Journal of Machine Learning Research 6 (2005) 81–127 Submitted 9/04; Revised 12/04; Published 1/05

Learning Hidden Variable Networks: The Information Bottleneck Approach

Gal Elidan GALEL@CS.HUJI.AC.IL
Department of Engineering and Computer Science
The Hebrew University
Jerusalem, 91904, Israel

Nir Friedman NIR@CS.HUJI.AC.IL
Department of Engineering and Computer Science
The Hebrew University
Jerusalem, 91904, Israel

Editor: David Maxwell Chickering

Abstract

A central challenge in learning probabilistic graphical models is dealing with domains that involve hidden variables. The common approach for learning model parameters in such domains is the *expectation maximization* (EM) algorithm. This algorithm, however, can easily get trapped in sub-optimal local maxima. Learning the model *structure* is even more challenging. The *structural EM* algorithm can adapt the structure in the presence of hidden variables, but usually performs poorly without prior knowledge about the cardinality and location of the hidden variables. In this work, we present a general approach for learning Bayesian networks with hidden variables that overcomes these problems. The approach builds on the *information bottleneck* framework of Tishby et al. (1999). We start by proving formal correspondence between the information bottleneck objective and the standard parametric EM functional. We then use this correspondence to construct a learning algorithm that combines an information-theoretic smoothing term with a continuation procedure. Intuitively, the algorithm bypasses local maxima and achieves superior solutions by following a continuous path from a solution of, an easy and smooth, target function, to a solution of the desired likelihood function. As we show, our algorithmic framework allows learning of the parameters as well as the structure of a network. In addition, it also allows us to introduce new hidden variables during model selection and learn their cardinality. We demonstrate the performance of our procedure on several challenging real-life data sets.

Keywords: Bayesian networks, hidden variables, information bottleneck, continuation, variational methods

1. Introduction

Probabilistic graphical models have been widely used to model real world domains and are particularly appealing due to their natural interpretation. Despite extensive research in learning these models from data (Pearl, 1988; Heckerman, 1998), learning with *hidden* (or *latent*) variables has

A preliminary version of this paper appeared in the Proceedings of the Nineteenth Conference on Uncertainty in Artificial, 2003 (UAI '03).

remained a central challenge in learning graphical models in general, and Bayesian networks in particular. Hidden entities play a central role in many real-life problems: an unknown regulating mechanism can be the key to complex biological systems; correlating symptoms might hint at a hidden fundamental problem in a diagnostic system; an intentionally masked economic power might be the cause of related financial phenomena. Indeed, hidden variables typically serve as a summarizing mechanism that "captures" information from some of the observed variables and "passes" this information to some other part the network. As such, hidden variables can simplify the network structure and consequently lead to better generalization.

When learning the parameters of a Bayesian network with missing values or hidden variables, the most common approach is to use some variant of the *expectation maximization* (EM) algorithm (Dempster et al., 1977; Lauritzen, 1995). This algorithm performs a greedy search of the likelihood surface and converges to a local stationary point (usually a local maximum). Unfortunately, in challenging real-life learning problems, there are many local maxima that can trap EM in a poor solution. Attempts to address this problem use a variety of strategies (*e.g.*, Glover and Laguna (1993); Kirkpatrick et al. (1983); Rose (1998); Elidan et al. (2002)). When learning structure, the *structural EM* (SEM) algorithm (Friedman, 1997; Meila and Jordan, 1998; Thiesson et al., 1998) can adapt the network topology. In this approach, as in the classical parametric EM algorithm, we use the distribution induced by our current model, to probabilistically *complete* the data. Unlike parametric EM, we then use the completed data to evaluate different candidate structures. This allows us to perform structure improvement steps in the *M-Step* of a structural EM iteration. As in the case of EM, while convergence is guaranteed, the algorithm typically converges to a local maximum.

An even more challenging problem is that of *model selection* with hidden variables. This involves choosing the number of hidden variables, their cardinalities and the dependencies between them and the observed entities of the domain. These decisions are crucial to achieve good generalization. In particular, in hard real-life learning problems, structural EM will perform poorly unless some prior knowledge of the interaction between the hidden and observed variables exists or if the cardinality of the hidden variables is not (at least approximately) known. These challenging problems have received surprisingly little attention.

In this paper, we introduce a new approach to learning Bayesian networks with hidden variables. We pose the learning problem as an the optimization of a target function that includes a tradeoff between two information theoretic objectives. The first objective is to compress information about the training data. Intuitively, this is required when we want to generalize from the training data to new unseen instances. The second objective is to make the hidden variables informative about the observed attributes to ensure they preserve the *relevant* information. This objective is directly related to maximizing the likelihood of the training data. By exploring different relative weightings of these two objectives, we are able to bypass local maxima and learn better models.

Our approach builds on the *information bottleneck* framework of Tishby et al. (1999) and its multivariate extension (Friedman et al., 2001). This framework provides methods for constructing a set of new variables $\mathbf{T}$ that are stochastic functions of one set of variables $\mathbf{Y}$ and at the same time provide information on another set of variables $\mathbf{X}$. The intuition is that the new variables $\mathbf{T}$ capture the relevant aspects of $\mathbf{Y}$ that are informative about $\mathbf{X}$. We show how to pose the learning problem within the multivariate information bottleneck framework and derive a target Lagrangian for the hidden variables. We then show that this Lagrangian is an extension of the Lagrangian formulation of EM of Neal and Hinton (1998), with an additional regularization term. By controlling the strength

of this information theoretic regularization term using a *scale parameter*, we can explore a range of target functions. On the one end of the spectrum there is a trivial target where compression of the data is total and all relevant information is lost. On the other extreme is the target function of EM.

This continuum of target functions allow us to learn using a procedure motivated by the *deterministic annealing* approach (Rose, 1998). We start with the optimum of the trivial target function and slowly change the scale parameter while tracking the local optimum solution at each step on the way. To do so, we present an alternative view of the optimization problem in the joint space of the model parameters and the scale parameter. This provides an appealing method for scanning the range of solutions as in *homotopy continuation* (Watson, 2000).

We generalize our *information bottleneck expectation maximization* (IB-EM) framework for multiple hidden variables and any Bayesian network structure. To make learning feasible for large, real-life problems we show how to introduce variational approximation assumptions into the framework. We further show that, similarly to the case of standard parametric EM, there is a formal relation between the information bottleneck objective in this case and the *variational EM* functional (Jordan et al., 1998).

We then extend the approach to deal with structure learning. As we show, we can easily incorporate our method into the structural EM framework to deal with *model selection* with hidden variables. In doing so, we perform continuation interleaved with model selection steps that change the structure and the scope of the model. On top of standard structure modification steps of adding and removing edges, we introduce two model enrichment operators that take advantage of emergent information cues during the continuation process. The first operator can adapt the cardinality of a hidden variable. Specifically, the cardinality of a hidden variable can increase during the continuation process, increasing the likelihood as long as it is beneficial to do so. The second operator introduces new hidden variables into the network structure. Intuitively, a hidden variable is introduced as a parent of a subset of nodes whose interactions are poorly explained by the current model.

We demonstrate the effectiveness of our information bottleneck EM algorithm in several learning scenarios. First, we learn parameters in general Bayesian networks for several challenging real-life data sets and show significant improvement in generalization performance on held-out test data. Second, we demonstrate the importance of cardinality adaptation for good generalization. We then show how our operator for enriching the network structure with new hidden variables leads to significantly superior models, for several complex real-life problems. Finally, we show that combining both structure enrichment and cardinality adaptation results in further improvement of test performance.

The paper is organized as follows. In Section 2, we give a short background on learning Bayesian networks and on the *Multivariate information bottleneck* of Friedman et al. (2001). In Section 3, we present the basic framework of our IB-EM algorithm. In Section 4, we show how to combine this algorithm with continuation to bypass local maxima. In Section 5 we extend the framework to multiple hidden variables. In Section 6, we demonstrate the method for parameter learning in real-life scenarios. In Section 7, we show how our method can be combined with the structural EM algorithm to learn the structure of a network with hidden variables. In Section 8, we take advantage of emergent structure during the continuation process, and present a method for learning the cardinality of the hidden variables. We apply this method to real-life data in Section 9. In Section 10, we address the model selection challenge of learning new hidden variables. We present experimental evaluation for several real-life problems in Section 11. In Section 12, we give

a brief overview of relevant works, and in Section Section 13 we end with a discussion and future directions.

2. Background

In this section we briefly present the basics of learning Bayesian networks from data followed by the essentials of the *multivariate information bottleneck* framework that forms the basis of our approach.

2.1 Bayesian Networks

Consider a finite set $\mathcal{X} = \{X_1, \ldots, X_n\}$ of random variables, where each variable X_i may take on values from a finite set, denoted by $Val(X_i)$. We use capital letter such as X, Y, Z for variable names and lower case letters such as x, y, z to denote specific values taken by those variables. We use bold letters such as $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ when referring to sets of variables. A *Bayesian network* (Pearl, 1988) is an annotated directed acyclic graph that encodes a joint probability distribution over $\mathcal{X}$. Formally, a Bayesian network over $\mathcal{X}$ is a pair $B = \langle \mathcal{G}, \Theta \rangle$. The first component, $\mathcal{G}$, is a directed acyclic graph whose vertices correspond to the random variables in $\mathcal{X}$. The edges in the graph represent direct dependencies between the variables. The graph $\mathcal{G}$ represents independence properties that are assumed to hold in the underlying distribution: Each X_i is independent of its non-descendants given its parents $\mathbf{Pa}_i$ denoted by $(X_i \perp \mathit{NonDescendants}_i \mid \mathbf{Pa}_i)$. The second component, Θ, represent the set of parameters that quantify the network. Each node is annotated with a *conditional probability distribution* $P(X_i \mid \mathbf{Pa}_i)$, representing the conditional probability of the node X_i given its parents in $\mathcal{G}$, defined by the parameters $\theta_{x_i|\mathbf{pa}_i}$ for each value of X_i and $\mathbf{Pa}_i$. A Bayesian network defines a unique joint probability distribution over $\mathcal{X}$ given by

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \mathbf{Pa}_i).$$

In this distribution, a variable X_i is independent of the rest of the variables given its *Markov blanket* variables. These include the variable's parents, direct children and the parents of those children (spouses).

Given a network structure $\mathcal{G}$, the problem of learning a Bayesian network can be stated as follows: Given a training set $\mathcal{D} = \{\mathbf{x}[1], \ldots, \mathbf{x}[M]\}$ of instances of $\mathbf{X} \subset \mathcal{X}$, we want to learn parameters for the network. In the *Maximum Likelihood* setting we want to find the parameter values θ that maximize the log-likelihood function

$$\log P(\mathcal{D} \mid \mathcal{G}, \theta) = \sum_m \log P(\mathbf{x}[m] \mid \mathcal{G}, \theta).$$

This function can be equivalently (up to a multiplicative constant) written as $\boldsymbol{E}_{\hat{P}}[\log P(\mathbf{X} \mid \mathcal{G}, \theta)]$ where $\hat{P}$ is the empirical distribution in $\mathcal{D}$. When all instances in $\mathcal{D}$ are complete, estimating the *maximum likelihood* parameters can be done efficiently using a closed form solution. This involves empirical sufficient statistics in the form of joint counts

$$N(x_i, \mathbf{pa}_i) = \sum_m 1\{X_i[m] = x_i, \mathbf{Pa}_i[m] = \mathbf{pa}_i\}, \tag{1}$$

where $1\{\}$ is the indicator function. When learning multinomial conditional parameterization, using Dirichlet priors (DeGroot, 1970) amounts to augmenting the empirical counts with *pseudo-counts*

$\alpha(x_i, \mathbf{pa}_i)$.[1] These can thought of as adding imaginary instances that are distributed according to a certain distribution (*e.g.*, uniform) to the training data (Heckerman, 1998). Consequently, from this point on we view priors as modifying the empirical distribution $\hat{P}$ with additional instances, and then apply the maximum likelihood principle.

When learning with hidden variables, the problem is more complex. Since we observe only partial instances, learning also involves "guessing" the values of the hidden variables. In the *expectation maximization* (EM) algorithm (Dempster et al., 1977; Lauritzen, 1995) and its variants (Neal and Hinton, 1998), this issue is addressed by using an auxiliary distribution Q that provides a proxy for the empirical distribution. In the M-step of EM we estimate parameters as though this was the true empirical distribution. In the E-step, we use the data and the current model to optimize the auxiliary distribution over the hidden values resulting in a *completed* empirical distribution. Each of these steps is simpler than the original problem and is guaranteed not to decrease the likelihood. Unfortunately, EM iterations are prone to getting trapped at local maxima, since each step is biased by the choices made by the previous ones. Attempts to address this problem use a variety of strategies (*e.g.*, Glover and Laguna (1993); Kirkpatrick et al. (1983); Rose (1998); Elidan et al. (2002)).

Learning the structure of a network poses additional challenges as the number of possible structures is super-exponential. In practice, structure learning is typically done using a local search procedure, which examines local structure changes that are easily evaluated (add, delete or reverse an edge). This search is usually guided by a scoring function such as the MDL principle based score (Lam and Bacchus, 1994) or the *Bayesian score* (BDe) (Heckerman et al., 1995). Both scores penalize the likelihood of the data to limit the model complexity. An important characteristic of these scoring functions is that when the data instances are complete (that is, each training instance assigns values to all of the variables) the score is *decomposable*. More precisely, the score can be rewritten as the sum

$$\mathcal{S}core(\mathcal{G} : \mathcal{D}) = \sum_i \text{FamScore}_{X_i}(\mathbf{Pa}_i \ : \ \mathcal{D}),$$

where FamScore_{X_i} is the *local* contribution of X_i to the total network score. This term depends only on values of X_i and $\mathbf{Pa}_{X_i}$ in the training instances. In particular, the BDe score is defined as

$$\text{Score}_{\text{BDe}}(\mathcal{G} : \mathcal{D}) = \sum_i \sum_{\mathbf{pa}_i} \left(\log \frac{\Gamma(\alpha(\mathbf{pa}_i))}{\Gamma(N(\mathbf{pa}_i) + \alpha(\mathbf{pa}_i))} + \sum_{x_i} \log \frac{\Gamma(N(x_i, \mathbf{pa}_i) + \alpha(x_i, \mathbf{pa}_i))}{\Gamma(\alpha(x_i, \mathbf{pa}_i))} \right), \quad (2)$$

where Γ is the Gamma function that generalizes the factorial function for real numbers, the terms $\alpha()$ are hyper-parameters of the prior distributions over the parameterizations and the terms $N()$ are the corresponding empirical *sufficient statistics*.

In the presence of incomplete data or hidden variables, the structural EM framework (Friedman, 1997; Meila and Jordan, 1998; Thiesson et al., 1998) can adapt the network structure. In this approach, as in classical *parametric* EM, we use the distribution induced by our current model to probabilistically complete the data. Unlike parametric EM, we then use the completed data to evaluate different candidate structures, and perform structure improvement steps in the *M-step* of the structural EM iteration. As in the case of EM, convergence is guaranteed, albeit to a local maximum. Scoring candidate structures in this scenario is more complex, and computation of the score is typically intractable. Thus, we need to resort to approximations such as the *Cheeseman-Stutz* (CS)

[1] The use of pseudo-counts is slightly different depending on whether we do MAP or Bayesian estimation and depends on the representation used (see (Thiesson, 1997) for more details).

score (Cheeseman et al., 1988; Chickering and Heckerman, 1997), which combines the likelihoods of the parameters found by EM, with an estimate of the penalty term associated with structure.

2.2 Multivariate Information Bottleneck

The *information bottleneck* method (Tishby et al., 1999) is a general non-parametric information-theoretic clustering framework. Given a joint distribution $Q(Y,X)$ of two variables, it attempts to extract the relevant information that Y contains about X. We can think of such information extraction as partitioning the possible values of Y into coarser distinctions that are still informative about X. (The actual details are more complex, as we shall see shortly). For example, we might want to partition the words (Y) appearing in several documents in a way that is most relevant to the topics (X) of these documents.

To achieve this goal, we first need a relevance measure between two random variables X and Y with respect to some probability distribution $Q(X,Y)$. The symmetric *mutual information* measure (Cover and Thomas, 1991)

$$\boldsymbol{I}_Q(X;Y) = \sum_{x,y} Q(x,y) \log \frac{Q(x,y)}{Q(x)Q(y)}$$

is a natural choice as it measures the average number of bits needed to convey the information X contains about Y and vice versa. It is bounded from below by zero when the variables are independent, and attains its maximum when one variable is a deterministic function of the other.

The next step is to introduce a new variable T. This variable provides the *bottleneck* relation between X and Y. In our words and documents example, we want T to maintain the distinctions between words (Y) that provide information for determining the topic of a document (X). For example, the words 'music' and 'lyrics' typically occur together and are typical of the same topic, and thus the distinction between them does not contribute to the prediction of the topic. At the same time, we want T to distinguish between 'music' and 'politics' as they correlate with markedly different topics. Formally, we define T using a stochastic function $Q(T \mid Y)$. On the one hand we want T to compress Y, while on the other hand we want it to preserve information that is relevant to X. Using the mutual information defined above, a balance between these two competing goals is achieved by minimization of the Lagrangian

$$\mathcal{L}[Q] = \boldsymbol{I}_Q(Y;T) - \beta \boldsymbol{I}_Q(T;X), \tag{3}$$

where the parameter β controls the tradeoff. Tishby et al. (1999) show that the optimal partition for a given value of β satisfies

$$Q(t \mid y) = \frac{Q(t)}{Z(y,\beta)} \exp\left\{-\beta \boldsymbol{D}(Q(X \mid y) \| Q(X \mid t))\right\},$$

where

$$\boldsymbol{D}(P(\mathbf{X}) \| Q(\mathbf{X})) = \sum_{\mathbf{x}} P(\mathbf{x}) \log \frac{P(\mathbf{x})}{Q(\mathbf{x})}$$

is the Kulback Leibler divergence between the distributions P and Q over the set of random variables $\mathbf{X}$ (Cover and Thomas, 1991). Repeated iterations of these equations for all t and y converge to a (local) maximum where all equations are satisfied. Practical application of this approach for various clustering problems was demonstrated in several works (e.g., (Slonim and Tishby, 2000, 2001)).

