of different algorithms based on 10-fold cross-validation done twice on the UCI datasets and a single train-test validation on the medical datasets. To avoid cluttering only the mean for each dataset is shown. The bottom three rows give the average logarithmic losses for the UCI datasets, for the medical datasets and for all the datasets respectively. Best results are underlined.

Dataset	PSMBl	PSMBl-MS	NPSMBl	PSMBg
australian	0.3589	0.3651	0.3607	0.3390
breast-cancer	0.1206	0.1248	0.1239	0.1068
cleveland	0.4205	0.4337	0.4276	0.3925
corral	0.0457	0.0290	0.0425	0.1018
crx	0.3683	0.3753	0.3673	0.3451
diabetes	0.4655	0.4671	0.4644	0.4601
flare	0.4365	0.4385	0.4346	0.4282
german	0.5221	0.5240	0.5221	0.5331
glass2	0.4571	0.4574	0.4611	0.4238
glass	0.7598	0.7650	0.7598	0.7112
heart	0.3998	0.4026	0.3999	0.3996
hepatitis	0.4200	0.4949	0.4960	0.2396
iris	0.1618	0.1614	0.1619	0.1560
lymphography	0.4164	0.4445	0.4328	0.4100
pima	0.4632	0.4641	0.4627	0.4647
postoperative	0.6882	0.6948	0.6881	0.7381
sonar	0.3334	0.3541	0.3310	0.3573
vehicle	0.5986	0.5994	0.5975	0.5863
vote	0.1848	0.2024	0.2040	0.1393
wine	0.0227	0.0286	0.0226	0.0418
zoo	0.1256	0.1256	0.1255	0.1297
pneumonia	0.5748	0.5751	0.5746	0.5728
sepsis-d	0.2586	0.2597	0.2610	0.2525
sepsis-s	0.4704	0.4717	0.4642	0.4726
heart failure-d	0.3170	0.3181	0.3170	0.3174
heart failure-c	0.1692	0.1697	0.1691	0.1690
UCI average	0.3700	0.3787	0.3755	0.3573
medical average	0.3580	0.3589	0.3572	0.3569
overall average	0.3695	0.3776	0.3744	0.3572

Table 5-16: Mean squared errors of different algorithms based on 10-fold cross-validation done twice on the UCI datasets and a single train-test validation on the medical datasets. To avoid cluttering only the mean for each dataset is shown. The bottom three rows give the average squared errors for the UCI datasets, for the medical datasets and for all the datasets respectively. Best results are in underlined.

Dataset	PSMBl	PSMBl-MS	NPSMBl	PSMBg
australian	0.2137	0.2166	0.2140	0.2054
breast-cancer	0.0445	0.0444	0.0458	0.0440
cleveland	0.2535	0.2596	0.2574	0.2433
corral	0.0086	0.0051	0.0074	0.0352
crx	0.2132	0.2164	0.2098	0.2081
diabetes	0.3012	0.3028	0.3007	0.2978
flare	0.2679	0.2691	0.2671	0.2619
german	0.3444	0.3450	0.3439	0.3526
glass2	0.2577	0.2574	0.2619	0.2469
glass	0.3745	0.3769	0.3745	0.3609
heart	0.2429	0.2449	0.2438	0.2444
hepatitis	0.2084	0.2315	0.2295	0.1410
iris	0.0758	0.0757	0.0759	0.0727
lymphography	0.2350	0.2441	0.2391	0.2391
pima	0.3009	0.3013	0.3001	0.3009
postoperative	0.4386	0.4429	0.4385	0.4772
sonar	0.2201	0.2301	0.2179	0.2349
vehicle	0.3580	0.3585	0.3577	0.3471
vote	0.0958	0.1024	0.1056	0.0788
wine	0.0160	0.0184	0.0156	0.0183
zoo	0.0632	0.0656	0.0632	0.0612
pneumonia	0.2452	0.2463	0.2451	0.2442
sepsis-d	0.1579	0.1581	0.1590	0.1501
sepsis-s	0.3077	0.3083	0.3030	0.3120
heart failure-d	0.0842	0.0849	0.0842	0.0839
heart failure-c	0.1882	0.1893	0.1884	0.1883
UCI average	0.2159	0.2159	0.2176	0.2129
medical average	0.1966	0.1974	0.1959	0.1957
overall average	0.2174	0.2207	0.2188	0.2145

Table 5-18: Mean CAL scores of different algorithms based on 10-fold cross-validation done twice on the UCI datasets and a single train-test validation on the medical datasets. To avoid cluttering only the mean for each dataset is shown. The bottom three rows give the average CAL scores for the UCI datasets, for the medical datasets and for all the datasets respectively. Best results are underlined.

Dataset	PSMBl	PSMBl-MS	NPSMBl	PSMBg
australian	0.0380	0.0436	0.0409	0.0470
breast-cancer	0.0139	0.0141	0.0147	0.0146
cleveland	0.0672	0.0775	0.0716	0.0497
corral	0.0255	0.0190	0.0240	0.0583
crx	0.0438	0.0473	0.0451	0.0452
diabetes	0.0451	0.0447	0.0391	0.0403
flare	0.0581	0.0600	0.0558	0.0551
german	0.0533	0.0576	0.0515	0.0684
glass2	0.0580	0.0466	0.0547	0.0359
glass	0.0274	0.0282	0.0274	0.0188
heart	0.0574	0.0625	0.0586	0.0498
hepatitis	0.0401	0.0445	0.0366	0.0422
iris	0.0139	0.0135	0.0140	0.0110
lymphography	0.0212	0.0264	0.0247	0.0226
pima	0.0496	0.0458	0.0542	0.0532
postoperative	0.0547	0.0482	0.0552	0.0404
sonar	0.0561	0.0650	0.0609	0.0437
vehicle	0.0300	0.0303	0.0306	0.0479
vote	0.0289	0.0322	0.0337	0.0247
wine	0.0085	0.0079	0.0078	0.0062
zoo	0.0058	0.0057	0.0058	0.0065
pneumonia	0.0994	0.0999	0.0995	0.0998
sepsis-d	0.0323	0.0314	0.0314	0.0353
sepsis-s	0.0608	0.0676	0.0519	0.0627
heart failure-d	0.0255	0.0264	0.0258	0.0263
heart failure-c	0.0530	0.0538	0.0531	0.0533
UCI average	0.0379	0.0391	0.0384	0.0372
medical average	0.0542	0.0558	0.0523	0.0555
overall average	0.0387	0.0400	0.0387	0.0382

Table 5-19: Wilcoxon paired-samples signed ranks test comparing the performance of PSMBl with other algorithms. For each performance measure the number on top is the Z statistic and the number at the bottom is the corresponding p-value. The Z statistic is negative when PSMBl has a lower score on a performance measure than the competing algorithm. On all measures except the AUC, a negative Z statistic indicates better performance by PSMBl; on the AUC a positive Z statistic indicates better performance by PSMBl. Underlined results indicate p-values of 0.05 or smaller.

