Journal of Machine Learning Research 10? (2005?) 1-48? Submitted 10/05; Revised 5/06; Published 00/06?

Bayesian Network Learning with Parameter Constraints

Radu Stefan Niculescu STEFAN.NICULESCU@SIEMENS.COM
Computer Aided Diagnosis and Therapy Group
Siemens Medical Solutions
51 Valley Stream Parkway
Malvern, PA, 19355, USA

Tom M. Mitchell TOM.MITCHELL@CMU.EDU
Center for Automated Learning and Discovery
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, 15213, USA

R. Bharat Rao BHARAT.RAO@SIEMENS.COM
Computer Aided Diagnosis and Therapy Group
Siemens Medical Solutions
51 Valley Stream Parkway
Malvern, PA, 19355, USA

Editor: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

The task of learning models for many real-world problems requires incorporating domain knowledge into learning algorithms, to enable accurate learning from a realistic volume of training data. This paper considers a variety of types of domain knowledge for constraining parameter estimates when learning Bayesian Networks. In particular, we consider domain knowledge that constrains the values or relationships among subsets of parameters in a Bayesian Network with known structure.

We incorporate a wide variety of parameter constraints into learning procedures for Bayesian Networks, by formulating this task as a constrained optimization problem. The assumptions made in Module Networks, Dynamic Bayes Nets and Context Specific Independence models can be viewed as particular cases of such parameter constraints. We present closed form solutions or fast iterative algorithms for estimating parameters subject to several specific classes of parameter constraints, including equalities and inequalities among parameters, constraints on individual parameters, and constraints on sums and ratios of parameters, for discrete and continuous variables. Our methods cover learning from both frequentist and Bayesian points of view, from both complete and incomplete data.

We present formal guarantees for our estimators, as well as methods for automatically learning useful parameter constraints from data. To validate our approach, we apply it to the domain of fMRI brain image analysis. Here we demonstrate the ability of our system to first learn useful relationships among parameters, and then to use them to constrain the training of the Bayesian Network, resulting in improved cross-validated accuracy of the learned model. Experiments on synthetic data are also presented.

Keywords: Bayesian Networks, Constrained Optimization, Domain Knowledge

1. Introduction

Probabilistic models have become increasingly popular in the last decade because of their ability to capture non-deterministic relationships among variables describing many real world domains. Among these models, Graphical Models have received significant attention because of their ability to compactly encode conditional independence assumptions over random variables and because of the development of effective algorithms for inference and learning based on these representations.

A Bayesian Network (Heckerman, 1999) is a particular case of a Graphical Model that compactly represents the joint probability distribution over a set of random variables. It consists of two components: a structure and a set of parameters. The structure is a Directed Acyclic Graph with one node per variable. This structure encodes the Local Markov Assumption: a variable is conditionally independent of its non-descendants in the network, given the value of its parents. The parameters describe how each variable relates probabilistically to its parents. A Bayesian Network encodes a unique joint probability distribution, which can be easily computed using the chain rule.

When learning Bayesian networks, the correctness of the learned network of course depends on the amount of training data available. When training data is scarce, it is useful to employ various forms of prior knowledge about the domain to improve the accuracy of learned models. For example, a domain expert might provide prior knowledge specifying conditional independencies among variables, constraining or even fully specifying the network structure of the Bayesian Network. In addition to helping specify the network structure, the domain expert might also provide prior knowledge about the values of certain parameters in the conditional probability tables (CPTs) of the network, or knowledge in the form of prior distributions over these parameters. While previous research has examined a number of approaches to representing and utilizing prior knowledge about Bayesian Network parameters, the type of prior knowledge that can be utilized by current learning methods remains limited, and is insufficient to capture many types of knowledge that may be readily available from experts.

One contribution of our previous work (Niculescu, 2005) was the development of a general framework to perform parameter estimation in Bayesian Networks in the presence of any parameter constraints that obey certain differentiability assumptions, by formulating this as a constrained maximization problem. In this framework, algorithms were developed from both a frequentist and a Bayesian point of view, for both complete and incomplete data. The optimization methods used by our algorithms are not new and therefore this general learning approach has serious limitations given by these methods, especially given arbitrary constraints. However, this approach constitutes the basis for the efficient learning procedures for specific classes of parameter constraints described in this paper. Applying these efficient methods allows us to take advantage of parameter constraints provided by experts or learned by the program, to perform more accurate learning of very large Bayesian Networks (thousands of variables) based on very few (tens) examples, as we will see later in the paper.

The main contribution of our paper consists of efficient algorithms (closed form solutions, fast iterative algorithms) for several classes of parameter constraints which current methods can not accommodate. We show how widely used models including Hidden Markov

Models, Dynamic Bayesian Networks, Module Networks and Context Specific Independence are special cases of one of our constraint types, described in subsection 4.3. Our framework is able to represent parameter sharing assumptions at the level of granularity of individual parameters. While prior work on parameter sharing and Dirichlet Priors can only accommodate simple equality constraints between parameters, our work extends to provide closed form solutions for classes of parameter constraints that involve relationships between groups of parameters (sum sharing, ratio sharing). Moreover, we provide closed form Maximum Likelihood estimators when constraints come in the form of several types of inequality constraints. With our estimators come a series of formal guarantees: we formally prove the benefits of taking advantage of parameter constraints to reduce the variance in parameter estimators and we also study the performance in the case when the domain knowledge represented by the parameter constraints might not be entirely accurate. Finally, we present a method for automatically learning parameter constraints, which we illustrate on a complex task of modelling the fMRI brain image signal during a cognitive task.

The next section of the paper describes related research on constraining parameter estimates for Bayesian Networks. Section 3 presents the problem and describes our prior work on a framework for incorporating parameter constraints to perform estimation of parameters of Bayesian Networks. Section 4 presents the main contribution of this paper: very efficient ways (closed form solutions, fast iterative algorithms) to compute parameter estimates for several important classes of parameter constraints. There we show how learning in current models that use parameter sharing assumptions can be viewed as a special case of our approach. In section 5, experiments on both real world and synthetic data demonstrate the benefits of taking advantage of parameter constraints when compared to baseline models. Some formal guarantees about our estimators are presented in section 6. We conclude with a brief summary of this research along with several directions for future work.

2. Related Work

The main methods to represent relationships among parameters fall into two main categories: Dirichlet Priors and their variants (including smoothing techniques) and Parameter Sharing of several kinds.

In Geiger and Heckerman (1997), it is shown that Dirichlet Priors are the only possible priors for discrete Bayes Nets, provided certain assumptions hold. One can think of a Dirichlet Prior as an expert's guess for the parameters in a discrete Bayes Net, allowing room for some variance around the guess. One of the main problems with Dirichlet Priors and related models is that it is impossible to represent even simple equality constraints between parameters (for example the constraint: $\theta_{111} = \theta_{121}$ where $\theta_{ijk} = P(X_i = x_{ij}|Parents(X_i) = pa_{ik})$) without using priors on the hyperparameters of the Dirichelet Prior, in which case the marginal likelihood can no longer be computed in closed form, and expensive approximate methods are required to perform parameter estimation. A second problem is that it is often beyond the expert's ability to specify a full Dirichlet Prior over the parameters of a Bayesian Network.

Extensions of Dirichlet Priors include Dirichlet Tree Priors (Minka, 1999) and Dependent Dirichlet Priors (Hooper, 2004). Although these priors allow for more correlation between the parameters of the model than standard Dirichlet Priors, essentially they face the same

issues. Moreover, in the case of Dependent Dirichlet Priors, parameter estimators can not be computed in closed form, although Hooper (2004) presents a method to compute approximate estimators, which are linear rational fractions of the observed counts and Dirichlet parameters, by minimizing a certain mean square error measure. Dirichlet Priors can be considered to be part of a broader category of methods that employ parameter domain knowledge, called smoothing methods. A comparison of several smoothing methods can be found in Zhai and Lafferty (2001).

A widely used form of parameter constraints employed by Bayesian Networks is *Parameter Sharing*. Models that use different types of Parameter Sharing include: Dynamic Bayesian Networks (Murphy, 2002) and their special case Hidden Markov Models (Rabiner, 1989), Module Networks (Segal et al., 2003), Context Specific Independence models (Boutilier et al., 1996) such as Bayesian Multinetworks, Recursive Multinetworks and Dynamic Multinetworks (Geiger and Heckerman, 1996; Pena et al., 2002; Bilmes, 2000), Probabilistic Relational Models (Friedman et al., 1999), Object Oriented Bayes Nets (Koller and Pfeffer, 1997), Kalman Filters (Welch and Bishop, 1995) and Bilinear Models (Tenenbaum and Freeman, 2000). Parameter Sharing methods constrain parameters to share the same value, but do not capture more complicated constraints among parameters such as inequality constraints or constraints on sums of parameter values. The above methods are restricted to sharing parameters at either the level of sharing a conditional probability table (CPT) (Module Networks, HMMs), at the level of sharing a conditional probability distribution within a single CPT (Context Specific Independence), at the level of sharing a state-to-state transition matrix (Kalman Filters) or at the level of sharing a style matrix (Bilinear Models). None of the prior models allow sharing at the level of granularity of individual parameters.

One additional type of parameter constraints is described by *Probabilistic Rules*. This kind of domain knowledge was used in Rao et al. (2003) to assign values to certain parameters of a Bayesian Network. We are not aware of Probabilistic Rules being used beyond that purpose for estimating the parameters of a Bayesian Network.