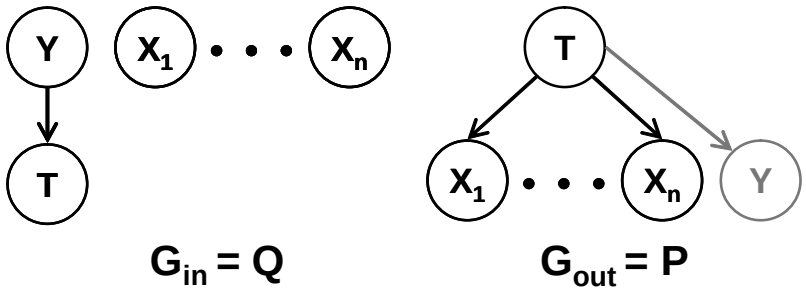


Figure 1: Definition of $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$ for the multivariate information bottleneck framework. $\mathcal{G}_{in}$ encodes the distribution Q that compresses Y. $\mathcal{G}_{out}$ encodes the distribution P that we want to approximate using Q.

The multivariate extension of this framework (Friedman et al., 2001) allows us to consider the interactions of multiple observed variables using several bottleneck variables. For example, we might want to compress words (Y) in a way that preserves information both on the topic of the document (X_1) and on the author of that document (X_2). In addition, there probably is a strong correlation between the author and the topics he writes about. Evidently, the number of possible interactions may be large, and so the framework allows us to specify the interactions we desire. These interactions are represented via two Bayesian networks. The first, called $\mathcal{G}_{in}$, represents the required compression, and the second, called $\mathcal{G}_{out}$, represents the independencies that we are striving for between the bottleneck variables and the target variables. In Figure 1, $\mathcal{G}_{in}$ specifies that T is a stochastic function of its parent in the graph Y. $\mathcal{G}_{out}$ specifies that we want T to make Y and the variables X_i's independent of each other.

Formally, the framework of Friedman et al. (2001), attempts to minimize the Lagrangian

$$\mathcal{L}^{(1)}[\mathcal{G}_{in}, Gout] = I^{\mathcal{G}_{in}} - \beta I^{\mathcal{G}_{out}},$$

where

$$I^{\mathcal{G}} = \sum_i \boldsymbol{I}(X_i; \mathbf{Pa}_i^{\mathcal{G}})$$

and the information is computed with respect to the probability distribution represented by the network $\mathcal{G}$. This objective is a direct generalization of Eq. (3), and as before, tractable self-consistent equations characterize the optimal partitioning. Note that, as in the basic information bottleneck formulation, the two objective of the above Lagrangian are competing. On the one hand we want to compress the information between all bottleneck variables $\mathbf{T}$ and their parents in $\mathcal{G}_{in}$. On the other hand we want to preserve, or maximize, the information between the variables and their parents in $\mathcal{G}_{out}$.

Friedman et al. (2001) also present an analogous variational principal that will be useful in our framework. Briefly, the problem is reformulated as a tradeoff between compression of mutual information in $\mathcal{G}_{in}$ so that the bottleneck variable(s) $\mathbf{T}$ help us describe a joint distribution that follows that form of a target Bayesian network $\mathcal{G}_{out}$. Formally, they attempt to minimize the following objective function

$$\mathcal{L}^{(2)}[Q,P] = \boldsymbol{I}_Q(Y;T) + \gamma \boldsymbol{D}(Q(Y,T,\mathbf{X}) \| P(Y,T,\mathbf{X})), \tag{4}$$

where Q and P are joint probabilities that can be represented by the networks of $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$, respectively. The two principals are analogous under the transformation $\beta = \frac{\gamma}{1+\gamma}$ and assuming $I^{Gin} = \boldsymbol{I}_Q(Y;T)$. See Friedman et al. (2001) for more details of the relation between the two principals.

The minimization of the above Lagrangian is over possible parameterizations of $Q(T \mid Y)$ (the marginal $Q(Y,\mathbf{X})$ is given and fixed) and over possible parameterizations of $P(Y,T,\mathbf{X})$ that can be represented by $\mathcal{G}_{out}$. In other words, we want to compress Y in such a way that the distribution defined by $\mathcal{G}_{in}$ is as close as possible to desired distribution of $\mathcal{G}_{out}$. The analogous principal gives us a new view on why these two objectives are conflicting: Consider a distribution that is consistent with $\mathcal{G}_{in}$ so that T is independent of X given Y. On the other hand, a distribution consistent with a specific choice of $\mathcal{G}_{out}$ may require that X is independent of Y given T. Constructing a distribution where both of these requirements actually hold is not useful, may results in T that is equal to either X or Y, making this bottleneck variable redundant.

The scale parameter γ balances the above two factors. When γ is zero we are only interested in compressing the variable Y and we resort to the trivial solution of a single cluster (or an equivalent parameterization). When γ is high we concentrate on choosing $Q(T \mid Y)$ that is close to a distribution satisfying the independencies encoded by $\mathcal{G}_{out}$. Returning to our word-document example. We might be willing to forgo the distinction between 'football' and 'baseball' in which case we would set γ to a relatively low value. On the other hand, we might even want to make a minute distinction between 'Pentium' and 'Celeron' in which case we would set γ to a high value. Obviously, there is no single correct value of γ but rather a range of possible tradeoffs. Accordingly, several approaches were devised to explore the spectrum of solutions as γ varies. These include Deterministic annealing like approaches that start with small value of γ and progressively increase it (Friedman et al., 2001), as well as agglomerative approaches that start with a highly refined solution and gradually compress it (Slonim and Tishby, 2000, 2001; Slonim et al., 2002).

3. Information Bottleneck Expectation Maximization

The main focus of the multivariate information bottleneck (see is on distribution $Q(T \mid Y)$ that is a local maxima solution of the Lagrangian This distribution can be thought of as a soft clustering of the original data. Our emphasis in this work is somewhat different. Given a data set $\mathcal{D} = \{\mathbf{x}[1],\ldots,\mathbf{x}[M]\}$ over the observed variables $\mathbf{X}$, we are interested in learning a better generative model describing the distribution of the observed attributes $\mathbf{X}$. That is, we want to give high probability to new data instances from the same source. In the learned network, the hidden variables will serve to summarize some part of the data while retaining the relevant information on (some) of the observed variables $\mathbf{X}$.

We start by extending the multivariate information bottleneck framework for the task of generalization where, in addition to the task of clustering, we are also interested in learning the generative model P. We emphasize that this is a conceptually different task. In particular, the common view of the information bottleneck framework is as a non-parametric information-theoretic method for clustering (the obvious exception is the work of Slonim and Weiss (2002) mentioned below). In generative learning, on the other hand, we are interested in modeling the distribution. That is, we are ultimately interested in *parameterizing* a specific model so that our generalization prediction on unseen future instances is improved. We start by considering this task for the case of a single hidden variable T and then, in Section 5, extend the framework to several hidden variables.

3.1 The Information Bottleneck EM Lagrangian

If we were only interested in the *training* data and the cardinality of the hidden variable allows it, each state of the hidden variable would have been assigned to a different instance. Consider, for example, a variable T with $|T|$ states that defines a soft clustering on the specific identity of words (Y) appearing in documents while preserving the information relevant to the topic (X) of these documents. Now suppose we are given a set of instances $\mathcal{D} = \{word[i], topic[i]\}$ where i goes from 1 to M, the number of instances. If $|T| = M$ then we could simply deterministically set $Q(T = i \mid word[i]) = 1$ and then predict $topic[i]$ perfectly. While this model achieves perfect training performance, it will clearly have no generalization abilities. Since we are also interested in unknown future samples, we intuitively require that the learned model "forget" the specifics of the training examples. However, in doing so we will also deteriorate the (previously deterministic) prediction of the observed variables. Thus, there is a tradeoff between the compression of the identity of specific instances and the preservation of the information relevant to the observed variables.

We now formalize this idea for the task of learning a generative model over the variables $\mathbf{X}$ and the hidden variable T. We define an additional variable Y to be the instance identity in the training data $\mathcal{D}$. That is, Y takes values in $\{1,\ldots,M\}$ and $Y[m] = m$. We define $Q(Y,\mathbf{X})$ to be the empirical distribution of the variables $\mathbf{X}$ in the data, augmented with the distribution of the new variable Y. For each instance y, $\mathbf{x}[y]$ are the values $\mathbf{X}$ take in the specific instance. We now apply the information bottleneck framework with the graph $\mathcal{G}_{in}$ of Figure 1. The choice of the graph $\mathcal{G}_{out}$ depends on the network model that we want to learn. We take it to be the target Bayesian network, augmented by the additional variable Y, where we set T as Y's parent. For simplicity, we consider as a running example the simple clustering model of $\mathcal{G}_{out}$ where T is the parent of $X_1,\ldots,X_n$. In practice, and as we show in Section 6 any choice of $\mathcal{G}_{out}$ can be used. We now want to optimize the Bottleneck objective as defined by these two networks. This will attempt to define a conditional probability $Q(T \mid Y)$ so that $Q(T,Y,\mathbf{X}) = Q(T \mid Y)Q(Y,\mathbf{X})$ can be approximated by a distribution that factorizes according to $\mathcal{G}_{out}$. This construction will aim to find T that captures the relevant information the instance identity has about the observed attributes. The following proposition concretely defines the objective function for the particular choice of $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$ we are dealing with.

Proposition 1
Let

1. *Y be the instance identity as defined above;*
2. *$\mathcal{G}_{in}$ be a Bayesian network structure such that such that T is independent of $\mathbf{X}$ given Y; and*
3. *$\mathcal{G}_{out}$ be a Bayesian network structure such that Y is a leaf with T as its only parent.*

Then, minimizing the information bottleneck objective function in Eq. (4) is equivalent to minimizing the Lagrangian

$$\mathcal{L}_{EM} = \boldsymbol{I}_Q(T;Y) - \gamma(\boldsymbol{E}_Q[\log P(\mathbf{X},T)] - \boldsymbol{E}_Q[\log Q(T)]),$$

as a function of $Q(T \mid Y)$ and $P(\mathbf{X},T)$.

Note that once the above conditions are satisfied, we can still arbitrarily choose the structure of $\mathcal{G}_{out}$, which encodes independencies of the distribution P we ultimately wish to learn.

Proof: Using the chain rule and the fact that Y and X are independent given T in $\mathcal{G}_{out}$), we can write $P(Y,\mathbf{X},T) = P(Y \mid T)P(\mathbf{X},T)$. Similarly, using the chain rule and the fact that X and T are independent given Y in $\mathcal{G}_{in}$, we can write $Q(Y,\mathbf{X},T) = Q(Y \mid T)Q(T)Q(\mathbf{X} \mid Y)$. Thus,

$$\begin{aligned} \boldsymbol{D}(Q(Y,\mathbf{X},T)\|P(Y,\mathbf{X},T)) &= \boldsymbol{E}_Q\left[\log \frac{Q(Y \mid T)Q(T)Q(\mathbf{X} \mid Y)}{P(Y \mid T)P(\mathbf{X},T)}\right] \\ &= \boldsymbol{D}(Q(Y \mid T)\|P(Y \mid T)) \\ &\quad + \boldsymbol{E}_Q[\log Q(\mathbf{X} \mid Y)] \\ &\quad + \boldsymbol{E}_Q[\log Q(T)] \\ &\quad - \boldsymbol{E}_Q[\log P(\mathbf{X},T)]. \end{aligned}$$

By setting $P(Y \mid T) = Q(Y \mid T)$, the first term reaches zero, its minimal value. The second term is a constant since we cannot change the input distribution $Q(\mathbf{X} \mid Y)$. Thus, we need to minimize the last two terms and the result follows immediately. ∎

An immediate question is how this target function relates to standard maximum likelihood learning. To explore the connection, we use a formulation of EM introduced by Neal and Hinton (1998). Although EM is usually thought of in terms of changing the parameters of the target function P, Neal and Hinton show how to view it as a dual optimization of P and an auxiliary distribution Q. This auxiliary distribution replaces the given empirical distribution $Q(\mathbf{X})$ with a completed empirical distribution $Q(\mathbf{X},T)$. Using our notation in the above discussion, we can write the functional defined by Neal and Hinton as

$$\mathcal{F}[Q,P] = \boldsymbol{E}_Q[\log P(\mathbf{X},T)] + \boldsymbol{H}_Q(T \mid Y), \tag{5}$$

where $\boldsymbol{H}_Q(T \mid Y) = \boldsymbol{E}_Q[-\log Q(T \mid Y)]$, and $Q(\mathbf{X},Y)$ is fixed to be the observed empirical distribution.

Theorem 2 (Neal and Hinton, 1998) *If (Q^*,P^*) is a stationary point of $\mathcal{F}$, then P^* is a stationary point of the log-likelihood function $\boldsymbol{E}_Q[\log P(\mathbf{X})]$.*

Moreover, Neal and Hinton show that an EM iteration corresponds to maximizing $\mathcal{F}[Q,P]$ with respect to $Q(T \mid Y)$ while holding P fixed, and then maximizing $\mathcal{F}[Q,P]$ with respect to P while holding $Q(T \mid Y)$ fixed. The form of $\mathcal{F}[Q,P]$ is quite similar to the IB-EM Lagrangian, and indeed we can relate the two.

Theorem 3 $\mathcal{L}_{EM} = (1-\gamma)\boldsymbol{I}_Q(T;Y) - \gamma\mathcal{F}[Q,P]$.

Proof: Plugging the identity $\boldsymbol{H}_Q(T \mid Y) = -\boldsymbol{E}_Q[\log Q(T)] - \boldsymbol{I}_Q(T;Y)$ into the EM functional we can write

$$\mathcal{F}[Q,P] = \boldsymbol{E}_Q[\log P(\mathbf{X},T)] - \boldsymbol{E}_Q[\log Q(T)] - \boldsymbol{I}_Q(T;Y).$$

If we now multiply this by γ, and re-arrange terms, we get the form of Proposition 1. ∎

As a consequence, *minimizing* the IB-EM Lagrangian is equivalent to *maximizing* the EM functional combined with an information theoretic regularization term. When $\gamma = 1$, the solutions of

the Lagrangian and the EM functional coincide and finding a local minimum of $\mathcal{L}_{EM}$ is equivalent to finding a local maximum of the likelihood function. Slonim and Weiss (2002) provide a similar result for the specific case where the generative model is a mixture model of a univariate X. Their formulation is different than ours in several subtle details that do not allow a direct relation between the two methods. Nonetheless, both Slonim and Weiss (2002) and Theorem 3 show that for a particular value of γ, the information bottleneck Lagrangian coincides with the likelihood objective of EM. The main difference between the two results is the choice of generative models, in our case general multi-variate Bayesian networks, and in the case of Slonim and Weiss (2002), univariate mixture models.

3.2 The Information Bottleneck EM Algorithm

Using the above results, we can now describe the *Information Bottleneck EM* algorithm given a specific value of γ. The algorithm can be described similarly to the EM iterations of Neal and Hinton (1998).

- **E-step**: Maximize $-\mathcal{L}_{EM}$ by varying $Q(T \mid Y)$ while holding P fixed.
- **M-step**: Maximize $-\mathcal{L}_{EM}$ by varying P while holding Q fixed.

Note that the algorithm is formulated in terms of maximizing $-\mathcal{L}_{EM}$ rather than minimizing $\mathcal{L}_{EM}$ to enhance the relation between the Lagrangian and the EM objective.

The M-Step is essentially the standard maximum likelihood optimization of Bayesian networks. To see that, note that the only term that involves P is $\boldsymbol{E}_Q[\log P(\mathbf{X},T)]$. This term has the form of a log-likelihood function, where Q plays the role of the empirical distribution. Since the distribution is over all the variables, we can use sufficient statistics of P for efficient estimates, just as in the case of complete data. Thus, the M step consists of computing expected sufficient statistics given Q, and then using a closed form formula for choosing the parameters of P.

The E-step is a bit more involved. We need to maximize with respect to $Q(T \mid Y)$. To do this we use the following two results that are variants of Theorem 7.1 and Theorem 8.1 of Friedman et al. (2001) and proved using similar techniques (see Appendix A for the full proof).

Proposition 4 *Let $\mathcal{L}_{EM}$ be defined via $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$ as in Proposition 1. $Q(T \mid Y)$ is a stationary point of $\mathcal{L}_{EM}$ with respect to a fixed choice of P if and only if for all values t and y of T and Y, respectively,*

$$Q(t \mid y) = \frac{1}{Z(y,\gamma)} Q(t)^{1-\gamma} P(\mathbf{x}[y],t])^{\gamma}, \tag{6}$$

where $Z(y,\gamma)$ is a normalizing constant:

$$Z(y,\gamma) = \sum_{t'} Q(t')^{1-\gamma} P(\mathbf{x}[y],t'])^{\gamma}.$$

Note that, as can be expected from Theorem 3, when $\gamma = 1$ the update equation reduces to $Q(t \mid y) \propto P(\mathbf{x}[y],t)$ which is equivalent to the standard EM update equation.