Performance measure	PSMBl-MS	NPSMBl	PSMBg
Misclassification error	-2.122	-1.350	-0.695
	0.034	0.177	0.487
AUC	2.235	0.282	-2.581
	0.025	0.778	0.010
Logarithmic loss	-3.458	-0.766	-1.430
	0.001	0.444	0.153
Squared error	-3.469	-0.539	-1.120
	0.001	0.539	0.263
CAL score	-1.278	-0.243	-0.532
	0.201	0.808	0.594

Table 5-20: Paired-samples *t* test comparing the performance of PSMBl with other algorithms. For each performance measure the number on top is the mean difference between PSMBl and the indicated algorithm and the number at the bottom is the corresponding p-value. The mean difference is negative when PSMBl has a lower score on a performance measure than the competing algorithm. On all measures except the AUC, a negative mean difference indicates better performance by PSMBl; on the AUC a positive mean difference indicates better performance by PSMBl. Underlined results indicate p-values of 0.05 or smaller.

Performance measure	PSMBl-MS	NPSMBl	PSMBg
Misclassification error	-0.002	-0.001	0.004
	0.056	0.163	0.343
AUC	0.001	0.000	-0.010
	0.193	0.881	0.049
Logarithmic loss	-0.008	-0.005	0.012
	0.033	0.169	0.222
Squared error	-0.003	-0.001	0.002
	0.007	0.213	0.478
CAL score	-0.001	0.000	0.000
	0.233	0.971	0.860

Table 5-21: Approximate running times of the various algorithms. For each algorithm, the time shown is the average running time for a single test case over all the UCI and medical datasets. For both population-wide and the patient-specific algorithms the running time includes the time for learning the model and for doing inference for the target variable of the test case.

Algorithm	Average running time for a test case
Naïve Bayes	< 1 minute
Decision Tree (Classification Tree)	< 1 minute
Logistic Regression	< 1 minute
Neural Networks	< 1 minute
k-Nearest Neighbor	< 1 minute
Lazy Bayesian Rule	~ 1 minute
PSMBg	~ 1 hour 30 minutes
PSMBl	~ 5 hours 30 minutes

