UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

LEARNING ENSEMBLED DYNAMIC BAYESIAN NETWORKS

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

SCOTT HELLMAN
Norman, Oklahoma
2012

LEARNING ENSEMBLED DYNAMIC BAYESIAN NETWORKS

A THESIS APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY

Acknowledgments

I am no meteorologist, and this thesis deals extensively with meteorological domains. David John Gagne II was my very patient source for answers to all things meteorological, and also provided ample help with the SSEF dataset used in Chapter 4. Branden Katona was also a source of help for meteorological confusion. Thanks to both, as their help was instrumental in generating meaningful empirical results.

Similarly, thanks to John Williams and Dave Ahijevych at the National Center for Atmospheric Research (NCAR) for providing the Reflectivity dataset used in Chapter 5, and for their help in working with the dataset and interpreting the results.

The process of writing this thesis was greatly aided by the distraction of terrible jokes, which Bradley Pirtle was kind enough to provide during the long evenings in the lab. And on a similar note, thanks to Joshua Southerland for the lunchtime conversations that gave me a reprieve from writing. Thanks to Tristin Forbus for his encouragement, and for his remarkable tolerance of my rambling 2 am emails.

And finally, thanks to my advisor, Amy McGovern, for her guidance and support throughout the process of researching and writing this thesis.

This material is based upon work supported by the National Science Foundation under Grant No. EPS-0919466.

Some of the data for this project was provided by the Federal Aviation Administration (FAA). Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

Table of Contents

List Of Tables

List Of Figures

Abstract

We introduce Ensembled Dynamic Bayesian Networks (EDBN), an ensemble approach to learning salient dependence relationships and to predicting values for continuous temporal data. By training individual Bayesian networks on both a subset of the data (bagging) and a subset of the attributes in the data (randomization), EDBN produces models for continuous domains that can be used to identify important variables in a dataset and to identify relationships between those variables. We use linear Gaussian distributions within our ensembles, allowing EDBN to handle continuous data while providing efficient network-level inference. By ensembling these networks, we are able to represent nonlinear relationships. EDBN can also be used to explore the properties of a domain, both by counting how frequently two variables are found to be dependent upon one another when creating networks, and by estimating the importance of each variable for prediction.

We empirically demonstrate the effectiveness of EDBN on two meteorological domains. The first is the Storm Scale Ensemble Forecast (SSEF), which contains multiple meteorological model forecasts. We show that EDBN can be used to combine these forecasts, resulting in rainfall prediction that is better than the mean prediction. In the second domain, we demonstrate EDBN's utility for storm prediction, with empirical results showing that EDBN achieves better prediction results than the model prediction and persistence prediction.

Chapter 1

Introduction

Random Forests, introduced by Breiman (2001), have been shown to perform well in a variety of settings, such as genetics (Diaz-Uriarte and Alvarez de Andres 2006), astrophysics (Albert et al. 2008), and land use classification (Chan and Paelinckx 2008). Recently, the Random Forest methodology has been successfully applied to decision trees that are not traditional C4.5 trees (Supinie et al. 2009; McGovern et al. 2010) and to discrete Bayesian Networks (Utz 2010). Inspired by this success, this thesis develops ensembles of Bayesian Networks that can handle continuous, temporal data, and that can be used to investigate the dependence relationships that exist within a domain. Because a major benefit of a Bayesian Network is its human-readability, and ensembles are inherently difficult for humans to interpret, we also provide techniques for analyzing the importance of variables within a domain and for analyzing common structural features in the ensemble. We empirically demonstrate the effectiveness of our Ensembled Dynamic Bayesian Networks (EDBN) on two real-world meteorological domains.

In Chapter 2, we provide the necessary background for the development of EDBN, including an overview of Bayesian Networks, Linear Gaussian Networks, and the key features of Random Forests. Chapter 3 describes the details of EDBN and develops an approach to prediction that allows EDBN to predict nonlinear relationships. Chapters 4 and 5 present empirical results on two meteorological domains. Chapter 4 focuses on the Storm Scale Ensemble Forecast data, where we use EDBN to combine individual meteorological model forecasts into a final forecast. Chapter 5 presents results for a storm prediction task, comprised of reflectivity prediction, vertically integrated liquid prediction, and echo top prediction. We demonstrate that EDBN performs well in both of these domains. Finally, Chapter 6 contains concluding remarks as well as avenues for future work.

Chapter 2

Background

2.1 Notation

We use capital letters to refer to random variables or sets of random variables, such as A or X. We use bold to distinguish vectors and matrices from scalars, and for a random variable A, $\mathbf{A}$ denotes a vector of values sampled from A, while a denotes a single value sampled from A. We refer to the unconditional mean of A as μ_A, and to the standard deviation of A as σ_A. For a set of random variables X, $\mathbf{\Sigma_X}$ denotes the covariance of X. For any vector or matrix, $\mathbf{A_i}$ refers to the i-th value or row in the vector or matrix. $||\mathbf{A}||$ represents the Euclidean norm of the vector $\mathbf{A}$.

We represent a Gaussian distribution with mean μ and standard deviation σ as $N(\mu, \sigma)$. The value of the Gaussian distribution with mean μ and standard deviation σ at a point x is represented by $N(x; \mu, \sigma)$.

2.2 Bayesian Networks

Bayesian Networks provide a compact representation of a joint probability distribution as a directed acyclic graph (DAG) (Pearl 1988, 2000). Each vertex in the graph corresponds to a random variable, and each edge in the graph corresponds to a dependence relationship between two variables. Each vertex has a corresponding local probability distribution associated with it. If the structure of the network is specified by a domain expert, or if the structure is learned from data under certain assumptions, edges can represent either causal relationships or correlations (Pearl 2000). However, in general, if a structure is learned from data, then edges only represent correlation.

To illustrate why Bayesian Networks are useful, consider the Slippery domain, from Pearl (2000). This domain contains five discrete random variables related to (and including) the slipperiness of the ground. The full conditional probability table is shown in Table 2.1, which shows the probability of the surface being slippery for every possible value of the other variables. Even for this small domain, the CPT is large (32 rows). In contrast, the Bayesian Network for the Slippery domain (Figure 2.1) contains a total of 18 rows across all vertices.

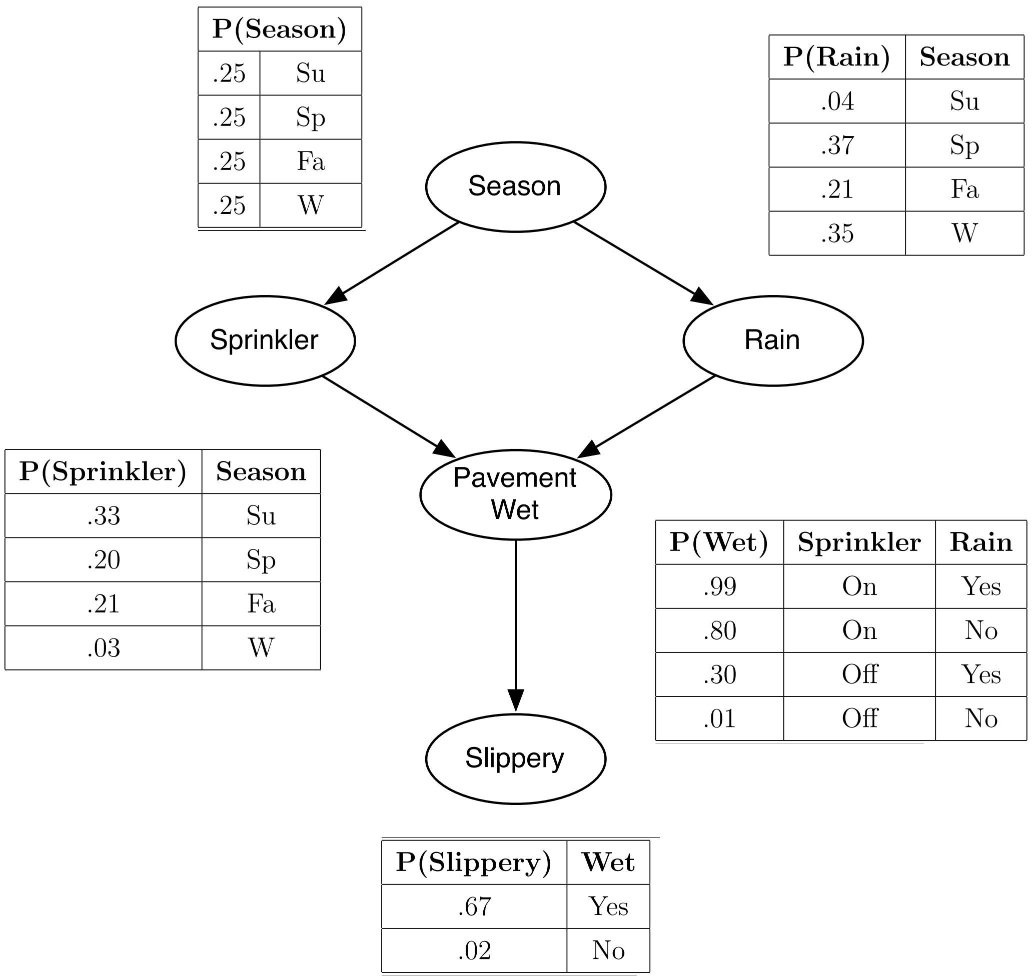
However, compression is only one benefit of Bayesian Networks. Both the underlying structure of a Bayesian Network and the parameters for the conditional probability distributions at each vertex can be learned from training data. Learning the structure of a Bayesian Network can help uncover important relationships within a dataset, and with learned parameters, Bayesian Networks can be used for prediction. However, finding the optimal structure for a dataset is NP-complete, forcing the use of heuristics (Chickering 1996).

We describe structure learning next. We postpone the discussion on parameter learning until Section 2.4, as the details of parameter learning depend on the type of probability distribution used at the vertices.

There are two main approaches to structure learning. The first approach is search and score, which searches through the set of possible structures and scores them according to some metric, eventually choosing the highest scoring model found during the search (Heckerman et al. 1995; Geiger and Heckerman 1994;

P(Wet)	**Sprinkler**	**Rain**	**Season**	**Slippery**
1.00	On	Yes	Su	Yes
0.99	On	Yes	Su	No
1.00	On	Yes	Sp	Yes
0.97	On	Yes	Sp	No
1.00	On	Yes	Fa	Yes
0.99	On	Yes	Fa	No
1.00	On	Yes	W	Yes
0.96	On	Yes	W	No
0.99	On	No	Su	Yes
0.58	On	No	Su	No
0.99	On	No	Sp	Yes
0.58	On	No	Sp	No
0.99	On	No	Fa	Yes
0.56	On	No	Fa	No
1.00	On	No	W	Yes
0.51	On	No	W	No
0.94	Off	Yes	Su	Yes
0.17	Off	Yes	Su	No
0.93	Off	Yes	Sp	Yes
0.13	Off	Yes	Sp	No
0.93	Off	Yes	Fa	Yes
0.13	Off	Yes	Fa	No
0.94	Off	Yes	W	Yes
0.13	Off	Yes	W	No
0.25	Off	No	Su	Yes
0.00	Off	No	Su	No
0.26	Off	No	Sp	Yes
0.00	Off	No	Sp	No
0.27	Off	No	Fa	Yes
0.00	Off	No	Fa	No
0.25	Off	No	W	Yes
0.00	Off	No	W	No

Table 2.1: *Probability distribution for* $p(Wet|Sprinkler, Rain, Season, Slippery)$. *From Pearl (2000)*



P(Season)	
.25	Su
.25	Sp
.25	Fa
.25	W

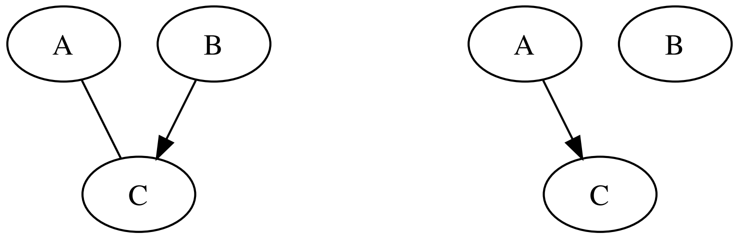
P(Rain)	**Season**
.04	Su
.37	Sp
.21	Fa
.35	W

P(Sprinkler)	**Season**
.33	Su
.20	Sp
.21	Fa
.03	W

P(Wet)	**Sprinkler**	**Rain**
.99	On	Yes
.80	On	No
.30	Off	Yes
.01	Off	No

P(Slippery)	**Wet**
.67	Yes
.02	No

Figure 2.1: *A Bayesian Network for the Slippery domain (Pearl 2000). Figure from Utz (2010)*



(a) Skeleton graph (b) Final network

Figure 2.2: *Example skeleton and final graphs, demonstrating hybrid structure learning. The final network does not necessarily contain all edges in the skeleton, but will never contain edges that are not in the skeleton.*

Cooper and Dietterich 1992). The second approach is constraint identification, which uses statistical independence tests to identify edges to place in the final graph (Cheng et al. 2002; Spirtes et al. 2001; Pearl 2000). In this thesis, we use a hybrid constraint and search based approach (Tsamardinos et al. 2006; Fast et al. 2008). Such approaches build a skeleton network that is refined with local search to generate the final network. Figure 2.2 shows a skeleton network and the resulting final network. The skeleton network contains edges between all vertices that cannot be shown to be conditionally independent at an α confidence level (that is, that given our statistical testing assumptions, we have an α chance of two vertices being conditionally independent when our test results show that they are not conditionally independent). α is a user-defined parameter.

During structure learning, the order in which the conditional independence tests are performed affects the structure of the skeleton. This is because we not only have to test every pair of verticies, but also every possible *conditioning set*, that is, set of other variables to condition the two vertices on. To order our tests, we use the Fast Adjacency Search (FAS) algorithm (Spirtes et al. 2001). FAS generates a skeleton by iteratively looking for edges to remove for larger and larger conditioning sets, until no larger sets are available. Algorithm 2.1 describes the details of FAS.

Once we have generated our skeleton network, which may contain both directed and undirected edges, we need to transform it into our final network. We use the hill-climbing approach used by Fast (2009). To generate a DAG, we use hill-climbing in the space of possible network structures to identify a locally optimal network structure. At each step of the search, the moves are: remove an edge in the final graph, add an edge that is in the skeleton to the final graph, or reverse an edge in the final graph if the corresponding edge in the skeleton is undirected. The search continues until no moves result in a higher scoring network. As the details of scoring a network depend on the distributions used in the network, we discuss the scoring used in this thesis in Section 2.4.

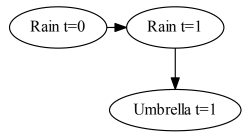
Prediction using Bayesian Networks, also referred to as *inference*, is NP-hard in the general case (Cooper 1990). Methods for approximate inference include belief propagation (Pearl 1988), likelihood weighting (Fung and Chang 1989; Shachter and Peot 1989), and Gibbs sampling (Geman and Geman 1984). We describe how we perform inference with our networks in Section 2.4.