Proposition 5 *A stationary point of $\mathcal{L}_{EM}$ is achieved by iteratively applying the self-consistent equations of Proposition 4.*

Combining this result with the result of Neal and Hinton that show that optimization of P increases $F(P,Q)$, we conclude that both the E-step and the M-step increase $-\mathcal{L}_{EM}$ until we reach a stationary point. As in standard EM, in most cases the stationary convergence point reached by applying these self-consistent equations will be a local maximum of $-\mathcal{L}_{EM}$, or a local minimum of $\mathcal{L}_{EM}$.

4. Bypassing Local Maxima using Continuation

As discussed in the previous section, the parameter γ balances between compression of the data and the fit of parameters to $\mathcal{G}_{out}$. When γ is close to 0, our only objective is compressing the data and the effective dimensionality of T will be 1, leading to a trivial solution (or an equivalent parameterization). At larger values of γ we pay more and more attention to the distribution of $\mathcal{G}_{out}$, and we can expect additional states of T to be utilized. Ultimately, we can expect each sample to be assigned to a different cluster (if the dimensionality of T allows it), in which case there is no compression of Y and the information about the Xs is fully preserved. Theorem 3 also tells us that at the limit of $\gamma = 1$ our solution will actually converge to one of the standard EM solutions. In this section we show how to utilize the inherent tradeoff determined by γ to bypass local maxima towards a better solution at $\gamma = 1$.

Naively, we could allow a large cardinality for the hidden variable, set γ to a high value and find the solution of the bottleneck problem. There are several drawbacks to this approach. First, we will typically converge to a sub-optimal solution for the given cardinality and γ, all the more so for $\gamma = 1$ where there are many such maxima. Second, we often do not know the cardinality that should be assigned to the hidden variable. If we use a cardinality for T that is too large, learning will be less robust and might become intractable. If T has too low a dimensionality, we will not fully utilize the potential of the hidden variable. We would like to somehow identify the beneficial number of clusters without having to simply try many options.

To cope with this task, we adopt the *deterministic annealing* strategy (Rose, 1998). In this strategy, we start with $\gamma = 0$ where a single cluster solution is optimal and compression is total. We then progress toward higher values of γ. This gradually introduces additional structure into the learned model. Intuitively, the algorithm starts at a place where a single, easy to compute solution exists, and tracks it through various stages of progressively complex solutions hopefully bypassing local maxima by staying close to the optimal solution at each value of γ. There are several ways of executing this general strategy. The common approach is simply to increase γ in fixed steps, and after each increment apply the iterative algorithm to re-attain a (local) maxima with the new value of γ. On the problems we examine in Section 6, this naive approach did not prove successful.

Instead, we use a more refined approach that utilizes *continuation methods* for executing the annealing strategy. This approach automatically tunes the magnitude of changes in the value of γ, and also tracks the solution from one iteration to the next. To perform continuation, we view the optimization problem in the joint space of the parameters and γ. In this space we want to follow a smooth path from the trivial solution at $\gamma = 0$ to a solution at $\gamma = 1$. Furthermore, we would like this path to follow a local maximum of $\mathcal{L}_{EM}$. As was shown above, this is equivalent to requiring that the fixed point equations hold at all points along the path. Continuation theory (Watson, 2000) guarantees that, excluding degenerate cases, such a path, free of discontinuities, indeed exists. Figure 2 shows a synthetic illustration of the setup. (a) shows the likelihood function of the two extremes of the easy solution at $\gamma = 0$ and the EM function at $\gamma = 1$ in the joint (γ, Q)-space. (b) shows the range of solutions between these extremes and marks the desired path we would like to follow.

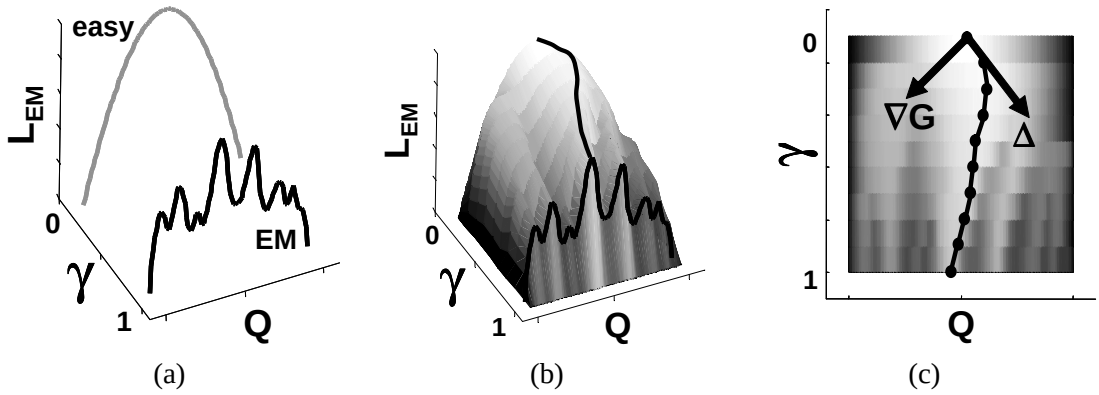


Figure 2: Synthetic illustration of the continuation process. (a) shows the easy likelihood function at $\gamma = 0$ and the complex EM function at $\gamma = 1$. (b) spans the full range of functions and marks the desired path for following the maximum. (c) demonstrates a single step in the continuation process. The gradient $\nabla_{Q,\gamma} G$ is computed and then the orthogonal direction is taken.

We start by characterizing such paths. Note that once we fix the parameters $Q(T \mid Y)$, the M-step maximization of the parameters in P is fully determined as a function of Q. Thus, we take $Q(T \mid Y)$ and γ as the only free parameters in our problem. As we have shown in Proposition 4, when the gradient of the Lagrangian is zero, Eq. (6) holds for each value of t and y. Thus, we want to consider paths where all of these equations hold. Rearranging terms and taking a log of Eq. (6) we define

$$G_{t,y}(Q,\gamma) = -\log Q(t \mid y) + (1-\gamma)\log Q(t) + \gamma \log P(\mathbf{x}[y], y) - \log Z(y,\gamma). \tag{7}$$

Clearly, $G_{t,y}(Q,\gamma) = 0$ exactly when Eq. (6) holds for all t and y. Our goal is then to follow an equi-potential path where all $G_{t,y}(Q,\gamma)$ functions are zero starting from some small value of γ up to the desired EM solution at $\gamma = 1$.

Suppose we are at a point (Q_0, γ_0), where $G_{t,y}(Q_0,\gamma_0) = 0$ for all t and y. We want to move in a direction $\Delta = (dQ, d\gamma)$ so that $(Q_0 + dQ, \gamma_0 + d\gamma)$ also satisfies the fixed point equations. To do so, we want to find a direction Δ, so that

$$\forall t, y, \;\; \nabla_{Q,\gamma} G_{t,y}(Q_0,\gamma_0) \cdot \Delta = 0, \tag{8}$$

where $\nabla_{Q,\gamma} G_{t,y}(Q_0,\gamma_0)$ is the gradient of $G_{t,y}(Q_0,\gamma_0)$ with respect to the parameters Q and γ. Computing these derivatives with respect to each of the parameters results in a derivative matrix

$$H_{t,y}(Q,\gamma) = \left(\frac{\partial G_{t,y}(Q,\gamma))}{\partial Q(t|y)} \;\middle|\; \frac{\partial G_{t,y}(Q,\gamma)}{\partial \gamma} \right). \tag{9}$$

Rows of the matrix correspond to each of the $L = |T| \times |Y|$ functions of Eq. (7), corresponding to joint combinations of the $|T|$ states of the bottleneck variable T and the $|Y| = M$ number of possible values of the instance identity variable Y. The columns correspond to the L parameters of Q as well as γ. The entries correspond to the partial derivative of the function associated with the row with respect to the parameter associated with the column.

To find a direction Δ that satisfies Eq. (8) we need to satisfy the matrix equation

$$H_{t,y}(Q_0,\gamma_0)\Delta = 0. \tag{10}$$

In other words, we are trying to find a vector in the null-space of $H_{t,y}(Q_0,\gamma_0)(Q_0,\gamma_0)$. The matrix H is an $L \times (L+1)$ matrix and its null-space is defined by the intersection of L tangent planes, and is of dimension $L+1-\text{Rank}(H_{t,y}(Q,\gamma))$. Numerically, excluding measure zero cases (Watson, 2000), we expect $\text{Rank}(H_{t,y}(Q_0,\gamma_0))$ to be full, *i.e.*, L. Thus, a unique line that (up to scaling) defines the null space, and we can choose any vector along it. To follow the path to our target objective at $\gamma = 1$ we choose the direction that always increases γ (we discuss the choice of the length of this vector below). Returning to Figure 2, (c) illustrates this process. Shown is joint (γ, Q)-space with the grey-level denoting the value of the likelihood function. At each point in the learning process the gradient of G is evaluated and the orthogonal direction is taken to follow the desired path.

Finding this direction, however, can be costly. Notice that $H_{t,y}(Q,\gamma)$ is of size $L(L+1)$. This number is quadratic in the training set size, and full computation of the matrix is impractical even for small data sets. Instead, we resort to approximating $H_{t,y}(Q,\gamma)$ by a matrix that contains only the diagonal entries $\frac{\partial G_{t,y}(Q,\gamma)}{\partial Q(t|y)}$ and the last column $\frac{\partial G_{t,y}(Q,\gamma)}{\partial \gamma}$. While we cannot bound the extent of this diagonal approximation, we note that the diagonal terms are also the most significant ones and many off diagonal terms are zero. Once we make the approximation, we can solve Eq. (10) in time linear in L. (See Appendix B for a full development of H and the computation of the orthogonal direction.)

Note that once we find a vector Δ that satisfies Eq. (10), we still need to decide on its length, or the size of the step we want to take in that direction. There are various standard approaches, such as normalizing the direction vector to a predetermined size. However, in our problem, we have a natural measure of progress that stems from the tradeoff defined by the target Lagrangian $\mathcal{L}_{EM}$, where $\boldsymbol{I}(T;Y)$ increases when T captures more and more information about the samples during the annealing procedure. That is, the "interesting" steps in the learning process occur when $\boldsymbol{I}(T;Y)$ grows. These are exactly the points where the balance between the two terms in the Lagrangian changes and the second term grows sufficiently to allow the first term to increase $\boldsymbol{I}(T;Y)$. Using $\boldsymbol{I}(T;Y)$ to gauge the progress of the annealing procedure is appealing since it is a non-parametric measure that does not involve the form of the particular distribution of interest P. In addition, in all runs $\boldsymbol{I}(T;Y)$ starts at 0, and is upper-bounded by the log of the cardinality of T and we are thus given a scale of progress.

With this intuition at hand, we want to normalize the step size by the expected change in $\boldsymbol{I}(T;Y)$. That is, we calibrate our progress with respect to the *actual* amount of regularization applied at the current value of γ. At regions where $\boldsymbol{I}(T;Y)$ is not sensitive to changes in the parameters, we can proceed rapidly. On the other hand, if small changes in the parameters result in significant changes of $\boldsymbol{I}(T;Y)$, then we want to carefully track the solution. Figure 3 illustrates the difference between using a predetermined step of γ and partitioning $\boldsymbol{I}(T;Y)$ in order to determine the step size. It is evident the using $\boldsymbol{I}(T;Y)$ causes the method to concentrate on the region of interest in terms of rapid change of the Lagrangian.

Formally, we compute $\nabla_{Q,\gamma}\boldsymbol{I}(T;Y)$ and rescale the direction vector so that

$$(\nabla_{Q,\gamma}\boldsymbol{I}_Q(T;Y))' \cdot \Delta = \varepsilon, \tag{11}$$

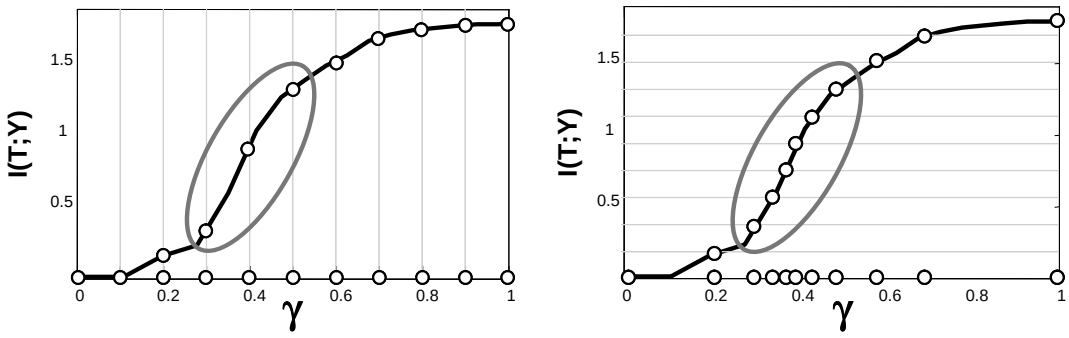


Figure 3: Illustration of the step size calibration process. Both graphs show the change in information between T and Y as a function of γ. The circles denote values of γ to be evaluated. (a) shows naive calibration when fixed steps are taken in the γ range. (b) shows calibration that uses fixed steps in the information range. The grey circle shows the region of dramatic change of the Lagrangian.

where ε is a predetermined step size that is a fraction of $\log|T|$. We also bound the minimal and maximal change in γ so that we do not get trapped in too many steps or alternatively overlook the regions of change.

Finally, although the continuation method takes us in the correct direction, the approximation as well as inherent numerical instability can lead us to a suboptimal path. To cope with this situation, we adopt a commonly used heuristic used in deterministic annealing. At each value of γ, we slightly perturb the current solution and re-iterate the self-consistent equations to converge on a solution. If the perturbation leads to a better value of the Lagrangian, we take it as our current solution.

To summarize, our procedure works as follows: we start with $\gamma = 0$ for which only trivial solutions exists. At each stage we compute the joint direction of γ and $Q(T \mid Y)$ that will leave the fixed point equations intact. We then take a small step in this direction and apply IB-EM iterations to attain the fixed point equilibrium at the new value of γ. We repeat these iterations until we reach $\gamma = 1$.

5. Multiple Hidden Variables

The framework we described in the previous sections can easily accommodate learning networks with multiple hidden variables simply by treating T as a vector of hidden variables. In this case, the distribution $Q(\mathbf{T} \mid Y)$ describes the *joint* distribution of the hidden variables for each value of Y, and $P(\mathbf{T}, \mathbf{X})$ describes their joint distribution with the attributes $\mathbf{X}$ in the desired network. Unfortunately, if the number of variables $\mathbf{T}$ is large, the representation of $Q(\mathbf{T} \mid Y)$ grows exponentially, and this approach becomes infeasible.

One strategy to alleviate this problem is to force $Q(\mathbf{T} \mid Y)$ to have a factorized form. This reduces the cost of representing Q and also the cost of performing inference. As an example, we can require that $Q(\mathbf{T} \mid Y)$ is factored as a product $\prod_i Q(T_i \mid Y)$. This assumption is similar to the *mean field variational approximation* (*e.g.*, Jordan et al. (1998)).

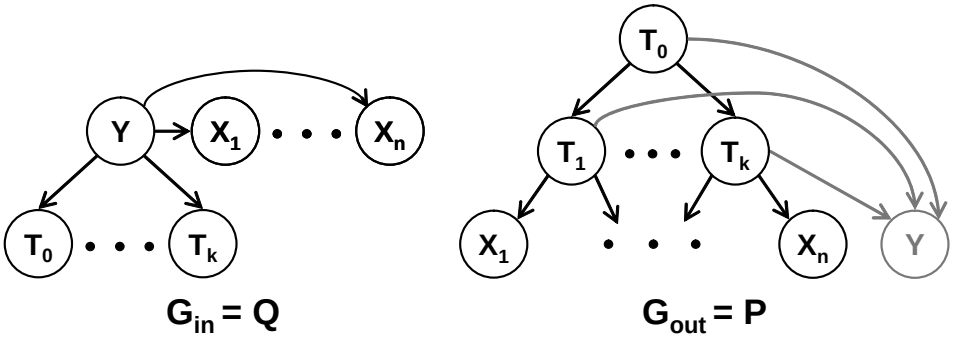


Figure 4: Definition of networks for the multivariate information bottleneck framework with multiple hidden variables. Shown are $\mathcal{G}_{in}$ with the *mean field* assumption, and a possible choice for $\mathcal{G}_{out}$.