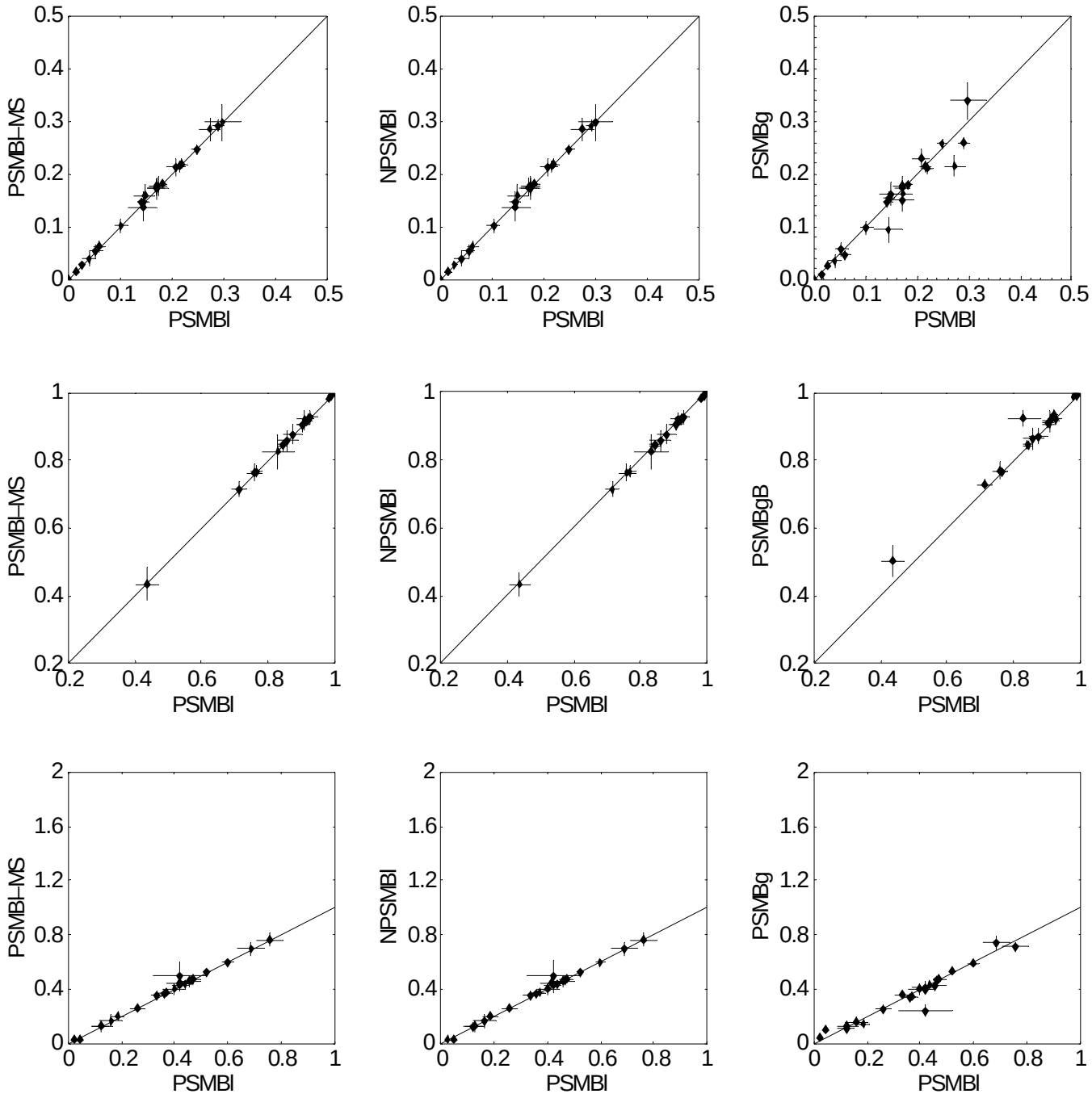


Figure 5-10: Pairwise plots of the mean misclassification errors (top row), mean AUCs (middle row) and mean logarithmic losses (bottom row) of PSMBl vs. competing algorithms. Each point represents the mean score of PSMBl and a competing algorithm on a single dataset, and the crosshairs represent one standard deviation on either side of the mean score. Points above the diagonal line represent better performance by PSMBl than the competing algorithm.

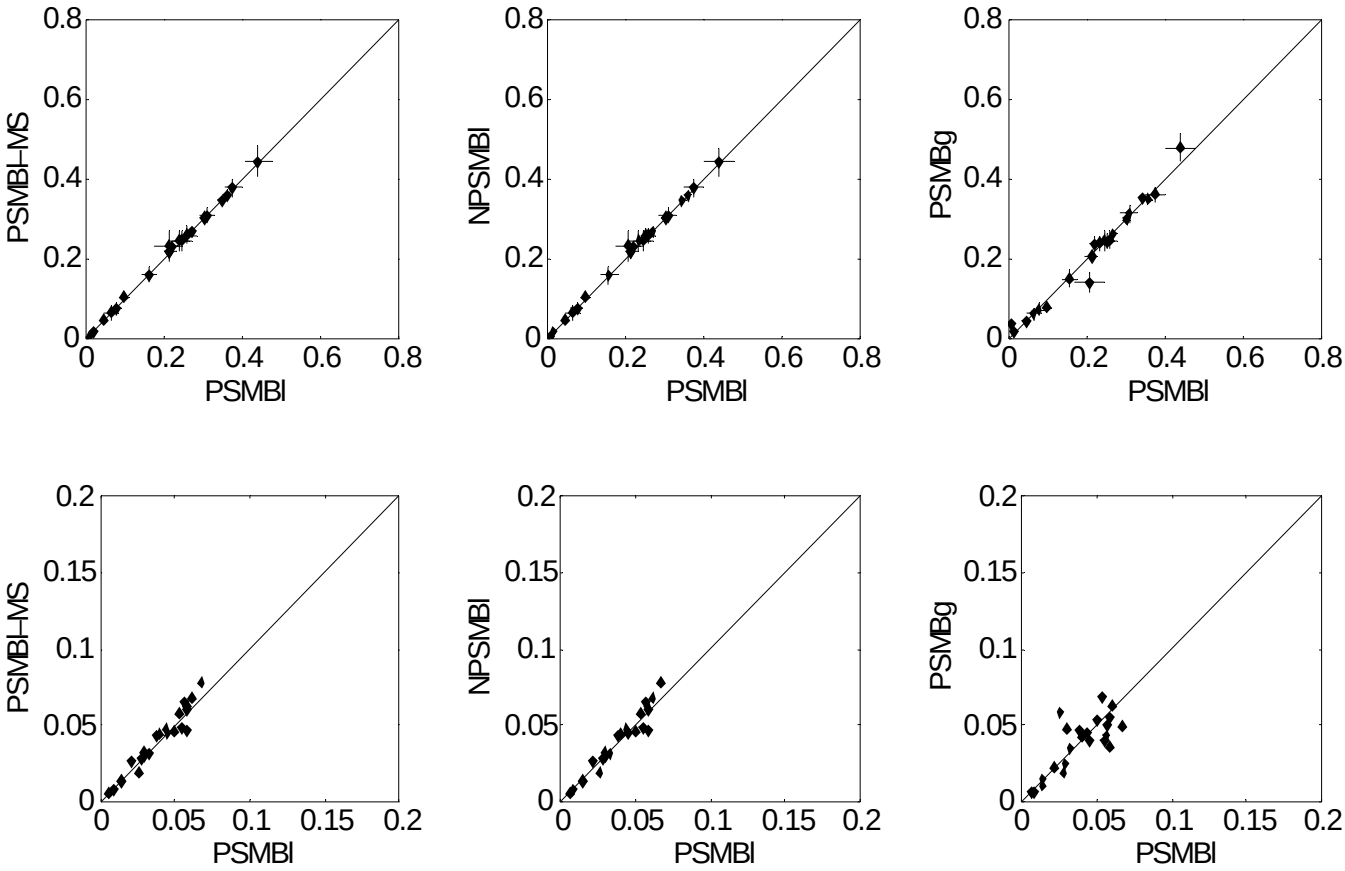


Figure 5-11: Pairwise plots of the mean squared errors (top row) and the mean CAL scores (bottom row) of PSMBl vs. competing algorithms. Each point represents the mean score of PSMBl and a competing algorithm on a single dataset, and the crosshairs represent one standard deviation on either side of the mean score. Points above the diagonal line represent better performance by PSMBl than the competing algorithm.

6.0 CONCLUSIONS

In this dissertation, I presented a new framework for learning predictive models that is characterized by the use of a patient-specific heuristic in selecting models for Bayesian model averaging (BMA). I implemented two basic algorithms using Bayesian network Markov blanket (MB) models and evaluated them extensively on several datasets. The patient-specific algorithms were able to better predict the target by improving on probabilistic, discriminative and calibration measures when compared to several commonly used machine learning methods. A summary of the findings is presented in the next section followed by some directions for future work in the last section.

6.1 CONTRIBUTIONS AND FINDINGS

This section summarizes the main contributions and the experimental results of the research presented in this dissertation.