3. Problem Definition and Approach

Here we define the problem and describe our previous work on a general optimization based approach to solve it. This approach has serious limitations when the constraints are arbitrary. However, it constitutes the basis for the very efficient learning procedures for the classes of parameter constraints described in section 4. While the optimization methods we use are not new, applying them to our task allows us to take advantage of expert parameter constraints to perform more accurate learning of very large Bayesian Networks (thousands of variables) based on very few (tens) examples, as we will see in subsection 5.2. We begin by describing the problem and state several assumptions that we make when deriving our estimators.

3.1 The Problem

Our task here is to perform parameter estimation in a Bayesian Network where the structure is known in advance. To accomplish this task, we assume a dataset of examples is available. In addition, a set of parameter equality and/or inequality constraints is provided by a

domain expert. The equality constraints are of the form $g_i(\theta) = 0$ for $1 \leq i \leq m$ and the inequality constraints are of the form $h_j(\theta) \leq 0$ for $1 \leq j \leq k$, where θ represents the set of parameters of the Bayesian Network.

Initially we will assume the domain knowledge provided by the expert is correct. Later, we investigate what happens if this knowledge is not completely correct. Next we enumerate several assumptions that must be satisfied for our methods to work. These are similar to common assumptions made when learning parameters in standard Bayesian Networks.

First, we assume that the examples in the training dataset are drawn independently from the underlying distribution. In other words, examples are conditionally independent given the parameters of the Graphical Model.

Second, we assume that all the variables in the Bayesian Network can take on at least two different values. This is a safe assumption since there is no uncertainty in a random variable with only one possible value. Any such variables in our Bayesian Network can be deleted, along with all arcs into and out of the nodes corresponding to those variables.

When computing parameter estimators in the discrete case, we additionally assume that all observed counts corresponding to parameters in the Bayesian Network are strictly positive. We enforce this condition in order to avoid potential divisions by zero, which may impact inference negatively. In the real world it is expected there will be observed counts which are zero. This problem can be solved by using priors on parameters, that essentially have the effect of adding a positive quantity to the observed counts and essentially create strictly positive virtual counts.

Finally, the functions $g_1, \ldots, g_m$ and $h_1, \ldots, h_k$ must be twice differentiable, with continuous second derivatives. This assumption justifies the formulation of our problem as a constrained maximization problem that can be solved using standard optimization methods.

3.2 A General Approach

In order to solve the problem described above, here we briefly mention our previous approach (Niculescu, 2005) based on already existing optimization techniques. The idea is to formulate our problem as a constrained maximization problem where the objective function is either the data log-likelihood $\log P(D|\theta)$ (for Maximum Likelihood estimation) or the log-posterior $\log P(\theta|D)$ (for Maximum Aposteriori estimation) and the constraints are given by $g_i(\theta) = 0$ for $1 \leq i \leq m$ and $h_j(\theta) \leq 0$ for $1 \leq j \leq k$. It is easy to see that, applying the Karush-Kuhn-Tucker conditions theorem (Kuhn and Tucker, 1951), the maximum must satisfy a system with the same number of equations as variables. To solve this system, one can use any of several already existing methods (for example the Newton-Raphson method (Press et al., 1993)).

Based on this approach, in Niculescu (2005) we develop methods to perform learning from both a frequentist and a Bayesian point of view, from both fully and partially observable data (via an extended EM algorithm). While it is well known that finding a solution for the system given by the KKT conditions is not enough to determine the optimum point, in Niculescu (2005) we also discuss when our estimators meet the sufficiency criteria to be optimum solutions for the learning problem. There we also describe how to use *Constrained Conjugate Parameter Priors* for the MAP estimation and Bayesian Model Averaging. A sampling algorithm was devised to address the challenging issue of computing the normal-

ization constant for these priors. Furthermore, procedures that allow the automatic learning of useful parameter constraints were also derived.

Unfortunately, the above methods have a very serious shortcoming in the general case. With a large number of parameters in the Bayesian Network, they can be extremely expensive because they involve potentially multiple runs of the Newton-Raphson method and each such run requires several expensive matrix inversions. Other methods for finding the solutions of a system of equations can be employed, but, as noted in Press et al. (1993), all these methods have limitations in the case when the constraints are arbitrary, non-linear functions. The worst case happens when there exists a constraint that explicitly uses all parameters in the Bayesian Network.

Because of this shortcoming and because the optimization methods we use to derive our algorithms are not new, we choose not to go into details here. We mention them to show how learning in the presence of parameter constraints can be formulated as a general constrained maximization problem. This general framework also provides the starting point for the efficient learning methods for the particular classes of parameter constraints presented in the next section.

4. Parameter Constraints Classes

In the previous section we mentioned the existence of general methods to perform parameter learning in Bayesian Networks given a set of parameter constraints. While these methods can deal with arbitrary parameter constraints that obey several smoothness assumptions, they can be very slow since they involve expensive iterative and sampling procedures.

Fortunately, in practice, parameter constraints usually involve only a small fraction of the total number of parameters. Also, the data log-likelihood can be nicely decomposed over examples, variables and values of the parents of each variable (in the case of discrete variables). Therefore, the Maximum Likelihood optimization problem can be split into a set of many independent, more manageable, optimization subproblems, which can either be solved in closed form or for which very efficient algorithms can be derived. For example, in standard Maximum Likelihood estimation of the parameters of a Bayesian Network, each such subproblem is defined over one single conditional probability distribution. In general, in the discrete case, each optimization subproblem will span its own set of conditional probability distributions. The set of Maximum Likelihood parameters will be the union of the solutions of these subproblems.

This section shows that for several classes of parameter constraints the system of equations given by the Karush-Kuhn-Tucker theorem can be solved in an efficient way (closed form or fast iterative algorithm). In some of these cases we are also able to find a closed form formula for the normalization constant of the corresponding Constrained Parameter Prior.

4.1 Parameter Sharing within One Distribution

This class of parameter constraints allows asserting that specific user-selected parameters within a single conditional probability distribution must be shared. This type of constraint allows representing statements such as: *"Given this combination of causes, several effects are equally likely"*. Since the scope of this constraint type does not go beyond the level of a single conditional probability distribution within a single CPT, the problem of maximizing the data likelihood can be split into a set of independent optimization subproblems, one for each such conditional probability distribution. Let us consider one of these subproblems (for a variable X and a specific value $PA(X) = pa$ of the parents). Assume the parameter constraint asserts that several parameters are equal by asserting that the parameter θ_i appears in k_i different positions in the conditional distribution. Denote by N_i the cumulative observed count corresponding to θ_i. The cumulative observed count is the sum of all the observed counts corresponding to the k_i positions where θ_i appears in the distribution. Let $N = \sum_i N_i$ be the sum of all observed counts in this conditional probability distribution i.e. the total number of observed cases with $PA(X) = pa$.

At first it may appear that we can develop Maximum Likelihood estimates for θ_i and the other network parameters using standard methods, by introducing new variables that capture the groups of shared parameters. To see that this is not the case, consider the following example. Assume a variable X with values $\{1, 2, 3, 4\}$ depends on Y. Moreover, assume the parameter constraint states that $P(X = 1|Y = 0) = P(X = 2|Y = 0)$ and $P(X = 3|Y = 0) = P(X = 4|Y = 0)$. Then one can introduce variable X_{12} which is 1 if $X \in \{1, 2\}$ and 0 otherwise. This variable is assumed dependent on Y and added as a parent of X. It is easy to see that $P(X|X_{12} = 0, Y = 0)$ must be equal to the distribution on $\{1, 2, 3, 4\}$ that assigns half probability to each of 3 and 4. Therefore, if Y takes only one value, the task of finding Maximum Likelihood estimators with Parameter Sharing is reduced to the one of finding standard Maximum Likelihood estimators for $X_{12}|Y = 0$. However, if Y takes only one value, then we can safely remove it as a parent of X. When Y can take two values, 0 and 1, assume the expert states the additional assumption that $P(X = 1|Y = 1) = P(X = 3|Y = 1) = P(X = 4|Y = 1)$. Now we need to introduce a new variable X_{134} that depends on Y and add it as a parent of X. There must be an edge between X_{12} and X_{134} because, otherwise, the structural assumption that X_{12} and X_{134} are conditionally independent given Y is obviously not true. Assuming X_{12} is the parent of X_{134}, the constraints given by the expert need to be modelled in the distribution $P(X_{134}|X_{12}, Y)$ instead. Not only did we fail to encode the constraints in the new structure, but we also complicated the problem by adding two nodes in our network. A similar argument holds for all discrete types of parameter constraints presented in this section.

Below we present closed form solutions for the Maximum Likelihood estimators from complete data and for the normalization constant for the corresponding Constrained Dirichlet Priors used to perform Maximum Aposteriori estimation. These priors are similar to the standard Dirichlet Priors, but they assign zero probability over the space where the expert's constraints are not satisfied. The normalization constant for the Constrained Dirichlet Prior can be computed over the scope of a certain constraint and then all such constants are multiplied to obtain the normalization constant for the prior over the whole set of parameters

of the Bayesian Network. We also present an EM algorithm to approach learning from incomplete data under this type of parameter sharing.