2.3 Dynamic Bayesian Networks

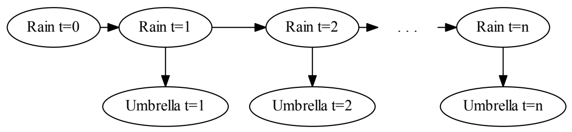
Temporal data has unique characteristics that can be modeled with a Bayesian Network. A Bayesian Network that includes temporal variables is known as a Dynamic Bayesian Network (DBN) (Dean and Kanazawa 1989). To faithfully

Algorithm 2.1: Fast Adjacency Search (FAS) (Spirtes et al. 2001). This description of FAS is adapted from Fast (2009)

```
Input: $D$ = Data, $V$ = Variables, $\alpha$ = Confidence Level
Output: Skeleton Graph
//P contains dependent pairs of variables
$P \leftarrow \emptyset$
$C \leftarrow$ Empty graph over $V$
//Begin by assuming everything is mutually dependent
for $v1, v2 \in V$ do
    $P \leftarrow P \cup \{(v1, v2)\}$
    Add edge (v1,v2) to $C$
end
//n is the size of the candidate conditioning sets
$n \leftarrow 0$
while $|P| > 0$ do
    for $(v_1, v_2) \in P$ do
        //Get neighbors of $v_1$, excluding $v_2$
        adj $\leftarrow$ Adjacencies$(C, v_1) \backslash v_2$
        ////If there are fewer adjacent vertices than our
        conditiong set size, then we have exhausted all tests
        for this pair
        if $|adj| < n$ then
            $P \leftarrow P \backslash \{(v_1, v_2)\}$
        end
        //Check all possible conditioning sets
        for $S \subset adj$ $s.t.$ $|S| = n$ do
            $pVal \leftarrow$ Conditional Independence Test $(v1, v2, S, D)$
            if $pVal > \alpha$ then
                // $(v_1 \perp\!\!\!\perp v_2 | S)$
                Remove edge between $v_1$ and $v_2$ in $C$
                $P \leftarrow P \backslash \{(v_1, v_2)\}$
            end
        end
        $n \leftarrow n + 1$
    end
end
return $C$
```



(a) DBN Slice



(b) Unrolled DBN

Figure 2.3: *An example Dynamic Bayesian Network. Figure 2.3(a) shows a slice of the DBN, while Figure 2.3(b) shows the slice unrolled over n timesteps. Rain at time 0 is the prior probability of rain. Adapted from Russell and Norvig (2003).*

represent the true underlying joint probability distribution, a DBN requires an nth-order Markov assumption to be made about the data, meaning that the current state has edges to at most the n previous states. A slice of a DBN is shown in Figure 2.3(a). By replicating the slice for each timestep under consideration (referred to as unrolling the DBN), a DBN can be used to find $P(X_t|Y_{\tau:t})$, where $1 \leq \tau \leq t$ and the subscript indicates the timestep(s) under consideration (Figure 2.3(b)). Through unrolling, a DBN can be used to predict the value of a temporal variable, even if that variable is never directly observed. Many common algorithms, including Kalman filters and Hidden Markov Models, can be represented as DBNs (Ghahramani 1998).

DBNs have been extended to continuous-time as well (Nodelman et al. 2002). However, in this thesis, we only consider discrete-time DBNs with a 1st order Markov assumption and full knowledge of the previous timestep.

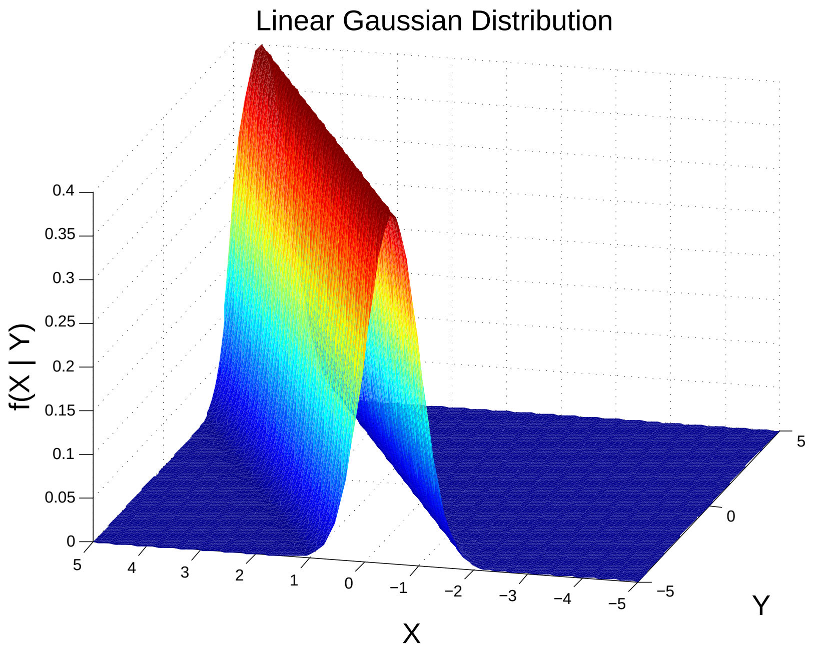


Figure 2.4: *An example linear Gaussian, showing* $N(x; \mu_x + 2(y - \mu_y), 1)$ *as the observed value* y *varies.*

2.4 Linear Gaussian Networks

Many domains feature continuous variables. For instance, consider rainfall prediction. We are interested in the amount of rainfall, and are provided information such as the temperature, wind speed, and dew point. None of these values are discrete, and so cannot be directly used in conventional discrete Bayesian Network. Fortunately, there are many ways to represent continuous probability distributions in a Bayesian Network. Linear Gaussian Networks (LGNs) are Bayesian Networks that use linear Gaussian distributions (Figure 2.4) to represent the conditional probability distributions of the vertices (Geiger and Heckerman 1994). Another approach for continuous variables is representing local probability distributions as Gaussian mixture models (Friedman et al. 1998). Another possible approach is to use kernel density estimation to estimate the probability distribution, which requires storing all training data (Hofmann and Tresp 1995; Pérez et al. 2009). Friedman and Nachman (2000) use Gaussian process priors in Bayesian networks, which requires specifying a family of prior covariances in order to learn a network.

LGNs are only capable of representing linear relationships. However, they feature tractable exact inference and, when ensembled, the ensemble can be viewed as a mixture of Gaussians. As we will discuss in Section 3.1, this allows for our ensembles to handle nonlinear relationships. Additionally, LGNs can be extended to create Dynamic Linear Gaussian Networks (Grzegorczyk and Husmeier 2009). For these reasons, we use Linear Gaussian Networks in this thesis.

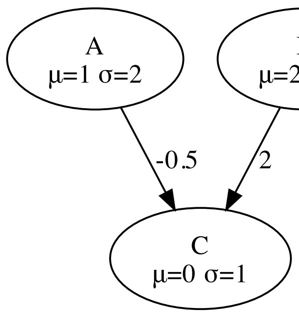


Figure 2.5: *An example Linear Gaussian Network with values* $\mu_A = 1$, $\mu_B = 2$, $\mu_C = 0$, $\sigma_A = 2$, $\sigma_B = 1$, $\sigma_C = 1$, $w_{AC} = -0.5$, *and* $w_{BC} = 2$.

Linear Gaussians provide a way of representing conditional distributions for continuous variables. If the pdf of a random variable A with parents $\mathbf{X}$ is a linear Gaussian, then given the values $\mathbf{x}$ of its parents the pdf takes the form

$$N\left(\mu_A + \sum_{i=1}^{|\mathbf{X}|} \mathbf{w}_i(\mathbf{x}_i - \mu_{\mathbf{X}_i}), \sigma_A\right)$$

where $\mathbf{w}$ is a vector of weights. That is, a linear Gaussian is a univariate Gaussian with a set standard deviation and a mean that is a linear combination of an unconditional mean and the values sampled from the parent vertices.

To see how linear Gaussians can be used within a Bayesian Network, consider the LGN $A \to C \leftarrow B$ (Figure 2.5). The following describes the conditional probability distributions of the three vertices:

$$\begin{aligned} A &\sim N(\mu_A, \sigma_A) \\ B &\sim N(\mu_B, \sigma_B) \\ C &\sim N(\mu_C + w_{AC}(a - \mu_A) + w_{BC}(b - \mu_B), \sigma_C) \end{aligned}$$

where w_{AC} is the weight of the edge between A and C. We learn the values for μ, σ, and w during parameter estimation. The details of parameter estimation appear later in this section.

As mentioned in Section 2.2, we generate a skeleton graph during structure learning. To create such a skeleton graph, we need to be able to test the conditional independence of two nodes in the graph. More specifically, we want to test whether or not the correlation between two continuous variables A and B given a third set of variables $\mathbf{X}$ is statistically significant when given N independent and identically distributed (i.i.d.) samples. To do this, we use the Fisher z-transformation of the partial correlation coefficient $\rho_{AB\cdot\mathbf{X}}$. The partial correlation coefficient is only useful for identifying independent variables when we assume that the variables were generated by a multivariate Gaussian (Baba et al. 2004). Since we are working within the framework of LGNs, we accept this assumption.

To compute $\rho_{AB\cdot\mathbf{X}}$ we use linear least squares regression to find $\mathbf{w_A}$ and $\mathbf{w_B}$ such that both

$$||\mathbf{Xw_A} - \mathbf{A}||^2$$

and

$$||\mathbf{Xw_B} - \mathbf{B}||^2$$

are minimized. From these weights, we can find the residuals,

$$\mathbf{r_A} = \mathbf{Xw_A} - \mathbf{A},$$

$$\mathbf{r_B} = \mathbf{Xw_B} - \mathbf{B},$$

which indicate how much variance in the data for A and B cannot be accounted for by the linear effects of $\mathbf{X}$. These residuals allow us to compute the partial correlation coefficient as follows:

$$\rho_{AB\cdot\mathbf{X}} = \frac{N\sum_{i=1}^{N}\mathbf{r}_{\mathbf{A}i}\mathbf{r}_{\mathbf{B}i} - \sum_{i=1}^{N}\mathbf{r}_{\mathbf{A}i}\sum_{i=1}^{N}\mathbf{r}_{\mathbf{B}i}}{\sqrt{N\sum_{i=1}^{N}\mathbf{r}_{\mathbf{A}i}^2\left(\sum_{i=1}^{N}\mathbf{r}_{\mathbf{B}i}\right)^2}\sqrt{N\left(\sum_{i=1}^{N}\mathbf{r}_{\mathbf{A}i}\right)^2\sum_{i=1}^{N}\mathbf{r}_{\mathbf{B}i}^2}} \tag{2.1}$$

$\rho_{AB\cdot\mathbf{X}}$ indicates the strength and direction of the linear relationship between A and B after removing the effects of $\mathbf{X}$, and when combined with the Fisher z-transformation:

$$z(\rho_{AB\cdot\mathbf{X}}) = \frac{1}{2}ln\left(1 + \frac{1+\rho_{AB\cdot\mathbf{X}}}{1-\rho_{AB\cdot\mathbf{X}}}\right) \tag{2.2}$$

can be treated as a z-score for determining statistical significance (Fisher 1915).

Given the skeleton graph, we need to be able to score potential networks in order to perform hill-climbing. We make the assumption that the underlying structure of our data is multivariate normal so that we can use *B*ayesian metric for *G*aussian Networks having score *e*quivalence (BGe) as our score (Geiger and Heckerman 1994).

BGe is a metric calculated from the likelihood of a LGN given the training data D. Given a LGN with structure S, our goal is to calculate $p(B_S|D)$, and we can utilize Bayes' Theorem to see that the likelihood of a network given a dataset is

$$p(B_S|D) = \frac{p(B_S)p(D|B_S)}{p(D)}. \tag{2.3}$$

We note that for a given training set, $p(D)$ is constant, and so we ignore it in our calculations. $p(B_S)$ is our prior probability for the structure of our network, and is specified by the user.

To compute $p(D|B_S)$, we must specify the following prior knowledge. μ_0 is the prior mean, and indicates our prior beliefs about the average values for the variables in the dataset. $\mathbf{T_0}$ is the prior precision matrix (the precision matrix is the inverse of the covariance matrix). This provides our prior knowledge of the covariance of the variables in the dataset. μ_0 and $\mathbf{T_0}$ are weighted by the user's confidence in their values, which we refer to as the equivalent sample size. The equivalent sample size for μ_0 is denoted ν, and the equivalent sample size for $\mathbf{T_0}$ is denoted α.

We also define our posterior precision $\mathbf{T_1}$ as:

$$\mathbf{T_1} = \mathbf{T_0} + \bar{\mathbf{\Sigma}} + \frac{\nu m}{\nu + m}(\mu_0 - \bar{\mathbf{x}})(\mu_0 - \bar{\mathbf{x}})^\top \tag{2.4}$$

where $\bar{x}$ denotes the sample mean, $\bar{\Sigma}$ denotes the sample covariance, and m denotes the number of samples in D.

Given the above, the probability of the data given a complete LGN B_{S_C} is:

$$p(D|B_{S_C}) = (2\pi)^{-nm/2}\left(\frac{\nu}{\nu+m}\right)^{n/2}\frac{c(n,\alpha)}{c(n,\alpha+1)}|\mathbf{T_0}|^{\alpha/2}|\mathbf{T_1}|^{-(\alpha+1)/2} \tag{2.5}$$

where a complete LGN is a LGN with an edge between every vertex and $c(n,\alpha)$ is defined as:

$$c(n,\alpha) = \left(2^{\alpha n/2}\pi^{n(n-1)/4}\prod_{i=1}^{n}\Gamma\left(\frac{\alpha+1-i}{2}\right)\right)^{-1} \tag{2.6}$$

Equation 2.5 is very closely related to the multivariate t-distribution, and arises because the sampling distribution of a single datapoint given a LGN follows the multivariate t-distribution (Geiger and Heckerman 1994).

Given $p(D|B_{S_C})$, the probability of the data given an arbitrary LGN is:

$$p(D|B_S) = \prod_{i=1}^{N}\frac{p(D^{x_i\Pi_i}|B_{s_C})}{p(D^{\Pi_i}|B_{S_C})} \tag{2.7}$$

where $D^{x_i\Pi_i}$ is the data corresponding to the random variable x_i and the parents of x_i in B_S, and D^{Π_i} is the data corresponding to the parents of x_i in B_S.

Combing equation 2.3 with our knowledge that $p(D)$ is constant, we define the BGe score for a LGN with structure S as

$$BGe(B_S) = p(B_S)p(D|B_S) \tag{2.8}$$

Assessing meaningful priors in large domains is difficult, and so in this thesis we use uninformative priors. We use a uniform prior for $p(B_S)$. We let α equal the number of attributes in the network, and ν equal 1. These are the minimum meaningful values for α and ν. We let μ_0 be a zero vector and T_0 be the identity matrix.

Once we have the structure of our network, we still need to estimate the parameters for the linear Gaussians at each vertex. Parameter estimation of LGNs is done by linear least-squares regression (Bøttcher 2004). Given a network structure and a set of training data, we traverse the graph in topological order. For each vertex visited, the effects of all parent vertices are subtracted out via least-squares regression, and then the mean and variance are computed from the residuals.

Assume that we are given training data in the form of a vector $\mathbf{A}$ and a matrix $\mathbf{X}$ containing N i.i.d. samples from a joint probability distribution over A and X. The variables in X are the topological parents of A. Since we are traversing in topological order, we already know μ_P for all $P \in X$.

To find our parameters, we first find $\mathbf{X}'$ such that

$$\mathbf{X}' = \mathbf{X} - \begin{pmatrix} \mu_{\mathbf{X}} \\ \vdots \\ \mu_{\mathbf{X}} \end{pmatrix},$$

that is, such that $\mathbf{X}'$ contains the values of $\mathbf{X}$ with the parent means subtracted out. We then augment $\mathbf{X}'$ with a constant column to allow for a constant factor in our regression.

We can then compute the edge weights for this node by linear least squares regression, minimizing:

$$||\mathbf{X}'\mathbf{w} - \mathbf{A}||^2,$$

where $\mathbf{w}$ contains the edge weights, and the weight on the constant factor is μ_A, the unconditional mean of A.

Once we know the edge weights and the mean of A, we can compute the variance of A. To do so, we compute the error vector $\mathbf{E} = \mathbf{X}'\mathbf{w} - \mathbf{A}$. Given $\mathbf{E}$, the variance σ_A^2 is

$$\sigma_A^2 = \frac{\sum_{i=1}^{N}(\mathbf{E}_i - \mu_A)^2}{N}.$$

By performing this estimation across the whole network in topological order, we can estimate the parameters for every vertex in the network.

With both the structure and the parameters, we are able to perform inference to predict values given a test example. As mention in Section 2.2, exact inference is hard in the general case. Fortunately, exact inference in LGNs is tractable, because a LGN is a multivariate Gaussian distribution (Shachter and Kenley 1989).

Shachter and Kenley (1989) first described the conversion from a LGN to a covariance representation. Our description here more closely follows the description provided by Heckerman et al. (1995). Given a topological ordering of the nodes in B_S, we can find the precision matrix for B_S through a recursive formula (the precision matrix is the inverse of the covariance matrix). Let v_i correspond to the variance of the ith node in the topological ordering, and $b_{i,j}$ correspond to the weight on the edge between nodes i and j (with a weight of 0 if no edge exists). Let $W(i)$ denote the $i \times i$ upper left sub matrix of the precision matrix W and $\mathbf{b}_i$ denote the row vector $(b_{1,i}, \cdots, b_{i-1,i})$. We can recursively construct W with the following formula:

$$W(i) = \begin{pmatrix} W(i-1) + \frac{\mathbf{b}_i^\top \mathbf{b}_i}{v_i} & -\frac{\mathbf{b}_i^\top}{v_i} \\ -\frac{\mathbf{b}_i}{v_i} & \frac{1}{v_i} \end{pmatrix}, \tag{2.9}$$

where $W(1) = \frac{1}{v_1}$. We can then find our covariance matrix $\mathbf{\Sigma}$ by inverting $\mathbf{W}$. The elements of the mean vector μ in our covariance representation are the means of the corresponding nodes in the network.

