# Exact model averaging with naive Bayesian classifiers

**Denver Dash** DDASH@SIS.PITT.EDU
Decision Systems Laboratory, Intelligent Systems Program, University of Pittsburgh, Pittsburgh, PA 15213 USA

**Gregory F. Cooper** GFC@CBMI.UPMC.EDU
Center for Biomedical Informatics, University of Pittsburgh, Pittsburgh, PA 15213 USA

## Abstract

The naive classifier is a well-established mathematical model whose simplicity, speed and accuracy have made it a popular choice for classification in AI and engineering. In this paper we show that, given $N$ features of interest, it is possible to perform tractable exact model averaging (MA) over all $2^N$ possible feature-set models. In fact, we show that it is possible to calculate parameters for a single naive classifier $C^*$ such that $C^*$ produces predictions equivalent to those obtained by the full model-averaging, and we show that $C^*$ can be constructed using the same time and space complexity required to construct a single naive classifier with MAP parameters. We present experimental results which show that on average the MA classifier typically outperforms the MAP classifier on simulated data, and we characterize how the relative performance varies with number of variables, number of training records, and complexity of the generating distribution. Finally, we examine the performance of the MA naive model on the real-world ALARM and HEPAR networks and show MA improved classification here as well.

## 1. Introduction

The general supervised classification problem seeks to create a model based on labelled data which can be used to classify future vectors of features $F = \{F_1, F_2, \ldots, F_N\}$ into one of various classes of interest. The naive classifier is a probabilistic model that accomplishes this goal by making the assumption that any feature $F_i \in F$ is conditionally independent of any other feature $F_j \in F$ given the value of the class variable $C$. The naive model can be represented by the Bayes net shown in Figure 1.

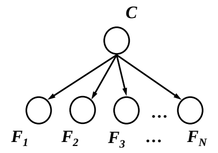

*Figure 1.* A naive network: $C$ is the class node which can take on one value for each possible class, and the $F_i$ denote features of interest.

Naive classifiers have several desirable features: First, they are simple to construct, requiring very little domain background knowledge, as opposed to general Bayesian networks which can require numerous intensive sessions with experts to produce the true dependence structure between features. Second, naive networks have very constrained space and time complexity: If $N_f$ is the largest number of states any feature can take on, $N_c$ is the number of class states, and $N$ is the total number of features, then constructing a network requires the estimation of a set of $O(N \cdot N_f \cdot N_c)$ parameters. Each of which can be estimated from data in time $O(N_r)$, where $N_r$ is the number of records in the database. Given a completely specified naive network, classification of a new feature vector $F'$ can be performed in time $O(|F'|)$, even if $F'$ is an incomplete instantiation of features. Finally, despite their naivety, these classifiers have been shown to perform surprisingly well in practice, for example in (Domingos & Pazzani, 1997; Friedman et al., 1997), comparing favorably to state-of-the-art neural network classifiers and even to the much more complex learning algorithms for general Bayes nets.

The construction of a single maximum *a posteriori* (MAP) naive classifier given a set $F$ of attributes requires only two general steps: (1) Select the subset of features $F' \subseteq F$ judged to be relevant to classification, and (2) Calculate the set $\hat{\boldsymbol{\theta}}$ of MAP parameters. The feature selection problem (1) is a difficult and central problem in machine learning in general. In terms of naive classifiers, the selection of the appropriate subset $F'$ has shown to be both important to classification and non-trivial to perform in practice (Langley & Sage, 1994; Kohavi & John, 1997; Friedman et al., 1997). Obviously eliminating features that do not bear on the classification is important, but also important is the ability to minimize redundant features.

In this paper we take a strict Bayesian approach to feature selection and, rather than finding a single "good" set $F'$, we consider the problem of model averaging predictions over all possible feature-set structures. For example, the four possible structures of a naive classifier with two features are $\{C \rightarrow F_1,\ C \rightarrow F_2\}$, $\{C \rightarrow F_1\}$, $\{C \rightarrow F_2\}$, and $\emptyset$.

If there exist $N$ total possible features then the number of possible naive structures is $2^N$; nonetheless, we show that the exact averaging over all structures can be performed in time linear in $N$ after construction of a single naive classifier $C^*$ bearing an appropriate set of parameters. Furthermore, we show that it is possible to construct $C^*$ in the same time and using the same space required to calculate the MAP parameters of a single naive network over all $N$ features.

Approximate techniques for model averaging with graphical models using pruning and stochastic Monte Carlo techniques have been previously studied by Madigan and Raftery (1994). Because of their generality, these approaches are both computationally expensive and not exact. Our approach, on the other hand is specific to averaging over the space of naive networks; however the solution is exact and efficient.

Friedman and Koller (2000) studied the ability to estimate structural features of a network (for example the probability of an arc from $X_i$ to $X_j$) by performing a MCMC search over orderings of nodes. The inner-loop of their method calculated in closed-form the posterior probability of a feature averaged over all networks consistent with a fixed ordering, and their ability to average over a given ordering relied on a decomposition of the posterior that is very similar in form to one that we use. Their work differs from ours in two key respects: (1) They perform model averaging only to calculate the probabilities of structural features, explicitly not for classification, and (2) their more general approach does not capture the simple single-network (and thus linear-time calculation efficiency) solution to the naive model-averaging problem.