4.1.1 Maximum Likelihood Estimation from Complete Data

Theorem 1 *The Maximum Likelihood Estimators for the parameters in the above conditional probability distribution are given by:*

$$\hat{\theta}_i = \frac{N_i}{k_i \cdot N}$$

Proof The problem of maximizing the data log-likelihood subject to the parameter sharing constraints can be broken down in subproblems, one for each conditional probability distribution. One such subproblem can be restated as:

$$P : argmax \ \{h(\theta) \mid g(\theta) = 0\}$$

$$\text{where } h(\theta) = \textstyle\sum_i N_i \log \theta_i \text{ and } g(\theta) = (\sum_i k_i \cdot \theta_i) - 1 = 0$$

When all counts are positive, it can be easily proved that P has a global maximum which is achieved in the interior of the region determined by the constraints. In this case the solution of P can be found using Lagrange Multipliers. Introduce Lagrange Multiplier λ for the constraint in P. Let $LM(\theta, \lambda) = h(\theta) - \lambda \cdot g(\theta)$. Then the point which maximizes P is among the solutions of the system $\nabla LM(\theta, \lambda) = 0$. Let $(\hat{\theta}, \lambda)$ be a solution of this system. We have: $0 = \frac{\partial LM}{\partial \theta_i} = \frac{N_i}{\theta_i} - \lambda \cdot k_i$ for all i. Therefore, $k_i \cdot \hat{\theta}_i = \frac{N_i}{\lambda}$. Summing up for all values of i, we obtain:

$$0 = \frac{\partial LM}{\partial \lambda} = (\sum_i k_i \cdot \hat{\theta}_i) - 1 = (\sum_i \frac{N_i}{\lambda}) - 1 = \frac{N}{\lambda} - 1$$

From the last equation we compute the value of $\lambda = N$. This gives us: $\hat{\theta}_i = \frac{N_i}{k_i \cdot N}$. The fact that $\hat{\theta}$ is the set of Maximum Likelihood estimators follows because the objective function is concave and because the constraint is a linear equality. ■

4.1.2 Constrained Dirichlet Priors

From a Bayesian point of view, each choice of parameters can occur with a certain probability. To make learning easier, for this type of parameter constraints, we employ conjugate Constrained Dirichlet Priors that have the following form for a given conditional probability distribution in the Bayesian Network:

$$P(\theta) = \begin{cases} \frac{1}{Z} \prod_{i=1}^{n} \theta_i^{\alpha_i - 1} & \text{if } \ \theta \geq 0, \sum k_i \cdot \theta_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

Maximum Aposteriori estimation can be now performed in exactly the same way as Maximum Likelihood estimation (see Theorem 1), with the only difference that the objective

function becomes $P(\theta|D) \propto P(D|\theta) \cdot P(\theta)$. The normalization constant Z can be computed by integration and depends on the elimination order. If θ_n is eliminated first, we obtain:

$$Z_n = \frac{k_n}{\prod_{i=1}^{n} k_i^{\alpha_i}} \cdot \frac{\prod_{i=1}^{n} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{n} \alpha_i)}$$

The above normalization should be thought of as corresponding to $P(\theta_1, \ldots, \theta_{n-1})$. If we eliminate a different parameter first when computing the integral, then we obtain a different normalization constant which corresponds to a different $(n-1)$-tuple of parameters. Note that having different constants is not an inconsistency, because the corresponding probability distributions over $n-1$ remaining parameters can be obtained from each other by a variable substitution based on the constraint $\sum k_i \cdot \theta_i = 1$. It is easy to see (Niculescu, 2005) that learning procedures are not affected in any way by which parameter is eliminated. In the case of no parameter sharing (that is $k_i = 1$ for all i), all these normalization constants are equal and we obtain the standard Dirichlet Prior.

4.1.3 Maximum Likelihood Estimation from Incomplete Data

It can be easily proved that learning with incomplete data can be achieved via a modified version of the standard Expectation Maximization algorithm used to train Bayesian Networks, where in the E-Step the expected counts are estimated and in the M-Step parameters are re-estimated using these expected counts based on Theorem 1.

Algorithm 1 (Expectation Maximization for discrete Bayesian Networks) *Randomly initialize the network parameters with a value $\hat{\theta}^0$. Repeat the following two steps until convergence is reached:*

E-Step*: At iteration $t+1$, use any inference algorithm to compute expected counts $E[N_i|\hat{\theta}^t]$ and $E[N|\hat{\theta}^t]$ for each distribution in the network under the current parameter estimates $\hat{\theta}^t$.*

M-Step*: Re-estimate the parameters $\hat{\theta}^{t+1}$ using Theorem 1, assuming that the observed counts are equal to the expected counts given by the E-Step.*

4.2 Parameter Sharing in Hidden Process Models

A *Hidden Process Model (HPM)* is a probabilistic framework for modelling time series data (Hutchinson et al., 2005, 2006) which predicts the value of a *target variable* X at a given point in time as the sum of the values of certain *Hidden Processes* that are active. The HPM model is inspired by our interest in modelling hidden cognitive processes in the brain, given a time series of observed fMRI images of brain activation. One can think of the observed image feature X as the value of the fMRI signal in one small cube inside the brain (also called a voxel). A hidden process may be thought of as a mental process that generates fMRI activity at various locations in the brain, in response to an external *stimulus*. For example, a *"ComprehendPicture"* process may describe the fMRI signal that happens in the brain starting when the subject is presented with a picture. A *"ComprehendSentence"* process may provide the same characterization for the situation when a subject is reading a sentence. HPMs assume several cognitive processes may be active at the some point in time,

and assume in such cases the observed fMRI signal is the sum of the corresponding processes, translated according to their starting times. Hidden Process Models can be viewed as a subclass of Dynamic Bayesian networks, as described in Hutchinson et al. (2006).

Formally, a *Hidden Process Model* is defined by a collection of time series (also called hidden processes): $P_1, \ldots, P_K$. For each process P_k with $1 \leq k \leq K$, denote by P_{kt} the value of its corresponding time series at time t after the process starts. Also, let X_t be the value of the target variable X at time t. If process P_k starts at time t_k, then a Hidden Process Model predicts the random variable X_t will follow the distribution:

$$X_t \sim N(\sum_k P_{k(t-t_k+1)}, \sigma^2)$$

where σ^2 is considered to be the variance in the measurement and is kept constant across time. For the above formula to make sense, we consider $P_{kt} = 0$ if $t < 0$. Figure 1 shows an example of a Hidden Process Model for the fMRI activity in a voxel in the brain during a cognitive task involving reading a sentence and looking at a picture.

In general HPMs allow modeling uncertainty about the timing of hidden processes, allow uncertainty about the types of the processes, and allow for multiple instances of the same process to be active simultaneously (Hutchinson et al., 2006). However, in the treatment and experiments in this paper we make three simplifying assumptions. We assume the times at which the hidden processes occur are known, that the types of the processes are known, and that two instances of the same types of process may not be active simultaneously. These three simplifying assumptions lead to a formulation of HPMs that is equivalent to the analysis approach of Dale (1999) based on multivariate regression within the General Linear Model.

In a typical fMRI experiment, the subject often performs the same cognitive task multiple times, on multiple *trials*, providing multiple observation sequences of the variable X. In our framework we denote by X_{nt} the value of X_t during trial n, and by t_{nk} the starting point of process P_k during trial n. Let N be the total number of observations. We can now write:

$$X_{nt} \sim N(\sum_k P_{k(t-t_{nk}+1)}, \sigma^2)$$

While not entirely necessary for our method to work, we assume that X is tracked for the same length of time in each trial. Let T be the length of every such trial (observation). Since we are not modelling what happens when $t > T$, we can also assume that each process has length T.

The natural constraints of this domain lead to an opportunity to specify prior knowledge in the form of parameter constraints, as follows: an external stimulus will typically influence the activity in multiple voxels of the brain during one cognitive task. For example, looking at a picture may activate many voxels in the visual cortex. The activation in these voxels may be different at each given point in time. Intuitively, that means the same stimulus may produce different hidden processes in different voxels. However, certain groups of voxels that are close together often have similarly shaped time series, but with different amplitude. In this case, we believe it is reasonable to assume that the underlying hidden processes corresponding to these voxels are proportional to one another. Experiments performed in