To perform inference, we need to be able to marginalize over random variables that we are not interested in, and to condition on evidence variables. Once we have our network in covariance form, we are able to marginalize and condition easily. Marginalizing a Gaussian involves dropping the corresponding elements from the mean vector and the corresponding rows and columns from the covariance matrix. More precisely, given a vector $\mathbf{X} = (\mathbf{X_1 X_2})$ such that $(\mathbf{X_1 X_2}) \sim N(\mu, \mathbf{\Sigma})$ where

$$\mu = \begin{pmatrix} \mu_{\mathbf{1}} \\ \mu_{\mathbf{2}} \end{pmatrix}$$

and

$$\mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma_{11}} & \mathbf{\Sigma_{12}} \\ \mathbf{\Sigma_{21}} & \mathbf{\Sigma_{22}} \end{pmatrix},$$

marginalizing over the set of random variables $\mathbf{X_2}$ results in a Gaussian distribution of the form:

$$N(\mu_{\mathbf{1}}, \mathbf{\Sigma_{11}}), \tag{2.10}$$

Similarly, conditioning a Gaussian distribution also results in a Gaussian distribution. Given the same μ, $\mathbf{\Sigma}$, and $\mathbf{X_2}$ as above, conditioning $N(\mu, \mathbf{\Sigma})$ on $\mathbf{X_2}$ results in a distribution

$$N(\mu_{\mathbf{X_1|X_2}}, \mathbf{\Sigma}_{\mathbf{X_1|X_2}})$$

where

$$\mu_{\mathbf{X_1}|\mathbf{X_2}} = m(\mathbf{X_2}) = \mu_{\mathbf{1}} + \mathbf{\Sigma_{12}}\mathbf{\Sigma_{22}}^{-1}(\mathbf{X_2} - \mu_{\mathbf{2}}) \tag{2.11}$$

and

$$\mathbf{\Sigma}_{\mathbf{X_1}|\mathbf{X_2}} = \mathbf{\Sigma_{11}} - \mathbf{\Sigma_{12}}\mathbf{\Sigma_{22}}^{-1}\mathbf{\Sigma_{21}}. \tag{2.12}$$

With the ability to marginalize and condition, and given a set of target variables **V**, a set of variables to ignore **Z**, and a set of evidence **X**, we are able to perform inference. Inference requires conditioning the network on **X** and then marginalizing over **Z**, which then leaves only the variables in **V** in the network. Since marginalization just removes elements from the mean vector, the expected value of **V** is $m(\mathbf{X})_V$, which are the corresponding elements in the conditioned mean vector. The fact that we never use the values of **Z** during inference means that LGNs are capable of making predictions even when some data are missing.

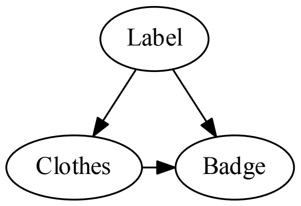
2.5 Ensembles

Ensembles have proven to be a powerful method in machine learning (Dietterich 2000). By training many models and combining their predictions (for instance, by averaging), this ensemble prediction will be more accurate than any of the individual ensemble components, assuming the individual components are better predictors than a random predictor (Dietterich 2000). More sophisticated approaches aim to reduce the correlation between the errors of individual components. For example, boosting creates an ensemble by iteratively training individual models, reweighting the training data after each iteration to focus more heavily on examples that were predicted incorrectly by the previous model (Freund and Schapire 1995).

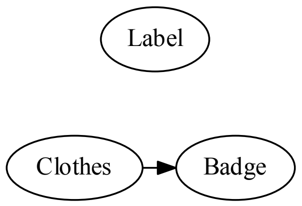
Breiman (2001) introduced Random Forests, which trains ensembles of decision trees using both bagging and randomization. Bagging alters the training data for each component by sampling new training sets from the original training set with replacement (Breiman 1996). The bagged training set is the same size as the original training set, but contains duplicates of some examples and is missing others. The examples that are not contained in the resampled training set are called "out-of-bag." These out-of-bag examples can provide a validation set at the component level. Bagging ensures that each individual component is trained on a slightly different set of data.

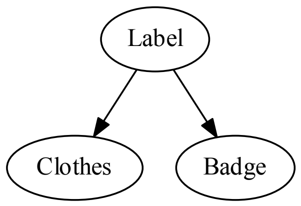
In contrast to both bagging and boosting, randomization does not alter the training set for different components. Instead, randomization involves training each component on a subset of the available features (Breiman 2001). The number of features to use per component is a parameter. The class feature is included in all components. The benefit of randomization is that it forces different networks to focus on different relationships within the domain, by providing each network with a different subset of the variables within the domain.

These approaches have been applied to Bayesian Networks. Liu et al. (2007) present an approach for combining bagging and Bayesian Networks, but their method ultimately results in a single network. Boosting has been applied to networks with a Naïve Bayes structure (that is, networks where the only edges are from the class variable to the other variables) (Jing et al. 2008). Taking inspiration from Random Forests, Utz (2010) combined bagging and randomization to learn ensembles of discrete Bayesian networks. In this thesis, we also combine bagging and randomization for learning ensembles of Bayesian Networks. Our approach differs from Utz (2010) in that we model dynamic continuous variables in our networks.



(a) Full Network





(b) Label=Worker (c) Label=Visitor or Spy

Figure 2.6: *Example of a Multinet versus a traditional Bayesian Network. Example adapted from Geiger and Heckerman (1996).*

The Bayesian Multinet provides a different approach to creating ensembles of Bayesian Networks (Geiger and Heckerman 1996). In contrast to conventional ensembles, Multinets (Figure 2.6) learn a single network for different partitions of the class label, such that each network corresponds to a disjoint subset of the possible class values. To better illustrate the effect of this partitioning, consider a domain in which we want to identify whether someone is a spy, worker, or visitor, based on their clothing and whether or not they have a badge. Figure 2.6(a) shows a traditional Bayesian Network for this domain. Figures 2.6(b) and 2.6(c) form a multinet for this domain. If the value of Label is Worker, then 2.6(b) is the network used, and if the value of Label is either Spy or Visitor, 2.6(c) is the network used. In the multinet, we can see that the relationship between badge and clothing only exists for workers, a fact which is hidden in the full network.

Unlike a conventional ensemble, multinets have a natural probabilistic interpretation, as the whole multinet corresponds to a single joint probability distribution, and they can find different dependence relations between variables for different class values. Multinets have been extended to handle temporal data (Bilmes 2000).

Chapter 3

Ensembled Dynamic Bayesian Networks

A Linear Gaussian Network can only represent linear relationships. Furthermore, as discussed in Chapter 2, ensembling can improve predictive strength. In this chapter, we show that by combining LGNs with ensembling, we can create an ensemble of dynamic LGNs that is capable of representing nonlinear relationships.

Similar to Utz (2010), we draw inspiration from Random Forests and improve EDBN's performance by using bagging and randomization (Breiman 2001). We construct the individual LGNs of our ensemble in the manner described in Section 2.4. The main issue we consider in this chapter is how to use EDBN for prediction.

3.1 Prediction Using Ensembles of Dynamic Bayesian Networks

Given an ensemble of LGNs, how do we combine individual predictions in such a way that we can represent nonlinear relationships? One approach to combining

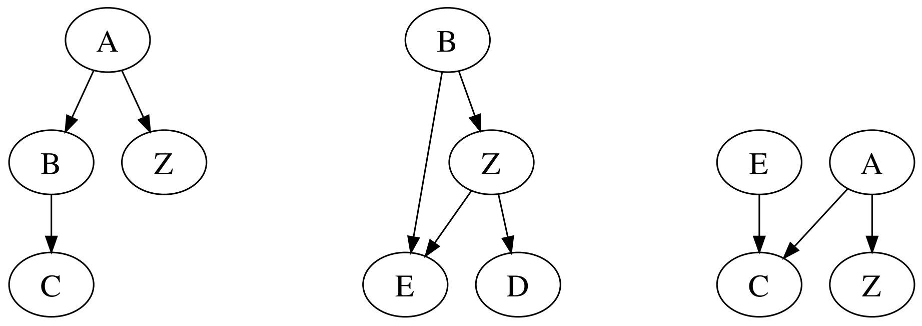


Figure 3.1: *An ensemble of size 3 with 4 attributes per model. The possible attributes are A,B,C,D,E, and Z. Z is the target variable and so is in every ensemble.*

predictions is to take the weighted average of the component predictions. For EDBN, this would result in a predicted value v such that

$$v = \frac{\sum_{i=1}^{k} w_i m_i(\mathbf{x})}{\sum_{i=1}^{k} w_i} \tag{3.1}$$

where w_i represents the weight given to the i-th model and m_i comes from Equation (2.11). However, this is a linear combination of the component predictions, which are also linear. Thus, only linear relationships can be accurately predicted using Equation (3.1).

In order to represent nonlinear relationships, we need a nonlinear weighting. A key insight is that, since each component of our ensemble is a multivariate Gaussian, our whole ensemble is a Gaussian mixture model (GMM).

A conventional GMM with M components has a probability density function (pdf) of the form

$$f(\mathbf{x}) = \sum_{i=1}^{M} \pi_i N(\mathbf{x}; \mu_i, \mathbf{\Sigma}_i)$$

where μ_i and $\mathbf{\Sigma}_i$ are the mean and covariance of the i-th Gaussian in the model, and π_i is the weight of the i-th Gaussian in the model. These weights must satisfy

$$\sum_{i=1}^{M} \pi_i = 1$$

in order for $f(\mathbf{x})$ to be a pdf.

Unlike a conventional GMM, EDBN does not have individual weights for its networks. So the pdf of an EDBN is

$$f(\mathbf{x}) = \frac{1}{M} \sum_{i=1}^{M} N(\mathbf{x}|\mathbf{x} \in B_i; \mu_i, \mathbf{\Sigma}_i)$$

where $(x|x \in B_i)$ refers to the subset of the random variables in the i-th network. This pdf is equivalent to the pdf of a randomized GMM with uniform weighting on its components.

Given that our ensemble is a GMM, we can utilize Gaussian Mixture Regression (GMR) (Sung 2004). GMR is similar to Equation (3.1), but each prediction is weighted by the likelihood of that network given the test example. As the likelihood is Gaussian, we have a nonlinear weighting, and are able to represent nonlinear relationships.

More specifically, to perform regression we first need to find component weights w_i of the form:

$$w_i(\mathbf{x}) = \frac{\pi_i N(\mathbf{x}; \mu_{ix}, \mathbf{\Sigma}_{ix})}{\sum_{j=1}^{k} \pi_j N(\mathbf{x}; \mu_{jx}, \mathbf{\Sigma}_{jx})} \tag{3.2}$$

In the case of our networks, the weights π_i are uniform.

Combining Equation (2.11) with Equation (3.2), the predicted value $\mathbf{v}$ of a subset of random variables $\mathbf{X_1}$ given a set of evidence $\mathbf{X_2} = \mathbf{x}$ using Gaussian mixture regression is

$$\mathbf{v} = E[\mathbf{X_1}|\mathbf{X_2} = \mathbf{x}] = \sum_{i=1}^{k} w_i(\mathbf{x}) m_i(\mathbf{x}) \tag{3.3}$$

Because Equation (3.2) relies on evaluating the underlying pdf of a network, we anticipate that it will interact poorly with randomization. If a network is a poor predictor for a given test example, the network may still evaluate to a high likelihood because of the removal of variables by randomization. To fix this problem, we learn weights for each network in our ensemble by least-squares regression. Given a vector of true target values $\mathbf{t}$ and a matrix of each network's predictions for every training example $\mathbf{V}$, we find our weights ω by minimizing

$$||\mathbf{V}\omega - \mathbf{t}||^2$$

These weights differ from those used in GMR in that these weights are not normalized.

To perform regression with our new weights, we augment Equation (3.3) with the new linear regression weights as follows

$$v = \sum_{i=1}^{k} w_i(\mathbf{x}) m_i(\mathbf{x}) \omega_{\mathbf{i}} \tag{3.4}$$

That is, we weight our predictions by both the least-squares regression weights and the conventional GMR weights.

For temporal data, Equations (3.3) and (3.4) can still be used. We unroll the ensemble by unrolling each individual network.

In the next two chapters, we investigate whether prediction using Equation (3.3) or (3.4) performs better.

3.2 Variable Importance

In addition to introducing Random Forests, Breiman (2001) described how to use Random Forests to compute the importance of different variables for prediction. This variable importance is computed by first evaluating each component on its out of bag data, getting the RMSE of the true data ($RMSE_{true}$). We then permute the values of variable A across the out of bag examples and recompute the RMSE on this permuted data ($RMSE_A$). We then define the variable importance score I to be

$$\mathrm{I}_A = RMSE_{true} - RMSE_A. \tag{3.5}$$

This procedure is performed for all variables A of interest. Higher variable importance scores indicate that a variable is highly important in prediction. Similarly, lower scores indicate that a variable is not as important in prediction.

As an example of the permutation step, Table 3.1 shows a portion a dataset inspired by the Sprinkler domain. Table 3.2 shows the same dataset with the Season variable permuted.

Breiman and Cutler (2012) provides a procedure for testing the significance of variable importance scores. This is done by converting the importance score of a variable into a z-score,

$$z_A = \frac{I_A}{\bar{\sigma}_A/\sqrt{n_A}} \tag{3.6}$$

where $\bar{\sigma}_A$ is the standard deviation of the importance scores for variable A, and n_A is the number of networks containing variable A. That is, the denominator is the standard error of I_A. Assuming that the networks are independent of one another, we can treat z_A as a typical z-score and perform hypothesis tests. However, Strobl and Zeileis (2008) show that the power of this test procedure decreases as the size of the out-of-bag set increases. Strobl et al. (2008) provide a test procedure that remedies this problem, but is designed specifically for decision trees and discrete variables. Fortunately, Strobl and Zeileis (2008) also show that the power can be increased by increasing the number of networks,

Wet	**Sprinkler**	**Rain**	**Season**	**Slippery**
Yes	On	Yes	Sp	Yes
Yes	Off	Yes	Su	No
Yes	On	Yes	Sp	Yes
Yes	Off	Yes	Fa	Yes
Yes	On	Yes	W	No
No	Off	No	Sp	No
No	On	No	Su	No
No	Off	Yes	Fa	No
No	Off	No	W	Yes
No	Off	No	Su	No
No	On	No	Su	No
No	Off	No	Su	No
No	Off	No	W	No
No	Off	No	Fa	No

Table 3.1: *A portion of the Sprinkler training data.*

Wet	**Sprinkler**	**Rain**	**Season**	**Slippery**
Yes	On	Yes	Fa	Yes
Yes	Off	Yes	Sp	No
Yes	On	Yes	Fa	Yes
Yes	Off	Yes	Sp	Yes
Yes	On	Yes	W	No
No	Off	No	Fa	No
No	On	No	Su	No
No	Off	Yes	Su	No
No	Off	No	Su	Yes
No	Off	No	Sp	No
No	On	No	Su	No
No	Off	No	Su	No
No	Off	No	W	No
No	Off	No	W	No

Table 3.2: *Sprinkler data with Season permuted.*

thereby decreasing the standard error. For this reason, in this thesis, we use a large number of networks for our variable importance significance testing.

We note that variable importance is performed on a network-by-network basis, and does not involve the entire ensemble. For this reason, even if we are predicting using weights learned from least squares regression (Equation 3.4), our networks are independent for variable importance calculations, and we can use this test procedure without violating its independence assumption.

3.3 Edge Counts

A benefit of Bayesian Networks is that they provide a human-readable model that can provide insights into the underlying structure of a dataset. However, ensembling and randomization make it harder to ascertain any underlying structure in the data. To compensate for this, we can take advantage of the large number of networks EDBN generates to analyze how often certain edges appear. The technique we use was first proposed by Utz (2010). Using all of the networks generated in an ensemble, we evaluate the importance of a relationship between two variables A and B by computing the edge frequency

$$e_{AB} = \frac{f_{AB}}{n_{AB}} \tag{3.7}$$

where f_{AB} is the number of edges between A and B in the ensemble and n_{AB} is the number of networks in the ensemble that contain both A and B. Given e for all pairs of variables in the ensemble, we can construct an edge frequency graph that contains all variables in the ensemble and contains an edge from A to B if e_{AB} is greater than a user-defined threshold.