This dissertation described the development and evaluation of a new approach for learning predictive models that are relevant to a single patient case. The new patient-specific methods that were developed use Bayesian network models, carry out selective Bayesian model averaging to predict the outcome of interest for the patient case at hand, and employ a patient-specific heuristic to locate a set of suitable models to average over. The main contribution is the

development of a new search heuristic for identifying models over which to perform Bayesian model averaging that is guided by the features of the patient case at hand. This heuristic was implemented in two algorithms, namely, the patient-specific Markov blanket (global) (PSMBg) and the patient-specific Markov blanket (local) (PSMBl) algorithms. Both algorithms employ the patient-specific heuristic to select models in the space of MB structures but differ in the model space. The difference between the model spaces of the two algorithms lies in the representation used for the CPDs: the PSMBg algorithm uses conditional probability tables (CPTs) that capture explicitly only the *global* structure while the PSMBl algorithm uses decision graphs that capture explicitly *global and local* structure. Given a set of parents, a node in a MB structure has only one global structure represented by a CPT but has several distinct local structures represented by decision graphs. Since the global structure is equivalent to one of the possible local structures, the model space of the PSMBg algorithm is a subset of that of the PSMBl algorithm.

A second contribution is the development of new operators to traverse the space of MB structures; these operators are modifications of operators used in existing techniques for learning general Bayesian network structures.

A third contribution is the development of a new hierarchical structure prior for BNs and MBs with local decision graph structures that penalizes complex decision graph structures over simpler ones. This structure prior is used in computing the Bayesian score that is used by the PSMBl algorithm.

The experimental results demonstrate that the PSMBg algorithm improves prediction of the target variable on a variety of performance measures when compared to several population-wide predictive algorithms. The greatest improvements occur in logarithmic loss and squared error, followed by good improvement in calibration and smaller improvements in

misclassification error and the AUC. Bayesian model averaging has better performance than Bayesian model selection, and within model averaging, patient-specific Bayesian model averaging has better performance than non-patient-specific Bayesian model averaging though the improvement is not as large as that of model averaging over model selection.

The PSMBl algorithm also improves prediction of the target variable when compared to several population-wide predictive algorithms. However, it did not significantly improve on the performance of the PSMBg algorithm on any measure.

Overall, Bayesian model averaging in conjunction with patient-specific search led to better performance than either non-patient-specific Bayesian model averaging or patient-specific search for a single good model. However, the use of local structure did not lead to significant improvements over the use of global structure alone. The implementation of patient-specific phase 2 in the PSMBl algorithm may have limited the performance of local structures.

6.2 DISCUSSION

This section summarizes some of the key aspects of the patient-specific method. The essence of the patient-specific method lies in the model score used in phase 2 of the search. This score is sensitive to both the posterior probability of the model and the predicted distribution for the outcome variable of the patient case at hand (see Equations 4.17 and 4.16). Typically, methods that evaluate models with a score employ a score that is sensitive only to the fit of the model to the training data and not to the prediction of the outcome variable.

Several situations are possible where the patient-specific method has no advantage over a population-wide method. As one example, in a domain where complete BMA is tractable and

model averaging is carried out over all models in the model space, a search heuristic that selects a subset of models such as the one used by the patient-specific method is superfluous. Typically, in real life domains complete BMA over all models is not tractable due to the enormous number of models in the model space. Thus, the patient-specific method is useful for selective model averaging where it identifies a potentially relevant set of models that is predictive of the patient case at hand. As another example, in a domain where features that are relevant are commonly present, selection of relevant variables may not be a problem. In such a situation, the variables selected by a population-wide method are likely to be relevant for predicting any future case and the patient-specific method that performs model selection will likely select the same set of variables for each new case.

There are several open questions regarding the behavior of the patient-specific method. Characterizing the bias of the selective model averaged prediction of the patient-specific method is an open problem. In contrast, the bias of selective BMA over models that are chosen randomly is low. However, the variance of selective BMA over models that are chosen randomly is likely to be much larger than the variance of selective BMA over models chosen by the patient-specific method which is constrained to prefer models that are good fit to the training data.

Another open issue is the comparison of the performance of patient-specific selective BMA to that of other ensemble methods. Ensemble methods construct a set of predictive models that predict the target variable for a new case by taking a weighted average of their predictions. In addition to BMA, examples of other ensemble methods include bagging, boosting and stacking [107, 108]. Boosting, in particular, has been shown to improve on the performance of a single model for classification and for predictive tasks. Boosting selects a new model by weighting more those training cases that have been misclassified by the models selected

previously. The patient-specific method described in this dissertation selects a new model that disagrees considerably with the predictions of models selected previously. Thus, both methods select a set of models that are diverse with respect to different heuristics.

6.3 FUTURE WORK

The experimental work presented in this research is intended as a first step in exploring the utility of the patient-specific framework. Several extensions and directions for future work are feasible.

Efficient computation of phase 2 score. In the current implementation of the PSMBl algorithm, the inner search procedure in phase 2 search evaluates candidate local structures with the non-patient-specific phase 1 score rather than the patient-specific phase 2 score. Only the local decision graph structure that has the highest phase 1 score is evaluated with the phase 2 score. Ideally, the patient-specific phase 2 score should be computed for every candidate local structure that is encountered by the inner search procedure. In the current implementation, computation of the phase 2 score is less efficient than the computation of the phase 1 score. More efficient implementations of the phase 2 score with caching of intermediate computational results will enable the scoring of all local structures.

Unified search. The encapsulated search strategy employed by the PSMBl algorithm decouples the problem of network structure learning (global structure) from that of learning the CPD structures (local structure). However, this strategy typically leads to considerable duplication of effort. This arises due to the repetitive characteristic of the inner search procedure: the search for local structure of the CPD is restarted with every addition of a new parent. Often,

the new parent is irrelevant and will be discarded when the local structure is learned. Unified search employs a single search procedure with operators that modify explicitly only the local structure, since a set of local structures is sufficient to determine the global structure. Unified search will typically learn the local structure of a MB node only once and is potentially more efficient. However, the downside of unified search is that operations like reversing an arc in the global structure are not well defined.