Similarly Meila and Jaakkola (2000) discuss the ability to perform exact model averaging over all trees. Again they use similar assumptions and similar decompositions that we use; however they also do not specialize their technique to naive networks and thus do not capture the linear-time classification efficiency. They do discuss the ability to build a single ensemble model to represent the average over all trees; however for this task they rely on the EM algorithm to estimate parameters of the ensemble model, a restriction which is not required for our solution and one which will not in general reproduce the exact calculation.

Our primary contributions in this paper are as follows: (1) we demonstrate a factorization that can be applied to the specialized task of classification while model averaging over the restricted space of naive Bayes nets, (2) we show that the exact model-averaged calculations can be performed with a single naive network which can be constructed efficiently, and (3) we provide extensive experimental investigations showing that model averaging under these conditions produces more accurate classifications than the MAP naive model.

In Section 2 we formally frame the problem and state our assumptions and notation. In Section 3 we derive the MA solution and show that the MA predictions are equivalent to those of a single naive structure bearing a particular set of parameters. In Section 4 we present the experimental results of applying the technique to synthetic and benchmark networks and show that on average the MA predictions almost always perform better than the MAP predictions. Finally, in Section 5 we present our conclusions and future directions.

## 2. Assumptions and Notation

The general supervised classification problem can be framed as follows: Given a set of features $F = \{F_1, F_2, \ldots, F_N\}$ and a set of classes $\mathcal{C} = \{C_1, C_2, \ldots C_{N_c}\}$, letting $\mathcal{F}_i$ denote the range of the $i$th feature, $F_i \in \mathcal{F}_i$ and letting $\mathcal{F}$ denote the joint space of features $\mathcal{F} = \mathcal{F}_1 \otimes \mathcal{F}_2 \otimes \ldots \otimes \mathcal{F}_N$, a classification model $M$ can be defined as a mapping from $\mathcal{F}$ to the set of classes: $M : \mathcal{F} \rightarrow \mathcal{C}$. A labelled database is a set $D = \{D_1, D_2, \ldots, D_R\}$ containing records $D_i = \{f_1^i, f_2^i, \ldots f_N^i, C^i\}$, where $C^i \in \mathcal{C}$ denotes the class into which the feature vector $\{F_1 = f_1^i, F_2 = f_2^i, \ldots F_N = f_N^i\}$ belongs. In these terms the supervised classification problem can be stated succinctly: Given a labelled database $D$, construct a classification model $M_D$.

A Bayes net plus a set of threshold parameters can be used to define a probabilistic classification model. For example, when $C$ is binary, given a Bayes net $B$ and a threshold parameter $t \in [0,1]$, a Bayes net classification model $M_B$ can be defined as follows:

$$M_B(\boldsymbol{f}) = \begin{cases} C_0 & \text{if } P(C = C_0 \mid \boldsymbol{F} = \boldsymbol{f}, B) > t \\ C_1 & \text{otherwise} \end{cases} \tag{1}$$

We associate the nodes in the network with the index $i$, using the convention that $i = 0$ refers to the class node and that $i \neq 0$ refers to feature $F_i$. We may use the notation $X_i$ to refer to the nodes when we do not care to distinguish between the class node and the feature nodes; thus, $X_i \equiv F_i$ and $X_0 \equiv C$. We use $\boldsymbol{X}$ to denote the collective set of nodes in the network, and we use $P_i$ to denote the parent set of $X_i$; for our naive networks $P_i \in \{\emptyset, \{C\}\}$.

**Assumption 1 (Multinomial variables)** *We assume that each node $X_i$ is a discrete variable with $r_i$ possible states.*

We let $q_i$ denote the number of columns in the conditional probability table (CPT) of node $X_i$. In the naive case, if $P_i = \emptyset$ then $q_i = 1$, and if $P_i = \{C\}$ then $q_i = N_c$. In general when $Q_i$ is some quantity associated with node $X_i$, we use $Q_i^{\emptyset}$ to denote $Q_i$ in the graph where $P_i = \emptyset$ and $Q_i^C$ to denote $Q_i$ in the graph for which $P_i = \{C\}$; thus $q_i^{\emptyset} = 1$ and $q_i^C = N_c$.

We use $\theta_{ijk}$ to denote the $(jk)$ component of node $X_i$'s CPT, the symbol $\boldsymbol{\theta_{ij}}$ to denote the entire probability distribution function for the $i$-th node and the $j$-th column, and the symbol $\boldsymbol{\theta}$ to denote the collective parameters of the network. In general we use the common $(ijk)$ coordinates notation to identify the *k-th* state and the *j-th* column of the *i-th* node in the network. We use the shorthand that if $Q_{ijk}$ is some quantity associated with coordinates $(ijk)$, then $Q_{ij} \equiv \sum_k Q_{ijk}$.