Chapter 4

Improving Storm Scale Ensemble Rainfall Forecasts

Rain is a common meteorological event, but it is also challenging to predict due to the nonlinear spatial and temporal nature of rainfall and the large number of factors that can affect it (Ebert 2001; Bremnes 2004). The consequences of rainfall can range from inconvenience to destructive flash flooding, and improving our ability to accurately forecast rainfall would help prevent the more disastrous outcomes and help plan around the more annoying outcomes. In this chapter, we investigate EDBN's ability to improve upon Storm Scale Ensemble Forecast (SSEF) rainfall predictions.

The SSEF dataset consists of quantile values for the outputs of variables of different meteorological models. The SSEF data is provided by the Center for the Analysis and Prediction of Storms (CAPS) (Xue et al. 2011; Kong et al. 2011). The dataset that we use in this chapter comes from Gagne II (2012).

A map showing where data were sampled from is shown in Figure 4.1. Table 4.1 lists the meteorological variables in the SSEF dataset. As in Gagne II (2012), this dataset is built from the outputs of 14 meteorological models. The variables that come directly from the component models have three values within the SSEF data: the 10th, 50th, and 90th percentile value of the member outputs. Percentiles are used to provide a more nuanced view of the distribution of the

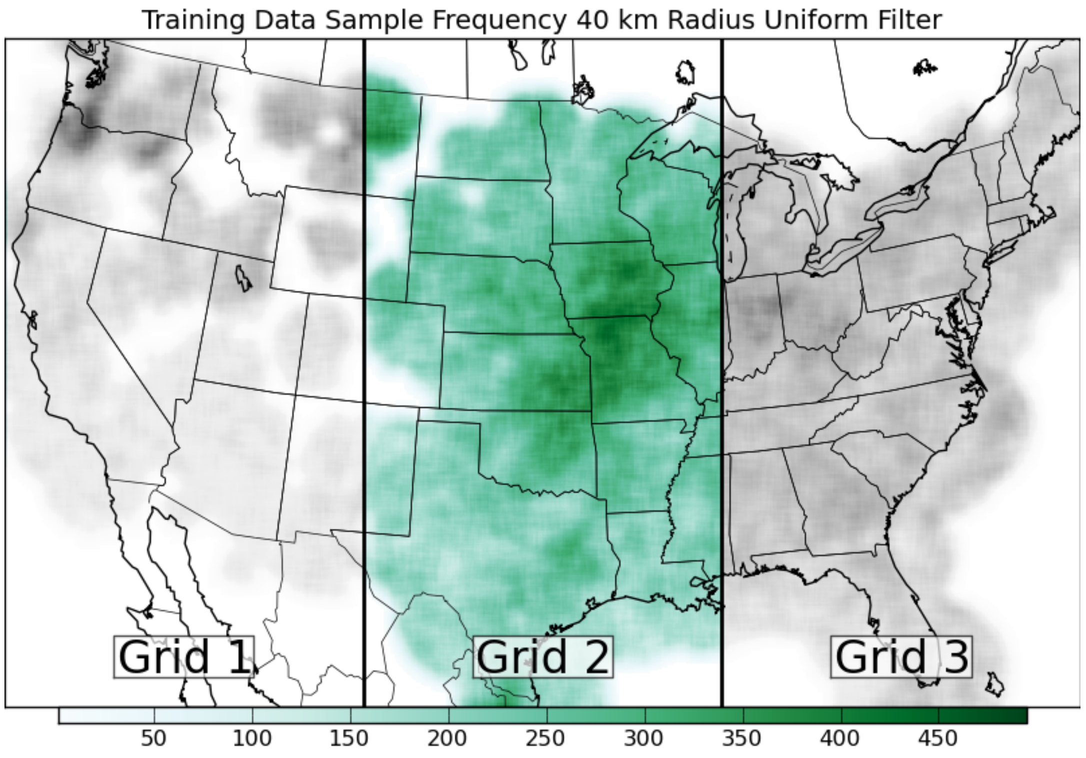


Figure 4.1: *Map showing locations and density of sampled datapoints in the SSEF dataset. In this thesis, we focus on grid 2 (Adapted from Gagne II (2012)).*

Variable	**Notes**
1-hour accumulated precipitation	Rainfall
Composite reflectivity	
2 m dew point temperature	Temperature at which condensation would occur
1-hour maximum reflectivity	
Mean sea level pressure	
Surface-based Convective Available Potential Energy (CAPE)	Indicator of atmospheric instability
Surface-based Convective Inhibition (CIN)	Indicator of atmospheric stability
Precipitable water	Amount of water vapor in a column of air
2 m air temperature	
700 mb temperature	
700 mb east-west wind	
700 mb north-south wind	
700 mb height	
500 mb north-south wind	
1-hour max upward vertical velocity	Maximum downward wind speed
1-hour max downward vertical velocity	Maximum upward wind speed

Table 4.1: *The model variables sampled from the SSEF runs. Adapted from Gagne II (2012)*

data than the mean and variance would give, while still reducing the total number of variables compared to using the raw model.

4.1 Rainfall Prediction

Our goal is to predict the true amount of rainfall given the SSEF forecast. We aim to improve upon the raw SSEF forecast for rainfall by intelligently combining the individual forecasts with EDBN. The SSEF data ranges from May 3rd 2010 to June 18th 2010, except for weekends. The model outputs are at a 4km grid spacing. We focus on the forecast for the central plains of the US, due to the prevalence of rainfall events occurring there.

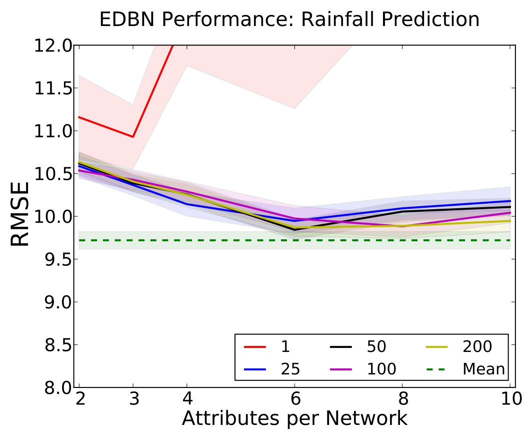
Due to the large amount of data in the SSEF dataset, we train each individual network on a small, random portion of the data, the size of which we vary in our experiments. We also vary the number of attributes in each network, up to 10 variables per network, as well as the number of models in the ensemble, from 1 to 200. To discover whether Gaussian Mixture Regression (Equation (3.3)) or Gaussian Mixture Regression with learned weights (Equation (3.4)) performs better, we perform prediction using both. For learning weights with least-squares regression, we perform regression on one tenth of the total training set. The significance cutoff for dependence testing during structure learning, α, was set to 0.05.

We compare our EDBN results to a mean predictor that guesses the mean rainfall amount. We evaluate model performance using root mean squared error (RMSE). All experiments were executed 30 times. Each run left 10 days out (chosen randomly) as a test set and trained on the other 24 days.

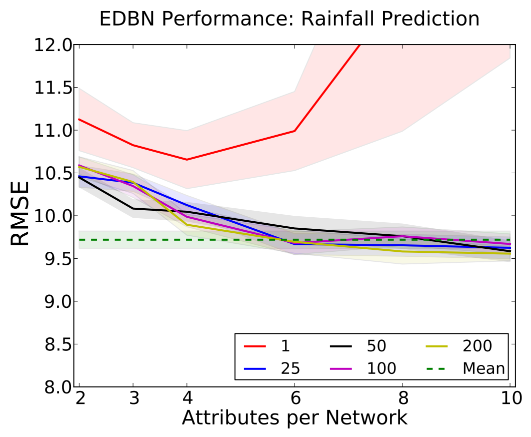
Results for prediction without learned weights are shown in Figure 4.2. Ensembles of size 1 perform poorly, and their performance degrades as the number of attributes increases. We hypothesize this is because the single LGN is trying to represent non-linear relationships, but when given only a few attributes, is not able to do much more than guess the mean. As the number of attributes increases, a LGN is capable of representing more complex linear relationships, which fail to properly fit the non-linear nature of the SSEF dataset. The difference between the other ensemble sizes is much less stark, and all follow a general trend of improved performance that levels off as the number of attribute increases. Increasing the number of training examples given to each network greatly impacts the performance of ensembles of size 1, and slightly improves the performance of other ensembles sizes.

Results for prediction with learned weights are shown in Figures 4.3 and 4.4 (Figure 4.4 places the model count on the x-axis and plots each attribute count as a line, to better illustrate performance at different attribute counts). Ensembles of size 1 perform comparably to the non-weighted results, which is expected since weighting does nothing if there is only one model. Performance degrades across all ensemble sizes greater than 1 as the number of attributes is increased. This is due to overfitting in the weight learning step. We hypothesize that models of size 2 perform best because their lack of complexity reduces the ability for the weights to overfit. For larger ensemble sizes, in particular 200, increasing the number of training examples given to each network hurts performance. This agrees with our hypothesis that our best performance occurs when we ensemble many weak models.

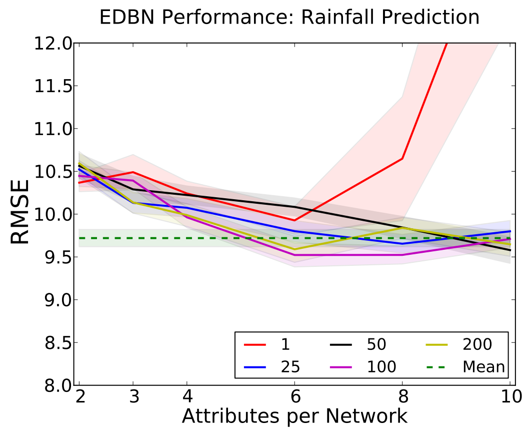
As our results indicate that overfitting is an issue with the least-squares weighting, we hypothesize that an approach that includes regularization, such



(a) 100 Training Examples per Network

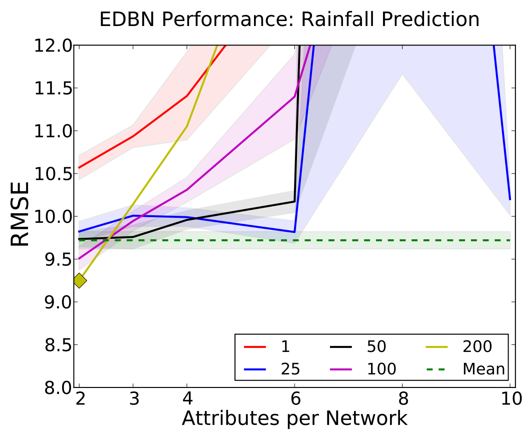


(b) 1000 Training Examples per Network

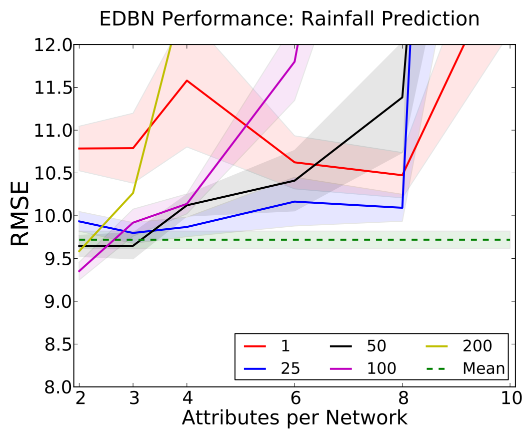


(c) 2000 Training Examples per Network

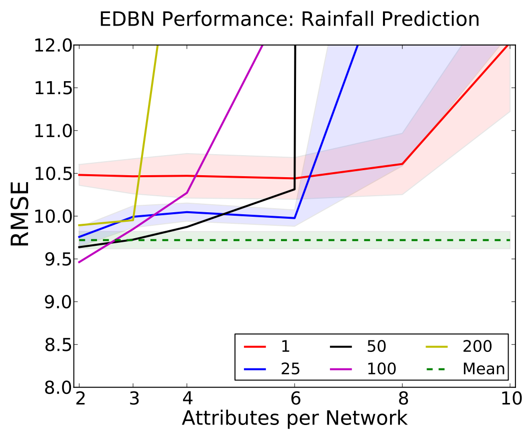
Figure 4.2: *RMSE for the SSEF domain without weighted prediction. Lines correspond to models used per ensemble. Shaded area represents standard error.*



(a) 100 Training Examples per Network

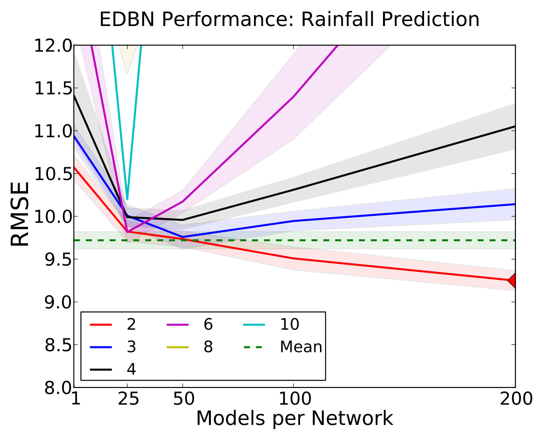


(b) 1000 Training Examples per Network

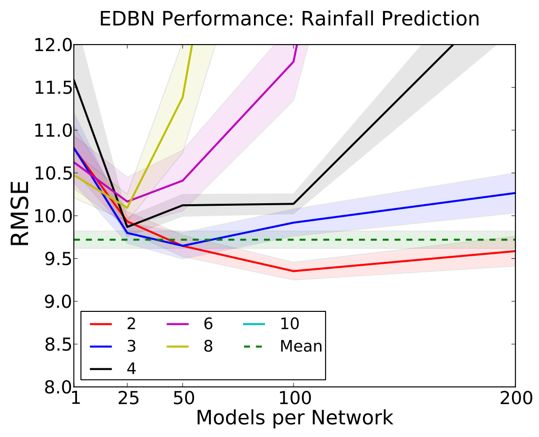


(c) 2000 Training Examples per Network

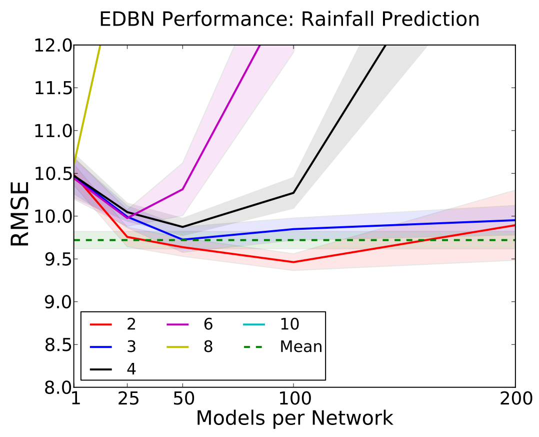
Figure 4.3: *RMSE for the SSEF domain at various parameter settings with least-squares weighted prediction.*



(a) 100 Training Examples per Network



(b) 1000 Training Examples per Network



(c) 2000 Training Examples per Network

Figure 4.4: *RMSE for the SSEF domain at various parameter settings with least-squares weighted prediction. Lines are attribute counts.*

as ridge regression, would reduce the overfitting (Hoerl and Kennard 1970). We discuss regularization further in our future work (Chapter 6)

We compare EDBN across all parameter settings to the mean predictor. However, we face a multiple comparisons problem because we have 90 distinct parameter settings for both prediction methods. Aiming for a 5% confidence level, we use Bonferroni correction, resulting in a much lower $\alpha = 0.05/90 = 0.00056$. Significant parameter settings are marked in Figures 4.2, 4.3, and 4.4 with diamonds. Using a two-sample one-tailed t-test, we find that only weighted prediction with 200 models, 2 attributes, 100 training examples performs significantly better than the mean predictor.

We compare our best weighted parameter setting (200 models, 2 attributes, 100 examples) to our best unweighted predictor (200 models, 10 attributes, 1000 training examples) using a two-sample one-tailed t-test with $\alpha = 0.00056$ and find that they are not significantly different ($p = 0.02$).