In the multivariate information bottleneck framework, different factorizations of $Q(\mathbf{T} \mid Y)$ correspond to different choices of networks $\mathcal{G}_{in}$. For example, the mean field factorization is achieved when $\mathcal{G}_{in}$ is such that the only parent of each T_i is Y, as in Figure 4. In general, we can consider other choices where we introduce edges between the different T_i's. For any such choice of $\mathcal{G}_{in}$, we get exactly the same Lagrangian as in the case of a single hidden variable. The main difference is that since Q has a factorized form, we can decompose $\boldsymbol{I}_Q(\mathbf{T};Y)$. For example, if we use the mean field factorization, we get

$$\boldsymbol{I}_Q(\mathbf{T};Y) = \sum_i \boldsymbol{I}_Q(T_i;Y).$$

Similarly, we can decompose $\boldsymbol{E}_Q[\log P(\mathbf{X},\mathbf{T})]$ into a sum of terms, one for each family in P. These two factorization can lead to tractable computation of the first two terms of the Lagrangian as written in Proposition 1. Unfortunately, the last term $\boldsymbol{E}_Q[\log Q(\mathbf{T})]$ cannot be evaluated efficiently. Thus, we approximate this term as $\sum_i \boldsymbol{E}_Q[\log Q(T_i)]$. For the mean field factorization, the resulting Lagrangian (with this lower bound approximation) has the form

$$\mathcal{L}^{+}_{EM} = \sum_i \boldsymbol{I}_Q(T_i;Y) - \gamma \left(\boldsymbol{E}_Q[\log P(\mathbf{X},T)] - \sum_i \boldsymbol{E}_Q[\log Q(T_i)] \right). \tag{12}$$

The form of $\mathcal{L}^{+}_{EM}$ is valid, if Proposition 1 still holds for the case of multiple hidden variables. This is immediate if we make the following requirements, similar to those made for the case of a single hidden variable:

1. Y is the instance identity;

2. $\mathcal{G}_{in}$ is a Bayesian network structure such that all of the variables $\mathbf{T}$ are independent of X given Y; and

3. $\mathcal{G}_{out}$ is a Bayesian network structure such that Y is a child of $\mathbf{T}$ and has no other parents. This implies that in $\mathcal{G}_{out}$, Y is independent of all $\mathbf{X}$ given $\mathbf{T}$.

The last requirement is needed so that we can set $P(Y \mid T) = Q(Y \mid T)$ in the proof of Proposition 1. As in the case of a single hidden variable, we can now characterize fixed point equations that hold in stationary points of the Lagrangian.

Proposition 6 *Let $\mathcal{L}^+_{EM}$ be defined via $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$ as in Eq. (12). Assuming a* mean field *approximation for $Q(\mathbf{T} \mid Y)$, a (local) maximum of $\mathcal{L}^+_{EM}$ is achieved by iteratively solving, independently for each hidden variable i, the self-consistent equations*

$$Q(t_i \mid y) \quad = \quad \frac{1}{Z(i,y,\gamma)} Q(t_i)^{1-\gamma} \exp\left\{\gamma \mathbf{EP}(t_i,y)\right\},$$

where

$$\mathbf{EP}(t_i,y) \equiv \boldsymbol{E}_{Q(\mathbf{T}|t_i,y)}[\log P(\mathbf{x}[y],\mathbf{T})]$$

and $Z(i,y,\gamma)$ is a normalizing constant that equals to

$$Z(i,y,\gamma) = \sum_{t'_i} Q(t'_i)^{1-\gamma} \exp\left\{\gamma \mathbf{EP}\left(t'_i,y\right)\right\}.$$

See Appendix A for the proof.

The only difference from the case of a single hidden variables is in the form of the expectation $\mathbf{EP}(t_i,y)$. It is easy to see that when a single hidden variable is considered, and $\mathbf{EP}(t_i,y) \equiv \log P(\mathbf{x}[y],t)$, the two forms coincide. It is also easy to see that this term decomposes into a sum of expectations, one for each factor in the factorization of P. We note that only terms that average over factors that involve T_i are of interest in $\mathbf{EP}(t_i,y)$. All other terms do not depend on the value of T_i, and can be absorbed by the normalizing constant. Thus, $\mathbf{EP}(t_i,y)$ can still be computed efficiently.

A more interesting consequence (see theorem below) of this discussion is that when $\gamma = 1$, maximizing $\mathcal{L}^+_{EM}$ is equivalent to performing *mean field EM* (Jordan et al., 1998). Thus, by using the modified Lagrangian we generalize this variational learning principle, and as we show below manage to reach better solutions.

The formulation is easily extensible to a general variational approximation of Q where $\mathcal{G}_{in}$ allows, in addition to the dependence of each T_i on Y, dependencies between the different T_i's. In this case, we get

$$\boldsymbol{I}_Q(\mathbf{T};Y) = \sum_i \boldsymbol{I}_Q(T_i;\mathbf{Pa}_i^{\mathcal{G}_{in}}).$$

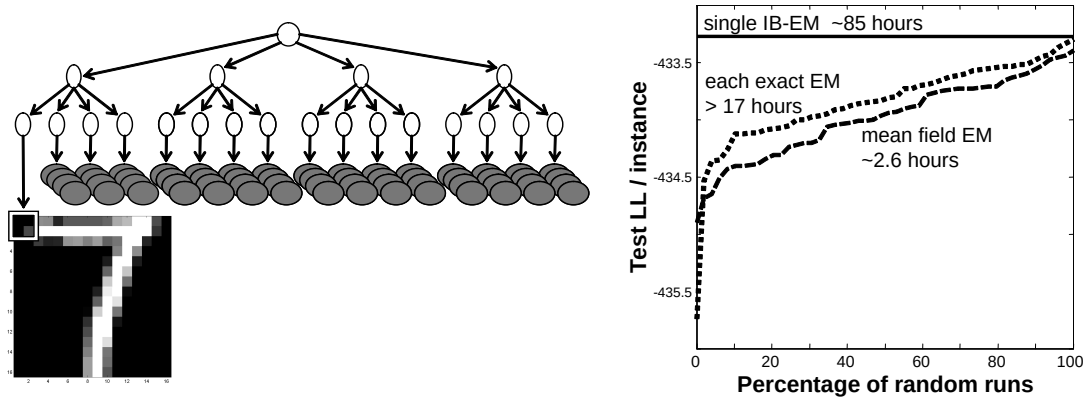
Similarly, $\boldsymbol{E}_Q[\log P(\mathbf{X},T)]$ decomposes according to the *joint* families of T_i in P and in Q. That is, each term in the decomposition depends on T_i, its parents $\mathbf{Pa}_i^{\mathcal{G}_{in}}$ in $\mathcal{G}_{in}$, and its parents $\mathbf{Pa}_i^{\mathcal{G}_{out}}$ in $\mathcal{G}_{out}$. As in the case of the mean field variational approximation, the last term $\boldsymbol{E}_Q[\log Q(\mathbf{T})]$ cannot be evaluated efficiently. We approximate it using a decomposition that follows the structure of $\mathcal{G}_{in}$ as

$$\boldsymbol{E}_Q[\log Q(\mathbf{T})] \approx \sum_i \boldsymbol{E}_Q\left[\log Q(T_i \mid \mathbf{T} \cap \mathbf{Pa}_i^{\mathcal{G}_{in}})\right]. \tag{13}$$

We can now reformulate the results of Theorem 3 for this general case:

Theorem 7 *Let $Q(\mathbf{T} \mid Y)$ decompose according to any structure $\mathcal{G}_{in}$ where all T_i's are descendents of Y and replace $\boldsymbol{E}_Q[\log Q(\mathbf{T})]$ by a decomposition as defined in Eq. (13). Then for the resulting Lagrangian*

$$\mathcal{L}^+_{EM} = (1-\gamma) \sum_i \boldsymbol{I}_Q(T_i;\mathbf{Pa}_i^{\mathcal{G}_{in}}) - \gamma \mathcal{F}^+[Q,P],$$



(a) (b)

Figure 5: (a) A quadrant based hierarchy structure with 21 hidden variables for modeling 16×16 images in the Digit domain. (b) Test log-loss of the **IB-EM** algorithm for the model of (a) compared to the cumulative performance of 50 random EM and mean field EM runs.

where $\mathcal{F}^+[Q,P]$ is defined as in Eq. (5), except that the above decomposition for both $\boldsymbol{E}_Q[\log P(\mathbf{X},T)]$ and $\boldsymbol{H}_Q(T \mid Y)$ is used.

Proof: This is a direct result of the fact that in the proof of Theorem 3, no assumptions were made of the form of Q. ∎

The above theorem extends the formal relation of the information bottleneck target Lagrangian and the EM functional for any form of variational approximation encoded by $\mathcal{G}_{in}$. In particular, when $\gamma = 1$, finding a local minimum of $\mathcal{L}^+_{EM}$ is equivalent to finding a local maximum of the likelihood function when the same variational approximation is used in the EM algorithm. Similarly, we can derive the fixed point equations with each for different choices of $\mathcal{G}_{in}$. The change to Proposition 6 is simply a different decomposition for $\mathbf{EP}(i,y)$

To summarize, the IB-EM algorithm of Section 3.2 can be easily generalized to handle multiple hidden variables by simply altering the form of $\mathbf{EP}(t_i,y)$ in the fixed point equations. All other details, such as the continuation method, remain unchanged.

6. Experimental Validation: Parameter Learning

To evaluate the IB-EM method for the task of parameter learning, we examine its generalization performance on several types of models on three real-life data sets. In each architecture, we consider networks with hidden variables of different cardinality, where for now we use the same cardinality for all hidden variables in the same network. We now briefly describe the data sets and the model architectures we use.

- The Stock data set records up/same/down daily changes of 20 major US technology stocks over a period of several years (Boyen et al., 1999). The training set includes 1213 samples and the test set includes 303 instances. We trained a Naive Bayes hidden variable model where the hidden variable is a parent of all the observations.

- The Digits data set contains 7291 training instances and 2007 test instances from the USPS (US Postal Service) data set of handwritten digits (see http://www.kernel-machines.org/data.html). An image is represented by 256 variables, each denoting the gray level of one pixel in a 16×16 matrix. We discretized pixel values into 10 equal bins.

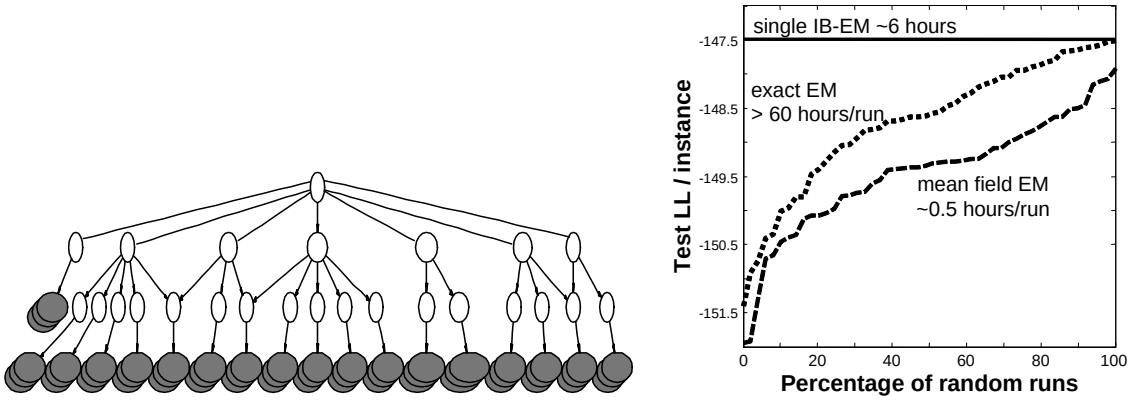
  On this data set we tried several network architectures. The first is a Naive Bayes model with a single hidden variable. In addition, we examined more complex hierarchical models. In these models we introduce a hidden parent to each quadrant of the image recursively. The 3-level hierarchy has a hidden parent to each 8x8 quadrant, and then another hidden variable that is the parent of these four hidden variables. The 4-level hierarchy starts with 4x4 pixel blocks each with a hidden parent. Every 4 of these are joined into an 8x8 quadrant by another level of hidden variables, totaling 21 hidden variables, as illustrated in Figure 5(a).

- The Yeast data set contains measurements of the expression of the Baker's yeast genes in 173 experiments (Gasch et al., 2000). These experiments measure the yeast response to changes in its environmental conditions. For each experiment the expression of 6152 genes were measured. We discretized the expression levels of genes into ranges down/same/up by using a threshold of one standard deviation from above and below the gene's mean expression across all experiments. In this data set, we treat each gene as an instance that is described by its behavior in the different experiments. We randomly partitioned the data into 4922 training instances (genes) and 1230 test instances.

  The model we use for this data set has an hierarchical structure with 19 hidden variables in a 4-level hierarchy that was determined by the biological expert based on the nature of the different experiments, as illustrated schematically in Figure 6. In this structure, 5–24 similar conditions (filled nodes) such as different hypo-osmotic shocks are children of a common hidden parent (unfilled nodes). These hidden parents are in their turn children of further abstraction of conditions. For example, the heat shock and heat shock with oxidative stress hidden nodes, are both children of a common more abstract heat node. A root hidden variable is the common parents of these high-level abstractions. Intuitively, each hidden variable encodes how the specific instance (a gene) is altered in the relevant groups of conditions.

As a first sanity check, for each model (and each cardinality of hidden variables) we performed 50 runs of EM with random starting points. The parameter sets learned in these different runs have a wide range of likelihoods both on the training set and the test set. These results (on which we elaborate below), indicate that these learning problems are challenging in the sense that EM runs can be trapped in markedly different local maxima.

Next, we considered the application of IB-EM on these problems. We performed a single IB-EM run on each problem and compared it to the 50 random EM runs, and also to 50 random mean field EM runs. For example, Figure 5 compares the test set performance (log-likelihood per instance) of these runs on the Digit data set with a 4-level hierarchy of 21 hidden variables with 2 states each. The solid line shows the performance of the IB-EM solution at $\gamma = 1$. The two dotted lines show the cumulative performance of the random runs. As we can see, the IB-EM model is superior to all the mean field EM runs, as well as all of the exact EM runs. Figure 6 shows the result for the biological expert constructed hierarchy of Yeast data set with binary variables. As can be seen, in this harder domain, the superiority of the exact EM runs over mean field EM runs is more evident. Yet, the IB-EM run which also use the mean field approximation, is still able to surpass all of the 50 random exact EM runs.



(a) (b)

Figure 6: (a) A structure constructed by the biological expert for the Yeast data set based on properties of different experiments. 5-24 similar conditions (filled nodes) are aggregated by a common hidden parent (unfilled nodes). These hidden nodes are themselves children of further abstraction nodes of similar experiments, which in their turn are children of the single root node. (b) Comparison of test performance when learning the parameters of the structure of (a) with binary variables. Shown is test log-likelihood per instance of the **IB-EM** algorithm and the cumulative performance of 50 random EM as well as 50 random mean field EM runs.

It is important to note the time required by these runs, all on a Pentium IV 2.4 GHz machine. For the Digit data set, a single mean field EM run requires approximately 2.5 hours, an exact EM run requires roughly 17 hours, and the single IB-EM run requires just over 85 hours. As the IB-EM run reaches a solution that is better than all of this runs, it offers an appealing performance to time tradeoff. This is even more evident for the Yeast data set where the structure is somewhat more complex and the difference between exact learning and the mean field approximation is greater. For this data set, the single IB-EM is still superior and takes significantly less time than a single exact EM.

Figure 7 compares the test log-likelihood per instance performance of our IB-EM algorithms and 50 random EM runs for a range of models for the Stock, Digit and Yeast data sets. In most cases, IB-EM is better than 80% of the EM runs and is often as good or better than the best of them. The advantage of IB-EM is particularly pronounced for the more complex models with higher cardinalities. Table 1 provides more details of these runs including train performance and comparison to 50 random mean field EM runs.

We also compared the IB-EM method to the perturbation method of Elidan et al. (2002). Briefly, their method alters the landscape of the likelihood by perturbing the relative weight of the samples and progressively diminishing this perturbation as a factor of the temperature parameter. In the **Stock** data set, the perturbation method initialized with a starting temperature of 4 and cooling factor of 0.95, had performance similar to that of IB-EM. However, the running time of the perturbation method was an order of magnitude longer. For the other data sets we considered above, running the perturbation method with the same parameters proved to be impractical. When we used more

Model	Train Log-Likelihood							Test Log-Likelihood						
	IB-EM	**Random EM**			**Mean Field EM**			**IB-EM**	**Random EM**			**Mean Field EM**		
		%<	100%	80%	%<	100%	80%		%<	100%	80%	%<	100%	80%
Stock														
C=3	-19.91	62%	-19.90	-19.90				-19.90	76%	-19.88	-19.89			
C=4	-19.47	98%	-19.46	-19.52				-19.52	96%	-19.52	-19.62			
C=5	-19.16	94%	-19.15	-19.24				-19.31	98%	-19.30	-19.39			
Digit														
C=5	-429.95	36%	-428.67	-429.11				-439.91	56%	-439.03	-439.47			
C=10	-411.44	100%	-411.72	-413.96				-425.33	100%	-425.36	-427.05			
DigH3														
C=2	-442.02	100%	-442.02	-442.29	100%	-442.03	-442.20	-450.812	92%	-450.76	-450.92	82%	-450.76	-450.84
C=3	-428.77	100%	-428.85	-429.02	100%	-428.83	-429.02	-437.798	98%	-437.74	-438.20	98%	-437.74	-438.04
DigH4														
C=2	-425.43	100%	-425.54	-425.81	100%	-425.61	-425.94	-433.279	100%	-433.30	-433.55	100%	-433.40	-433.71
C=3	-407.60	100%	-407.75	-408.56	100%	-408.49	-408.83	-415.798	100%	-415.88	-416.48	100%	-416.37	-416.77
Yeast														
C=2	-148.13	100%	-148.32	-148.79	100%	-148.89	-149.71	-147.48	100%	-147.51	-147.87	100%	-147.92	-148.78
C=3	-139.44	100%	-139.58	-140.05	100%	-140.09	-140.87	-138.38	100%	-138.57	-139.00	100%	-139.06	-139.92
C=4	-136.36	100%	-136.72	-136.97	100%	-137.72	-138.28	-135.65	100%	-135.96	-136.16	100%	-136.92	-137.34

Table 1: Comparison of the IB-EM algorithm, 50 runs of EM with random starting points, and 50 runs of mean field EM from the same random starting points. Shown are train and test log-likelihood per instance for the best and 80th percentile of the random runs. Also shown is the percentile of the runs that are worse than the IB-EM results. Data sets shown include a Naive Bayes model for the Stock data set and the Digit data set; a 3 and 4 level hierarchical model for the Digit data set (DigH3 and DigH4); and an hierarchical model for the Yeast data set. For each model we show several cardinalities for the hidden variables, shown in the first column.