**Assumption 2 (Complete Labelled Training Data)** *The training data set $D$ contains no record $D_l \in D$ such that $D_l$ is missing data: $D_l \in D \Rightarrow |D_l| = N + 1$.*

We take this assumption initially but will show how to relax it in Section 5. We let $N_{ijk}$ denote the number of times in the database that the node $X_i$ achieved state $k$ when $P_i$ was in the $j$-th configuration.

**Assumption 3 (Dirichlet priors)** *The prior beliefs over parameter values are given by a Dirichlet distribution.*

We let $\alpha_{ijk}$ denote the Dirichlet hyperparameters corresponding to the network parameter $\theta_{ijk}$. We assume the existence of both complete sets of hyperparameters $\alpha_{ijk}^{\emptyset}$ and $\alpha_{ijk}^{C}$.

**Assumption 4 (Parameter independence)** *For any given network structure $S$, each probability distribution $\boldsymbol{\theta_{ij}}$ is independent of any other probability distribution $\boldsymbol{\theta_{i'j'}}$:*

$$P(\boldsymbol{\theta} \mid S) = \prod_{i=0}^{N} \prod_{j=1}^{q_i} P(\boldsymbol{\theta_{ij}} \mid S) \tag{2}$$

**Assumption 5 (Structure modularity)** *We assume that the prior over structures, $P(S)$, can be factorized according to the network:*

$$P(S) \propto \prod_{i=0}^{N} p_s(X_i, P_i), \tag{3}$$

*where $p_s(X_i, P_i)$ is some function that depends only on attributes of $X_i$ and $P_i$.*

This assumption was used in (Meila & Jaakkola, 2000) and (Friedman & Koller, 2000) the latter introducing the name "structure modularity". We assume that $p_s(X_i, P_i)$ can be calculated in time constant in $N$ and $N_r$. For example, any metric that scores $P(S \mid K)$ based on a difference in arcs between $S$ and some prior network (as suggested by (Heckerman et al., 1995)) will satisfy this condition. Another possibility is the uniform distribution which is obviously $O(1)$.

## 3. Theoretical Results

In this section we show how to efficiently calculate the quantity $P(\boldsymbol{X} = \boldsymbol{x} \mid D)$ averaged over all possible naive structures. From this quantity we can readily calculate $P(C_i \mid \boldsymbol{f}, B)$ in $O(N_c)$ time using $P(C_i \mid \boldsymbol{f}, B) = P(C_i, \boldsymbol{f} \mid B) / \sum_j P(C_j, \boldsymbol{f} \mid B)$.

### 3.1 Fixed Network Structure

For a fixed network structure $S$ and a fixed set of network parameters $\boldsymbol{\theta}$, the quantity $P(\boldsymbol{X} = \boldsymbol{x} \mid S, \boldsymbol{\theta})$ can be calculated in $O$(N) time:

$$P(\boldsymbol{X} = \boldsymbol{x} \mid S, \boldsymbol{\theta}) = \prod_{i=0}^{N} \theta_{iJK}, \tag{4}$$

where all $(j,k)$ coordinates are fixed by the configuration of $\boldsymbol{X}$ to the value $(j,k) = (J,K)$.

When, rather than a fixed set of parameters, a database $D$ is given, from an ideal Bayesian perspective it is necessary to average over all possible configurations of the parameters $\boldsymbol{\theta}$:

$$\begin{aligned} P(\boldsymbol{X}=\boldsymbol{x} \mid S,D) &= \int P(\boldsymbol{X}=\boldsymbol{x} \mid S,\boldsymbol{\theta}) \cdot P(\boldsymbol{\theta} \mid S,D) \cdot d\boldsymbol{\theta} \\ &= \int \prod_{i=0}^{N} \theta_{iJK} \cdot P(\boldsymbol{\theta} \mid S,D) \cdot d\boldsymbol{\theta} \end{aligned}$$

where the second line follows from Equation 4. Given the assumption of parameter independence and Dirichlet priors, this quantity can be written just in terms of sufficient statistics and Dirichlet hyperparameters (Cooper & Herskovits, 1992; Heckerman et al., 1995):

$$P(\boldsymbol{X}=\boldsymbol{x} \mid S,D) = \prod_{i=0}^{N} \frac{\alpha_{iJK}+N_{iJK}}{\alpha_{iJ}+N_{iJ}} \qquad (5)$$

Comparing this result to Equation 4 illustrates the well-known result that a single network with a fixed set of parameters $\hat{\boldsymbol{\theta}}$ given by

$$\hat{\theta}_{ijk} = \frac{\alpha_{ijk}+N_{ijk}}{\alpha_{ij}+N_{ij}} \qquad (6)$$

will produce predictions equivalent to those obtained by averaging over all parameter configurations. Heckerman (1998) makes the claim that under a suitable coordinate transformation these parameters represent a maximum *a posteriori* (MAP) configuration.