We therefore conclude that, for this domain, weighted predictions can perform significantly better than guessing the mean when many weak LGNs are ensembled, and that we can draw no conclusions about the relative performance of unweighted prediction.

Similar to Gagne II (2012) we perform a case study using data from May 25, 2010 at 0100 UTC. Figure 4.5 shows the EDBN forecast across along with the median SSEF forecast and the true observed precipitation. The EDBN forecast underestimates the rainfall in the southern plains, and overestimates the rainfall in the northern plains. EDBN also estimates large areas of very slight rainfall, which is an effect that Gagne II (2012) notes in his regression work and resolves by combining regression with a classification algorithm to determine whether or not the regression model applies to a given location.

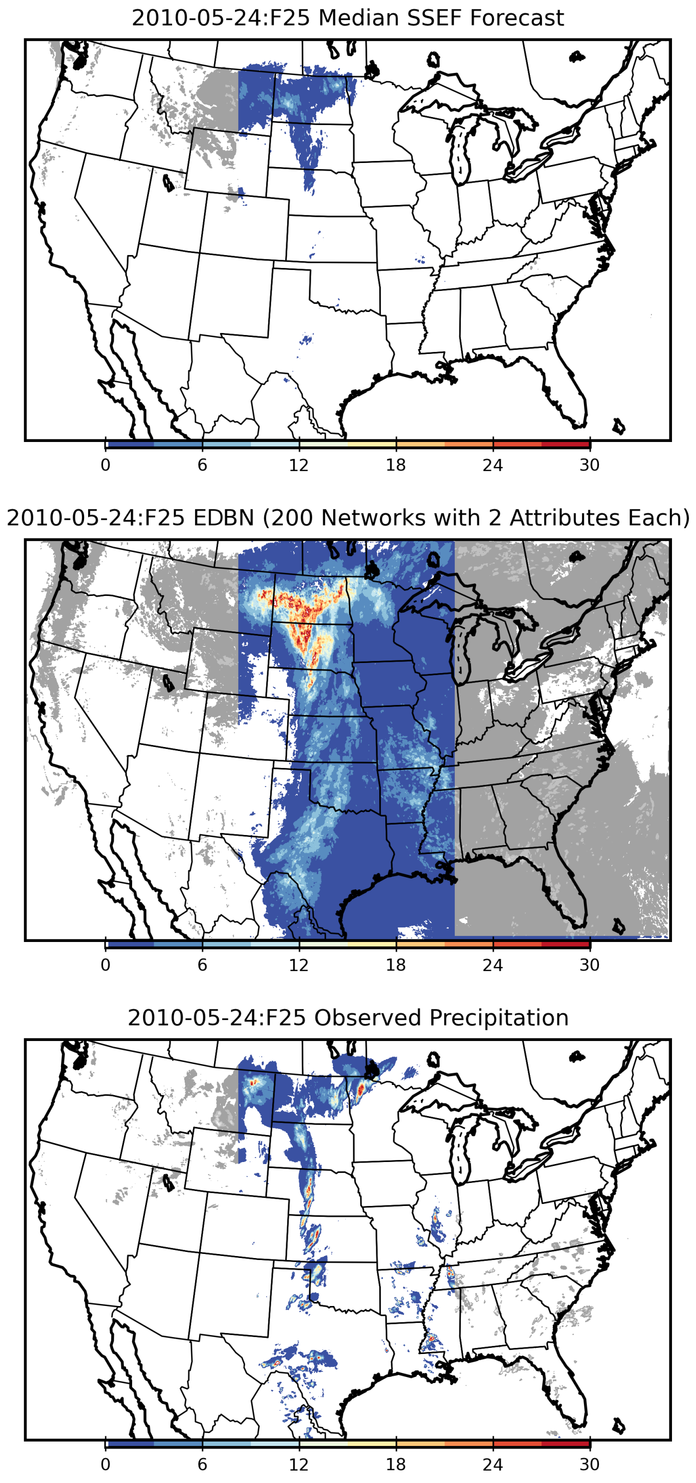

2010-05-24:F25 Median SSEF Forecast
0
6
12
18
24
30
2010-05-24:F25 EDBN (200 Networks with 2 Attributes Each)
0
6
12
18
24
30
2010-05-24:F25 Observed Precipitation
0
6
12
18
24
30

Figure 4.5: *Maps of SSEF prediction, EDBN prediction, and observed rainfall at 25 May 2010 0100 UTC. EDBN was trained on the middle third of the US.*

4.2 Variable Importance

The variable importance results for the SSEF data are shown in Table 4.2. We sort by z-score rather than raw score, so scores should be interpreted as a measure of confidence that the variable impacts predictive abilities, rather than as a measure of how much a variable impacts predictive abilities. The top five variables are downward vertical velocity, composite reflectivity, maximum reflectivity, surface-based convective available potential energy, longitude, and accumulated precipitation. These results indicate that EDBN is using the atmospheric ingredients for storms to predict rainfall. The accumulated precipitation forecast is in the top ten variables, but EDBN is using the 90th percentile value for it, indicating that EDBN is compensating for low SSEF precipitation forecasts. Our variable importance results are similar to those reported by Gagne II (2012) for random forests.

4.3 Edge Frequency Analysis

We visualize the edge frequency graphically by creating a network that contains edges between all variables that were connected more than a user specified percentage of the time. Figure 4.6 shows such a network for a subset of the SSEF variables drawn from Table 4.2, with edges between variables that were connected more than 90% of the time. We can quickly see that edges are common in this domain, as even at a 90% threshold many edges remain in the edge graph. Of particular note is that longitude is not frequently connected to any of the other variables, despite appearing in the variable importance analysis, and that surface-based Convective Available Potential Energy (CAPE) is only connected to rain. Other than CAPE and downward vertical velocity,

Variable	z-score
Downward Vertical Velocity 10%	21.4
Composite Reflectivity 90%	20.3
Maximum Reflectivity 90%	19.8
Upward Vertical Velocity 90%	16.0
Downward Vertical Velocity 50%	14.1
Surface-Based CAPE 90%	10.8
Longitude	10.0
Composite Reflectivity 50%	9.7
Accumulated Precipitation 90%	9.6
Downward Vertical Velocity 90%	8.8

Table 4.2: *Top 10 variable importance scores for the SSEF data. 10%, 50%, and 90% refer to the percentile.*

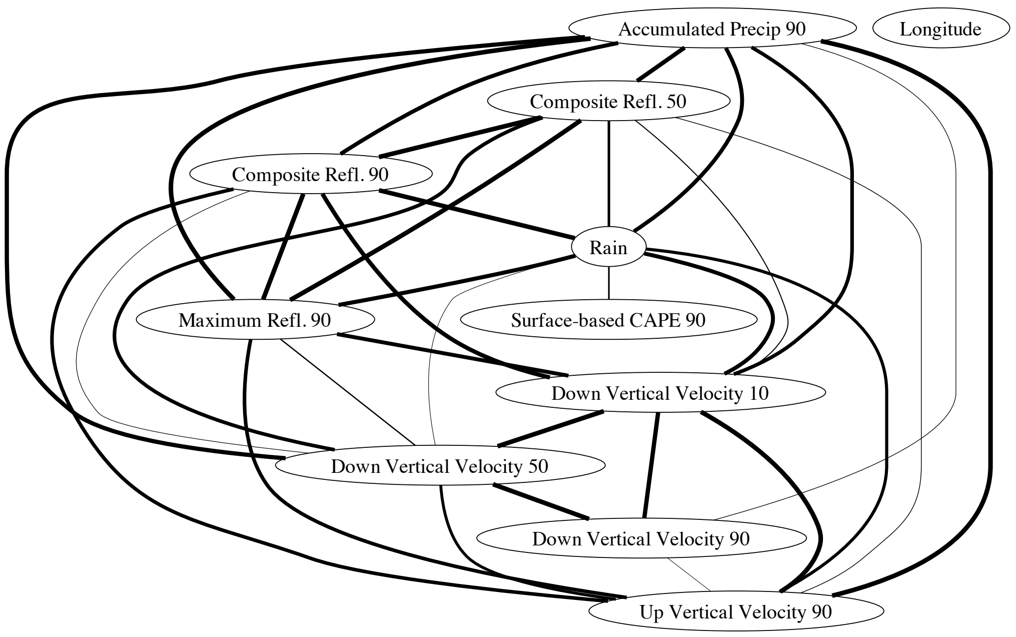


Figure 4.6: *Edge graph of a subset of the attributes in SSEF. Edge thickness indicates how often a pair of edges appeared in the ensemble, with the thinnest being edges that occurred 90% of the time and thicker edges occurring more frequently.*

accumulated precipitation and rain are connected to the same variables, as we would expect given that accumulated precipitation is the forecast value for rain.

Chapter 5

Reflectivity, Vertically Integrated Liquid, and Echo Top Prediction

Severe storms bring with them the threat of economic loss as well as the loss of human life. From 1995 to 2000, an average of 70 people were killed each year due to lightning or hail, and the average annual economic loss due to lightning and hail was \$976 million (Pielke Jr and Carbone 2002). Our ability to understand and predict storms is of great importance in reducing these numbers. In this chapter, we focus on EDBN's ability to predict reflectivity, and also investigate EDBN's ability to predict two other meteorological values: vertically integrated liquid (VIL) and echo top. VIL is a measure of the total amount of liquid in a column of air, and echo top is the highest altitude with a reflectivity of 18 dBZ or higher.

The Reflectivity domain focuses on the prediction of composite reflectivity based on a number of meteorological forecasts. In this domain, we have access to the predictions and true reflectivity of the previous timestep, so we are able to use the dynamic aspect of EDBN. Table 5.1 shows the full set of variables in the Reflectivity domain. Some additional fields are added for the dynamic networks, corresponding to true values from the previous timestep for a small number of measurements. These additional fields are shown in Table 5.2. The Reflectivity data are hourly and covers the Continental United States from June

Variable	**Notes**
Latitude	
Longitude	
Upward Long-Wave Radiation Flux (Forecast)	Measure of infrared radiation from surface reaching top of the atmosphere
Surface Temperature (Forecast)	
Sensible Heat Net Flux (Forecast)	Heat transfered from surface to atmosphere by convection
Relative Humidity (Forecast)	
Precipitable Water (Forecast)	Amount of water vapor in a column of air
Surface Pressure (Forecast)	
Latent Heat Net Flux (Forecast)	Heat transfered from surface to atmosphere by evaporation
Planetary Boundary Layer Heigh (Forecast)	Height at which atmosphere is not directly affected by surface changes
Vertical Velocity (Forecast)	Upward or downward wind speed
Downward Short-Wave Radiation Flux (Forecast)	Incoming solar radiation
Dew Point Temperature (Forecast)	Temperature at which condensation would occur
Convective Inhibition (CIN) (Forecast)	Indicator of atmospheric stability
Convective Available Potential Energy (CAPE) (Forecast)	Indicator of atmospheric instability
1-Hour Accumulated Precipitation (Forecast)	Rainfall
Wind Speed (Forecast Hour 1)	

Table 5.1: *The variables in the Reflectivity dataset. (Forecast Hour 1) indicates that the value is the previous hour's forecast for the present hour's value. (Forecast) indicates that the variable has three values in the dataset, corresponding to forecast hour 1, forecast hour 2, and forecast hour 3.*

Maximum Reflectivity (3 hours old)
Maximum Reflectivity (2 hours old)
Maximum Reflectivity (1 hour old)
Maximum Reflectivity (1 hour old, motion adjusted)
MCS_grid
MCS_CI_20km
MCS_CI_grid_plsmns1

Table 5.2: *Extra variables provided to the dynamic networks, in addition to the current and previous hour's values for all variables in Table 5.1. Maximum Reflectivity (1 hour old) is the true label for the previous timestep.*

1st, 2011 to August 31, 2011. Model forecasts were sampled every 0.007 degrees latitude and 0.007 degrees longitude.

Reflectivity that is greater than 0 dBZ is a rare occurrence within this dataset. Approximately 95% of the Reflectivity data have reflectivity less than or equal to 0, so we undersample these cases to achieve a rate of 80%. Due to the large amount of data (approximately 50 million cases), we create our final dataset of 250,000 examples via random sampling.

5.1 Reflectivity Prediction

For this domain, it is reasonable to expect knowledge of the true values from the previous timestep before predicting the next timestep, and so we use our networks to predict $P(X_t | Y_t \cup Y_{t-1} \cup X_{t-1})$.

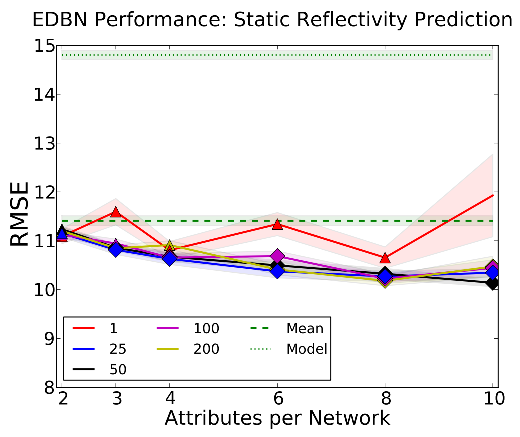
As in the experiments in Chapter 4, we vary the number of examples used to train each network, the number of attributes per network, and the size of the ensembles, with 30 runs for each parameter setting. We again compare Equations (3.3) and (3.4).

We compare our results to the mean predictor, which predicts the mean reflectivity value, and the model reflectivity forecast, which predicts the forecast reflectivity. For our dynamic networks, we also compare to the persistence prediction, which predicts the previous timestep's reflectivity.

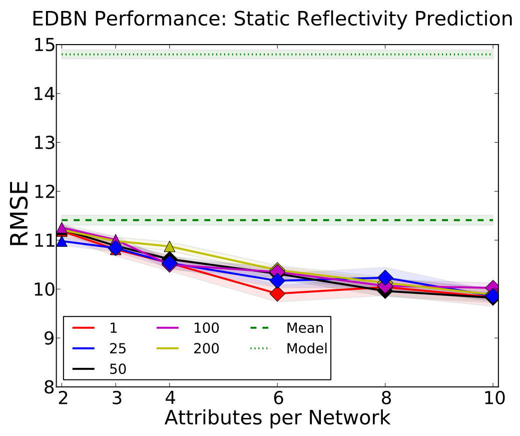
Figure 5.1 shows the results for EDBN on the static Reflectivity domain without least-squares weighting, and Figures 5.2 and 5.3 show the results for weighted prediction (Figure 5.3 places the model count on the x-axis and plots each attribute count as a line, to better illustrate performance at different attribute counts). Figures 5.4, 5.5, and 5.6 show the unweighted and weighted prediction results on the dynamic domain. As attribute counts are increased, the performance of weighted prediction degrades, while the performance of unweighted prediction improves. Surprisingly, without least-squares regression, single networks perform comparably to ensembles in this domain, both for static prediction and dynamic prediction. Combined with the fact that weighted predictions do much better, this indicates that Gaussian Mixture Regression by itself is not enough to model the nonlinear aspects of this domain.

As in Chapter 4, we use a one-tailed two-sample t-test with $\alpha = 0.00056$ for our statistical tests.