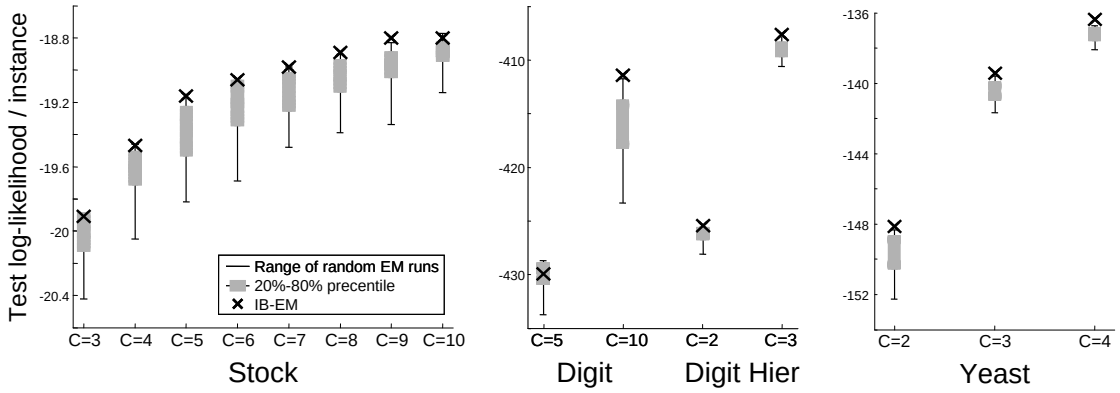


Figure 7: Comparison of log-likelihood per instance test performance of the IB-EM algorithm (black 'X') and 50 runs of EM with random starting points. The vertical line shows the range of the random runs and boxes mark the 20%-80% range. Data sets shown (x-axis) include a Naive Bayes model for the Stock data set and the Digit data set; a 4 level hierarchical model for the Digit data set (Digit Hier); a hierarchical model for the Yeast data set. For each model we show several cardinalities for the hidden variables, shown in the x-axis.

efficient parameter settings, the perturbation method's performance was significantly inferior to that of IB-EM. These results do not contradict those of Elidan et al. (2002) who showed some improvement for the case of parameter learning but mainly focused on structure learning, with and without hidden variables.

To demonstrate the effectiveness of the continuation method we examine **IB-EM** during the progress of γ. Figure 8 illustrates the progression of the algorithm on the Stock data set. (a) shows training log-likelihood per instance of parameters in intermediate points in the process. This panel also shows the values of γ evaluated during the continuation process (circles). These were evaluated using the predicted change in $\boldsymbol{I}(T;Y)$ shown in (b). As we can see, the continuation procedure focuses on the region where there are significant changes in $\boldsymbol{I}(T;Y)$ approximately corresponding the areas of significant changes in the likelihood. For both the Stock and Digit data sets, we also tried changing γ naively from 0 to 1 as in standard annealing procedures, without performing continuation. This procedure often "missed" the superior local maxima even when a large number (1000) of γ values were used in the process. In fact, in most runs the results were no better than the average random EM run emphasizing the importance of the continuation in the annealing process.

7. Learning Structure

Up until now, we were only interested in parameter learning. However, in real life it is often not the case that the structure is given. A structure that is too simple will not be able to faithfully capture the distribution, while an overly complex structure will deteriorate our ability to learn. In this section we consider the case where the set of hidden variables is fixed and their cardinalities are known, and we want to learn the network structure. Clearly, this problem is harder than simple

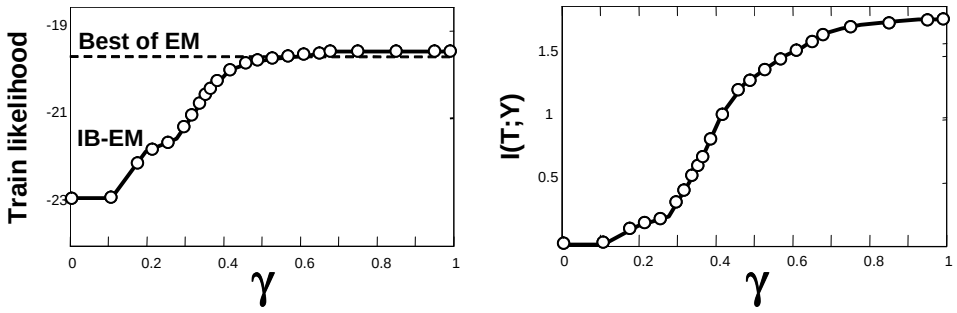


Figure 8: The continuation process for a Naive Bayes model on the Stock data set. (a) Shows the progress of training likelihood as a function of γ compared to the best of 50 EM random runs. Black circles illustrate the progress of the continuation procedure by denoting the value of γ at the end of each continuation step. Calibration is done using information between the hidden variable T and the instance identity Y shown in (b) as a function of γ.

parameter learning, which is just one of the tasks we have to perform in this scenario. The common approach to this model selection task is to use a *score-based approach* where we search for a structure that maximizes some score. Common scores such as the BDe score (Heckerman et al., 1995) balance the likelihood achieved by the model and its complexity. Thus, model selection is achieved independently of the search procedure used (see Section 2.1 for more details).

We now aim to extend the **IB-EM** framework for the task of structure learning using a score-based approach. Naively, we could simply consider different structures and for each one apply the IB-EM procedure to estimate parameters, and then evaluate its generalization ability using the score. Such an approach is extremely inefficient, since it spends a non-trivial amount of time to evaluate each potential candidate structure. In this work we advocate a strategy that is based on the structural EM framework of Friedman (1997). In structural EM, we use the completion distribution Q that is a result of the E-Step to compute *expected sufficient statistics*. That is, instead of Eq. (1), we use

$$\boldsymbol{E}_{Q(\mathbf{T}|Y)}[N(x_i,\mathbf{pa}_i)] = \sum_m \sum_{\mathbf{t}} Q(\mathcal{X}[m] = x_i, \mathbf{Pa}_i = \mathbf{pa}_i, \mathbf{t} \mid Y = m).$$

These statistics are then used in the *M-step* when structure modification steps are evaluated. Thus, instead of assuming that the target structure $\mathcal{G}_{out}$ is fixed, we define the Lagrangian as a function of the pair $(\mathcal{G}_{out}, \theta)$. Then, in the M-step, we can consider different choices of $\mathcal{G}_{out}$ and evaluate how each of them changes the score. Given the expected statistics, the problem is identical in form to learning from a fully observed data set and computation of the score is similar. This facilitates an efficient greedy search procedure that uses local edge modification to the network structure. The EM procedure of Section 3.2 is thus revised as follows:

- **E-step** : Maximize $-\mathcal{L}_{EM}$ by varying $Q(T \mid Y)$ while holding P fixed.
- **M-step**: While holding Q fixed:

- Search for the structure $\mathcal{G}_{out}$ of P that maximizes $\text{Score}_{\text{BDe}}(\mathcal{G} : \mathcal{D})$, using the sufficient statistics of Q.
- Maximize $-\mathcal{L}_{EM}$ by varying the parameters of P using the structure $\mathcal{G}_{out}$ selected.

In practice, since the BDe score is not a linear function of the sufficient statistics, we approximate it in the **M-step** using the Cheeseman-Stutz (Cheeseman et al., 1988) approximation. It is important to note the distinction between the optimization of the Lagrangian and that of the score. Specifically, optimizing the Lagrangian involves maximization of the likelihood along with an information theoretic regularization term that does not depend on P. On the other hand, optimization of the structure is performed using the BDe model selection score. This is mathematically valid since each optimization step is ignorant of the inner mechanics of the other step. However, one might wonder why the use of a score is needed at all if regularization is already present in the form of the information theoretic term in the Lagrangian. It is easy to understand the reason for this if we look at the final stage of learning when $\gamma = 1$. At this point, as we have shown, optimizing the Lagrangian is equivalent to optimizing the EM objective. Using the same objective to adapt structure will result in dense structures. In particular, it will be beneficial to add an edge between any two variables that are not perfectly independent in the training data. Thus, while the regularization encoded in the Lagrangian is needed to smooth the parametric EM problem, a model selection regularization via a score is also needed to constrain the network structure.

Using the structural EM framework allows us to apply our framework to structure learning and to use various search operators as simple plug-ins. For general Bayesian networks, for example, one can consider the standard add, delete and reverse edge operators. The only requirement in this case is that a hidden variable is constrained to be non-leaf, in which case it becomes redundant and can be marginalized out. In addition, as in the case of learning parameters, we are still guaranteed to converge for a given value of γ. However, as in parametric EM, convergence is typically to a local maximum. In fact, the problem now has two facets: First, local maxima that result from evaluation of Q in the E-step. Second, local maxima in the discrete structure search space due to the greedy nature of the search algorithm.

Although the method described above applies for any Bayesian network structure, for concreteness we focus on learning *hierarchies* of hidden variables in the following sections. In this sub-class of networks each variable has at most one parent, and that parent has to be a hidden variable. This implies that the hierarchy of hidden variables captures the dependencies between the observed attributes. Since we are dealing with hierarchies we consider search steps that replace the parent of a variable by one of the hidden variables. Such moves preserve the overall hierarchy structure, repositioning a single observed variable, or a sub-hierarchy. We apply these steps in a greedy manner, from the one that leads to the largest improvement, as long as the resulting hierarchy is acyclic.

8. Learning Cardinality

In real life, it is often the case that we do not know the cardinality of a hidden variable. In a clustering application, for example, we typically do not know of a beneficial number of clusters and need to either use some arbitrary choice or spend time evaluating several possibilities. Naively, we might try to set a high cardinality so that we can capture all potential clusters. However, this approach can lead to bad generalization performance due to over-representation. The discussion in Section 4 on the behavior of the model as a function of γ provides insight on the effect of cardinality

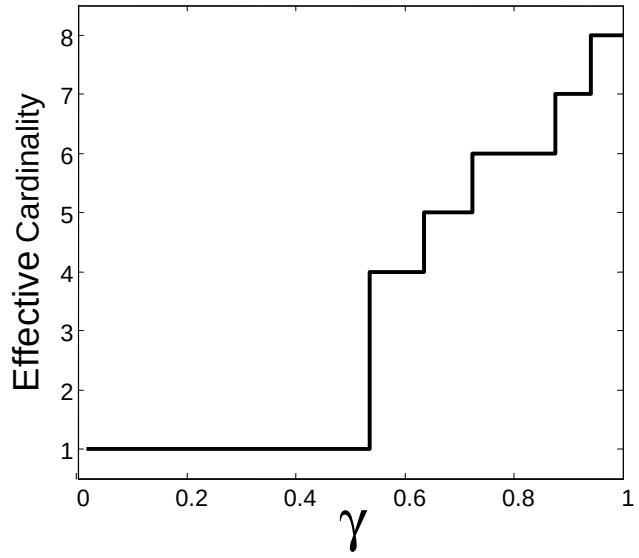


Figure 9: Effective cardinality as a function of γ during the learning process for the Stock data set using a Naive Bayes model. Cardinality is evaluated using local decomposition of the BDe score.

selection. When examining the models during the continuation process, we observe that for lower values of γ the *effective* cardinality of the hidden variable is smaller than its cardinality in the model (we elaborate on how this is measured below). Figure 9 shows an example of this phenomenon for the Naive Bayes model of the Stock data set. Thus, limiting the cardinality of the hidden variable is in effect similar to stopping the continuation process at some $\gamma < 1$. This is, by definition, equivalent to using a regularized version of the EM objective, which can avoid overfitting.

The most straightforward approach to learning the cardinality of a hidden variable is simply to try a few values, and for each value apply IB-EM independently. We can then compare the value of the EM objective (at $\gamma = 1$) corresponding to the different cardinalities. However, models with higher cardinality will achieve a higher likelihood and will thus always be chosen as preferable by the Lagrangian, at the risk of overfitting the training data. In the previous section we discussed the use of a model selection score as a measure for preferring one network structure over another. The same score can also be readily applied for this scenario of cardinality selection. Whether the complexity is a result of a dense structure or an increased number of parameters due to a high cardinality of a variable, all common scores balance the likelihood with the model complexity, either explicitly as in the case of the MDL score (Lam and Bacchus, 1994) or implicitly as in the case of the Bayesian (BDe) score (Heckerman et al., 1995). Thus, similarly to structure learning, we use the Lagrangian when estimating parameters and turn to the score when performing the black-box model selection step. One problem with this simple approach is that it can be extremely time consuming. If we want to try K different cardinalities for each hidden variable, we have to carry out $|H|^K$ independent IB-EM runs, where $|H|$ is the number of hidden variables.

The intuition that the "effective" cardinality of the hidden variable will increase as we consider larger values of γ suggests that we increasing the model complexity during the continuation process.

A simple method is as follows. At each stage allow the model an extra, seemingly redundant, state for the hidden variable. As soon as this state is utilized, we increase the cardinality by adding a new "spare" state. The annealing process, by nature, automatically utilizes this new state when it is beneficial to do so. The task we face is to determine when all the states of a hidden variables are being utilized and therefore a new redundant state is needed. Intuitively, a state of a variable is being used if it captures a distinct behavior that is not captured by other states. That is, for any state i, no other state j is similar.

To determine whether state i is different than all other states, we start by evaluating the cost that we incur due to the merging of state i with another state j. We denote by $\widehat{ij}$ a new state that combines both i and j and alter Q so that

$$Q(T = \widehat{ij} \mid Y = y) = Q(T = i \mid Y = y) + Q(T = j \mid Y = y). \tag{14}$$

We then use this to reestimate the parameters of P in the M-step, and examine the resulting change to the Lagrangian. As shown in Slonim et al. (2002), the difference in the Lagrangian before and after the merge is a sum of Jensen-Shannon divergence terms that measure the difference between the conditional distribution of each child variable given the two states of the hidden variable. This is in fact the change in likelihood of the model resulting from merging the states and can be computed efficiently.

Now that we have the change in the Lagrangian due to the merging of state i with state j, we have to determine whether this change is significant. As already noted, using more states will always improve the likelihood so that the difference in the Lagrangian is not sufficient for model selection. Instead, we can use the BDe score to take into account both the improvement to the likelihood and the change in the model complexity as in Elidan and Friedman (2001). One appealing property of the BDe score is that it is *locally decomposable*. That is, Eq. (2) decomposes according to the different values of each variables. Thus, the difference between the BDe score after and before the merge of states i and j is only in the terms where T appears:

$$\begin{aligned}
&\text{Score}_{\text{BDe}}(\mathcal{G}_{\widehat{ij}} : \mathcal{D}) - \text{Score}_{\text{BDe}}(\mathcal{G}_{i,j} : \mathcal{D}) = \\
&\sum_{pa_t} \left[\log \frac{\Gamma(N^+(T{=}i,j,pa_t))}{\Gamma(\alpha(T{=}i,j,pa_t))} - \log \frac{\Gamma(N^+(T{=}i,pa_t))}{\Gamma(\alpha(T{=}i,pa_t))} - \log \frac{\Gamma(N^+(T{=}j,pa_t))}{\Gamma(\alpha(T{=}j,pa_t))} \right] + \\
&\sum_C \sum_{pa_c} \Bigg[\log \frac{\Gamma(\alpha(pa_c,T{=}i,j))}{\Gamma(N^+(pa_c,T{=}i,j))} + \sum_c \log \frac{\Gamma(N^+(c,pa_c,T{=}i,j))}{\Gamma(\alpha(c,pa_c,T{=}i,j))} \\
&\qquad - \log \frac{\Gamma(\alpha(pa_c,T{=}i))}{\Gamma(N^+(pa_c,T{=}i))} - \sum_c \log \frac{\Gamma(N^+(c,pa_c,T{=}i))}{\Gamma(\alpha(c,pa_c,T{=}i))} \\
&\qquad - \log \frac{\Gamma(\alpha(pa_c,T{=}j))}{\Gamma(N^+(pa_c,T{=}j))} - \sum_c \log \frac{\Gamma(N^+(c,pa_c,T{=}j))}{\Gamma(\alpha(c,pa_c,T{=}j))} \Bigg],
\end{aligned}$$

where the first summation correspond to the family of T and its parents, and the second summation is over all $\mathbf{C}$ that are children of T and corresponds to the families of the children of T and their parents. $N^+(x) = N(x) + \alpha(x)$ and correspond to *total* count statistics that include the imaginary prior counts (see Section 2.1). As all the terms are functions of these simple sufficient statistics, the above difference can be computed efficiently. Moreover, as in the case of the likelihood computation, the sufficient statistics needed when merging two states are simply the sum of the statistics needed for scoring the individual states. Thus, we can easily evaluate all pairwise state merges to determine if *any* two states of T are similar.