Alternative dissimilarity metric for phase 2 scores. The computation of the phase 2 score (See Equation 4.17) requires a dissimilarity metric to compare the predictive distributions of the target variable in candidate MB structures. The current implementation of the PSMB algorithms use KL divergence as the dissimilarity metric. The experimental results indicate that KL divergence optimizes most logarithmic loss since the largest improvement in performance is observed on this measure. Alternative dissimilarity metrics may optimize other performance measures. The following dissimilarity metric, for example, has the potential for optimizing misclassification error:

$$f(M, M') = \left| \max P(Z^t \mid \boldsymbol{X}, M) - \max P(Z^t \mid \boldsymbol{X}, M') \right|,$$

where the phase 2 score, $f(M, M')$, for the candidate model *M'* is the absolute value of the difference in the maximum probabilities achieved in the distributions for the target Z^t estimated by the current model *M* and the candidate model *M'*.

Alternative models. The patient-specific framework is a general formulation and any statistical model can be substituted for MB structures. Alternative models that would be interesting to explore include logistic regression models and decision trees.

Comparison with other ensemble techniques. Several non-Bayesian ensemble techniques such as bagging, boosting and stacking have been shown to improve on the

performance of a single model for classification and for predictive tasks. Further comparisons between patient-specific selective Bayesian model averaging and these non-Bayesian ensemble techniques would be worthwhile.

APPENDIX

COUNTING MARKOV BLANKET STRUCTURES

This section gives the derivation of a formula for counting the number of possible Markov blanket (MB) structures with respect to a specified domain variable.

The MB of a node in a Bayesian network (BN) consists of its parents, its children, and the parents of its children (spouses). With respect to a MB, the nodes can be categorized into five groups: (1) the target node, (2) parent nodes of the target, (3) child nodes of the target, (4) spousal nodes, which are parent nodes of the children, and (5) other nodes, which are not part of the MB. The node under consideration is called the *target* node. A *parent* node is one that has an outgoing arc to the target node and may have additional outgoing arcs to one or more child nodes. A *child* node is one that has an incoming arc from the target node, may have additional incoming arcs from parent nodes, spousal nodes and other child nodes, and may have outgoing arcs to other child nodes. A *spousal* node is one that has outgoing arcs to one or more child nodes and has neither an incoming arc from the target node nor an outgoing arc to the target node. An *other* node is one that is not in the MB and is considered to be a potential spousal node. An example demonstrating the various types of nodes in a MB is given in Figure A-1.

The number of possible Markov blanket structures for a domain with m variables (where m excludes the target variable) is given by the following equation:

$$MB(m) = \sum_{m_P=0}^{m} \sum_{m_{so}=0}^{m-m_p} \left(\frac{m!}{m_p! m_c! m_{so}!} \right) 2^{m_c \cdot m_p} 2^{m_c \cdot m_{so}} BN(m_c) \quad for\ m > 0 \quad , \quad \text{(A.1)}$$
$$MB(0) = 1$$

where, m_p is number of parent nodes, m_c is the number of child nodes, m_{so} is the number of spousal and other nodes, and $m = m_p + m_c + m_{so}$. $BN(m_c)$ is the number of DAGs that can be constructed from m_c nodes. The number of DAGs that can be constructed from n variables is given by the following recurrence formula [73, 74]:

$$BN(n) = \sum_{k=1}^{n} (-1)^{k-1} C(n,k) 2^{k(n-k)} BN(n-k) \quad for\ n > 0 \quad , \quad \text{(A.2)}$$
$$BN(0) = 1$$

where $C(n, k)$ is the count of the number ways to choose k objects from n distinct objects.

Equation A.1 is derived as follows. The terms inside the double summation count the number of MB structures for a specified number of m_p, m_c and m_{so} nodes. The first term gives the number of ways m can be partitioned into m_p parent nodes, m_c child nodes and m_{so} spousal and other nodes. The second term gives the number of distinct MB structures that differ only in the

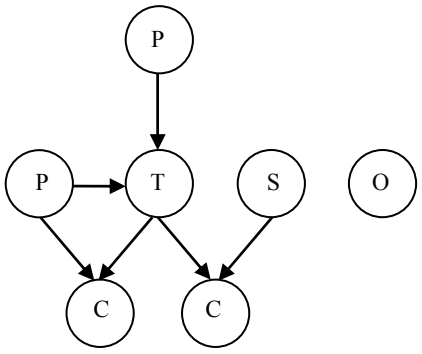


Figure A-1: An example of a Markov blanket demonstrating various node types. T is the *target* node, P is a *parent* node, C is a *child* node, S is a *spousal* node, and O is an *other* node.

presence or absence of arcs from parent nodes to child nodes. Each parent node can have an arc to none, one or more child nodes for a total of 2^{m_c} distinct MB structures. For m_p parent nodes, the number of distinct MB structures that differ only in the presence or absence of arcs from parent nodes to child nodes is $2^{m_c m_p}$. The third term gives the number of distinct MB structures that differ only in the presence or absence of arcs from spousal and other nodes to child nodes. This derivation is similar to the derivation of the previous term. Each spousal or other node can have an arc to none, one or more child nodes for a total of $2^{m_{so}}$ distinct MB structures. For m_{so} spousal or other nodes, the number of distinct MB structures that differ only in the presence or absence of arcs from spousal or other nodes to child nodes is $2^{m_c m_{so}}$. The fourth and last term gives the number of DAGs that can be formed with m_c child nodes. The summation is carried over all possible values of m_p and m_c; selection of particular values for m_p and m_c determines the value of m_{so} and hence no explicit summation is required over the values of m_{so}.