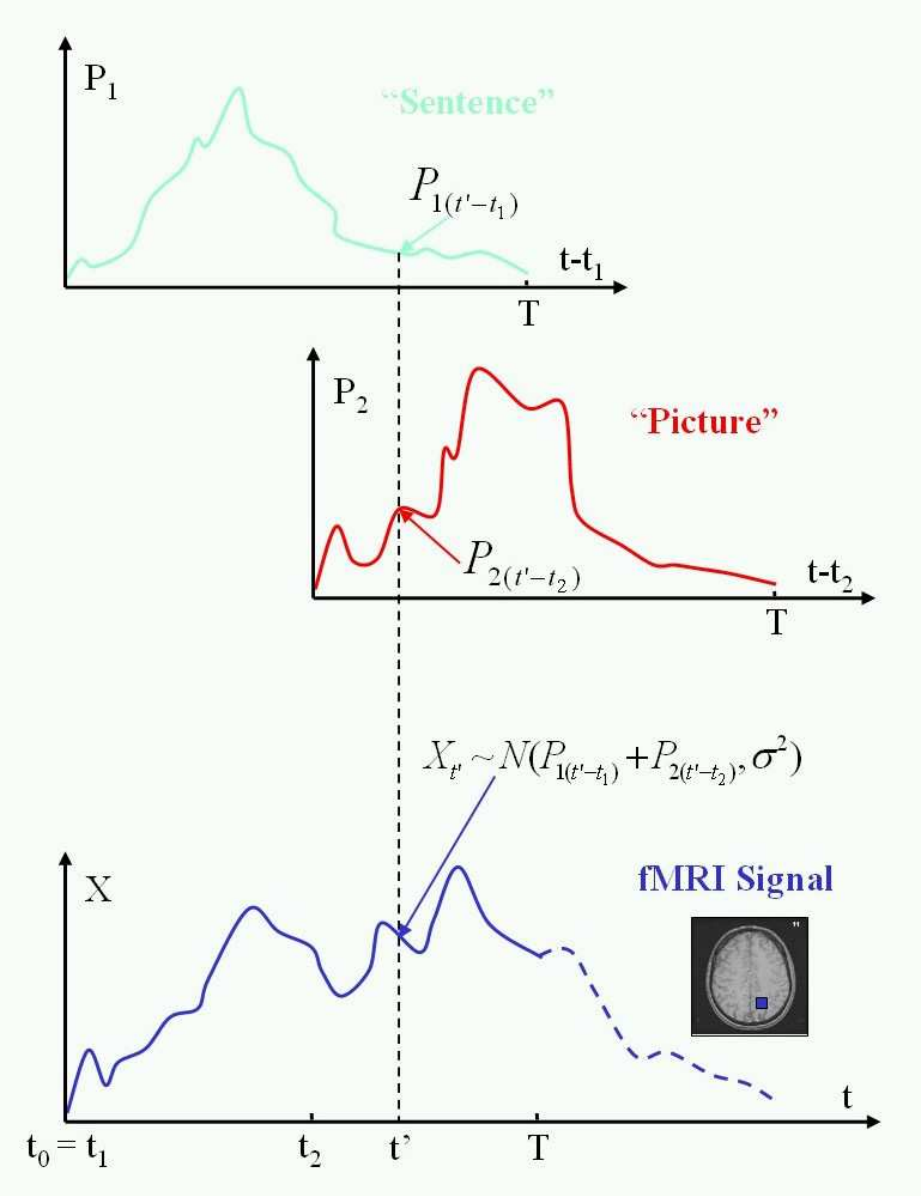

Figure 1: A Hidden Process Model to model a human subject who is asked to read a sentence and to look at a picture. In half of the observations, the sentence is presented first, then the picture is shown. In the other half of the observations, the picture is presented first. The activity in a given voxel X in the brain is modelled as a Hidden Process Model with two processes: "Sentence" (P_1) and "Picture" (P_2). Each observation has length $T = 32$ fMRI snapshots (16 seconds) and the same holds for both processes. This figure shows an observation where the sentence is presented at time $t_1 = 1$ and the picture is shown at $t_2 = 17$ (8 seconds after t_1). After time t_2, the two processes overlap and the fMRI signal $X_{t'}$ is the sum of the corresponding values of the two processes plus $N(0, \sigma^2)$ measurement variance. The blue dotted line represents the fMRI activity that would happen after time T.

Section 5 will prove that this assumption will help learn better models than the ones that choose to ignore it.

In the above paragraph we explained intuitively that sometimes it makes sense to share the same base processes across several time-varying random variables, but allow for different scaling factors. Formally, we say that time-varying random variables $X^1, \ldots, X^V$ share their corresponding *Hidden Process Models* if there exist base processes $P_1, \ldots, P_K$ and constants c_k^v for $1 \leq v \leq V$ such that:

$$X_{nt}^v \sim N(\sum_k c_k^v \cdot P_{k(t-t_{nk}^v+1)}, \sigma^2)$$

and the values of different variables X^v are independent given the parameters of the model. Here σ^2 represents the variance in measurement which is also shared across these variables.

We now consider how to efficiently perform Maximum Likelihood estimation of the parameters of the variables $X^1, \ldots, X^V$, assuming that they share their corresponding Hidden Process Model parameters as described above. The parameters to be estimated are the base process parameters P_{kt} where $1 \leq k \leq K$ and $1 \leq t \leq T$, the scaling constants c_k^v (one for each variable V and process k) where $1 \leq v \leq V$ and the common measurement variance σ^2. Let $P = \{P_{kt} \mid 1 \leq k \leq K,\ 1 \leq t \leq T\}$ be the set of all parameters involved in the base processes and let $C = \{c_k^v \mid 1 \leq k \leq K,\ 1 \leq v \leq V\}$ be the set of scaling constants. Subsequently, we will think of these sets as column vectors. Recall that N represents the number of observations. After incorporating the parameter sharing constraints in the log-likelihood function, our optimization problem becomes:

$$P : argmax \ \ l(P, C, \sigma)$$

where

$$l(P, C, \sigma) \quad = \quad -\frac{NTV}{2} \cdot \log(2\pi) - NTV \cdot \log(\sigma) - \frac{1}{2 \cdot \sigma^2} \cdot \sum_{n,t,v} (x_{nt}^v - \sum_k c_k^v \cdot P_{k(t-t_{nk}^v+1)})^2$$

It is easy to see that the value of (P, C) that maximizes l is the same for all values of σ. Therefore, to maximize l, we can first minimize $l'(P, C) = \sum_{n,t,v} (x_{nt}^v - \sum_k c_k^v \cdot P_{k(t-t_{nk}^v+1)})^2$ with respect to (P, C) and then maximize l with respect to σ based on the minimum point for l'. One may notice that l' is a sum of squares, where the quantity inside each square can be seen as a linear function in both P and C. Therefore one can imagine an iterative procedure that first minimizes with respect to P, then with respect to C using the Least Squares method. Once we find $M = \min l'(P, C) = l'(\hat{P}, \hat{C})$, the value of σ that maximizes l is given by $\hat{\sigma}^2 = \frac{M}{NVT}$. This can be derived in a straightforward fashion by enforcing $\frac{\partial l}{\partial \sigma}(\hat{P}, \hat{C}, \hat{\sigma}) = 0$. With these considerations, we are now ready to present an algorithm to compute Maximum Likelihood estimators $(\hat{P}, \hat{C}, \hat{\sigma})$ of the parameters in the shared Hidden Process Model:

Algorithm 2 (Maximum Likelihood Estimators in a Shared Hidden Process Model) *Let $\bar{X}$ be the column vector of values x^v_{nt}. Start with a random guess $(\hat{P}, \hat{C})$ and then repeat Steps 1 and 2 until they converge to the minimum of the function $l'(P, C)$.*

STEP 1. *Write $l'(\hat{P}, \hat{C}) = ||A \cdot \hat{P} - \bar{X}||^2$ where A is a NTV by KT matrix that depends on the current estimate $\hat{C}$ of the scaling constants. More specifically, each row of A corresponds to one of the squares from l' and each column corresponds to one of the KT parameters of the base processes (the column number associated with such a parameter must coincide with its position in column vector P). Minimize with respect to $\hat{P}$ using ordinary Least Squares to get a new estimate $\hat{P} = (A^T \cdot A)^{-1} \cdot A^T \cdot \bar{X}$.*

STEP 2. *Write $l'(\hat{P}, \hat{C}) = ||B \cdot \hat{C} - \bar{X}||^2$ where B is a NTV by KV matrix that depends on the current estimate $\hat{P}$ of the base processes. More specifically, each row of B corresponds to one of the squares from l' and each column corresponds to one of the KV scaling constants (the column number associated with such a constant must coincide with its position in column vector C). Minimize with respect to $\hat{C}$ using ordinary Least Squares to get a new estimate $\hat{C} = (B^T \cdot B)^{-1} \cdot B^T \cdot \bar{X}$.*

STEP 3. *Once convergence is reached by repeating the above two steps, let $\hat{\sigma}^2 = \frac{l'(\hat{P}, \hat{C})}{NVT}$.*

It might seem that this is a very expensive algorithm because it is an iterative method. However, we found that when applied to fMRI data in our experiments, it usually converges in 3-5 repetitions of Steps 1 and 2. We believe that the main reason why this happens is because at each partial step during the iteration we compute a closed form global minimizer on either $\hat{P}$ or $\hat{C}$ instead of using a potentially expensive gradient descent algorithm. In Section 5 we will experimentally prove the benefits of this algorithm over methods that do not take advantage of parameter sharing assumptions.

One may suspect that it is easy to learn the parameters of the above model because it is a particular case of bilinear model. However, this is not the case. In the bilinear model representation (Tenenbaum and Freeman, 2000), the style matrices will correspond to process parameters P and the content vectors will correspond to scaling constants. It is easy to see that in our case the style matrices have common pieces, depending on when the processes started in each example. Therefore, the SVD method presented in Tenenbaum and Freeman (2000) that assumes independence of these style matrices is not appropriate in our problem.

4.3 Other Classes of Parameter Constraints

In the above subsections we discussed efficient methods to perform parameter estimation for two types of parameter constraints: one for discrete variables and one for continuous variables. These methods bypass the need for the potentially expensive use of methods such as Newton-Raphson. There are a number of additional types of Parameter Constraints for which we have developed closed form Maximum Likelihood and Maximum Aposteriori estimators: equality and inequality constraints, on individual parameters as well as on sums and ratios of parameters, for discrete and continuous variables. Moreover, in some of these cases, we were able to compute the normalization constant in closed form for the

corresponding constrained priors, which allows us to perform parameter learning from a Bayesian point of view. All these results can be found in Niculescu (2005). We briefly describe these types of parameter constraints below, and provide real-world examples of prior knowledge that can be expressed by each form of constraint.