## 3.2 Averaging over Structure

For the case when the structure $S$ is not fixed, we must not only integrate over all sets of parameters, but also average over all possible network structures:

$$\begin{aligned} &P(\boldsymbol{X}=\boldsymbol{x} \mid D) \\ &= \sum_S \int P(\boldsymbol{X}=\boldsymbol{x} \mid S,\boldsymbol{\theta}) \cdot P(\boldsymbol{\theta} \mid S,D) \cdot P(S \mid D) \cdot d\boldsymbol{\theta} \\ &= \sum_S \prod_{i=0}^{N} \hat{\theta}_{iJK} \cdot P(S \mid D) \end{aligned}$$

which according to Bayes' rule can be written as:

$$P(\boldsymbol{X}=\boldsymbol{x} \mid D) = \kappa \sum_S \prod_{i=0}^{N} \hat{\theta}_{iJK} \cdot P(D \mid S) \cdot P(S) \qquad (7)$$

where $\kappa$ is a constant depending only on the data set $D$.

Given the assumptions of complete data, multinomial variables, Dirichlet priors and parameter independence, the marginal likelihood $P(D \mid S)$ can also be written just in terms of hyperparameters and sufficient statistics (Cooper & Herskovits, 1992; Heckerman et al., 1995):

$$P(D \mid S) = \prod_{i=0}^{N} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+N_{ij})} \cdot \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk}+N_{ijk})}{\Gamma(\alpha_{ijk})}. \qquad (8)$$

Combining Equations 7 and 8 with Assumption 5 yields:

$$P(\boldsymbol{X}=\boldsymbol{x} \mid D) = \kappa \sum_S \prod_{i=0}^{N} \rho_{iJK} \qquad (9)$$

where the $\rho_{iJK}$ functions are given by:

$$\rho_{iJK} = \hat{\theta}_{iJK} \cdot p_s(X_i, P_i) \cdot \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+N_{ij})} \cdot \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk}+N_{ijk})}{\Gamma(\alpha_{ijk})} \qquad (10)$$

Equation 9 includes a summation over $2^N$ network structures. Expanding the summation and using our notation $\rho^{\emptyset}_{ijk}$ and $\rho^{C}_{ijk}$ to denote $\rho_{ijk}$ when $P_i = \emptyset$ and $P_i = \{C\}$, respectively, yields:

$$P(\boldsymbol{X}=\boldsymbol{x} \mid D) \propto \left.\begin{array}{ll} & \rho_{0JK} \cdot \rho^{\emptyset}_{1JK} \cdot \rho^{\emptyset}_{2JK} \cdot \dots \cdot \rho^{\emptyset}_{NJK} \\ + & \rho_{0JK} \cdot \rho^{C}_{1JK} \cdot \rho^{\emptyset}_{2JK} \cdot \dots \cdot \rho^{\emptyset}_{NJK} \\ \vdots & \qquad \vdots \\ + & \rho_{0JK} \cdot \rho^{C}_{1JK} \cdot \rho^{C}_{2JK} \cdot \dots \cdot \rho^{C}_{NJK} \end{array}\right\} 2^N \text{ terms.}$$

We define the symbol $\Sigma^{JK}_m$ to denote the structure sum of the product up to and including the $m$-th node:

$$\begin{aligned} \Sigma^{JK}_m \equiv\ & \rho_{0JK} \cdot \rho^{\emptyset}_{1JK} \cdot \rho^{\emptyset}_{2JK} \cdot \dots \cdot \rho^{\emptyset}_{mJK} \\ +\ & \rho_{0JK} \cdot \rho^{C}_{1JK} \cdot \rho^{\emptyset}_{2JK} \cdot \dots \cdot \rho^{\emptyset}_{mJK} \\ \vdots\ & \vdots \\ +\ & \rho_{0JK} \cdot \rho^{C}_{1JK} \cdot \rho^{C}_{2JK} \cdot \dots \cdot \rho^{C}_{mJK} \end{aligned}$$

Using this notation, $\Sigma^{JK}_N$ be written in terms of $\Sigma^{JK}_{N-1}$:

$$\begin{aligned} \Sigma^{JK}_N &= \Sigma^{JK}_{N-1} \cdot \rho^{\emptyset}_{NJK} \\ &+ \Sigma^{JK}_{N-1} \cdot \rho^{C}_{NJK} \end{aligned}$$

which yields the recurrence relations:

$$\Sigma^{jk}_i = \Sigma^{jk}_{i-1} \cdot (\rho^{\emptyset}_{ijk} + \rho^{C}_{ijk}), \qquad \Sigma^{jk}_0 = \rho_{0jk}$$

Finally, expanding out the recurrence relation yields the expression for $P(\boldsymbol{X}=\boldsymbol{x} \mid D)$:

$$P(\boldsymbol{X}=\boldsymbol{x} \mid D) = \kappa \prod_{i=0}^{N} (\rho^{\emptyset}_{iJK} + \rho^{C}_{iJK}) \qquad (11)$$

Once the $\{\rho^{\emptyset}_{iJK}, \rho^{C}_{iJK}\}$ terms are calculated, the product in Equation 11 can be performed in $O(N)$ time.