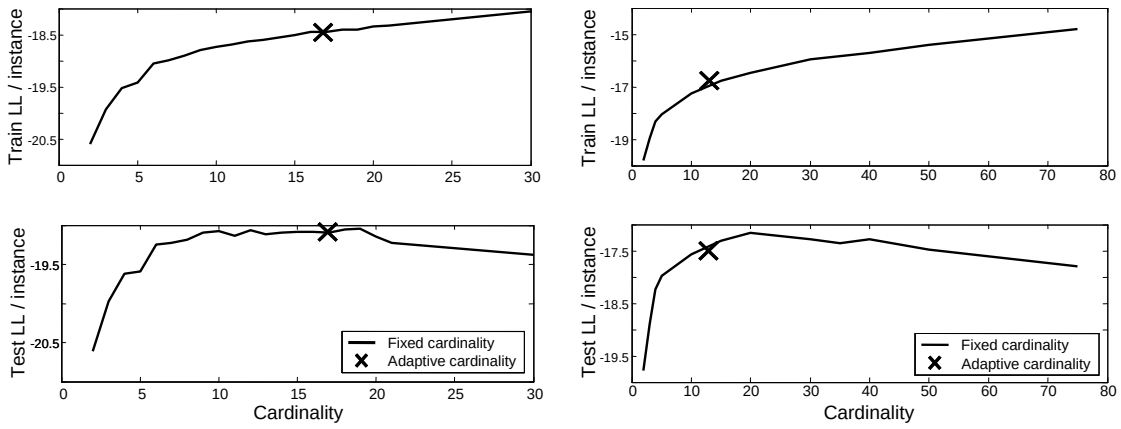


Figure 10: Evaluating adaptive cardinality selection for the Stock and the Yeast data sets with a Naive Bayes model. The 'X' marks the performance of runs with adaptive cardinality selection. The line shows performance of individual runs with a fixed cardinality. The top panel shows training set performance, and the bottom one test set performance.

To summarize, the resulting procedure is as follows. We start with a binary cardinality for the hidden variables at $\gamma = 0$. At each stage, before γ is increased, we determine for each hidden variable if all its states are utilized: For each pair of states we evaluate the BDe score difference between the model with the two states and the model with the states merged. If the difference is positive for all pairs of states then all states are considered utilized and a new state is added. Optimizing the Lagrangian using IB-EM will utilize this new state automatically when it will be beneficial to do so, causing the introduction of a new "spare" state, and so on.

In an early work leading to the formulation of the Information Bottleneck framework, (Pereira et al., 1993) used a similar idea to gauge the effective number of clusters. Briefly, for each cluster a slightly perturbed cluster (twin state) was incorporated in the model allowing each cluster to split into two distinct ones. Similar procedures were used in deterministic annealing (Rose, 1998) and later information bottleneck implementations (Tishby et al., 1999; Slonim et al., 2002). The method we presented in this section differs in two important aspects. First, we use a model selection score to determine when it is beneficial to declare that a redundant cluster is actually being used. This allows us to avoid using an arbitrary distance measure to determine if two clusters diverge. Second, the above allows us to use a single redundant cluster rather than a twin for each state, which significantly reduces the model complexity. While this may not be crucial in standard clustering scenario, it is of great importance for the large models with many hidden variables that we consider in this paper.

9. Experimental Validation: Learning Cardinality

We now want to evaluate the effectiveness of our method for adapting cardinality during the annealing process. For this, we would like to compare the cardinality and model achieved by the method to naive selection of the cardinality. To make this feasible, we look at the context of a Naive Bayes model with a single hidden variable for the Stock and the Yeast data set introduced in Section 6.

We trained the models using the IB-EM algorithm where the hidden variable was assigned a fixed cardinality, and repeated this for different cardinalities. We then applied our adaptive cardinality method to the same model. Figure 10 compares the adaptive cardinality selection run ('X' mark) vs. the fixed cardinality runs for both data sets. As we can see, the adaptive run learns models that generalize nearly as well as the best models learned with fixed cardinality. These results indicate that our method manages to increase cardinality while tracking a high likelihood solution, and that the decision when to add a new state manages to avoid adding spurious states.

A more complex scenario is where, for the Yeast data set, we learn the hierarchy supplied by the biological expert for 62 of the experiments. In this hierarchy there are 6 hidden variables that aggregate similar experiments, a *Heat* node that aggregates 5 of these hidden variables and a root node that is the parent of both *Heat* and the additional *Nitrogen Depletion* node. Figure 11 shows the structure along with the cardinalities of the hidden variables learned by our method and compares the performance of our method to model learned with different fixed cardinalities. As can be seen in (b), the performance of our final model is close to the optimal performance with fixed cardinality. (c) shows that this is achieved with a similar complexity to the simpler of the superior models (at a fixed cardinality of 10).

10. Learning New Hidden Variables

The ideas presented in Section 7 are motivated by the fact that in real life we are typically not given the structure of the Bayesian network. The situation is often even more complex. Hidden variables, as their name implies, are not only unobserved but can also be unknown entities. In this case, we do not even know which variables to include in our model. Thus, we want to determine the number of hidden variables, their cardinality, their relation to the observed variables, and their inter-dependencies. This situation is clearly much more complex than structure learning and might seem hopeless at first. However, as in the case of cardinality adaptation discussed in Section 8, we can use emergent cues of the continuation process to suggest an effective method.

Recall the behavior of our target Lagrangian as a function of γ. For small values of γ, the emphasis of the Lagrangian is on compressing the instance identity, and the hidden variables are (almost) independent of the observed attributes. Thus, at this stage, a simple model would be able to perform just as well as a complex one. In fact, to increase learning robustness, we will want to favor the simpler model and avoid redundant representational complexity. As we increase γ, the hidden variables start capturing properties of the data. In this scenario the need for the more complex structure becomes relevant as it will allow the learning procedure to improve performance.

The above intuition suggests that at small values of γ we start with a simple hierarchy (say, one with only a single hidden variable). When the continuation reaches larger values of γ, the Lagrangian can tolerate more complex structures. Thus, we want to adapt the complexity of the hierarchy as we progress. To do so, we consider a search operator that enriches the structure of hierarchy with a new hidden variable. (This operator is much in the spirit of the "top-down" strategy explored by Adachi and Hasegawa (1996) in learning evolutionary trees.)

Suppose that we want to consider the addition of a new hidden variables into the network structure. For simplicity, consider the scenario shown in Figure 12, where we start with a Naive Bayes network with a hidden variable T_1 as root and want to add a hidden variable T_2 as a parent of a subset $\mathbf{C}$ of T_1's children. Intuitively, we want to select a subset $\mathbf{C}$ that is not "explained well" by T_1 and where we expect to gain a lot by the introduction of T_2. Formally, we evaluate the change in

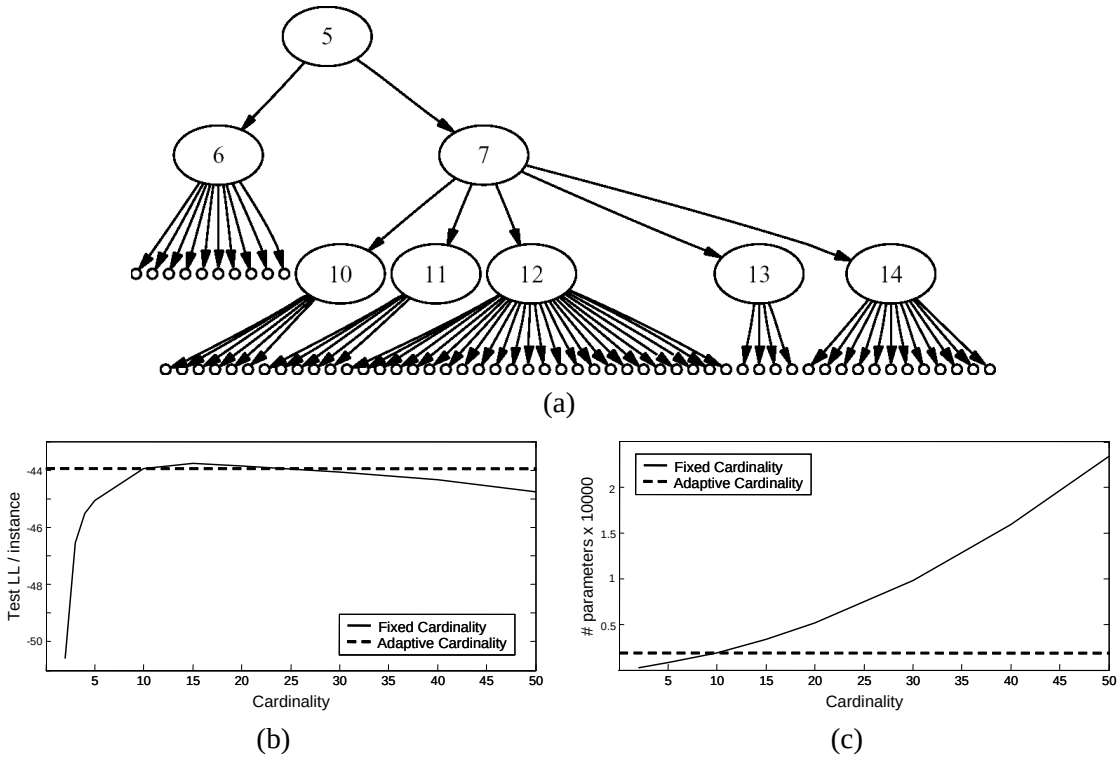


Figure 11: Cardinality learning for the Yeast data set on the structure provided by the biological expert. (a) shows the structure along with the nodes annotated with the cardinality learned by our adaptive approach. (b) shows the test set log-likelihood performance of models learned with different fixed cardinalities (solid line). The horizontal dashed line marks the performance of our adaptive cardinality method. (c) shows plot the number of parameters for each of these models (solid line) with the dashed horizontal line marking the number of parameters of the model learned by our method.

our target Lagrangian as the result of inserting T_2 into the network structure

$$\mathcal{L}_{EM} - \mathcal{L}'_{EM} =$$
$$-\boldsymbol{I}_Q(T_2;Y) + \gamma \boldsymbol{E}_Q[\log P'(T_2 \mid T_1) - \log Q(T_2) + \textstyle\sum_{i \in \mathbf{C}} [\log P'(X_i \mid T_2) - \log P(X_i \mid T_1)]],$$

where P and P' are the models before and after the change to the network, respectively. The term $\log P(X_i \mid T_1)$ can be readily evaluated from the current model for each $X \in \mathbf{C}$ and the terms $\boldsymbol{I}_Q(T_2;Y)$ and $\boldsymbol{E}_Q[\log Q(T_2)]$ can be easily bounded. However, to evaluate $\log P'(T_2 \mid T_1)$ or $\sum_{i \in \mathbf{C}} \log P'(X \mid T_2)$ we need to actually choose $\mathbf{C}$, add T_2 to the current structure and optimize $Q(T_2 \mid Y)$. This can be too costly as the number of possible subsets $\mathbf{C}$ can be large even for a relatively small number of variables. Thus, we want to somehow approximate the above terms efficiently using only the current model. The following bound allows us to do so by bounding the contribution of a hidden variable.

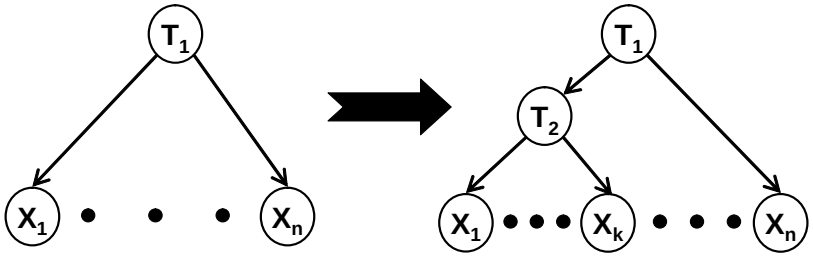


Figure 12: Example of enrichment with new hidden variables T_2 as parent of a subset $\mathbf{C}$ of the observed variables $X_1 \ldots X_n$.

Proposition 8 *Let P be a Bayesian network model with a hidden variable T_1 and denote by $\mathbf{C}$ an observed subset of T_1's children. Let P' be the result of replacing T_1 as a parent of $\mathbf{C}$ by T_2, making T_2 a child of T_1 and optimizing the parameters of the model using the IB-EM algorithm for any value of γ. Then*

$$\boldsymbol{E}_Q[\log Q(\mathbf{C} \mid T_1)] \geq \boldsymbol{E}_Q\left[\sum_{i \in \mathbf{C}} \log P'(X_i \mid T_2) + \log P'(T_2 \mid T_1)\right].$$

Proof: Using the chain rule and positivity of entropy, we can write

$$\begin{aligned}
\boldsymbol{E}_Q[\log Q(\mathbf{C} \mid T_1)] &\equiv -\boldsymbol{H}_Q(\mathbf{C} \mid T_1) \\
&= -\Big[\boldsymbol{H}_Q(\mathbf{C}, T_2 \mid T_1) - \boldsymbol{H}_Q(T_2 \mid \mathbf{C}, T_1)\Big] \\
&\geq -\boldsymbol{H}_Q(\mathbf{C}, T_2 \mid T_1) \\
&= -\Big[\boldsymbol{H}_Q(\mathbf{C} \mid T_2, T_1) + \boldsymbol{H}_Q(T_2 \mid T_1)\Big] \\
&= -\Big[\sum_{i \in \mathbf{C}} \boldsymbol{H}_Q(X_i \mid X_1 \ldots X_{i-1}, T_2, T_1) + \boldsymbol{H}_Q(T_2 \mid T_1)\Big] \\
&\geq -\Big[\sum_{i \in \mathbf{C}} \boldsymbol{H}_Q(X_i \mid T_2) + \boldsymbol{H}_Q(T_2 \mid T_1)\Big] \\
&\equiv \boldsymbol{E}_Q\left[\sum_{i \in \mathbf{C}} \log P'(X_i \mid T_2) + \log P'(T_2 \mid T_1)\right].
\end{aligned}$$

The last inequality result from the fact that entropy conditioned on less variables can only increase. The final equivalence is a result of the construction of the M-Step of IB-EM, where Q is used when in the optimization of the parameters of P'. ∎

The above proposition provides a bound on the extent to which a hidden variable induces correlations in the marginal distribution. The result is intuitive — the contribution of insertion of a new hidden variable cannot exceed the entropy of its children given their current hidden parent. If we use the bound instead of the original term, we get an over-optimistic estimate of the potential profitability of adding a new hidden variable. However, the scenarios we are interested in are those in which the information between the hidden variable and its children is high and the entropy of

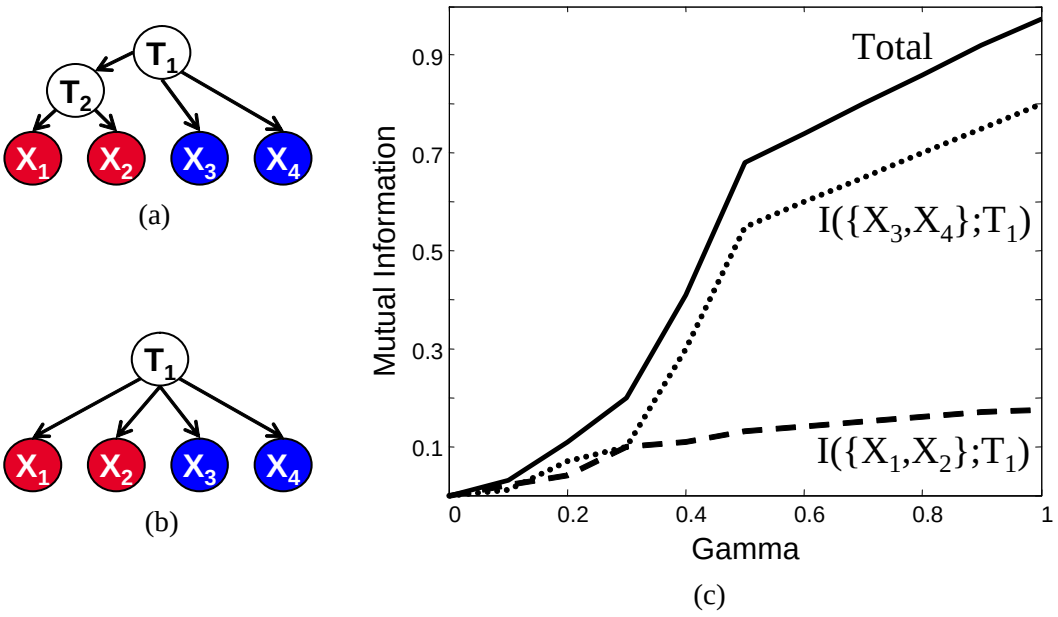


Figure 13: Synthetic example demonstrating the information signal for adding new hidden variables. (a) shows the original structure that generated the samples. (b) shows the structure used in learning without the hidden variable T_2. (c) shows the information as a function of γ between the hidden variables and the observed variables. As learning progresses, the total information rises and the distribution of the direct children of T_1 is captured significantly better (dotted). The information with the original children of T_2 (dashed) remains small.

the hidden variable is low (or there would be no need for it in the network). In such cases, we can expect the bound to be tight in both inequalities.