- *Constraint Type 1: Known Parameter values, Discrete.* Example: If a patient has a heart attack (Disease = "Heart Attack"), then there is a 90% probability that the patient will experience chest pain.
- *Constraint Type 2: Parameter Sharing, One Distribution, Discrete.* Example: Given a combination of risk factors, several diseases are equally likely.
- *Constraint Type 3: Proportionality Constants, One Distribution, Discrete.* Example: Given a combination of risk factors, disease A is twice as likely to occur as disease B.
- *Constraint Type 4: Sum Sharing, One Distribution, Discrete.* Example: A patient who is a smoker has the same chance of having a Heart Disease (Heart Attack or Congestive Heart Failure) as having a Pulmonary Disease (Lung Cancer or Chronic Obstructive Pulmonary Disease).
- *Constraint Type 5: Ratio Sharing, One Distribution, Discrete.* Example: In a bilingual corpus, the relative frequencies of certain groups of words are the same, even though the aggregate frequencies of these groups may be different. Such groups of words can be: "words about computers" ("computer", "mouse", "monitor", "keyboard" in both languages) or "words about business", etc. In some countries computer use is more extensive than in others and one would expect the aggregate probability of "words about computers" to be different. However, it would be natural to assume that the relative proportions of the "words about computers" are the same within the different languages.
- *Constraint Type 6: General Parameter Sharing, Multiple Distributions, Discrete.* Example: The probability that a person will have a heart attack given that he is a smoker with a family history of heart attack is the same whether or not the patient lives in a polluted area.
- *Constraint Type 7: Hierarchical Parameter Sharing, Multiple Distributions, Discrete.* Example: The frequency of several *international words* (for instance "computer") may be shared across both Latin languages (Spanish, Italian) and Slavic languages (Russian, Bulgarian). Other Latin words will have the same frequency only across Latin languages and the same holds for Slavic Languages. Finally, other words will be language specific (for example names of country specific objects) and their frequencies will not be shared with any other language.
- *Constraint Type 8: Sum Sharing, Multiple Distributions, Discrete.* Example: The frequency of nouns in Italian is the same as the frequency of nouns in Spanish.
- *Constraint Type 9: Ratio Sharing, Multiple Distributions, Discrete.* Example: In two different countries (A and B), the relative frequency of Heart Attack to Angina

Pectoris as the main diagnosis is the same, even though the the aggregate probability of Heart Disease (Heart Attack and Angina Pectoris) may be different because of differences in lifestyle in these countries.

- *Constraint Type 10: Inequalities between Sums of Parameters, One Distribution, Discrete.* Example: The aggregate probability mass of adverbs is no greater than the aggregate probability mass of the verbs in a given language.

- *Constraint Type 11: Upper Bounds on Sums of Parameters, One Distribution, Discrete.* Example: The aggregate probability of nouns in English is no greater than 0.4.

- *Constraint Type 12: Parameter Sharing, One Distribution, Continuous.* Example: The stock of computer maker *DELL* as a Gaussian whose mean is a weighted sum of the stocks of software maker *Microsoft (MSFT)* and chip maker *Intel (INTL)*. Parameter sharing corresponds to the statement that *MSFT* and *INTL* have the same importance (weight) for predicting the value of stock *DELL*.

- *Constraint Type 13: Proportionality Constants, One Distribution, Continuous.* Example: Suppose we also throw in the stock of a Power Supply maker (PSUPPLY) in the linear mix in the above example. The expert may give equal weights to INTL and MSFT, but five times lower to PSUPPLY.

- *Constraint Type 14: Parameter Sharing for Hidden Process Models.* Example: Several neighboring voxels in the brain exhibit similar activation patterns, but with different amplitudes when a subject is presented with a given stimulus.

Note that general parameter sharing (Constraint Type 6) encompasses models including HMMs, Dynamic Bayesian Networks, Module Networks and Context Specific Independence as particular cases, but allows for much finer grained sharing, at the level of individual parameters, across different variables and across distributions of different lengths. Briefly, this general parameter sharing allows for a group of conditional probability distributions to share some parameters across all distributions in the group, but not share the remaining parameters. This type of parameter constraint is described in more detail in Niculescu et al. (2005) where we demonstrate our estimators on a task of modelling synthetic emails generated by different subpopulations.

It is also important to note that different types of parameter constraints can be mixed together when learning the parameters of a Bayesian Network as long as the scopes of these constraints do not overlap.

5. Experiments

In this section we present experiments on both synthetic and real world data. Our experiments demonstrate that Bayesian Network models that take advantage of prior knowledge in the form of parameter constraints outperform similar models which choose to ignore this kind of knowledge.

5.1 Synthetic Data - Estimating Parameters of a Discrete Variable

This section describes experiments involving one of the simplest forms of parameter constraint: Parameter Sharing within One Distribution, presented in subsection 4.1. The purpose of these experiments is purely demonstrative and a more complicated scenario, on real world data, will be presented in subsection 5.2.

5.1.1 Experimental Setup

Here, our task is to estimate the set of parameters of a Bayesian Network which consists of one discrete variable X. We assume that prior knowledge is available that the distribution of X shares certain parameters. Without loss of generality, we consider that the parameter constraint states that the parameters to estimate are given by $\theta = \{\theta_1, \ldots, \theta_n\}$ where θ_i appears in $k_i \geq 1$ known places in the distribution of X.

Our synthetic dataset was created as follows: first, we randomly generated a distribution T (the "true distribution") that exhibits parameter sharing. This distribution described a variable X with 50 values, which had a total of roughly 50% shared parameters i.e. $\sum_{k_i>1} k_i \approx \sum_{k_i=1} k_i$. Each distinct parameter appeared at most 5 times. We start with an empty distribution and generate a uniformly random parameter v between 0 and 1. Then we generate a random integer s between 2 and 5 and share v in the first s places of the distribution. We continue to generate shared parameters until we reach 25 (50% of 50 parameters). After that, we generate the rest of parameters uniformly randomly between 0 and 1. After all 50 parameters are obtained using this procedure, we normalize to yield a valid probability distribution. Once this distribution was generated, we sampled it to obtain a dataset of 1000 examples which were used subsequently to perform parameter estimation.

In our experiments we compare two models that estimate the parameters of distribution T over X. One is a standard Bayesian Network (STBN) that is learned using standard Bayesian Networks Maximum Likelihood estimators with no parameter sharing. The second model (PDKBN) is a Bayesian Network that is learned using the results in 4.1 assuming the correct parameter sharing was specified by an oracle. While STBN needs to estimate $\sum_{i=1}^{n} k_i$ parameters, PDKBN only needs to estimate n parameters. To deal with potentially zero observed counts, we used priors on the parameters of the two models and then performed Maximum Aposteriori estimation. For STBN we introduced a Dirichlet count of 2 for each parameter while for PDKBN we used a Constrained Dirichlet count of k_i+1 for each distinct parameter θ_i in the network. The role of these priors is simply to assure strictly positive counts.

5.1.2 Results and Discussion

We performed parameter estimation for the STBN and PDKBN models, varying the number of examples in the training set from 1 to 1000. Since we were using synthetic data, we were able to assess performance by computing KL(T,STBN) and KL(T,PDKBN), the KL divergence from the true distribution T.

Figure 2 shows a graphical comparison of the performance of the two models. It can be seen that our model (PDKBN) that takes advantage of parameter constraints consistently outperforms the standard Bayesian Network model which does not employ such constraints. The difference between the two models is greatest when the training data is most sparse.

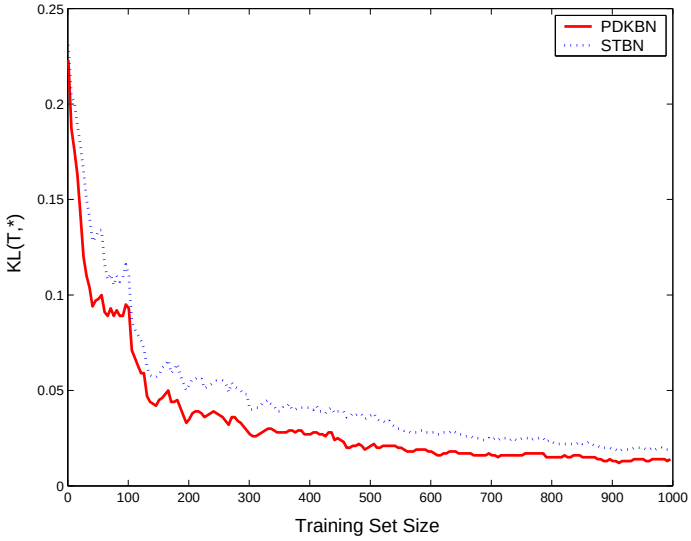

Figure 2: KL divergence of PDKBN and STBN with respect to correct model T.

The highest observed difference between KL(T,STBN) and KL(T,PDKBN) was 0.05, which was observed when the two models were trained using 30 examples. As expected, when the amount of training data increases, the difference in performance between the two models decreases dramatically.

Training Examples	KL(T,PDKBN)	Examples needed by STBN
5	0.191	16
40	0.094	103
200	0.034	516
600	0.018	905
650	0.017	> 1000

Table 1: Equivalent training set size so that STBN achieves the same performance as PDKBN.