Calculating the $\rho$ terms themselves requires the calculation of hyperparameters $\{\alpha^{\emptyset}_{iJK}, \alpha^{C}_{iJK}\}$, which are assumed given, and statistics $\{N^{\emptyset}_{iJK}, N^{C}_{iJK}\}$, which can be calculated in $O(N_r)$ time. To calculate the complete set for all $(j,k)$ combinations thus requires $O(N_r \cdot N \cdot N_f \cdot N_c)$ time and $O(N \cdot N_f \cdot N_c)$ space, the same as that required for calculating the parameters of a single MAP naive network over all $N$ features using Equation 6.

The final observation we make about Equation 11 is that, on comparison with Equation 4, it can be seen that a single naive network over all $N$ features bearing parameters $\boldsymbol{\theta}^*$ defined by:

$$\theta^*_{ijk} = \frac{1}{\kappa^N}(\rho^{\emptyset}_{ijk} + \rho^{C}_{ijk}) \tag{12}$$

will produce predictions equivalent to those produced by model averaging over all naive structures. These $\boldsymbol{\theta}^*$ parameters can be calculated without direct evaluation of the constant $\kappa$ by using the normalization condition on parameters: $\sum_k \theta_{ijk} = 1$. Thus, rather than performing the calculation of the $\rho_{iJK}$ for each case $\boldsymbol{X} = \boldsymbol{x}$ to be classified, we need only construct a single naive model and use standard $O(N)$ Bayes net inference for each case. This fact was verified empirically for $N = 10$ by comparing model-averaged predictions using brute-force enumeration to those produced by the single network with the $\boldsymbol{\theta}^*$ parametrization. The predictions were found to be identical.

By inspecting the definition of the $\rho$ functions in Equation 10, the $\theta^*_{ijk}$ parameters in Equation 12 can be written as:

$$\theta^*_{ijk} \;\propto\; \hat{\theta}^{\emptyset}_{ijk} \cdot f(\emptyset \mid D) \;+\; \hat{\theta}^{C}_{ijk} \cdot f(C \mid D) \tag{13}$$

where $f(\emptyset \mid D)$ and $f(C \mid D)$ are proportional to the local posterior probability of $P_i = \emptyset$ and $P_i = \{C\}$, respectively. Thus Equation 12 represents a structure-based smoothing of MAP parameters where each MAP parameter is weighted based on how likely an arc from $C$ to $F_i$ is, given the data and the priors.

## 4. Experimental Tests

In this section we describe several experimental investigations that were meant to test the performance of the MA technique described in Section 3.

### 4.1 Experimental Setup

Our experiments were aimed at characterizing the performance of the model-averaged naive predictions versus the MAP naive predictions. We sought trends over three experimental parameters: the number of nodes $N$, the number of records $N_r$, and the maximum in-degree $N_k$ (maximum number of parents of any one node) of the (not necessarily naive) generating network.

For the initial set of experiments, networks were generated randomly from a uniform distribution over directed acyclic graphs. Specifically, the network generation process was performed using the following procedure:

**Procedure 1 (Random structure generation)**
**Given:** $N$ and $N_k$.
*Do:*

```
1. Create $N+1$ nodes $X_1, X_2, \ldots, X_{N+1}$.
2. For each node $X_i$ do:
   (a) Let $N_{pmax} \equiv \min(i-1, N_k)$.
   (b) Generate a random integer $N_{pa} \in [0, N_{pmax}]$.
   (c) Randomly pick $N_{pa}$ parents uniformly from the list $\{X_1, \ldots, X_{i-1}\}$
```

Once a network structure had been generated, each node distribution $\boldsymbol{\theta_{ij}}$ was sampled independently from a uniform distribution over parameters. However, as we were generating data with extremely dense networks (looking at maximum in-degrees up to 500), it was necessary to employ a lazy-evaluation approach whereby the individual $\boldsymbol{\theta_{ij}}$ columns were only generated if and when they were actually required during the data generation process. Finally, once the generating network was constructed, we designated a class node from $\{X_1 \ldots X_{N+1}\}$ uniformly at random.

We use $M$ and $\widehat{M}$ to denote the naive and the MA naive models, respectively. Using this network generation process, three general tests were performed. The inner-loop of each test performed basically the same procedure: Given the number of nodes $N$, number of training records $N_r$, number of testing records $N_{test}$, maximum density of generating graphs $N_k$ and a total number of trials $N_{trials}$, we did the following:

**Procedure 2 (Basic testing loop)**
**Given:** $N$, $N_r$, $N_{test}$, $N_k$ and $N_{trials}$.
*Do:*

```
1. Generate $N_{trials}$ random graphs $G(N, N_k)$.
2. For each graph $G(N, N_k)$ do:
   (a) Generate $N_r$ training records and $N_{test}$ test records.
   (b) Train $M$ and $\widehat{M}$ on the training records.
   (c) Test $M$ and $\widehat{M}$ on the test data, measuring the ROC areas $R$ and $\widehat{R}$ of each.
```

(d) *Calculate the quantities* $\Delta_{ROC} = \frac{\widehat{R}-R}{R}$ *and* $\delta_{ROC} = \frac{\widehat{R}-R}{T-R}$, *where* $T$ *is the ROC area of a perfect classifier.*

3. *Average* $\Delta_{ROC}$ *and* $\delta_{ROC}$ *over all* $N_{trials}$.