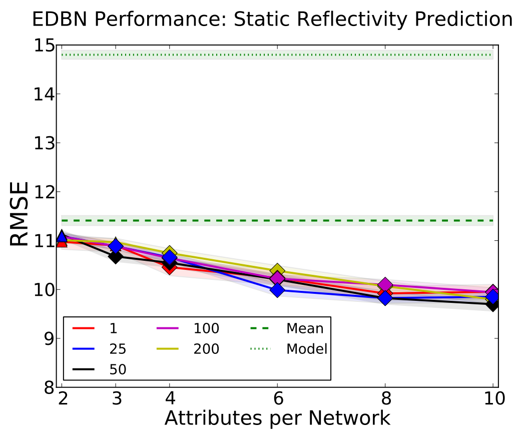
In the static domain figures (Figures 5.1, 5.2, and 5.3), parameter settings that are significantly better than the model predictions are marked with a triangle, and parameter settings that are significantly better than the mean predictor are marked with a diamond. We find that almost all settings are significantly better than the model predictions. The settings that are not significantly better all occur at 10 attributes per network, indicating that performance is hurt when



(a) 100 Training Examples per Network

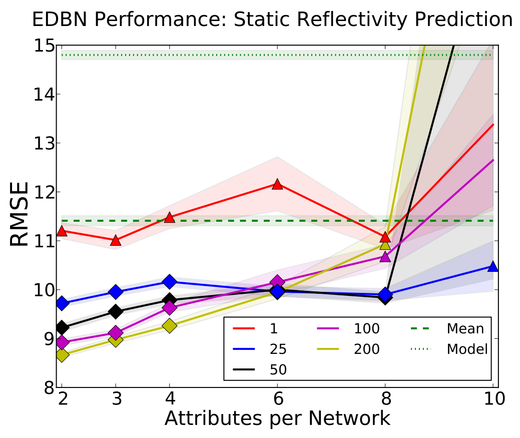


(b) 1000 Training Examples per Network

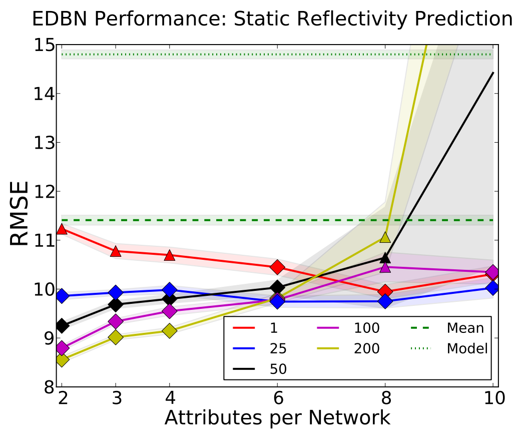


(c) 2000 Training Examples per Network

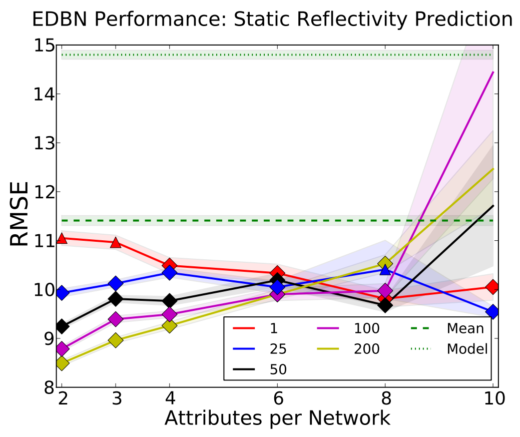
Figure 5.1: *RMSE for the static Reflectivity domain without least-squares weighted prediction. Shaded area represents standard error.*



(a) 100 Training Examples per Network

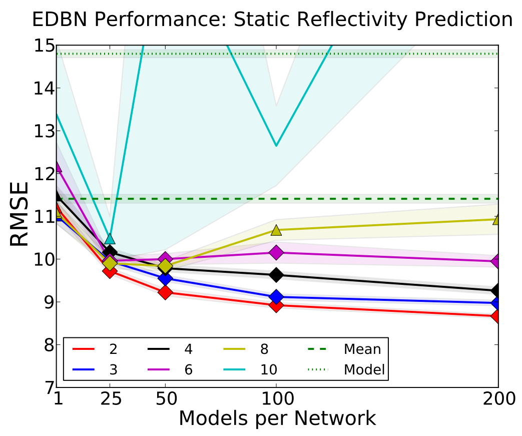


(b) 1000 Training Examples per Network

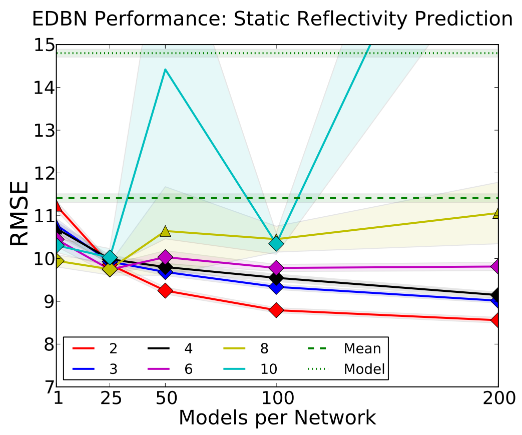


(c) 2000 Training Examples per Network

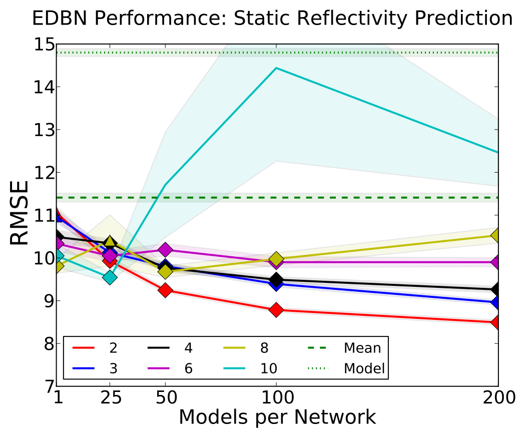
Figure 5.2: *RMSE for the static Reflectivity domain at various parameter settings with least-squares weigted prediction.*



(a) 100 Training Examples per Network

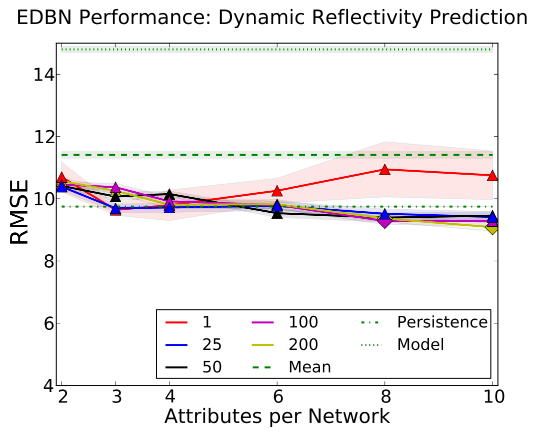


(b) 1000 Training Examples per Network

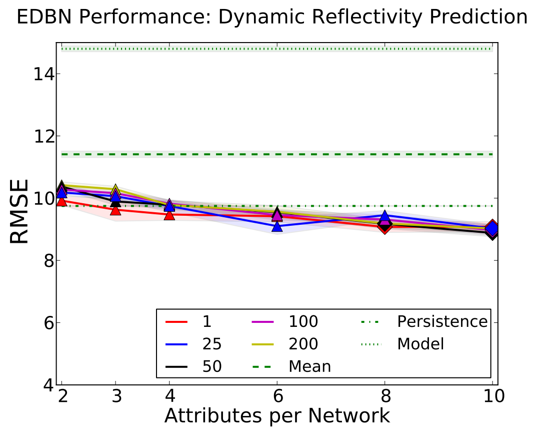


(c) 2000 Training Examples per Network

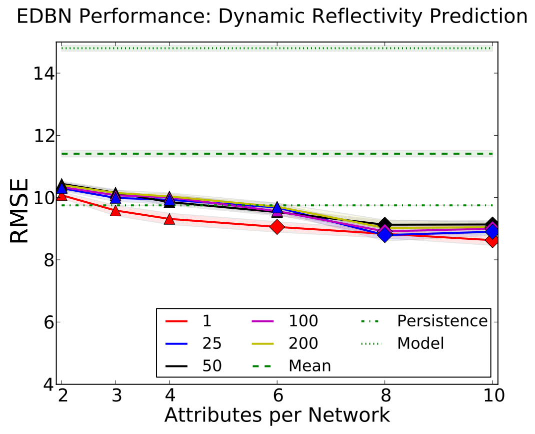
Figure 5.3: *RMSE for the static Reflectivity domain at various parameter settings with least-squares weigted prediction. Lines are attribute counts.*



(a) 100 Training Examples per Network

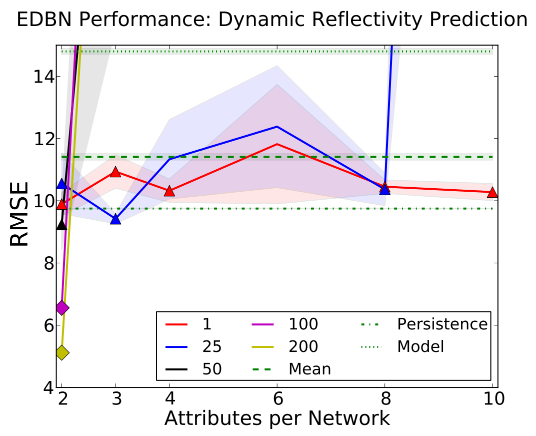


(b) 1000 Training Examples per Network

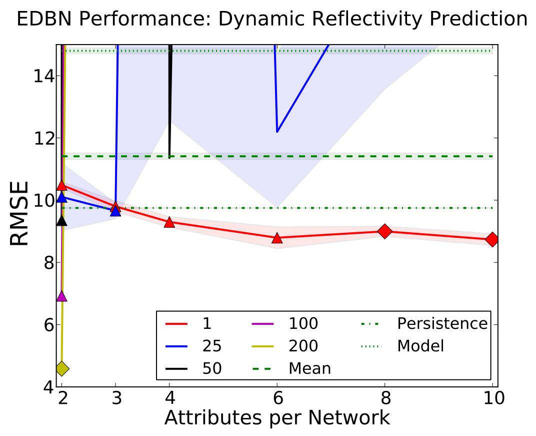


(c) 2000 Training Examples per Network

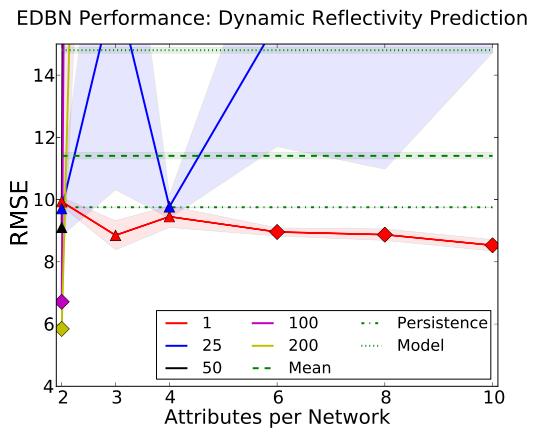
Figure 5.4: *RMSE for the dynamic Reflectivity domain at various parameter settings without least-squares weighted prediction.*



(a) 100 Training Examples per Network

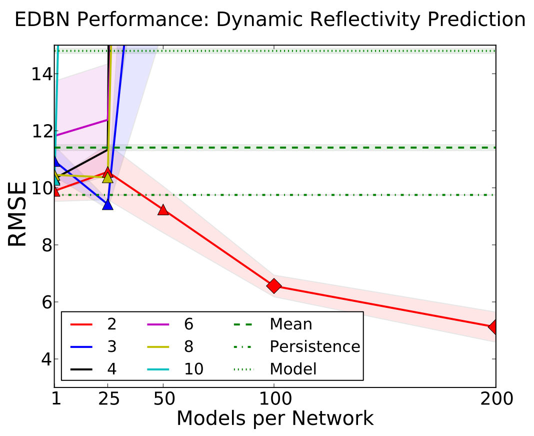


(b) 1000 Training Examples per Network

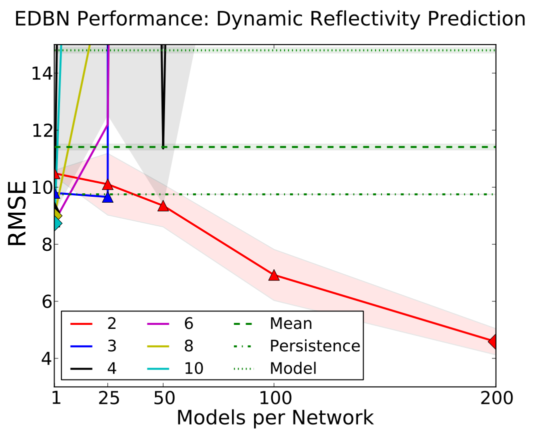


(c) 2000 Training Examples per Network

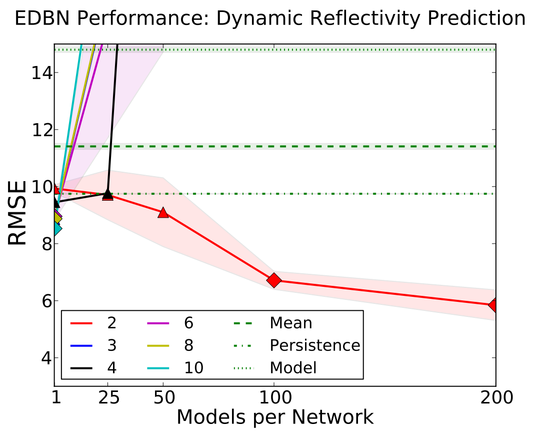
Figure 5.5: *RMSE for the dynamic Reflectivity domain at various parameter settings with least-squares weighted prediction.*



(a) 100 Training Examples per Network



(b) 1000 Training Examples per Network



(c) 2000 Training Examples per Network

Figure 5.6: *RMSE for the dynamic Reflectivity domain at various parameter settings with least-squares weigted prediction. Lines are attribute counts.*

EDBN attempts to represent complex relationships. Similar results are seen when we compare to the mean predictor. For unweighted predictions, EDBN does not perform significantly better than the mean predictor only at one model or at low attribute counts. For weighted predictions, the behavior of one model is the same, and non-significant results only occur large attribute counts. This agrees with our hypothesis from Chapter 4 that increasing attribute counts leads to overfitting when using weighted prediction.

We compare our best unweighted parameter setting (10 attributes, 50 models, and 2000 training examples) to our best weighted parameter setting (2 attributes, 200 models, and 2000 training examples) using a one-tailed two-sample t-test and find that the weighted prediction performs significantly better ($p = 3 \times 10^{-11}$).

In the dynamic domain figures (Figures 5.4, 5.5, 5.6), parameter settings that are significantly better than the model predictions are marked with a triangle, and parameter settings that are significantly better than the persistence prediction are marked with a diamond. For unweighted prediction, we find that all parameter settings perform significantly better than the model prediction. We also find that EDBN only performs significantly better than the persistence prediction at higher attribute counts. For weighted prediction, we find that at model counts higher than 25, all attribute counts higher than 3 do not perform significantly better than either the model prediction or the persistence prediction. However, performance at 2 attributes is significantly better than the persistence prediction at high model counts. This reinforces our hypothesis that EDBN with weighted prediction overfits as model complexity increases and indicates that some form of regularization is needed.

We compare our best unweighted parameter setting (10 attributes, 1 model, and 2000 training examples) to our best weighted parameter setting (2 attributes, 200 models, and 1000 training examples) using a one-tailed two-sample t-test and find that the weighted prediction performs significantly better ($p = 2 \times 10^{-8}$).

From these results, we conclude that for both the static and dynamic version of the Reflectivity domain, EDBN performs well. Both the weighted and unweighted predictions can perform better than the model predictions and persistence predictions, but weighted prediction is the superior approach despite the problematic overfitting.

5.2 Variable Importance and Edge Counts

We report the variable importance scores for networks of 10 attributes and 2000 samples, due to this parameter setting's effectiveness at non-weighted prediction. The top variables for static prediction are shown in Table 5.3. As we would expect, the forecast reflectivity is among the top variables, as is precipitable water. Similarly, the dynamic results (Table 5.4) are dominated by reflectivity. The only non-reflectivity variable is upward long-wave radiation flux, which is also found in the top scores for static prediction. This information, combined with our above prediction results, verifies our natural expectation that knowledge of previous reflectivity and reflectivity forecasts is important for predicting current reflectivity.

The edge graphs for static and dynamic reflectivity prediction are shown in Figures 5.7 and 5.8. As with our results in Chapter 4, these edge graphs are

Variable	z-score
Composite Reflectivity (Forecast Hour 1)	24.0
Composite Reflectivity (Forecast Hour 3)	21.9
Upward Long-Wave Radiation Flux (Forecast Hour 1)	20.7
Composite Reflectivity (Forecast Hour 2)	20.0
Upward Long-Wave Radiation Flux (Forecast Hour 3)	16.2
Upward Long-Wave Radiation Flux (Forecast Hour 2)	16.0
Precipitable Water (Forecast Hour 1)	14.7
Wind Speed (Forecast Hour 1)	11.8
Precipitable Water (Forecast Hour 2)	11.6
Precipitable Water (Forecast Hour 3)	11.2

Table 5.3: *Top 10 Variable Importance scores for the static Reflectivity domain.*

almost complete. In both graphs, the observed reflectivity is fully connected to the rest of the graph.

5.3 Vertically Integrated Liquid

In addition to the Reflectivity dataset, we also have observed Vertically Integrated Liquid (VIL) values for the same time period. Since EDBN performs well at reflectivity prediction, we anticipate that it will perform well at VIL prediction as well, and at similar parameter settings.