BIBLIOGRAPHY

1. van Bemmel, J.H. and M.A. Musen, *Handbook of Medical Informatics*. 1st ed. 1997, New York: Springer-Verlag.
2. Abu-Hanna, A. and P.J. Lucas, *Prognostic models in medicine. AI and statistical approaches.* Methods Inf Med, 2001. **40**(1): p. 1-5.
3. Cooper, G.F., et al., *An evaluation of machine-learning methods for predicting pneumonia mortality.* Artif Intell Med, 1997. **9**(2): p. 107-38.
4. Gottrup, C., et al., *Applying instance-based techniques to prediction of final outcome in acute stroke.* Artificial Intelligence in Medicine, 2005. **33**(3): p. 223-236.
5. Pearl, J., *Probabilistic Reasoning in Intelligent Systems*. 1988, San Mateo, California: Morgan Kaufmann.
6. Neapolitan, R.E., *Learning Bayesian Networks*. 1st ed. 2003, Upper Saddle River, New Jersey: Prentice Hall.
7. Russell, S.J. and P. Norvig, *Artificial Intelligence: Modern Approach*. 2002: Prentice Hall.
8. Coiera, E., *The Guide to Health Informatics*. 2nd ed. 2003, London: Hodder Arnold.
9. Wyatt, J.C. and D.G. Altman, *Commentary: Prognostic models: Clinically useful or quickly forgotten?* BMJ, 1995. **311**: p. 539-1541.
10. Bleich, H.L., *Computer evaluation of acid-base disorders.* Trans Assoc Am Physicians, 1968. **81**: p. 184-9.
11. Buchanan, B.G. and E.H. Shortliffe, *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. 1984, Reading, MA: Addison-Wesley Publishing Company.
12. Ledley, R.S. and L.B. Lusted, *Reasoning foundations of medical diagnosis; symbolic logic, probability, and value theory aid our understanding of how physicians reason.* Science, 1959. **130**(3366): p. 9-21.
13. Heckerman, D.E., E.J. Horvitz, and B.N. Nathwani, *Toward normative expert systems: Part I. The Pathfinder project.* Methods Inf Med, 1992. **31**(2): p. 90-105.
14. Heckerman, D.E. and B.N. Nathwani, *Toward normative expert systems: Part II. Probability-based representations for efficient knowledge acquisition and inference.* Methods Inf Med, 1992. **31**(2): p. 106-16.
15. Middleton, B., et al., *Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base. II. Evaluation of diagnostic performance.* Methods Inf Med, 1991. **30**(4): p. 256-67.
16. Shwe, M.A., et al., *Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base. I. The probabilistic model and inference algorithms.* Methods Inf Med, 1991. **30**(4): p. 241-55.

17. Dreiseitl, S. and L. Ohno-Machado, *Logistic regression and artificial neural network classification models: a methodology review.* J Biomed Inform, 2002. **35**(5-6): p. 352-9.
18. Wasserman, L., *Bayesian Model Selection and Model Averaging.* J Math Psychol, 2000. **44**(1): p. 92-107.
19. Hoeting, J.A., et al., *Bayesian model averaging: A tutorial.* Statistical Science, 1999. **14**(4): p. 382-401.
20. Quinlan, R. *Combining instance-based and model-based learning*. in *Proceedings of the Tenth International Conference on Machine Learning*. 1993. University of Massachusetts, Amherst: Morgan Kaufmann.
21. Ali, K. and M.J. Pazzani, eds. *Classification using Bayes averaging of multiple, relational rule-based models*. Learning from Data: Artificial Intelligence and Statistics, Vol. 5, ed. D. Fisher and H. Lenz. 1995, Springer-Verlag.
22. Mitchell, T.M., *Machine Learning*. 1st ed. 1997, New York: McGraw Hill.
23. Aha, D.W., *Feature weighting for lazy learning algorithms*, in *Feature extraction, construction and selection: A data mining perspective*, L. Huan and M. Hiroshi, Editors. 1998, Kluwer Academic Publisher: Norwell, MA. p. 13-32.
24. Zhang, J.P., Y.S. Yim, and J.M. Yang, *Intelligent selection of instances for prediction functions in lazy learning algorithms.* Artificial Intelligence Review, 1997. **11**(1-5): p. 175-191.
25. Cover, T. and P. Hart, *Nearest neighbor pattern classification.* IEEE Transactions on Information Theory, 1967. **13**(1): p. 21-27.
26. Dasarathy, B., *Nearest Neighbor (NN) norms: NN Pattern Classification Techniques*. 1991, Los Alamitos, California: IEEE Computer Society Press.
27. Atkeson, C.G., A.W. Moore, and S. Schaal, *Locally weighted learning.* Artificial Intelligence Review, 1997. **11**(1-5): p. 11-73.
28. Wettschereck, D., D.W. Aha, and T. Mohri, *A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms.* Artificial Intelligence Review, 1997. **11**(1-5): p. 273-314.
29. Kohavi, R. and G.H. John, *Wrappers for feature subset selection.* Artificial Intelligence, 1997. **97**(1-2): p. 273-324.
30. Friedman, J.H., R. Kohavi, and Y. Yun. *Lazy decision trees*. in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*. 1996. Portland, Oregon: AAAI Press.
31. Ting, K.M., Z. Zheng, and G.I. Webb. *Learning lazy rules to improve the performance of classifiers*. in *Proceedings of the Nineteenth SGES International Conference on Knowledge Based Systems and Applied Artificial Intelligence*. 1999. Cambridge, UK: Springer-Verlag.
32. Zheng, Z.J. and G.I. Webb, *Lazy learning of Bayesian rules.* Machine Learning, 2000. **41**(1): p. 53-84.
33. Kohavi, R. *Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid*. in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. 1996. Portland, Oregon: AAAI Press.
34. Pazzani, M.J., *Constructive induction of cartesian product attributes*, in *Feature Extraction, Construction and Selection: A Data Mining Perspective*, L. Huan and M. Hiroshi, Editors. 1998, Kluwer Academic Publisher: Norwell, MA.