$\Delta_{ROC}$ represents the fraction increase in ROC area achieved by MA, and $\delta_{ROC}$ represents the fraction of remaining ROC area covered by MA. The possible range of $\Delta_{ROC}$ depends on the value of $R$, for example if $R = 0.99T$ then $\Delta_{ROC} \le 0.01$; whereas $\delta_{ROC}$ always can take on values in the range $-\infty \le \delta_{ROC} \le 1$. In our experimental results we present these quantities as percentages. To avoid a singularity in $\delta_{ROC}$, we use the convention that if $R = T$ then $\delta_{ROC} = 0$. This was an exceedingly rare occurrence, and almost always when this did occur it was the case that $\widehat{R} = T$ as well.

The tests that we performed differed in which parameters were fixed and which were varied (see Section 4.2 for specific values):

1. **(Nodes test)** $N_r$, $N_{test}$, $N_k$, and $N_{trials}$ were fixed to constant values , and $N$ was systematically varied to test the sensitivity of MA performance as the number of nodes in the network is varied.

2. **(Training-records test)** $N$, $N_{test}$, $N_k$, and $N_{trials}$ were fixed to constant values, and $N_r$ was systematically varied to test the sensitivity on the number of training records.

3. **(Density test)** $N$, $N_r$, $N_{test}$, and $N_{trials}$ were fixed to constant values, and $N_k$ was systematically varied test the sensitivity of MA performance on the complexity of the generating distribution.

In all cases we assume a uniform prior over structures and thus allow $p_s(X_i, P_i) = 1$ for all $i$. We also adopted the K2 metric (Cooper & Herskovits, 1992) which sets $\alpha_{ijk} = 1$ for all $(i, j, k)$. This criterion has the property of weighting all distributions of parameters equally. All variables in our tests were binary: $N_c = N_f = 2$.

## 4.2 Results

The nodes test was performed with $N_r$ fixed to 200 records, $N_{test}$ fixed to 100 records, $N_k$ fixed to 5 parents, $N_{trials}$ set to 500, and $N$ systematically varied through the values $\{5, 10, 20, 40, 80, 100, 150, 200, 500\}$. The mean values of $\Delta_{ROC}$ and $\delta_{ROC}$ for this test are shown in Figure 2 (a). In these and all subsequent figures, the small solid error bars denote the 99% confidence interval of the mean value. The dashed bars indicate the $\pm 1\sigma$ level and are shown to give a feel for the spread of the data independent of the measurement error of the mean.

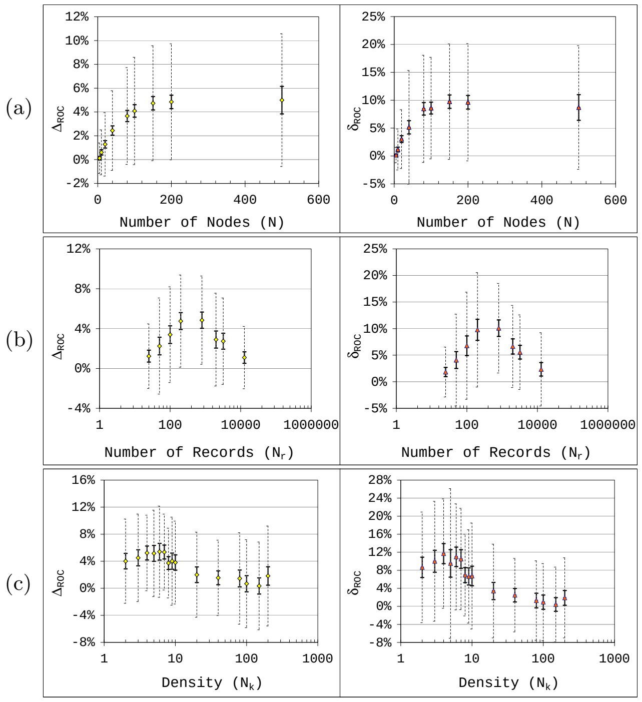

*Figure 2.* Results as a function of (a) the number of nodes, (b) the number of records, and (c) the generating density. The solid error bars denote the 99% confidence intervals for the mean. The dashed bars indicate the spread of the data ($\pm 1\sigma$).

In the nodes test, both $\Delta_{ROC}$ and $\delta_{ROC}$ were greater than 0 with probability $p > 0.99$ for all values of $N$ save one, which was significant at the 0.95 level. Model averaging performed increasingly better as the number of nodes increased, converging to about $\Delta_{ROC} = 5\%$, $\delta_{ROC} = 10\%$ for $N > 150$ nodes. Due to the large standard deviation of the results, about 70% ($2\sigma$) of the probability mass is concentrated in the region from $-1\% \lesssim \Delta_{ROC} \lesssim +11\%$ and $-2\% \lesssim \delta_{ROC} \lesssim +20\%$ for $N \gtrsim 100$.