To get a better idea of how beneficial the prior knowledge in these parameter constraints can be in this case, let us examine *"how far STBN is behind PDKBN"*. For a model PDKBN learned from a dataset of a given size, this can be measured by the number of examples that STBN requires in order to achieve the same performance. Table 1 provides these numbers for several training set sizes for PDKBN. For example, STBN uses 16 examples to achieve the same KL divergence as PDKBN at 5 examples, which is a factor of 3.2 (the maximum observed) increase in the number of training samples required by STNB. On average, STBN needs 1.86 times more examples to perform as well as PDKBN.

As mentioned previously, this subsection was intended to be only a proof of concept. We next provide experimental results on a very complex task involving several thousand random variables and prior knowledge in the form of parameter constraints across many conditional probability distributions.

5.2 Real World Data - fMRI Experiments

As noted earlier, functional Magnetic Resonance Imaging (fMRI) is a technique for obtaining three-dimensional images of activity in the brain, over time. Typically, there are ten to fifteen thousand voxels (three dimensional pixels) in each image, where each voxel covers a few tens of millimeters of brain tissue. Due to the nature of the signal, the fMRI activation observable due to neural activity extends for approximately 10 seconds after the neural activity, resulting in a temporally blurred response (see Mitchell et al. (2004) for a brief overview of machine learning approaches to fMRI analysis).

This section presents a generative model of the activity in the brain while a human subject performs a cognitive task, based on the Hidden Process Model and parameter sharing approach discussed in section 4.2. In this experiment involving real fMRI data and a complex cognitive task, domain experts were unable to provide parameter sharing assumptions in advance. Therefore, we have developed an algorithm to automatically discover clusters of voxels that can be more accurately learned with shared parameters. This section describes the algorithm for discovering these parameter sharing constraints, and shows that training under these parameter constraints leads to Hidden Process Models that far outperform the baseline Hidden Process Models learned in the absence of such parameter constraints.

5.2.1 Experimental Setup

The experiments reported here are based on fMRI data collected in a study of sentence and picture comprehension (Carpenter et al., 1999). Subjects in this study were presented with a sequence of 40 trials. In 20 of these trials, the subject was first presented with a sentence for 4 seconds, such as "The plus sign is above the star sign.", then a blank screen for 4 seconds, and finally a picture such as

for another 4 seconds. During each trial, the subject was required to press a "yes" or "no" button to indicate whether the sentence correctly described the picture. During the remaining 20 trials the picture was presented first and the sentence presented second, using the same timing.

In this dataset, the voxels were grouped into 24 anatomically defined spatial regions of interest (ROIs), each voxel having a resolution of 3 by 3 by 5 millimeters. An image of the brain was taken every half second. For each trial, we considered only the first 32 images (16 seconds) of brain activity. The results reported in this section are based on data from a single human subject (04847). For this particular subject, our dataset tracked the activity of 4698 different voxels.

We model the activity in each voxel by a Hidden Process Model with two processes, corresponding to the cognitive processes of comprehending a *Sentence* or a *Picture*. The start time of each processes is assumed to be known in advance (i.e., we assume the process begins immediately upon seeing the sentence or picture stimulus). We further assume that the activity in different voxels is independent given the hidden processes corresponding to these voxels. Since the true underlying distribution of voxel activation is not known, we use the Average Log-Likelihood Score (the log-likelihood of the test data divided by the number

of test examples) to assess performance of the trained HPMs. Because data is scarce, we can not afford to keep a large held-out test set. Instead, we employ a leave-two-out cross-validation approach to estimate the performance of our models.

In our experiments we compare three HPM models. The first model *StHPM*, which we consider a baseline, consists of a standard Hidden Process Model learned independently for each voxel. The second model *ShHPM* is a Hidden Process Model, shared for all the voxels within an ROI. In other words, all voxels in a specific ROI share the same shape hidden processes, but with different amplitudes (see Section 4.2 for more details). *ShHPM* is learned using Algorithm 2. The third model (*HieHPM*) also learns a set of Shared Hidden Process Models, but instead of assuming a priori that a particular set of voxels should be grouped together, it chooses these voxel groupings itself, using a nested cross-validation hierarchical approach to both come up with a partition of the voxels in clusters that form a Shared Hidden Process Model. The algorithm is as follows:

Algorithm 3 (Hierarchical Partitioning and Hidden Process Models learning)

STEP 1. *Split the 40 examples into a set containing 20 folds* $F = \{F_1, \ldots, F_{20}\}$*, each fold containing one example where the sentence is presented first and one example where the picture is presented first.*

STEP 2. *For all* $1 \leq k \leq 20$*, keep fold* F_k *aside and learn a model from the remaining folds using Steps 3-5.*

STEP 3. *Start with a partition of all voxels in the brain by their ROIs and mark all subsets as* Not Final*.*

STEP 4. *While there are subsets in the partition that are* Not Final*, take any such subset and try to split it using equally spaced hyperplanes in all three directions (in our experiments we split each subset into 4 (2 by 2) smaller subsets. If the cross-validation Average Log Score of the model learned from these new subsets using Algorithm 2 (based on folds* $F \setminus F_k$*) is lower than the cross-validation Average Log Score of the initial subset for folds in* $F \setminus F_k$*, then mark the initial subset as* Final *and discard its subsets. Otherwise remove the initial subset from the partition and replace it with its subsets which then mark as* Not Final*.*

STEP 5. *Given the partition computed by STEPS 3 and 4, based on the 38 data points in* $F \setminus F_k$*, learn a Hidden Process Model that is shared for all voxels inside each subset of the partition. Use this model to compute the log score for the examples/trials in* F_k*.*

STEP 6. *In Steps 2-4 we came up with a partition for each fold* F_k*. To come up with one single model, compute a partition using STEPS 3 and 4 based on all 20 folds, then, based on this partition learn a model as in STEP 5 using all 40 examples. The Average Log Score of this last model can be estimated by averaging the numbers obtained in STEP 5.*

5.2.2 Results and Discussion

We estimated the performance of our three models using the Average Log Score, based on a leave two out cross-validation approach, where each fold contains one example in which the sentence is presented first, and one example in which the picture is presented first.

Training Trials	No Sharing (StHPM)	All Shared (ShHPM)	Hierarchical (HieHPM)	Cells (HieHPM)
6	-30497	-24020	-24020	1
8	-26631	-23983	-23983	1
10	-25548	-24018	-24018	1
12	-25085	-24079	-24084	1
14	-24817	-24172	-24081	21
16	-24658	-24287	-24048	36
18	-24554	-24329	-24061	37
20	-24474	-24359	-24073	37
22	-24393	-24365	-24062	38
24	-24326	-24351	-24047	40
26	-24268	-24337	-24032	44
28	-24212	-24307	-24012	50
30	-24164	-24274	-23984	60
32	-24121	-24246	-23958	58
34	-24097	-24237	-23952	61
36	-24063	-24207	-23931	59
38	-24035	-24188	-23921	59
40	-24024	-24182	-23918	59

Table 2: The effect of training set size on the Average Log Score of the three models in the Visual Cortex (CALC) region.

Our first set of experiments, summarized in Table 2, compared the three models based on their performance in the Visual Cortex (CALC). This is one of the ROIs actively involved in this cognitive task and it contains 318 voxels. The training set size was varied from 6 examples to all 40 examples, in multiples of two. Sharing the parameters of Hidden Process Models proved very beneficial and the impact was observed best when the training set size was the smallest. With an increase in the number of examples, the performance of *ShHPM* starts to degrade because it makes the biased assumption that all voxels in CALC can be described by a single Shared Hidden Process Model. While this assumption paid off with small training set size because of the reduction in variance, it definitely hurt in terms of bias with larger sample size. Even though the bias was obvious in CALC, we will see in other experiments that in certain ROIs, this assumption holds and in those cases the gains in performance may be quite large.

As expected, the hierarchical model *HieHPM* performed better than both *StHPM* and *ShHPM* because it takes advantage of Shared Hidden Process Models while not making the restrictive assumption of sharing across entire ROIs. The largest difference in performance

between *HieHPM* and *StHPM* is observed at 6 examples, in which case *StHPM* basically fails to learn a reasonable model while the highest difference between *HieHPM* and *ShHPM* occurs at the maximum number of examples, presumably when the bias of *ShHPM* is most harmful. As the number of training examples increases, both *StHPM* and *HieHPM* tend to perform better and better and one can see that the marginal improvement in performance obtained by the addition of two new examples tends to shrink as both models approach convergence. While with an infinite amount of data, one would expect $StHPM$ and $HieHPM$ to converge to the true model, at 40 examples, *HieHPM* still outperforms the baseline model *StHPM* by a difference of 106 in terms of Average Log Score, which is an improvement of e^{106} in terms of data likelihood.

Probably the measure that shows best the improvement of *HieHPM* over the baseline *StHPM* is the number of examples needed by *StHPM* to achieve the same performance as *HieHPM*. It turns out that on average, *StHPM* needs roughly 2.9 times the number of examples needed by *HieHPM* in order to achieve the same level of performance in the Visual Cortex (CALC).