The above bound provides us with an information signal for putative new hidden variables. In practice, searching for the best subset $\mathbf{C}$ can be impractical even for relatively small networks. Instead, we use the following greedy approach: first, for each hidden variable, we limit our attention to up to K (we use 20) of its children with the highest entropy individually. We then consider *all* three-node subsets of these children whose entropy level passes some threshold (see details in the experiments below). Intuitively, such seeds will capture the core of the signal needed to attract other nodes when structure change is allowed.[2]

Another complication in using the above signal is a consequence of the annealing process itself. For small values of γ we can expect, and indeed we want, Q to smooth out all statistical signals. This will make most subsets appear equally appealing for adding a hidden variable, since T_1 will not be informative about them. In Section 4, we have shown that $\boldsymbol{I}_Q(Y;T)$ is a natural measure for the progress of the continuation process. To demonstrate the phenomenon in the structure learning scenario, Figure 13 shows a simple synthetic experiment where the samples were generated from

[2] In synthetic experiments for different structures where the network size still made computations feasible, these three node seeds always included two or three variables of the optimal larger subset.

the structure shown in (a) and a Naive Bayes model without T_2 was used when learning. (c) shows the information between the hidden variable T_1 and the observed children (solid), its direct children in the generating distribution (dotted) and the children of T_2 (dashed). Up to some point in the annealing process, the information content of the hidden variable is low and the information with both subsets of variables is low. When the hidden variable starts to capture the distribution of the observed variables, the two subsets diverge and while T_1 captures its original direct children significantly better, the children of T_2 still have high entropy given T_1. Thus, we want to start considering our information "cue" only when the hidden parent becomes meaningful, that is only when $\boldsymbol{I}_Q(Y;T_1)$ passes some threshold.

Finally, we note that although the discussion so far assumed that we have a Naive Bayes model and considered the addition of a single new hidden variable, it is easily generalized for any forms of P where in P' we separate a hidden variables in P from its observed children by introducing a new hidden variable.

To summarize, our approach for learning a new hidden variable T (or several such variables) is as follows: At each value of γ, we first evaluate $\boldsymbol{I}_Q(Y;T)$ to determine if it is above the threshold, signifying that the hidden variable is capturing some of the distribution over the rest of the variables. If this is the case, we greedily search for subsets of children of the hidden variable that have high entropy. These are subsets that are not predicted well by their hidden parent. For the subset with the highest entropy, we suggest a putative new hidden variable that is the parent of the variables in the subset. The purpose of this new variable is to improve the prediction of the subset variables, which are not sufficiently explained by the current model. We then continue with the parameter estimation and structure learning procedure as is. If, after structure search, a hidden variable has at most one child, it is in fact redundant and can be removed from the structure. We iterate the entire procedure until no more hidden variable are added and the structure learning procedure converges.

11. Full Learning — Experimental Validation

We want to evaluate the effectiveness of our method when learning structure with and without the introduction of new hidden variables into the model. We examined two real-life data sets: The Stock data set and the Yeast data set (see Section 6). For the Yeast data set we look at a subset of 62 experiments related to heat conditions and Nitrogen depletion.

In Figure 14 we consider average test set performance on the Stock data set. To create a baseline for hierarchy performance, we train a Naive hierarchy with a single hidden variable and cardinality of 3 totaling 122 parameters. We start by evaluating structure learning without the introduction of new hidden variables. To do this, we generated 25 random hierarchies with 5 binary hidden variables that are parents of the observed variables and a common root parent totaling 91 parameters. We then use structural EM (Friedman, 1997) to adapt the structure by using a *replace-parent* operator where at each local step an observed node can replace its hidden parents. As can be seen in Figure 14, standard structure learning applied to the IB-EM framework significantly improves the model's performance. In fact, many of the 25 random runs with the Search operator surpass the performance of the Naive model using fewer parameters.

Next, we evaluate the ability of the new hidden variable enrichment operator to improve the model. We denote by Enrich the IB-EM run with the automatic enrichment operator. We denote by Enrich+Search the run with this operator augmented with structure search operators in the M-steps. As can be seen in Figure 14, the performance of Enrich by itself was not able to compete with the

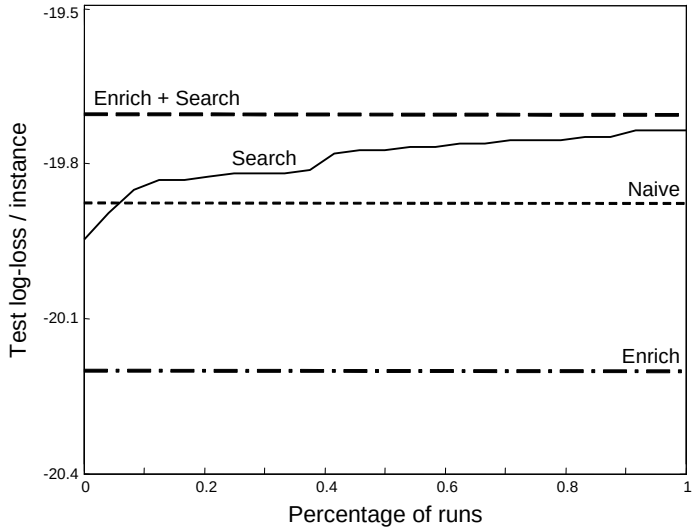
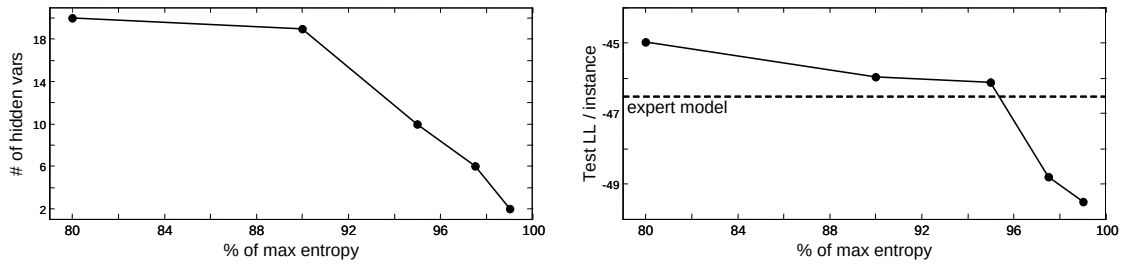


Figure 14: Comparison of performance on the Stock data set of Naive hierarchy (Naive), 25 hierarchies with replace-parent search (Search) , hierarchy learned with enrichment operator (Enrich) and hierarchy learned with enrichment and replace-parent search (Enrich+).

Naive or the Search method. This is not surprising as we cannot expect the information signal to introduce "perfect" hidden variables into the hierarchy. Indeed, when combining the enrichment operator with structure adaptation (Enrich+Search), our method was able to exceed all other runs. The learned hierarchy had only two hidden variables (requiring only 85 parameters). These results show the enrichment operator effectively added useful hidden variables and that the ability to adapt the structure of the network is crucial for utilizing these hidden variables to the best extent.

There are two thresholds used by our algorithm for learning new hidden variables. First, as noted in Section 10, due to the nature of the annealing process we consider adding new hidden variable only when the information $\boldsymbol{I}_Q(Y;T)$ of a hidden variable T in the current structure passes some threshold. In the results presented in this section we use a threshold of 20% of the maximum value the information can reach which is limited by the cardinality of T. Lowering this threshold to as far as 10% or raising it to 40% had negligible effect on the results. We hypothesize that this robustness is caused by the fact that, typically, the cardinality of T will be much lower than Y. Thus, when T undergoes the transition from being redundant to being informative, its information content rises drastically, even if it captures only a small aspect of Y.

The threshold used to limit the number of candidate subsets, however, is more interesting. Recall from Section 10 that the greedy procedure only considers subsets whose entropy passes some threshold. More precisely, we consider only subsets whose entropy passes some percentage of the maximum entropy possible for this subset. Thus, using a lower threshold potentially allows more hidden variables. This is observed empirically in Figure 15(a) for the Yeast data set. A possible concern is that lowering the threshold too much will results in many hidden variables leading to overfitting. However, as is evident in Figure 15(b), even when the number of hidden variables is 20, these new variables are effective in that they improve the generalization performance on unseen test



(a) (b)

Figure 15: Learning new hidden variables for the Yeast data set. (a) shows the number of variables learned as a function of the threshold on the percentage of entropy of a subset used in the greedy procedure. (b) shows the corresponding test set log-likelihood per instance performance and the performance of the model supplied by the biological expert.

data. In fact, with just a few extra variables, our method successfully surpassed the performance of the structure supplied by the biological expert. Obviously, at some point, having too many variables will lead to overfitting. We could not examine this scenario due to the running time required to learn such large networks.

To qualitatively assess the value of our method, we show in Figure 16 the structure learned for the Stock data set with binary variables and the entropy threshold set at 95% (structures at 92.5% and 97.5% were almost identical for this data set). The emergent structure is evident with the "High-tech giants" and "Internet" group dominating the model. The "Varied" group contains "Canon" and "Sony" that manufacture varied technology products such as electronics, photographic, computer peripheral, etc. The "Japanese" relation of "Toyota" to these companies was interestingly stronger than the relation to the "Car" group.

Finally, we applied runs that combine both automatic cardinality adaptation and enrichment of the structure with new hidden variables. Table 2 shows the train and test performance for the Stock data set. Shown are several runs with the Enrich operator and fixed cardinality. For each run, the number of hidden variables added during the learning process (excluding the initial root node) is noted. Also shown is the automatic cardinality method using the BDe score along with the different cardinalities of the 6 hidden variables introduced into the network structure. The combined method was able to surpass the best of the fixed cardinality models in terms of test set performance with fewer than 70% of the parameters. In addition, the fact that the combined method improves test performance but has worse training likelihood, demonstrates its ability to avoid overfitting.

12. Related Work

To define the IB-EM algorithm, we introduced a formal relation between the information bottleneck (IB) target Lagrangian and the EM functional. This allowed us to formulate an information-theoretic regularization for our learning problem. Given this objective, we used two central ideas to make learning feasible. First, following all annealing methods, we slowly diminish the level of "perturbation" as a way to reach a solution of the hard objective. Second, we use continuation to define a stable traversal from an easy problem to our goal problem.

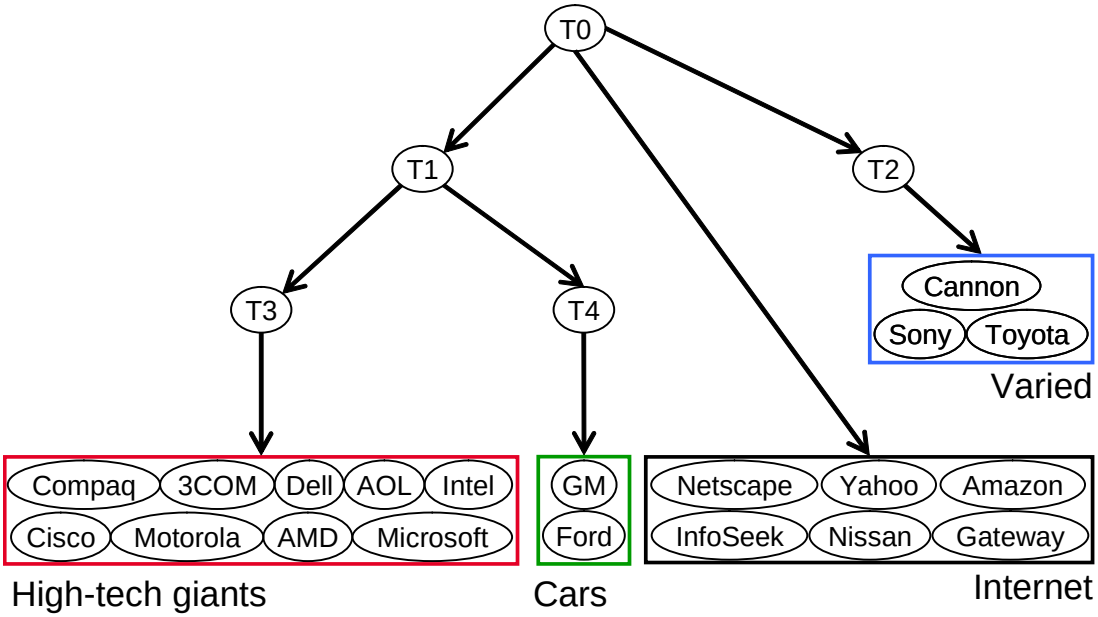


Figure 16: Structure learned for the Stock data set using the enrichment operator augmented with structure search that use the replace-parent operator. All the hidden variables (circles) are binary and the subset entropy threshold was set at 95%. The children of each leaf are annotated with a plausible interpretation.

A multitude of regularization forms are used in machine learning, typically depending on the specific form of the target function (see Bishop (1995) and references within). Information-theoretic regularization has been used for classification with partially labeled data by Szummer and Jaakkola (2002) and for general scenarios in deterministic annealing (Rose, 1998).

Of the annealing methods, the well known *Simulated annealing* (Kirkpatrick et al., 1983) is least similar to ours. Rather than changing the form of the objective function, Simulated annealing allows the search procedure to make "downhill" moves with some diminishing probability. This changes the way the procedure traverses the search space and allows it to potentially reach previously unattainable solutions. Several papers (Heckerman et al., 1994; Chickering, 1996; Elidan et al., 2002) have shown that Simulated annealing is not effective when learning Bayesian networks.

Weight annealing (Elidan et al., 2002), on the other hand, skews the target function directly by perturbing the weights of instances in diminishing magnitudes. Thus, like our method it changes the form of Q directly but does not use an information-theoretic regularization. Weight annealing can actually be applied to a wider variety of problems than our method, including structure search with complete data. However, like other annealing methods, it requires a cooling scheme. For the large problems with hidden variables we explored in this paper, Weight annealing proved inferior with similar running times, and impractical with the settings of Elidan et al. (2002).

Finally, like our method, deterministic annealing (Rose, 1998) alters the problem by explicitly introducing an information-theoretic regularization term. Specifically, following the widely recognized *maximum entropy principle* (Jaynes, 1957), deterministic annealing penalizes the objective

Cardinality	Log-likelihood Train	Test	# of hiddens	# of parameters
2	-19.62	-19.62	5	89
3	-19.32	-19.37	5	146
5	-18.87	-19.04	6	304
10	**-18.53**	**-18.96**	**5**	**769**
20	-18.43	-18.98	5	2340
BDe (9,6,7,7,7,7)	**-18.65**	**-18.94**	**6**	**526**

Table 2: Effect of cardinality when inserting new hidden variables into the network structure with the Enrich operator for the Stock data set. A 95% entropy threshold was used for the hidden variable discovery algorithm. The table shows results for several fixed cardinalities as well as the automatic cardinality method using the BDe score. Shown is the log-likelihood per instance for training as well as test data, the number of hidden variables and the number of parameters in the model. For the automatic method, the cardinalities of each hidden variable is noted.

with a term that is the entropy of the model. A concrete application of deterministic annealing to graphical models was suggested by Ueda and Nakano (1998). However, when learning graphical models, the deterministic annealing was not found to be superior to standard EM (e.g., (Smith and Eisner, 2004)).[3] In particular, Whiley and Titterington (2002); Smith and Eisner (2004)) show why applying deterministic annealing to standard unsupervised learning of Bayesian networks with hidden variables is problematic. One possible explanation for why our method works well for these methods is the difference in motivation of the regularization term. Specifically, our term was motivated by the need for generalization where one want to compress the identity of specific instances. Another important difference between the two methods is that, like Weight annealing, deterministic annealing requires the specification of a cooling policy which makes it potentially impractical for large generative problems. This problem may be avoided using a method similar to the one we used in this work. We leave this prospect as well as the challenge of better understanding the relation between the entropy and information regularization terms for future study.

Continuation methods are a well developed field in mathematics (Watson, 2000). While these methods are used extensively and successfully to solve practical engineering challenges such as complex polynomial systems, they have not been frequently used in machine learning. Recently, Corduneanu and Jaakkola (2002) used continuation to determine a beneficial balance between labeled and unlabeled data. To our knowledge this is the first work in learning graphical models to use continuation to traverse from an easy solution to the desired maximum likelihood problem.

A complementary aspect of our work is the introduction of modification operators for hidden variables. Our method both for learning the cardinality of a hidden variable, and for introducing new hidden variables into the network structure, relies on the annealing process and utilizes emergent signals. The problem of evaluating the cardinality of a hidden variable in a graphical model

[3] Smith and Eisner (2004) also suggest a variant of the deterministic annealing algorithm that appears to work well but is only applicable in the context of semi-supervised learning or when an initial informed starting point for the EM algorithm is at hand.

was explored in several works (*e.g.*, Chang and Fung (1990); Elidan and Friedman (2001)). The work of Stolcke and Omohundro (1993) for HMMs was the first to use evaluation of pairwise state merges to determine adapt the cardinality. In Elidan and Friedman (2001), we extend their method for general Bayesian networks, and Slonim et al. (2002) used a similar approach within the information bottleneck framework. All of these methods start with a large number of states, and then apply bottom-up agglomeration to merge overlaps in the state space and reduce redundancies. By contrast, our method is able to take an "add-when-needed" approach and state mergers are evaluated not to collapse states but rather to determine if a new one is needed. Several papers also explored methods for introducing new hidden variables into the network structure, either for specific classes of Bayesian networks (*e.g.*, Martin and VanLehn (1995); Spirtes et al. (1993); Zhang (2004)) or for general models using a structural signature approach (Elidan et al., 2001). Our contribution in enriching the structure with new hidden variables is twofold. First, we suggested a natural information signature as a "cue" for the presence of a hidden variable. Unlike the structural signature this signature is flexible and is able to weight the influence of different child nodes. Second, we use the enrichment approach in conjunction with the continuation approach for bypassing local maxima. As in cardinality learning, we are able to utilize emergent signals allowing the introduction of new hidden variables into simpler models rendering them more effective.