The training-records test was performed with $N$ fixed to 200, $N_{test}$ fixed to 100 records, $N_k$ fixed to 5 parents, $N_{trials}$ set to 200, and $N_r$ varied through the values $\{25, 50, 100, 200, 800, 2000, 3200, 12800\}$. The outcomes of this test are shown in Figure 2 (b). The mean values were statistically significant at the 99% level for all tested values of $N_r$; however, unlike the nodes test where the results seemed to converge as the number of nodes increased, here we observed a local maximum at $\Delta_{ROC} \simeq 5\%$, $\delta_{ROC} \simeq 10\%$ somewhere between 200 and 800 records. Again, due to the large standard deviation, $\sim 70\%$ of the results are generally concentrated between $-4\% \lesssim \Delta_{ROC} \lesssim +9\%$, $-5\% \lesssim \delta_{ROC} \lesssim +15\%$.

The density test was performed with $N$ fixed to 200 nodes, $N_r$ fixed to 200 records, $N_{test}$ fixed to 100 records, $N_{trials}$ set to 500, and $N_k$ systematically varied through the values $\{1-10, 20, 40, 60, 80, 100, 150, 200\}$. The outcomes of this test are shown in Figure 2 (c). For sparse graphs ($N_k < 10$), again with probability $> 99\%$, the results are favorable to model averaging with means around $\Delta_{ROC}$ =4–6% and $\delta_{ROC}$ =4–8% with most of the probability lying between $-2\% \lesssim \Delta_{ROC} \lesssim +12\%$, $-4\% \lesssim \delta_{ROC} \lesssim +20\%$. For very dense graphs the mean drops by a factor of about 2 or 3, and some $\Delta_{ROC}$ points are no longer statistically significant.

To test the robustness of these results, we performed a worst-case training-records test with the number of nodes set to $N = 5$ and the density set to $N_k = 5$, both chosen to be the least optimal values for MA (based on the previous experimental results reported above). This run used the same parameters as the training-records test in Figure 2 (b) only with $N = 5$ and $N_{trials} = 1000$. As shown in Figure 3, with this extreme test the MA classifier showed little if any significant improvement over the MAP naive classifier; conversely, even as all parameters achieved the least optimal values for our measurement range, the single MAP naive model never performed significantly better on average than the MA classifier.

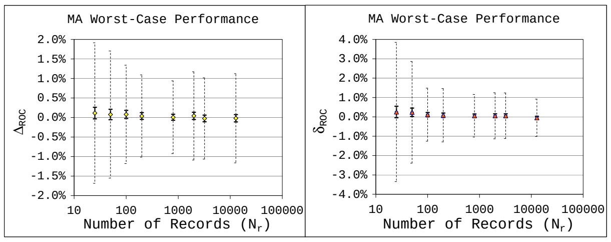

*Figure 3.* The worst-case configuration for model averaging still does not produce statistically significant gains for the single MAP naive model.

The performance of model averaging was also tested by generating training and test data with the benchmark ALARM network. In this case, $N = 36$ and $N_k = 4$ were fixed by the network and $N_{test}$ was set to 1000 records, $N_{trials}$ was set to 100 and $N_r$ was systematically varied in a manner similar to the previous training-records tests. In this network there are several nodes which could realistically be considered the class nodes because they represent diagnostic conditions. The results in Figure 4 are shown for the *anaphylaxis* and *hypovolemia* nodes. These two nodes were

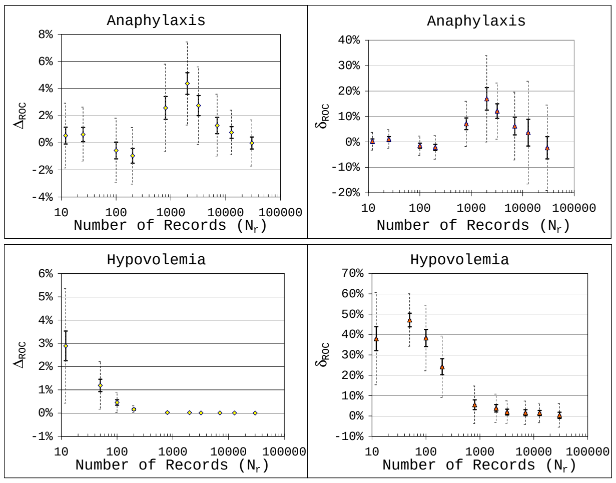

*Figure 4.* Results for the ALARM network classifying on the anaphylaxis and hypovolemia nodes.

selected for the diversity of their results, the differences indicating that the MA performance depends on local features of the class node as well as on general features of the generating network itself. Although these results are consistent with our randomly-generated results, there is a general difference in how MA responds to varying the number of training records for the two different nodes. There exist many data points for both nodes that did not achieve the 99% significance level, and two points in the anaphylaxis test where the MAP naive model performed statistically better at the 95% level than MA, indicating that although our promise of increased ROC area holds for the average network, the topology is important and can have a bearing on that result.