We colocated the VIL values with the Reflectivity examples by nearest neighbor. We ran a limited set of experiments to verify that EDBN performs well in a similar domain to Reflectivity, and to investigate how the variable importance

Variable	z-score
Motion Adjusted Reflectivity (1 Hour Old) (Previous)	27.9
True Reflectivity (2 Hours Old)	23.1
True Reflectivity (1 Hours Old)	20.0
Composite Reflectivity (Forecast Hour 2)	14.6
Composite Reflectivity (Forecast Hour 1)	13.7
True Reflectivity (3 Hours Old)	12.9
Upward Long-Wave Radiation Flux (Forecast Hour 1)	12.9
Composite Reflectivity (Forecast Hour 4)	12.4
Composite Reflectivity (Forecast Hour 3)	11.8
Upward Long-Wave Radiation Flux (Forecast Hour 2)	11.7

Table 5.4: *Top 10 Variable Importance scores for the dynamic Reflectivity domain.*

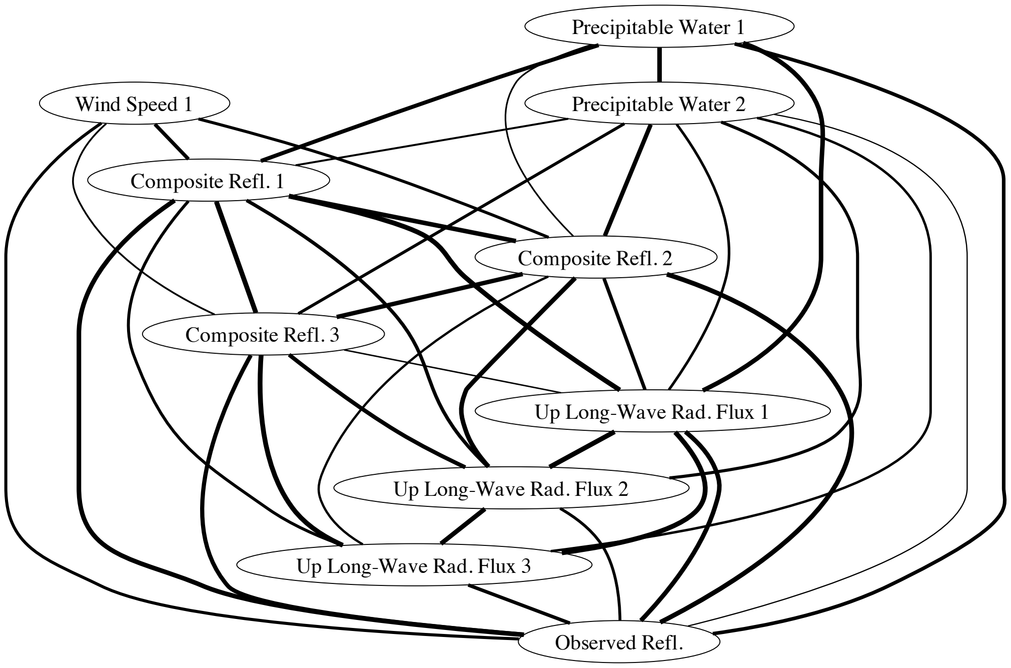


Figure 5.7: *Edge graph of a subset of the attributes in the static Reflectivity. Edge thickness indicates how often a pair of edges appeared in the ensemble, with the thinnest being edges that occurred 90% of the time and thicker edges occurring more frequently.*

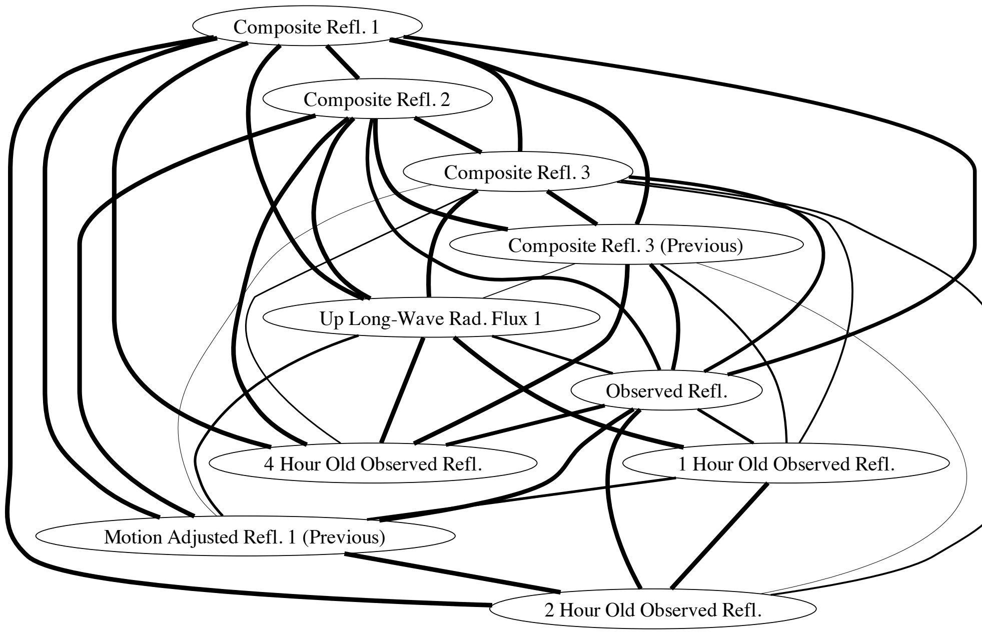

Composite Refl. 1
Composite Refl. 2
Composite Refl. 3
Composite Refl. 3 (Previous)
Up Long-Wave Rad. Flux 1
Observed Refl.
4 Hour Old Observed Refl.
1 Hour Old Observed Refl.
Motion Adjusted Refl. 1 (Previous)
2 Hour Old Observed Refl.

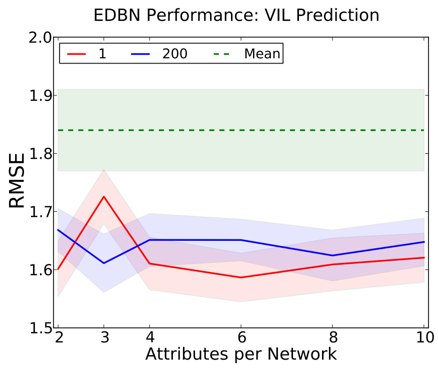
Figure 5.8: *Edge graph of a subset of the attributes the dynamic Reflectivity domain.*

and edge graphs compare to those for Reflectivity. We do not vary training set size in these experiments, and instead fix it to 2000 examples.

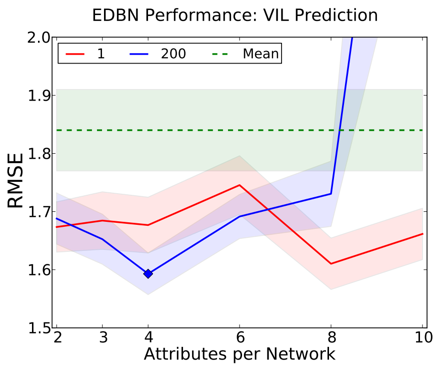
Results for VIL prediction are shown in Figure 5.9. The best results occur at 4 attributes per network with 200 networks, using weighted prediction. As in our other results, overfitting can be seen in the weighted predictions as the attribute count increases. Similar to our unweighted reflectivity results, the difference between 1 model and 200 models is slight.

As both unweighted and weighted prediction achieve similar best performance, we do not compare unweighted and weighted predictions to each other. This gives 24 parameter settings, so we correct our α from 0.05 to 0.021. In Figure 5.9, diamonds indicate parameter settings that perform significantly better than the mean predictor. We find that only one parameter setting performs significantly better than the mean predictor: 200 models with 4 attributes per model. From this we conclude that EDBN performs better than the mean predictor for VIL. However, our results also indicate that VIL prediction using the Reflectivity dataset is difficult, as only one parameter setting does significantly better and 1 model ensembles and 200 models ensembles perform comparably.

The variable importance results for VIL prediction are shown in Table 5.5. All three values of precipitable water are present, which we would expect since VIL measures the amount of liquid in a column of air and precipitable water measures the amount of water vapor in a column of air. As in the reflectivity experiments, composite reflectivity and upward long-wave radiation flux are present. In light of the edge graph for VIL (Figure 5.10), upward long-wave radiation in particular appears to be important. In the edge graph, it is the only vertex that is connected to VIL, indicating that the relationship of the other variables to VIL is more indirect.



(a) Unweighted Prediction



(b) Weighted Prediction

Figure 5.9: *RMSE for VIL prediction.*

Variable	z-score
Upward Long-Wave Radiation Flux (Forecast Hour 2)	8.67
Upward Long-Wave Radiation Flux (Forecast Hour 1)	7.31
Composite Reflectivity (Forecast Hour 3)	6.71
Composite Reflectivity (Forecast Hour 1)	6.66
Upward Long-Wave Radiation Flux (Forecast Hour 3)	6.32
Composite Reflectivity (Forecast Hour 2)	5.58
Precipitable Water (Forecast Hour 2)	4.30
Precipitable Water (Forecast Hour 1)	4.21
Precipitable Water (Forecast Hour 3)	4.02
Surface Pressure (Forecast Hour 1)	3.61

Table 5.5: *Top 10 Variable Importance scores for VIL prediction.*

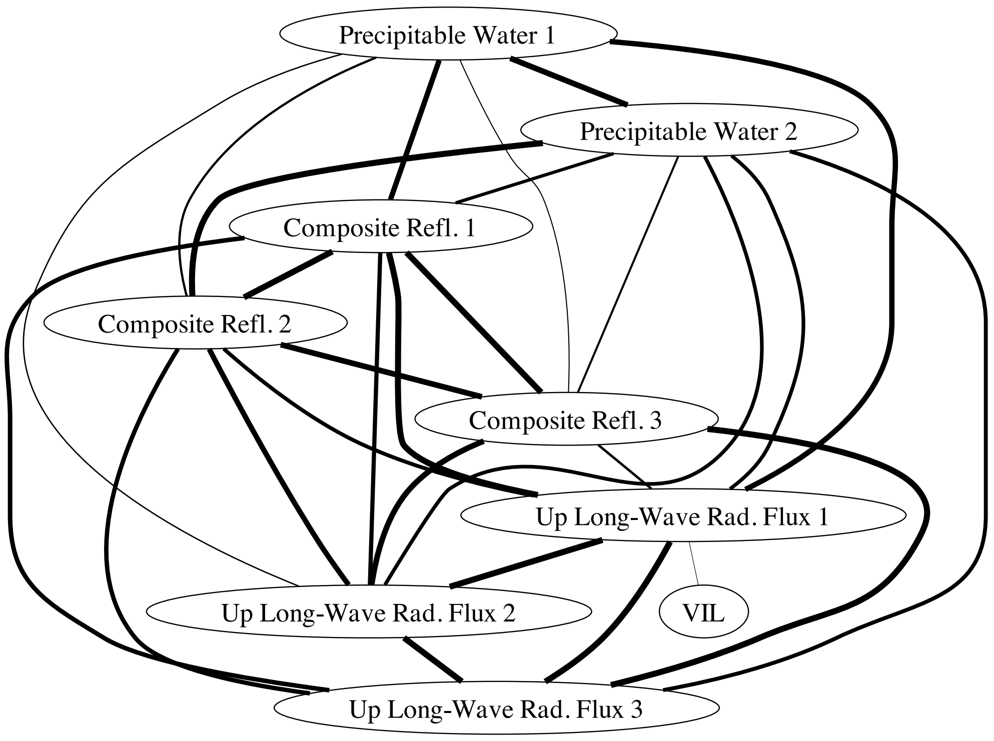
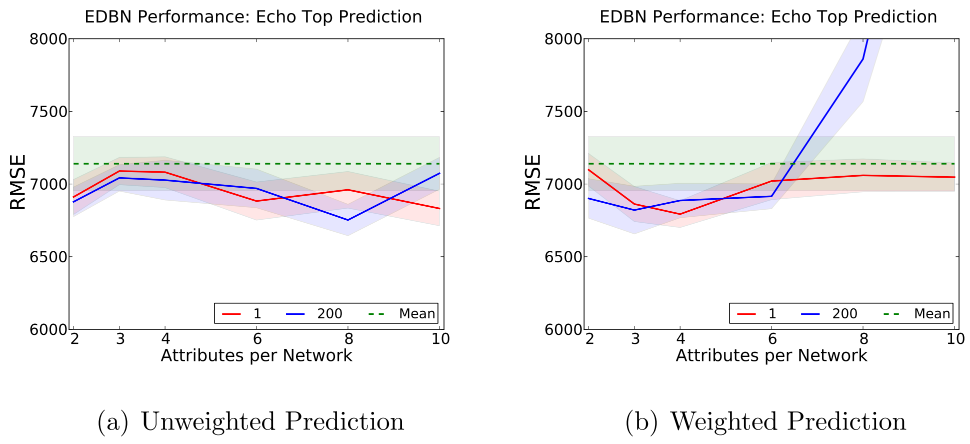


Figure 5.10: *Edge graph of a subset of the attributes used in VIL prediction.*

5.4 Echo Top

Finally, we also have observed echo top measurements for the time period of the Reflectivity dataset. As before, we anticipate that EDBN will perform well at echo top prediction. Data preparation and experimental design is the same as in Section 5.3.

Results for echo top prediction are shown in Figure 5.11. Overfitting is once again present in the weighted predictions, and the difference between weighted and unweighted predictions is minimal for both weighted and unweighted predictions. We follow the test procedure from Section 5.3, and we indicate significant parameter settings with a diamond in Figure 5.11. However, we find that all



(a) Unweighted Prediction (b) Weighted Prediction

Figure 5.11: *RMSE for echo top prediction.*

parameter settings, EDBN does not perform significantly better than the mean predictor. As with VIL prediction, we conclude that echo top prediction using the Reflectivity dataset is a difficult proposition.

Despite the poor predictive performance, we can still use EDBN to investigate how echo tops prediction compares to reflectivity and VIL prediction. The variable importance results are shown in Table 5.6. Once again, composite reflectivity, precipitable water, and upward long-wave radiation are all present. The edge graph is shown in Figure 5.12. Similar to VIL, echo top is only directly connected to the upward long-wave radiation flux vertices. Also of note is that convective inhibition and longitude are both absent from the reflectivity and VIL results, but present here, and in the edge graph they are connected only to each other. This indicates that they have a direct, definite, but weak impact on echo top prediction. Direct because they do not appear in the reflectivity and VIL results, definite because they are among the top variable importance scores for echo top, and weak because at the 90% threshold there is no edge from either convective inhibition or longitude to echo top in the edge graph.