The last column of Table 2 displays the number of clusters of voxels in which *HieHPM* partitioned CALC. As can be seen, at small sample size *HieHPM* draws its performance from reductions in variance by using only one cluster of voxels. However, as the number of examples increases, *HieHPM* improves by finding more and more refined partitions. This number of shared voxel sets tends to stabilize around 60 clusters once the number of examples reaches 30, which yields an average of more than 5 voxels per cluster given that CALC is made of 318 voxels. For a training set of 40 examples, the largest cluster has 41 voxels while many clusters consist of only one voxel.

The second set of experiments (see Table 3) describes the performance of the three models for each of the 24 individual ROIs of the brain, and trained over the entire brain. While we have seen that *ShHPM* was biased in CALC, we see here that there are several ROIs where it makes sense to characterize all of its voxels by a single Shared Hidden Process Model. In fact, in most of these regions, *HieHPM* finds only one cluster of voxels. Actually, *ShHPM* outperforms the baseline model *StHPM* in 18 out of 24 ROIs while *HieHPM* outperforms *StHPM* in 23 ROIs. One may ask how *StHPM* can possibly outperform *HieHPM* on a ROI, since *HieHPM* may also represent the case when there is no sharing. The explanation is that the hierarchical approach can get stuck in a local maximum of the data log-likelihood over the search space if it cannot improve by splitting at a specific step, since it is a greedy process that does not look beyond that split for a finer grained partition. Fortunately, this problem appears to be rare in these experiments.

Over the whole brain, *HieHPM* outperforms *StHPM* by a factor 1792 in terms of log likelihood while *ShHPM* outperforms *StHPM* only by a factor of 464. The main drawback of the *ShHPM* is that it also makes a very restrictive sharing assumption and therefore we suggest *HieHPM* as the recommended approach. Next we give the reader a feel of what the learned *HieHPM* model looks like.

As mentioned above, *HieHPM* automatically learns clusters of voxels that can be represented using a Shared Hidden Process Model. Figure 3 shows the portions of these learned clusters in slice five of the eight vertical slices that make up the 3D brain image captured by the fMRI scanner. Neighboring voxels that were assigned by *HieHPM* to the same cluster are pictured with the same color. Note that there are several very large clusters in this

ROI	Voxels	No Sharing (StHPM)	All Shared (ShHPM)	Hierarchical (HieHPM)	Cells Hierarchical
CALC	318	-24024	-24182	-23918	59
LDLPFC	440	-32918	-32876	-32694	11
LFEF	109	-8346	-8299	-8281	6
LIPL	134	-9889	-9820	-9820	1
LIPS	236	-17305	-17187	-17180	8
LIT	287	-21545	-21387	-21387	1
LOPER	169	-12959	-12909	-12909	1
LPPREC	153	-11246	-11145	-11145	1
LSGA	6	-441	-441	-441	1
LSPL	308	-22637	-22735	-22516	4
LT	305	-22365	-22547	-22408	18
LTRIA	113	-8436	-8385	-8385	1
RDLPFC	349	-26390	-26401	-26272	40
RFEF	68	-5258	-5223	-5223	1
RIPL	92	-7311	-7315	-7296	11
RIPS	166	-12559	-12543	-12522	20
RIT	278	-21707	-21720	-21619	42
ROPER	181	-13661	-13584	-13584	1
RPPREC	144	-10623	-10558	-10560	1
RSGA	34	-2658	-2654	-2654	1
RSPL	252	-18572	-18511	-18434	35
RT	284	-21322	-21349	-21226	24
RTRIA	57	-4230	-4208	-4208	1
SMA	215	-15830	-15788	-15757	10
All Brain	4698	-352234	-351770	-350441	299

Table 3: Per ROI performance (Average Log Score) of the three models when learned using all 40 examples.

picture. This may be because of the fact that it makes sense to represent an entire ROI using a single Shared Hidden Process Model if the cognitive process does not activate voxels in this ROI. However, large clusters are also found in areas like CALC, which we know is directly involved in visual processing.

In Figure 4 we can see the learned *Sentence* hidden process for the voxels in the Visual Cortex (CALC). Again, the graphs corresponding to voxels that belong to the same cluster have been painted in the same color, which is also the same as the color used in Figure 3. To make these graphs readable, we only plotted the base process, disregarding the scaling (amplitude) constants corresponding to each voxel within a given cluster (consult Section 4.2 for more details about Shared Hidden Process Models).

Figure 3: Parameter Sharing found using model *HieHPM*. Slice five of the brain is showed here. Shared neighboring voxels have the same color.

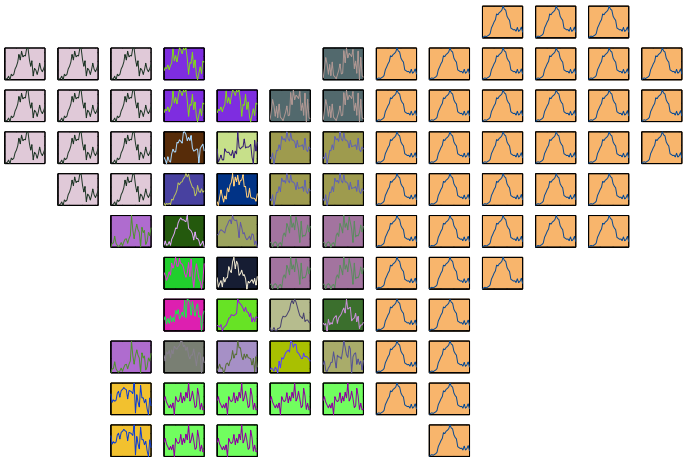

Figure 4: Per voxel base *Sentence* processes in the Visual Cortex(CALC).

To summarize, this subsection presented experiments training different generative models for the fMRI signal during a cognitive task, all based on Hidden Process Models. We demonstrated experimentally that Parameter Sharing for Hidden Process Models (as de-

fined in Section 4.2) can greatly benefit learning, and that it is possible to automatically discover useful parameter sharing constraints in this domain using our hierarchical partitioning algorithm.

6. Formal Guarantees

Taking advantage of parameter constraints can be beneficial to learning because, intuitively, it has the effect of lowering the variance in parameter estimators by shrinking the degrees of freedom of the model. In this section we provide a formal proof of this fact. In order for our proof to work, we make the assumption that the true distribution factors according to the given Bayesian Network structure and that it obeys the parameter constraints provided by the expert. The second interesting result presented in this section will give theoretical guarantees in the case when the constraints provided by the expert are not entirely accurate. While we only investigate this issue for one type of constraint, Parameter Sharing within One Distribution (introduced in subsection 4.1), we believe similar formal guarantees describe all other types of parameter constraints presented in this paper.

6.1 Variance Reduction by Using Parameter Constraints

Assume we want to learn a Bayesian Network in the case when a domain expert provides parameter constraints specifying that certain parameters appear multiple times (are shared) within a conditional probability distribution. Each conditional probability distribution in the Bayesian Network can have its own such constraints. Also, the case when all parameters are distinct within one such distribution may be seen as a particular case of Parameter Sharing within One Distribution, where each parameter is shared exactly once.

There are two ways to perform Maximum Likelihood parameter learning in the above Bayesian Network. First, one may choose to ignore the constraints given by the expert and compute standard Maximum Likelihood estimators. A second option is to incorporate the constraints in the learning method, in which case we can use the results described in subsection 4.1. One would intuitively expect that taking advantage of the constraints provided by the expert would reduce the variance in parameter estimates when compared to the first approach. In Niculescu (2005) we prove the following result:

Theorem 2 *Assuming a domain expert can specify parameter sharing assumptions that take place inside the conditional probability distributions of a Bayesian Network, the Maximum Likelihood estimators that use this domain knowledge as computed with Theorem 1 have lower variance than standard Maximum Likelihood estimators computed ignoring the domain knowledge. More specifically, for one parameter θ_{ijk} that is shared $s \geq 1$ times within $P(X_i|PA_i = pa_{ik})$, denote by $\hat{\theta}_{ijk}^{ML}$ the Maximum Likelihood estimator that ignores domain knowledge and by $\hat{\theta}_{ijk}^{PS}$ the Maximum Likelihood estimator that uses the parameter sharing assumptions specified by the expert. We have the following identity:*

$$Var[\hat{\theta}_{ijk}^{ML}] - Var[\hat{\theta}_{ijk}^{PS}] = \theta_{ijk} \cdot (1 - \frac{1}{s}) \cdot E[\frac{1}{N_{ik}} | N_{ik} \neq 0] \geq 0$$

6.2 Performance with Potentially Inaccurate Constraints

Sometimes it may happen that the parameter constraints provided by an expert are not completely accurate. In all our methods so far, we assumed that the parameter constraints are correct and therefore errors in domain knowledge can prove detrimental to the performance of our learned models. In this section we investigate the relationship between the true, underlying distribution of the observed data and the distribution estimated using our methods based on parameter constraints. In particular, we come up with an upper bound on how well our estimated model can perform given a set of potentially incorrect parameter constraints.

Assume an expert provides a set of potentially incorrect Parameter Sharing assumptions as described in subsection 4.1. In other words, for each conditional probability distribution c in the Bayesian Network, the expert is stating that parameter θ_{ic} is shared in k_{ic} given positions. We denote by N_{ic} the cumulative observed count corresponding to the presumably shared parameter θ_{ic} and by N_c the cumulative observed count corresponding to the conditional distribution c. Essentially, we follow the notations in subsection 4.1, to which we add an additional index corresponding to the conditional probability distribution that a parameter belongs to.