35. Pazzani, M. *Searching for dependencies in Bayesian classifiers*. in *Proceedings of the Fifth International Workshop on Artificial Intelli-gence and Statistics*. 1995. Fort Lauderdale, Florida: Springer-Verlag.
36. Knaus, W.A., et al., *APACHE II: a severity of disease classification system.* Crit Care Med, 1985. **13**(10): p. 818-29.
37. Anbeek, P., et al., *Automatic segmentation of different-sized white matter lesions by voxel probability estimation.* Med Image Anal, 2004. **8**(3): p. 205-15.
38. Nutt, C.L., et al., *Gene expression-based classification of malignant gliomas correlates better with survival than histological classification.* Cancer Res, 2003. **63**(7): p. 1602-7.
39. Chang, R.F., et al., *Support vector machines for diagnosis of breast tumors on US images.* Acad Radiol, 2003. **10**(2): p. 189-97.
40. Lee, Y. and C.K. Lee, *Classification of multiple cancer types by multicategory support vector machines using gene expression data.* Bioinformatics, 2003. **19**(9): p. 1132-9.
41. Cristianini, N. and J. Shawe-Taylor, *An Introduction to Support Vector Machines (and other kernel-based learning methods)*. 1st ed. 2000, Cambridge, UK: Cambridge University Press.
42. Qu, H. and J. Gotman, *A patient-specific algorithm for the detection of seizure onset in long-term EEG monitoring: possible use as a warning device.* IEEE Trans Biomed Eng, 1997. **44**(2): p. 115-22.
43. Shoeb, A., et al., *Patient-specific seizure onset detection.* Epilepsy Behav, 2004. **5**(4): p. 483-98.
44. Sheiner, L.B., et al., *Forecasting individual pharmacokinetics.* Clin Pharmacol Ther, 1979. **26**(3): p. 294-305.
45. Sheiner, L.B. and S.L. Beal, *Bayesian individualization of pharmacokinetics: simple implementation and comparison with non-Bayesian methods.* J Pharm Sci, 1982. **71**(12): p. 1344-8.
46. Sheiner, L.B., et al., *Improved computer-assisted digoxin therapy. A method using feedback of measured serum digoxin concentrations.* Ann Intern Med, 1975. **82**(5): p. 619-27.
47. Dreiseitl, S. and M. Binder, *Do physicians value decision support? A look at the effect of decision support systems on physician opinion.* Artificial Intelligence in Medicine, 2005. **33**(1): p. 25-30.
48. Cooper, G.F., *An Overview of the Representation and Discovery of Causal Relationships using Bayesian Networks*, in *Computation, Causation, and Discovery*, C. Glymour and G.F. Cooper, Editors. 1999, AAAI Press and MIT Press: Menlo Park, CA.
49. Lauritzen, S.L., et al., *Independence properties of directed Markov fields.* Networks, 1990. **20**: p. 491-505.
50. Lauritzen, S.L., *Graphical Models*. 1996, Oxford: Clarendon Press.
51. Friedman, N. and M. Goldszmidt, *Learning Bayesian networks with local structure*, in *Learning in graphical models*. 1999, MIT Press. p. 421-459.
52. Chickering, D.M., D. Heckerman, and C. Meek. *A Bayesian approach to learning Bayesian networks with local structure*. in *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*. 1997. Brown University, Providence, Rhode Island: Morgan Kaufmann.

53. Chickering, D.M., *Learning Bayesian networks is NP-complete*, in *Learning from Data: Artificial Intelligence and Statistics V*, D. Fisher and H. Lenz, Editors. 1996, Springer. p. 121-130.
54. Spiegelhalter, D.J. and S.L. Lauritzen, *Sequential updating of conditional probabilities on directed graphical structures.* Networks, 1990. **20**: p. 579-605.
55. Cooper, G.F. and E. Herskovits, *A Bayesian method for the induction of probabilistic networks from data.* Machine Learning, 1992. **9**(4): p. 309-347.
56. Heckerman, D., D. Geiger, and D.M. Chickering, *Learning Bayesian networks - the combination of knowledge and statistical data.* Machine Learning, 1995. **20**(3): p. 197-243.
57. Heckerman, D., *A Tutorial on Learning with Bayesian Networks*, in *Learning in Graphical Models*, M. Jordan, Editor. 1999, MIT Press: Cambridge, MA.
58. Cooper, G.F. and E. Herskovits. *A Bayesian method for constructing Bayesian belief networks from databases*. in *Seventh Annual Conference on Uncertainty in Artifical Intelligence*. 1991. Los Angeles, California: Morgan Kaufmann Publishers.
59. Buntine, W., *Theory refinement on Bayesian networks* in *Proceedings of the seventh conference (1991) on Uncertainty in artificial intelligence* 1991 Morgan Kaufmann Publishers Inc.: Los Angeles, California, United States p. 52-60
60. Chow, C.K. and C.N. Liu, *Approximating discrete probability distributions with dependence trees.* IEEE Transactions on Information Theory 1968. **14**(3): p. 462-467.
61. Chickering, D.M. *Large-sample learning of Bayesian networks is hard*. in *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*. 2003. Acapulco, Mexico: Morgan Kaufmann.
62. Teyssier , M. and D. Koller. *Ordering-based Search: A Simple and Effective Algorithm for Learning Bayesian Networks*. in *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence*. 2005. Edinburgh, Scotland: AUAI Press.
63. Glover, F., *Tabu search, Part I.* ORSA Journal on Computing, 1989. **1**: p. 190-206.
64. Koivisto, M. *Advances in exact Bayesian structure discovery in Bayesian networks*. in *Proceedings of the 22nd Conference in Uncertainty in Artificial Intelligence*. 2006. Cambridge, MA: AUAI Press.
65. Koivisto, M. and K. Sood, *Exact Bayesian structure discovery in Bayesian networks.* Journal of Machine Learning Research, 2004. **5**: p. 549–573.
66. Bennett, P.N., *Assessing the Calibration of Naive Bayes' Posterior Estimates*. 2000, Carnegie Mellon University: Pittsburgh, PA.
67. Warner, H.R., et al., *A mathematical approach to medical diagnosis. Application to congenital heart disease.* Jama, 1961. **177**: p. 177-83.
68. Friedman, N., D. Geiger, and M. Goldszmidt, *Bayesian Network Classifiers.* Machine Learning, 1997. **29**(2-3): p. 131-163.
69. Greiner, R., et al., *Structural Extension to Logistic Regression: Discriminative Parameter Learning of Belief Net Classifiers.* Machine Learning, 2005. **59**(3): p. 297-322.
70. Grossman, D. and P. Domingos. *Learning Bayesian network classifiers by maximizing conditional likelihood*. in *International Conference on Machine Learning*. 2004. Banff, Canada.
71. Madigan, D. and A.E. Raftery, *Model Selection and Accounting for Model Uncertainty in Graphical Models using Occam's Window.* Journal of the American Statistical Association, 1994. **89**: p. 1335-1346.