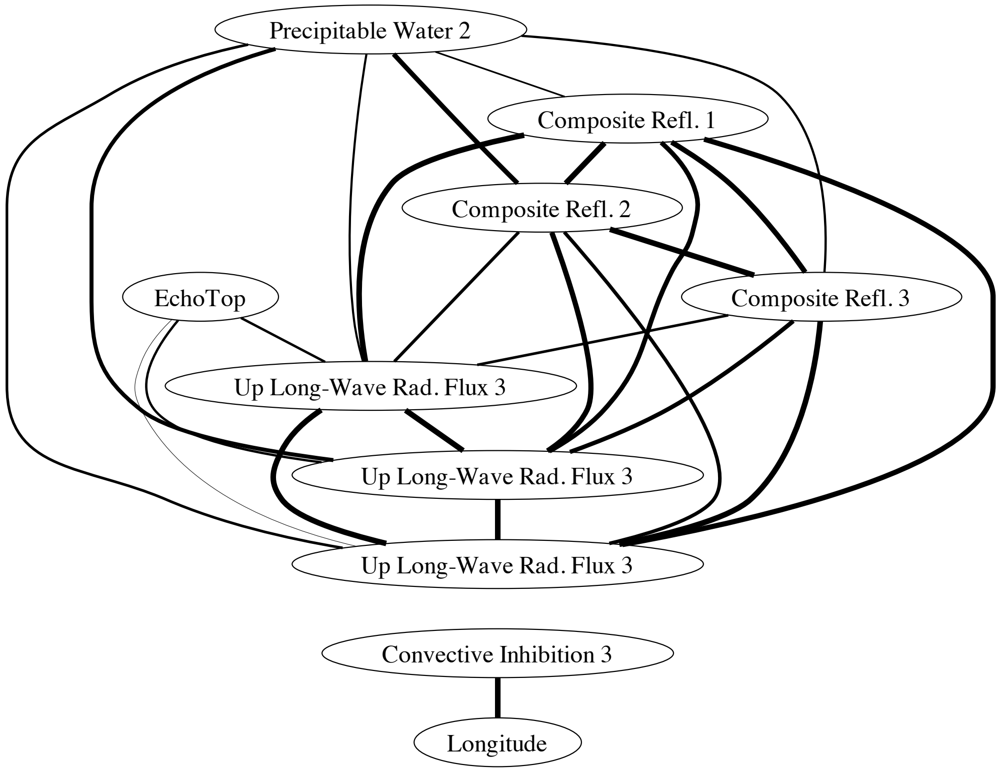


Figure 5.12: *Edge graph of a subset of the attributes used for echo top prediction.*

Variable	z-score
Upward Long-Wave Radiation Flux (Forecast Hour 3)	16.2
Upward Long-Wave Radiation Flux (Forecast Hour 2)	15.1
Upward Long-Wave Radiation Flux (Forecast Hour 1)	14.4
Composite Reflectivity (Forecast Hour 2)	12.9
Composite Reflectivity (Forecast Hour 3)	12.5
Composite Reflectivity (Forecast Hour 1)	10.7
Longitude	9.0
Convective Inhibition (Forecast Hour 3)	6.3
Precipitable Water (Forecast Hour 2)	5.6
Precipitable Water (Forecast Hour 3)	5.3

Table 5.6: *Top 10 Variable Importance scores for echo top prediction.*

Chapter 6

Conclusions and Future Work

We introduced Ensembled Dynamic Bayesian Networks, an approach that allows for the prediction of continuous random variables, the discovery of important relationships in a dataset, and the discovery of important variables for prediction. We have empirically shown that EDBN, in combination with learned weights on the individual network predictions, performs well on two real-world meteorological domains. We have also demonstrated that through variable importance and edge graphs, EDBN provides insights into the dependence relationships within a domain. EDBN is able to predict rainfall better than the mean predictor, and is able to predict reflectivity better than the model predictions, the mean predictor, and the persistence predictor. Our results indicate that EDBN is capable of performing well in large domains featuring many variables and complex relationships between those variables.

Our current approach features learned weights that are prone to overfitting. As discussed in the results, regularization is a likely solution to this problem (Breiman 1998). Instead of using linear least-squares regression to learn our weights, we could use ridge regression (Hoerl and Kennard 1970), LASSO regression (Tibshirani 1994), or some other regularized method. These approaches include extra terms when learning weights which penalize complex models. This penalization helps to prevent overfitting to the training data.

As we have shown, EDBN handles continuous data with nonlinear relationships well, but has no way of working with discrete random variables. Utz (2010) demonstrated that discrete Bayesian Networks can be ensembled effectively, and so a natural extension of EDBN would be to allow for hybrid networks. Hybrid networks are Bayesian Networks that contain both discrete and continuous variables. However, this would come at the expense of more difficult structure learning, parameter learning, and inference.

The fact that EDBNs are Gaussian Mixture Models suggests that expectation-maximization (EM) could be used to learn the parameters of an EDBN. This could eliminate the need for least-squares regression on the whole ensemble. Preliminary work indicates that EM performs well on small datasets, but overfits on more complex domains. We hypothesize that this could be overcome by learning ensembles of EDBNs trained with EM. Since EM learns all parameters for an ensemble simultaneously, the individual networks become dependent on one another. Ensembles of EM EDBNs would result in a top-level ensemble that consists of independent models.

In this work, we performed structure learning by finding a skeleton with the Fast Adjacency Search and then using hill-climbing to find a locally optimal network that matched the skeleton. Many other structure learning algorithms exist, and it would be interesting to see the effect of using other structure learning techniques on the performance of EDBN. In preliminary work, we implemented the Max-Min Hill Climbing algorithm (MMHC) (Tsamardinos et al. 2006), which resulted in worse prediction performance. While we did not perform full experiments, and thus cannot claim that FAS is better than MMHC for EDBN, these results do indicate that the specific structure learning algorithm can have a meaningful impact on prediction performance.

Finally, EDBN could be extended to include spatial information and relational information. Both domains discussed in this thesis implicitly included spatial and relational information that we ignored. For example, we would expect that the amount of rainfall in one location would be useful knowledge for predicting the rainfall at a nearby location. Bayesian Networks have been applied to spatial domains before (e.g., Dereszynski and Dietterich 2011), and such an approach could be extended into an ensemble of networks. Knowledge of relations, such as whether a storm is nearby another storm, is also useful. Bayesian Networks that are capable of representing relationships have been developed (e.g., Maier et al. 2010; Jaeger 1997), and could also be ensembled in the much same way that EDBN is.

Reference List

Albert, J., E. Aliu, H. Anderhub, P. Antoranz, A. Armada, M. Asensio, C. Baixeras, J. Barrio, H. Bartko, D. Bastieri, J. Becker, W. Bednarek, K. Berger, C. Bigongiari, A. Biland, R. Bock, P. Bordas, V. Bosch-Ramon, T. Bretz, I. Britvitch, M. Camara, E. Carmona, A. Chilingarian, S. Ciprini, J. Coarasa, S. Commichau, J. Contreras, J. Cortina, M. Costado, V. Curtef, V. Danielyan, F. Dazzi, A. D. Angelis, C. Delgado, R. de los Reyes, B. D. Lotto, E. Domingo-Santamara, D. Dorner, M. Doro, M. Errando, M. Fagiolini, D. Ferenc, E. Fernndez, R. Firpo, J. Flix, M. Fonseca, L. Font, M. Fuchs, N. Galante, R. Garca-Lpez, M. Garczarczyk, M. Gaug, M. Giller, F. Goebel, D. Hakobyan, M. Hayashida, T. Hengstebeck, A. Herrero, D. Hhne, J. Hose, S. Huber, C. Hsu, P. Jacon, T. Jogler, R. Kosyra, D. Kranich, R. Kritzer, A. Laille, E. Lindfors, S. Lombardi, F. Longo, J. Lpez, M. Lpez, E. Lorenz, P. Majumdar, G. Maneva, K. Mannheim, M. Mariotti, M. Martnez, D. Mazin, C. Merck, M. Meucci, M. Meyer, J. Miranda, R. Mirzoyan, S. Mizobuchi, A. Moralejo, D. Nieto, K. Nilsson, J. Ninkovic, E. Oa-Wilhelmi, N. Otte, I. Oya, M. Panniello, R. Paoletti, J. Paredes, M. Pasanen, D. Pascoli, F. Pauss, R. Pegna, M. Persic, L. Peruzzo, A. Piccioli, N. Puchades, E. Prandini, A. Raymers, W. Rhode, M. Rib, J. Rico, M. Rissi, A. Robert, S. Rgamer, A. Saggion, T. Saito, A. Snchez, P. Sartori, V. Scalzotto, V. Scapin, R. Schmitt, T. Schweizer, M. Shayduk, K. Shinozaki, S. Shore, N. Sidro, A. Sillanp, D. Sobczynska, F. Spanier, A. Stamerra, L. Stark, L. Takalo, P. Temnikov, D. Tescaro, M. Teshima, D. Torres, N. Turini, H. Vankov, A. Venturini, V. Vitale, R. Wagner, T. Wibig, W. Wittek, F. Zandanel, R. Zanin, and J. Zapatero, 2008: Implementation of the random forest method for the imaging atmospheric cherenkov telescope MAGIC. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, **588**, 424 – 432.

Baba, K., R. Shibata, and M. Sibuya, 2004: Partial correlation and conditional correlation as measures of conditional independence. *Australian and New Zealand Journal of Statistics*, **46**, 657–664.

Bilmes, J., 2000: Dynamic Bayesian multinets. *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, UAI '00, 38–45.

Bøttcher, S. G., 2004: *Learning Bayesian Networks with Mixed Variables*. Ph.D. thesis, AALBORG University.

Breiman, L., 1996: Bagging predictors. *Machine Learning*, **24**, 123–140.

—, 1998: Bias-variance, regularization, instability and stabilization. *Neural Networks and Machine Learning*, 27–560.

—, 2001: Random forests. *Machine Learning*, **45**, 5–32.

Breiman, L. and A. Cutler, 2012: Random forests - classification manual. URL `http://www.math.usu.edu/ adele/forests/`

Bremnes, J. B., 2004: Probabilistic forecasts of precipitation in terms of quantiles using NWP model output. *Monthly Weather Review*, **132**, 338–347.

Chan, J. C.-W. and D. Paelinckx, 2008: Evaluation of random forest and adaboost tree-based ensemble classification and spectral band selection for ecotope mapping using airborne hyperspectral imagery. *Remote Sensing of Environment*, **112**, 2999 – 3011.

Cheng, J., R. Greiner, J. Kelly, D. Bell, and W. Liu, 2002: Learning Bayesian networks from data: an information-theory based approach. *Artificial Intelligence*, **137**, 43–90.

Chickering, D. M., 1996: Learning Bayesian networks is NP-complete. *Learning from Data: Artificial Intelligence and Statistics V*, Springer-Verlag, 121–130.

Cooper, G. F., 1990: The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, **42**, 393 – 405.

Cooper, G. F. and T. Dietterich, 1992: A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 309–347.

Dean, T. and K. Kanazawa, 1989: A model for reasoning about persistence and causation. *Computational Intelligence*, **5**, 142–150.

Dereszynski, E. W. and T. G. Dietterich, 2011: Spatiotemporal models for data-anomaly detection in dynamic environmental monitoring campaigns. *ACM Transactions Sensor Networks*, **8**, 3:1–3:36.

Diaz-Uriarte, R. and S. Alvarez de Andres, 2006: Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, **7**, 3.

Dietterich, T. G., 2000: Ensemble methods in machine learning. *International Workshop on Multiple Classifier Systems*, Springer-Verlag, 1–15.

Ebert, E., 2001: Ability of a poor man's ensemble to predict the probability and distribution of precipitation. *Monthly Weather Review*, **129**, 2461–2480.

Fast, A., 2009: *Learning the Structure of Bayesian Networks with Constraint Satisfaction*. Ph.D. thesis, University of Massachusetts Amherst.

Fast, A., M. Hay, and D. Jensen, 2008: Statistical power analysis for improved learning Bayesian network structure.

Fisher, R. A., 1915: Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population. *Biometrika*, **10**, 507–521.

Freund, Y. and R. E. Schapire, 1995: A decision-theoretic generalization of on-line learning and an application to boosting.

Friedman, N., M. Goldszmidt, and T. J. Lee, 1998: Bayesian network classification with continuous attributes: Getting the best of both discretization and parametric fitting. *In Proceedings of the International Conference on Machine Learning*, Morgan Kaufmann, 179–187.

Friedman, N. and I. Nachman, 2000: Gaussian process networks. *Proceedings of the Sixteenth conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, UAI'00, 211–219.

Fung, R. M. and K.-C. Chang, 1989: Weighing and integrating evidence for stochastic simulation in Bayesian networks. *Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence*, North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, UAI '89, 209–220.

Gagne II, D. J., 2012: *Machine Learning Enhancement of Storm Scale Ensemble Precipitation Forecasts*. Master's thesis, University of Oklahoma.

Geiger, D. and D. Heckerman, 1994: Learning gaussian networks. *Proceedings of the Tenth international conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, UAI'94, 235–243.

—, 1996: Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence*, **82**, 45 – 74.

Geman, S. and D. Geman, 1984: Readings in computer vision: issues, problems, principles, and paradigms. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, chapter Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, 564–584.

Ghahramani, Z., 1998: Learning dynamic Bayesian networks. *Adaptive Processing of Sequences and Data Structures*, Springer-Verlag, 168–197.

Grzegorczyk, M. and D. Husmeier, 2009: Non-stationary continuous dynamic Bayesian networks. *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, eds., 682–690.

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

Hoerl, A. E. and R. W. Kennard, 1970: Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, **12**, 55–67.

Hofmann, R. and V. Tresp, 1995: Discovering structure in continuous variables using Bayesian networks. *Advances in Neural Information Processing Systems 8*, MIT Press, 500–506.

Jaeger, M., 1997: Relational Bayesian networks. Morgan Kaufmann, 266–273.

Jing, Y., V. Pavlović, and J. M. Rehg, 2008: Boosted Bayesian network classifiers. *Machine Learning*, **73**, 155–184.

Kong, F., M. Xue, K. W. Thomas, Y. Wang, K. Brewster, X. Wang, J. Gao, S. J. Weiss, J. S. Kain, and J. Du, 2011: CAPS multi-model storm-scale ensemble forecast for the NOAA HWT 2010 spring experiment. *24th Conference on Weather Forecasting/20th Conference Numerical Weather Prediction Seattle, WA*, American Meteorological Society, PAPER 457.

Liu, F., F. Tian, and Q. Zhu, 2007: Ensembling Bayesian network structure learning on limited data. *CIKM '07: Proceedings of the sixteenth ACM conference on information and knowledge management*, Association of Computing Machinary, New York, NY, USA, 927–930.

Maier, M., B. Taylor, H. Oktay, and D. Jensen, 2010: Learning causal models of relational domains.

McGovern, A., D. John Gagne, II, N. Troutman, R. A. Brown, J. Basara, and J. K. Williams, 2010: Understanding severe weather processes through spatiotemporal relational random forests. *Proceedings of the NASA Conference on Intelligent Data Understanding*, CIDU.

Nodelman, U., C. Shelton, and D. Koller, 2002: Continuous time Bayesian networks. *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, 378–387.

Pearl, J., 1988: *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

—, 2000: *Causality. Models, Reasoning, and Inference*. Cambridge University Press.

Pérez, A., P. Larraòaga, and I. Inza, 2009: Bayesian classifiers based on kernel density estimation: Flexible classifiers. *International Journal of Approximate Reasoning*, **50**, 341–362.

Pielke Jr, R. and R. E. Carbone, 2002: Weather impacts, forecasts and policy: An integrated perspective. *Bulletin of the American Meteorological Society*, 393–403.

Russell, S. and P. Norvig, 2003: *Artificial Intelligence: A Modern Approach*, Prentice-Hall, Englewood Cliffs, NJ, chapter Probabilistic Reasoning over Time. 2nd edition edition, 537–583.

Shachter, R. and C. Kenley, 1989: Gaussian influence diagrams. *Management Science*, **35**, 527–550.

Shachter, R. D. and M. Peot, 1989: Simulation approaches to general probabilistic inference on belief networks. *Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann Publishers Inc., UAI '89, 209–220.

Spirtes, P., C. Glymour, and R. Scheines, 2001: *Causation, Prediction, and Search*. The MIT Press, Cambridge, MA, USA, second edition.

Strobl, C., A.-L. Boulesteix, T. Kneib, T. Augistin, and A. Zeileis, 2008: Conditional variable importance for random forests. Technical report, Department of Statistics, University of Munich.

Strobl, C. and A. Zeileis, 2008: Danger, high power! - exploring the statistical properties of a test for random forest variable importance. Technical report, Department of Statistics, University of Munich.

Sung, H., 2004: *Gaussian Mixture Regression and Classification*. Ph.D. thesis, Rice University.

Supinie, T., A. Mcgovern, J. Williams, and J. Abernethy, 2009: Spatiotemporal relational random forests. *Proceedings of the 2009 IEEE International Conference on Data Mining (ICDM) workshop on Spatiotemporal Data Mining.*.

Tibshirani, R., 1994: Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, **58**, 267–288.

Tsamardinos, I., L. Brown, and C. Aliferis, 2006: The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, Kluwer Academic Publishers, Hingham, MA, USA, volume 65, 31–78.

Utz, C., 2010: *Learning Ensembles of Bayesian Network Structures Using Random Forest Techniques*. Master's thesis, University of Oklahoma.

Xue, M., F. Kong, K. W. Thomas, Y. Wang, K. Brewster, J. Gao, X. Wang, S. J. Weiss, A. J. Clark, J. S. Kain, M. C. Coniglio, J. Du, L. Jensen, and Y. H.

Kuo, 2011: CAPS realtime storm scale ensemble and high resolution forecasts for the NOAA hazardous weather testbed 2010 spring experiment. *24th Conference on Weather Forecasting/20th Conference on Numerical Weather Prediction Seattle, WA*, American Meteorological Society, PAPER 9A.2.