Finally, the performance was tested on the HEPAR network (Oniśko et al., 2000), another real-world network of $N = 70$ nodes and $N_k = 6$ that diagnoses liver disorders. In this case, $N_{test}$ was set to 500 records and $N_{trials}$ was set to 200. The results are shown in Figure 5 for the *toxic hepatitis* and *hyperbilirubinemia* nodes. Typically MA achieves gains at the 99% confidence level, but not always. Again, there are qualitative differences between classifications based on the different nodes despite the fact that they belong to the same network.

## 5. Discussion

We have shown that an alternative parametrization of the MAP naive network model will produce a naive model $C^*$ whose predictions are identical to those pro-

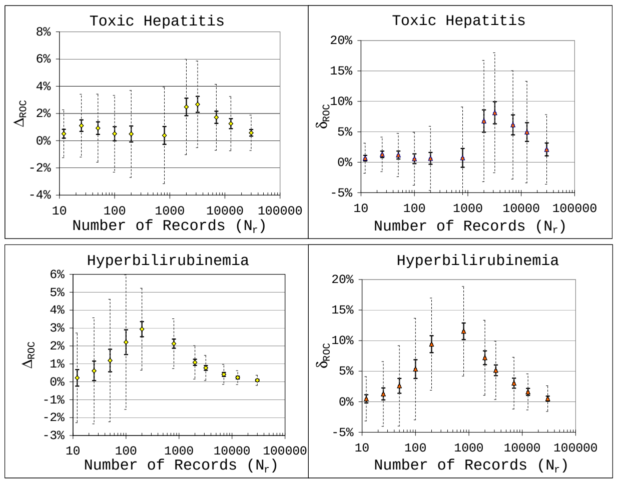

*Figure 5.* Results for the HEPAR network classifying on the toxic hepatitis and hyperbilirubinemia nodes.

duced by exact model averaging over the space of naive networks with a fixed class node. We have shown that calculating the $C^*$ parameters requires the two complete sets of priors and sufficient statistics: $\{\alpha^{\emptyset}_{ijk}, \alpha^{C}_{ijk}\}$ and $\{N^{\emptyset}_{ijk}, N^{C}_{ijk}\}$, respectively. These correspond to the priors and statistics for a naive network with no arcs and one with all arcs, respectively. Once these quantities have been calculated each parameter in $C^*$ can be calculated in constant time. This observation makes it apparent that our assumptions of both complete and labelled data can be relaxed by using the EM algorithm to calculate the expected sufficient statistics $\hat{N}^{\emptyset}_{ijk}$ and $\hat{N}^{C}_{ijk}$. These statistics can then be used without further ado to calculate the expected parameters of $C^*$ in constant time.

Our MA technique is especially interesting because of its simplicity. Any classification system that currently employs a naive classifier could trivially be adapted to use model-averaging by simply recalculating the parameters of the naive network. We have demonstrated through empirical studies that this simple adjustment very often leads to an increase in ROC area over the MAP naive model; specifically how much of an improvement is dependent on features of the training set, and both local and global properties of the generating network.

Possible topics of future investigation include identifying more general classes of networks for which exact model averaging would be possible, perhaps even applying the technique together with a stochastic search to perform more efficient approximate model averaging for the general class of DAGs.

## 6. Acknowledgements

Some of the experimental work in this paper was performed by Denver Dash while visiting the Machine Learning and Perception group at Microsoft Research, Cambridge, UK. This work was also supported in part by grant number S99-GSRP-085 from the National Aeronautics and Space Administration under the Graduate Students Research Program, by grant IIS-9812021 from the National Science Foundation and by grants LM06696 and LM06759 from the National Library of Medicine.

## References

Cooper, G. F., & Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, *9*, 309–347.

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

Friedman, N., Geiger, D., Goldszmidt, M., Provan, G., Langley, P., , & Smyth, P. (1997). Bayesian network classifiers. *Machine Learning*, *29*, 131.

Friedman, N., & Koller, D. (2000). Being Bayesian about network structure. *Uncertainty in Artificial Intelligence: Proceedings of the Sixteenth Conference (UAI-2000)* (pp. 201–210). San Francisco, CA: Morgan Kaufmann Publishers.

Heckerman, D. (1998). A tutorial on learning with Bayesian networks. In M. I. Jordan (Ed.), *Learning in graphical models*. Cambridge, Massachusetts: The MIT Press.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, *20*, 197–243.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, *97*, 273–324.

Langley, P., & Sage, S. (1994). Induction of selective Bayesian classifiers. *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-94)* (pp. 399–406). San Francisco, CA: Morgan Kaufmann Publishers.

Madigan, D., & Raftery, E. (1994). Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association*, *89*, 1535–1546.

Meila, M., & Jaakkola, T. S. (2000). Tractable Bayesian learning of tree belief networks. *Uncertainty in Artificial Intelligence: Proceedings of the Sixteenth Conference (UAI-2000)* (pp. 380–388). San Francisco, CA: Morgan Kaufmann Publishers.

Oniśko, A., Druzdzel, M. J., & Wasyluk, H. (2000). Extension of the Hepar II model to multiple-disorder diagnosis. *Intelligent Information Systems,* Advances in Soft Computing *Series* (pp. 303–313). Heidelberg: Physica-Verlag (A Springer-Verlag Company).