72. Raftery, A.E., D. Madigan, and J.A. Hoeting, *Model Selection and Accounting for Model Uncertainty in Linear Regression Models.* Journal of the American Statistical Association, 1997(92): p. 179-191.
73. Robinson, R.W., *Counting Labeled Acyclic Digraphs*, in *New Directions in Graph Theory*, F. Harary, Editor. 1973, Academic Press: New York.
74. Harary, F. and E.M. Palmer, *Graphical Enumeration*. 1973, New York: Academic Press.
75. Boutilier, C., et al. *Context-specific independence in Bayesian networks*. in *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence*. 1996. Reed College, Portland, Oregon: Morgan Kaufmann.
76. Friedman, N. and M. Goldszmidt. *Learning Bayesian networks with local structure*. in *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence*. 1996. Reed College, Portland, Oregon: Morgan Kaufmann.
77. Bell, E.T., *Exponential Numbers.* American Mathematical Monthly, 1934. **41**: p. 411-419.
78. Graham, R.L., D.E. Knuth, and P. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*. 2 ed. 1994: Addison-Wesley.
79. Cover, T.M. and A.T. Joy, *Elements of Information Theory*. 2nd ed. 2006: Wiley-Interscience.
80. Blake, C.L. and C.J. Merz, *UCI Repository of machine learning databases*. 1998, University of California, Department of Information and Computer Science: Irvine, CA.
81. USDHHS, *Health United States 1988, DHHS Pub. No. (CDC) 89-8411*. 1988, Public Health Service, Centers for Disease Control: Washington, DC.
82. USGPO, *Statistical Abstract of the United States, 108th ed.* 1988, U.S. Dept. of Commerce, Bureau of the Census.
83. Fine, M.J., et al., *A prediction rule to identify low-risk patients with community-acquired pneumonia.* N Engl J Med, 1997. **336**(4): p. 243-50.
84. Cooper, G.F., et al., *Predicting dire outcomes of patients with community acquired pneumonia.* Journal of Biomedical Informatics, 2005. **38**(5): p. 347-366.
85. Wheeler, A.P. and G.R. Bernard, *Treating patients with severe sepsis.* N Engl J Med, 1999. **340**(3): p. 207-14.
86. *From the Centers for Disease Control. Increase in National Hospital Discharge Survey rates for septicemia--United States, 1979-1987.* JAMA, 1990. **263**(7): p. 937-8.
87. Schulman, K.A., et al., *Cost-effectiveness of HA-1A monoclonal antibody for gram-negative sepsis. Economic assessment of a new therapeutic agent.* Jama, 1991. **266**(24): p. 3466-71.
88. Barriere, S.L. and S.F. Lowry, *An overview of mortality risk prediction in sepsis.* Crit Care Med, 1995. **23**(2): p. 376-93.
89. NHLBI, *Morbidity and Mortality: 2002 Chartbook on Cardiovascular, Lung, and Blood Diseases*. 2002, National Institutes of Health.
90. Popovich, J.R. and M.J. Hall, *1999 National Hospital Discharge Survey, Advance Data, No. 319*. 2001, Centers for Disease Control and Prevention, National Center for Health Statistics.
91. Hsieh, M., T.E. Auble, and D.M. Yealy, *Evidence-based emergency medicine. Predicting the future: can this patient with acute congestive heart failure be safely discharged from the emergency department?* Ann Emerg Med, 2002. **39**(2): p. 181-9.

92. Auble, T.E., et al., *A prediction rule to identify low-risk patients with heart failure.* Acad Emerg Med, 2005. **12**(6): p. 514-21.
93. Caruana, R. *A non-parametric EM-style algorithm for imputing missing values*. in *Proceedings of Artificial Intelligence and Statistics 2001*. 2001.
94. Fayyad, U.M. and K.B. Irani. *Multi-interval discretization of continuous-valued attributes for classification*. in *Proceedings of the International Joint Conference on Artificial Intelligence*. 1993. Chambry, France: Morgan Kaufmann.
95. Adams, N.M. and D.J. Hand, *Comparing classifiers when the misallocation costs are uncertain.* Pattern Recognition, 1999. **32**: p. 1139-1147.
96. Caruana, R. and N.-M. Alexandru, *Data mining in metric space: an empirical analysis of supervised learning performance criteria*. Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. 2004, Seattle, WA, USA: ACM Press. 69-78.
97. Hanley, J.A. and B.J. McNeil, *The meaning and use of the area under a receiver operating characteristic (ROC) curve.* Radiology, 1982. **143**(1): p. 29-36.
98. Hand, D.J. and R.J. Till, *A simple generalisation of the area under the ROC curve for multiple class classification problems.* Machine Learning, 2001. **45**(2): p. 171-186.
99. Mossman, D., *Three-way ROCs.* Medical Decision Making 1999. **19**: p. 78–89.
100. Provost, F. and P. Domingos, *Tree induction for probability-based rankings.* Machine Learning, 2003. **52**(3): p. 199 - 215
101. Brier, G.W., *Verification of forecasts expressed in terms of probability.* Monthly Weather Review, 1950. **78**: p. 1-3.
102. Yeung, K.Y., R.E. Bumgarner, and A.E. Raftery, *Bayesian model averaging: development of an improved multi-class, gene selection and classification tool for microarray data.* Bioinformatics, 2005. **21**(10): p. 2394-402.
103. Witten, I.H. and E. Frank, *Data Mining: Practical machine learning tools with Java implementations*. 2000, San Francisco: Morgan Kaufmann.
104. Witten, I.H. and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. 2nd ed. 2005: Morgan Kaufmann.
105. DeGroot, M.H. and S.E. Fienberg, *The comparison and evaluation of forecasters.* The Statistician, 1983. **32**: p. 14-22.
106. Daniel, W., *Applied Nonparametric Statistics*. 2nd ed. 1990: PWS-KENT Publishing Company.
107. Schapire, R.E. *A brief introduction to boosting*. in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. 1999. Stockholm, Sweden: Morgan Kaufmann.
108. Breiman, L., *Bagging predictors.* Machine Learning, 1996. **24**(2): p. 123--140.