Let us introduce the notion of *True Probabilistic Counts (TPC)*. Suppose P is the true distribution from which data is sampled. If, for example, the expert states that θ_{ic} is the shared parameter that describes the set $\{P(X = x_1|PA(X) = pa), \ldots, P(X = x_{k_{ic}}|PA(X) = pa)\}$, let $TPC_{ic} = \sum_{i=1}^{k_{ic}} P(X = x_i, PA(X) = pa)$. Let P^* be the distribution that factorizes according to the structure provided by the expert and has parameters given by theorem 1 where the observed counts are replaced by the *True Probabilistic Counts*.

Theorem 3 *P^* is the closest distribution to P(in terms of $KL(P, \cdot)$) that factorizes according to the given structure and obeys the expert's parameter sharing assumptions.*

Proof Let Q be such a distribution. Minimizing $K(P, Q)$ is equivalent to maximizing $\sum_d P(d) \cdot \log Q(d)$. Let θ be the set of parameters that describe this distribution Q. After breaking the logarithms into sums of logarithms based on the factorization given by the provided structure, our optimization problem reduces to the maximization of $\sum TPC_{ic} \cdot \log \theta_{ic}$. This is exactly the objective function used in theorem 1. This is equivalent to the fact that P^*(see the definition above) minimizes $KL(P, \cdot)$ out of all the distributions that factorize according to the given structure and obey the expert's sharing assumptions. ■

Theorem 4 *With an infinite amount of data, the distribution $\hat{P}$ given by the Maximum Likelihood estimators in Theorem 1 converges to P^* with probability 1.*

Proof Assume the number of data points in a dataset sampled from P is denoted by n. According to the Law of Large Numbers, we have $lim_{n\to\infty} \frac{N_{ic}}{n} = TPC_{ic}$. This implies that $\hat{P}$ converges to P^* with probability 1. ■

Corollary 5 *If the true distribution P factorizes according to the given structure and if the parameter sharing provided by the expert is completely accurate, then the distribution $\hat{P}$ given by the estimators computed in Theorem 1 converges to P with probability 1.*

Again, we mention that we analyzed the formal guarantees presented in this section using only one type of parameter constraints. We are confident that these results can be extended to all other types of constraints for which we computed closed form solutions.

7. Conclusions and Future Work

Building accurate models from limited training data is possible only by using some form of prior knowledge to augment the data. In this paper we have demonstrated both theoretically and experimentally that the standard methods for parameter estimation in Bayesian Networks can be naturally extended to accommodate parameter constraints capable of expressing a wide variety of prior domain knowledge.

We mentioned our previous work on methods for incorporating general parameter constraints into estimators for the parameters of a Bayesian Network, by framing this task as a constrained optimization problem. In the general case, solving the resulting optimization problem may be very difficult. Fortunately, in practice the optimization problem can often be decomposed into a set of many smaller, independent, optimization subproblems. We have presented parameter estimators for several types of constraints, including constraints that force various types of parameter sharing, and constraints on sums and other relationships among groups of parameters. Subsection 4.3 provides a comprehensive list of the parameter constraint types we have studied, along with brief examples of each. We considered learning with both discrete and continuous variables, in the presence of both equality and inequality constraints. While for most of these types of parameter constraints we can derive closed form Maximum Likelihood estimators, we developed a very efficient iterative algorithm to perform the same task for Shared Hidden Process Models. In many of these cases, for discrete variables, we are also able to compute closed form normalization constants for the corresponding Constrained Parameter Priors, allowing one to perform closed form MAP and Bayesian estimation when the data is complete.

The General Parameter Sharing domain knowledge type (Constraint Type 6 defined in subsection 4.3) encompasses models including HMMs, Dynamic Bayesian Networks, Module Networks and Context Specific Independence as particular cases, but allows for much finer grained sharing, at the parameter level, across different variables and across distributions of different lengths. It is also important to note that one can combine different types of parameter constraints when learning the parameters of a Bayesian Network as long as the scopes of these constraints do not overlap.

Experimental results using an fMRI brain imaging application demonstrate that taking advantage of parameter constraints can be very beneficial for learning in this high-dimensional, sparse-data domain. In the context of this application we developed methods to automatically discover parameter sharing constraints. Using these methods our program discovered clusters of voxels whose parameters can be shared. Our results showed that the impact of these learned parameter constraints can be equivalent to almost tripling the size of the training set on this task. Experiments on synthetic data demonstrated the same beneficial effect of incorporating parameter constraints.

A basic theoretical result is that the estimators taking advantage of a simple form of parameter sharing achieve variance lower than that achieved by estimators that ignore such constraints. We conjecture that similar results hold for other types of parameter constraints, but their proof is left as future work. In addition, we proved that even when the asserted parameter constraints turn out to be incorrect, given an infinite amount of training data our Maximum Likelihood estimators converge to the best describable distribution; that is, the distribution closest in terms of KL distance from the true distribution, among all distributions that obey the parameter constraints and factor according to the given structure.

We see many useful directions for future work. In this paper we have considered only how to take advantage of deterministic parameter constraints when the structure of the Bayesian Network is known in advance. It would be interesting to investigate methods to incorporate probabilistic constraints in learning algorithms for Bayesian Networks. A second direction to explore is to also use parameter constraints to perform structure learning. This might be achieved by specifying an initial set of parameter constraints, then at each step of the hill climbing during structure search performing a change of variable to adapt the constraints to the new parameterization of the network. Finally, we would like to extend our results to undirected graphical models, to the extent that it is intuitive to acquire domain knowledge from an expert about the much harder to interpret parameters of such models.

Acknowledgments

We would like to thank the following people for their useful comments and suggestions during the development of this research: John Lafferty, Andrew Moore, Russ Greiner, Zoubin Ghahramani, Sathyakama Sandilya and Balaji Krishnapuram. As a student at Carnegie Mellon University, Radu Stefan Niculescu was sponsored the National Science Foundation under grant nos. CCR-0085982 and CCR-0122581, by the Darpa PAL program under contract NBCD030010, and by a generous gift from Siemens Medical Solutions.

References

J. Bilmes. Dynamic bayesian multinets. In *Proceedings of UAI*, pages 38–45, 2000.

C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in bayesian networks. In *Proceedings of 12th UAI*, pages 115–123, 1996.

P. A. Carpenter, M. A. Just, T. A. Keller, W. F. Eddy, and K. R. Thulborn. Time course of fMRI-activation in language and spatial networks during sentence comprehension. *NeuroImage*, 10:216–224, 1999.

A. M. Dale. Optimal experimental design for event-related fMRI. *Human Brain Mapping*, 8:109–114, 1999.

N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of 16th IJCAI*, pages 1300–1307, 1999.

D. Geiger and D. Heckerman. Knowledge representation and inference in similarity networks and bayesian multinets. *Artificial Intelligence*, 82:45–74, 1996.

D. Geiger and D. Heckerman. A characterization of the dirichlet distribution through global and local parameter independence. *The Annals of Statistics*, 25:1344–1369, 1997.

D. Heckerman. A tutorial on learning with bayesian networks. In M. Jordan, editor, *Learning in Graphical Models*. MIT Press, Cambridge, MA, 1999.

P. Hooper. Dependent dirichlet priors and optimal linear estimators for belief net parameters. In AUAI Press, editor, *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI-04)*, pages 251–259, 2004.

R. Hutchinson, T.M. Mitchell, and I. Rustandi. Learning to identify overlapping and hidden cognitive processes from fMRI data. In *11th Conference on Human Brain Mapping*, June 2005.

R. Hutchinson, T.M. Mitchell, and I. Rustandi. Hidden process models. Technical Report CS-CALD-05-116, Carnegie Mellon University, February 2006.

D. Koller and A. Pfeffer. Object oriented bayesian networks. In *Proceedings of 13th UAI*, pages 302–313, 1997.

H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492. University of California Press, 1951.

T. Minka. The dirichlet-tree distribution. This unpublished paper is available online at http://research.microsoft.com/∼minka/papers/dirichlet/minka-dirtree.pdf, 1999.

T. Mitchell, R. Hutchinson, M. Just, S. Newman, R. S. Niculescu, F. Pereira, and X. Wang. Learning to decode cognitive states from brain images. *Machine Learning*, 57(1-2):145–175, 2004.

K. P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, 2002.

R. S. Niculescu. Exploiting parameter domain knowledge for learning in bayesian networks. Technical Report CMU-TR-05-147, Carnegie Mellon University, 2005.

R. S. Niculescu, T. Mitchell, and R. B. Rao. Parameter related domain knowledge for learning in graphical models. In *Proceedings of SIAM Data Mining conference*, 2005.

J. M. Pena, J. A. Lozano, and P. Larranaga. Learning recursive bayesian multinets for data clustering by means of constructive induction. *Machine Learning*, 47(1):63–89, 2002.

W. H. Press, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1993.

R. L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

R. B. Rao, S. Sandilya, R. S. Niculescu, C. Germond, and H. Rao. Clinical and financial outcomes analysis with existing hospital patient records. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 416–425, 2003.

E. Segal, D. Pe'er, A. Regev, D. Koller, and N. Friedman. Learning module networks. In *Proceedings of 19th UAI*, pages 525–534, 2003.

J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural Computation*, 12(6):1247–1283, 2000.

G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report TR 95-041, University of North Carolina, 1995.

C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR*, pages 334–342, 2001.