13. Discussion and Future Work

In this work we addressed the challenge of learning models with hidden variables in real-life scenarios. We presented a general approach for learning the parameters of hidden variables in Bayesian networks and introduced model selection operators that allow learning of new hidden variables and their cardinality. We showed that the method achieves significant improvement on challenging real-life problems.

The contribution of this work is threefold. First, we made a formal connection between the objective functionals of the information bottleneck framework (Tishby et al., 1999; Friedman et al., 2001) and maximum likelihood learning for graphical models. The information bottleneck and its extensions are originally viewed as methods to understand the structure of a distribution. We showed that in some sense the information bottleneck and maximum likelihood estimation are two sides of the same coin. The information bottleneck focuses on the distribution of variables in each instance, while maximum likelihood focuses on the projection of this distribution on the estimated model. This understanding extends to general Bayesian networks the recent results of Slonim and Weiss (2002) that relate the original information bottleneck and maximum likelihood estimation in univariate mixture distributions.

Second, the introduction of the IB-EM principle allowed us to use an approach that starts with a solution at $\gamma = 0$ and progresses toward a solution in the more complex landscape of $\gamma = 1$. This general scheme is common in *deterministic annealing* approaches (Rose, 1998; Ueda and Nakano, 1998). These approaches "flatten" the posterior landscape by raising the likelihood to the power of γ. The main technical difference of our approach is the introduction of a regularization term that is derived from the structure of the approximation of the probability of the latent variables in each instance. This was combined with a continuation method for traversing the path from the trivial solution at $\gamma = 0$ to a solution at $\gamma = 1$. Unlike standard approaches in deterministic annealing and information bottleneck, our procedure can automatically detect important regions where the

solution changes drastically and ensure that they are tracked closely. In preliminary experiment the continuation method was clearly superior to standard annealing strategies.

Third, we introduced model enrichment operators for inserting new hidden variables into the network structure and adapting their cardinality. These operators were specifically geared toward utilizing the emergent cues resulting from the annealing procedure. This resulted in models that generalize better and achieve equivalent or better results with a relatively simple model.

The methods presented here can be extended in several directions. First, we can improve the introduction of new hidden variables into the structure by formulating better "signals" that can be efficiently calculated for larger clusters. Second, we can use alternative variational approximations as well as adaptive approximation during the learning process. Third, we want to explore methods for stopping at $\gamma < 1$ as an alternative way for improving generalization performance.

Acknowledgments

We thank R. Bachrach, Y. Barash, G. Chechik, T. Kaplan, D. Koller, M. Ninio, D. Pe'er, A. Regev, T. Tishby, and Y. Weiss for discussions and comments on earlier drafts of this paper. This work was supported, in part, by a grant from the Israeli Ministry of Science and a grant by Intel Corporation. G. Elidan was also supported by the Horowitz fellowship. N. Friedman was also supported by an Alon Fellowship and by the Harry & Abe Sherman Senior Lectureship in Computer Science.

Appendix A. Fixed Point Equations

We now develop the fixed point equations use for solving the target Lagrangian of our approach. We start with the case of a single hidden variables and then address the more general scenario of multiple hidden variables.

A.1 Single Hidden Variable

Proposition 4: *Let $\mathcal{L}_{EM}$ be defined via $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$ as in Proposition 1. $Q(T \mid Y)$ is a stationary point of $\mathcal{L}_{EM}$ with respect to a fixed choice of P if and only if for all values t and y of T and Y, respectively,*

$$Q(t \mid y) = \frac{1}{Z(y,\gamma)} Q(t)^{1-\gamma} P(\mathbf{x}[y], t])^{\gamma},$$

where $Z(y,\gamma)$ is a normalizing constant and equals to

$$Z(y,\gamma) = \sum_{t'} Q(t')^{1-\gamma} P(\mathbf{x}[y], t'])^{\gamma}. \tag{15}$$

To prove the proposition we use the following

Lemma 9 *(El-Hay and Friedman, 2001) Let $Q(\mathbf{X})$ be a joint distribution over a set of random variables $\mathbf{X}$, that decomposes according to $Q(\mathbf{X}) = \prod_i Q(X_i \mid \mathbf{U}_i)$. Then*

$$\frac{\partial \boldsymbol{E}_Q[f(\mathbf{X})]}{\partial Q(x_i \mid \mathbf{u}_i)} = Q(\mathbf{u}_i) \boldsymbol{E}_{Q(\cdot \mid x_i, \mathbf{u}_i)}[f(\mathbf{X})] + \boldsymbol{E}_Q \left[\frac{\partial f(\mathbf{x})}{\partial Q(x_i, \mathbf{u}_i)} \right].$$

The following is an immediate results of that fact that $Q(t) = \sum_{y'} Q(y')Q(t|y')$

$$\frac{\partial Q(T)}{\partial Q(t_0 \mid y_0)} = Q(y_0) 1\{T = t_0\}. \tag{16}$$

We use this and an instantiation of the above lemma to prove the following:

Lemma 10

$$\frac{\partial \boldsymbol{I}_Q(T;Y)}{\partial Q(t_0 \mid y_0)} = Q(y_0) \log \frac{Q(t_0|y_0)}{Q(t_0)}.$$

Proof: We define $f(T,Y) \equiv \log \frac{Q(T,Y)}{Q(T)Q(Y)} = \log \frac{Q(T|Y)}{Q(T)}$ so that using Eq. (16), we can write

$$\begin{aligned}
\frac{\partial f(T,Y)}{\partial Q(t_0 \mid y_0)} &= \frac{\partial \log Q(T \mid Y)}{\partial Q(t_0 \mid y_0)} - \frac{\partial \log Q(T)}{\partial Q(t_0 \mid y_0)} \\
&= \frac{1}{Q(t_0 \mid y_0)} 1\{T = t_0, Y = y_0\} - \frac{Q(y_0)}{Q(t_0)} 1\{T = t_0\}.
\end{aligned}$$

Plugging this into Lemma 9, we get

$$\begin{aligned}
\frac{\partial \boldsymbol{I}_Q(T;Y)}{\partial Q(t_0 \mid y_0)} &= Q(y_0) \boldsymbol{E}_{Q(\cdot|t_0,y_0)} \left[\log \frac{Q(T \mid Y)}{Q(T)} \right] + \boldsymbol{E}_Q \left[\frac{\partial \log \frac{Q(T|Y)}{Q(T)}}{\partial Q(t_0,y_0)} \right] \\
&= Q(y_0) \log \frac{Q(t_0 \mid y_0)}{Q(t_0)} + Q(y_0) \frac{Q(t_0 \mid y_0)}{Q(t_0 \mid y_0)} - \sum_y Q(y) Q(t_0 \mid y) \frac{Q(y_0)}{Q(t_0)} \\
&= Q(y_0) \log \frac{Q(t_0 \mid y_0)}{Q(t_0)} + Q(y_0) \left[1 - \frac{1}{Q(t_0)} \sum_y Q(y) Q(t_0 \mid y) \right] \\
&= Q(y_0) \log \frac{Q(t_0 \mid y_0)}{Q(t_0)} + Q(y_0) [1 - 1] \\
&= Q(y_0) \log \frac{Q(t_0 \mid y_0)}{Q(t_0)}.
\end{aligned}$$

∎

Using Eq. (16) and Lemma 9 with $f(T,Y) \equiv \log Q(T)$, the following is immediate.

Lemma 11

$$\frac{\partial \boldsymbol{E}_Q[\log Q(T)]}{\partial Q(t_0 \mid y_0)} = Q(y_0) \log Q(t_0) + Q(t_0) \frac{1}{Q(t_0)} Q(y_0) = Q(y_0) [\log Q(t_0) + 1].$$

Proof of the proposition: We want to find $Q(T \mid Y)$ that are stationary points of the Lagrangian $\mathcal{L}_{EM}$ and where the constraints $\sum_t Q(t \mid y) = 1$ hold for any y. Thus, using Lagrange multipliers, we want to optimize

$$\mathcal{L} = \boldsymbol{I}_Q(T;Y) - \gamma(\boldsymbol{E}_Q[\log P(\mathbf{X},T)] - \boldsymbol{E}_Q[\log Q(T)]) + \sum_y \lambda_y \left(\sum_{t'} Q(t' \mid y) - 1 \right).$$

Since P is fixed, using Lemma 9 with $f(Y,\mathbf{X},T) \equiv \log P(\mathbf{X},T)$, we can write

$$\frac{\partial \boldsymbol{E}_Q[\log P(\mathbf{X},T)]}{\partial Q(t_0 \mid y_0)} = Q(y_0)\log P(\mathbf{x}[y_0],t_0).$$

Combining this with Lemma 10 and Lemma 11, we get

$$\frac{\partial \mathcal{L}_{EM}}{\partial Q(t_0 \mid y_0)} = Q(y_0)\left[\log Q(t_0 \mid y_0) - (1-\gamma)\log Q(t_0) + \gamma - \gamma \log P(\mathbf{x}[y_0],t_0)\right] + \lambda_{y_0}.$$

Dividing by $Q(y_0)$ and equating to 0, we get after rearranging of terms

$$Q(t_0|y_0) = e^{\lambda_{y_0}/Q(y_0)+\gamma} Q(t_0)^{1-\gamma} P(\mathbf{x}[y_0],t_0)^{\gamma}. \tag{17}$$

This must hold for any value t_0 and y_0. Using $\sum_t Q(t \mid y_0) = 1$ we get

$$e^{\lambda_{y_0}/Q(y_0)+\gamma} = \frac{1}{\sum_t Q(t)^{1-\gamma} P(\mathbf{x}[y_0],t)^{\gamma}}.$$

We get the desired result by plugging this into Eq. (17).

A.2 Multiple Hidden Variables

Proposition 6: *Let $\mathcal{L}^+_{EM}$ be defined via $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$ as in Eq. (12). Assuming a* mean field *approximation for $Q(\mathbf{T} \mid Y)$, a (local) maximum of $\mathcal{L}^+_{EM}$ is achieved by iteratively solving, independently for each hidden variable i, the self-consistent equations*

$$Q(t_i \mid y) \quad = \quad \frac{1}{Z(i,y,\gamma)} Q(t_i)^{1-\gamma} \exp^{\gamma \mathbf{EP}(t_i,y)},$$

where

$$\mathbf{EP}(t_i,y) \equiv \boldsymbol{E}_{Q(\mathbf{T}|t_i,y)}[\log P(\mathbf{x}[y],\mathbf{T})]$$

and $Z(i,y,\gamma)$ is a normalizing constant that equals to

$$Z(i,y,\gamma) = \sum_{t'_i} Q(t'_i)^{1-\gamma} \exp^{\gamma \mathbf{EP}(t'_i,y)}.$$

Proof: Using the *mean field* assumption, the information and entropy terms in the Lagrangian decompose as follows

$$\mathcal{L}^+_{EM} = \sum_i \boldsymbol{I}_Q(T_i;Y) - \gamma \left(\boldsymbol{E}_Q[\log P(\mathbf{X},T)] - \sum_i \boldsymbol{E}_Q[\log Q(T_i)] \right).$$

When computing the derivative with respect to the parameters of a specific variables T_i, the only change from the case of single hidden variable, is in the derivative of $\boldsymbol{E}_Q[\log P(\mathbf{X},T)]$ given fixed P. Again using Lemma 9 with $f(Y,\mathbf{X},T) \equiv \log P(\mathbf{X},T)$ we get

$$\frac{\partial \boldsymbol{E}_Q[\log P(\mathbf{X},\mathbf{T})]}{\partial Q(t_{i0} \mid y_0)} = \boldsymbol{E}_{Q(T|t_{i0},y_0)}[\log P(\mathbf{x}[y_0],\mathbf{T})],$$

from which we get the change from Proposition 4 to Proposition 6 for the case of multiple hidden variables. ∎

Appendix B. Computing the Continuation Direction

We now develop the precise computations needed to perform continuation as described in Section 4. We start with the case of a single hidden variables T.

B.1 Single Hidden Variable

Consider again Eq. (7), where we now write the normalization term $Z(y,\gamma)$ explicitly:

$$\begin{aligned} G_{t,y}(Q,\gamma) = \quad & -\log Q(t \mid y) + (1-\gamma)\log Q(t) + \gamma \log P(\mathbf{x}[y],t) \\ & - \log \underbrace{\sum_{t'} \exp^{(1-\gamma)\log Q(t') + \gamma \log P(\mathbf{x}[y],t')}}_{Z(y,\gamma)}. \end{aligned} \tag{18}$$

We want to compute the derivative of $G_{t,y}(Q,\gamma)$ with respect to the parameters and γ, and and then use the orthogonal direction for continuation. The will follow a direction in which the fix point equations remain unchanged, and the local maximum is tracked. To do so, we start by expressing $\log P(\mathbf{x}[y],t)$ as a function of the parameters Q.

The maximum likelihood parameters of $\log P(\mathbf{X},T)$ for the conditional distribution of the children X_i of T in $\mathcal{G}_{out}$ are

$$\theta_{x_i|\mathbf{pa}_i,t} = \frac{\sum_y Q(y)Q(t|y)1\{x_i[y]=x_i,\mathbf{pa}_i[y]=\mathbf{pa}_i\} + \alpha(x_i,\mathbf{pa}_i,t)}{\sum_y Q(y)Q(t|y)1\{\mathbf{pa}_i[y]=\mathbf{pa}_i\} + \alpha(\mathbf{pa}_i,t)} \equiv \frac{\mathcal{N}(x_i,\mathbf{pa}_i,t)}{\mathcal{N}(\mathbf{pa}_i,t)}, \tag{19}$$

where $1\{\}$ is the indicator function, $\alpha()$ are the hyper-parameters of the Dirichlet prior distribution (see Section 2.1) and $\mathcal{N}$ are used to denote the total counts (including prior) used for estimation. Similarly the maximum likelihood parameters of the distribution of T given its parents are

$$\theta_{t|\mathbf{pa}_t} = \frac{\sum_y Q(y)Q(t|y)1\{\mathbf{pa}_t[y]=\mathbf{pa}_t\} + \alpha(\mathbf{pa_t},t)}{\sum_y Q(y)1\{\mathbf{pa}_t[y]=\mathbf{pa}_t\} + \alpha(\mathbf{pa}_t)} \equiv \frac{\mathcal{N}(\mathbf{pa}_t,t)}{\mathcal{N}(\mathbf{pa}_t)}. \tag{20}$$

We now consider each term in $G_{t,y}(Q,\gamma)$ and compute its derivative with respect to these parameters of Q.

COMPUTATION OF $\frac{\partial \log P(\mathbf{x}[y],t)}{\partial Q(t_0|y_0)}$

The derivatives of the parameters expressed in Eq. (19) are

$$\begin{aligned} &\tfrac{\partial \theta_{x_i|\mathbf{pa}_i,t}}{\partial Q(t_0|y_0)} \\ &\quad = \tfrac{Q(y_0)}{\mathcal{N}(\mathbf{pa}_i,t)^2}\left[\ 1\{x_i[y_0]=x_i,\mathbf{pa}_i[y_0]=\mathbf{pa}_i\}\mathcal{N}(\mathbf{pa}_i,t) - 1\{\mathbf{pa}_i[y_0]=\mathbf{pa}_i\}\mathcal{N}(x_i,\mathbf{pa}_i,t)\ \right] \\ &\quad = \tfrac{Q(y_0)1\{\mathbf{pa}_i[y_0]=\mathbf{pa}_i\}}{\mathcal{N}(\mathbf{pa}_i,t)^2}\Big(1\{x_i[y_0]=x_i\}\mathcal{N}(\mathbf{pa}_i,t) - \mathcal{N}(x_i,\mathbf{pa}_i,t)\Big) \end{aligned} \tag{21}$$

for $t=t_0$ and are zero otherwise. Similarly, the derivatives of the parameters of Eq. (20) are

$$\frac{\partial \theta_{t|\mathbf{pa}_t}}{\partial Q(t_0 \mid y_0)} = \frac{Q(y_0)}{\mathcal{N}(\mathbf{pa}_t)^2}[1\{\mathbf{pa}_t[y_0]=\mathbf{pa}_t\}\mathcal{N}(\mathbf{pa}_t) - 0] = \frac{Q(y_0)}{\mathcal{N}(\mathbf{pa}_t)}1\{\mathbf{pa}_t[y_0]=\mathbf{pa}_t\} \tag{22}$$

for $t = t_0$, and are zero otherwise. The log-probability of a specific instance can be written as

$$\log P(\mathbf{x}[y],t) = \log \theta_{t|\mathbf{pa}_t}[y] + \sum_{i \in Ch_t} \log \theta_{x_i|\mathbf{pa}_i,t}[y] + \sum_{i \neq t, Ch_t} \log \theta_{x_i|\mathbf{pa}_i}[y], \tag{23}$$

where Ch_t denotes the children of T in $\mathcal{G}_{out}$ and $\theta_{t|\mathbf{pa}_t}[y]$ is the parameter corresponding to the values appearing in instance y. We note that the last summation does not depend on the parameters $Q(t \mid y)$, and by plugging Eq. (21) and Eq. (22) into Eq. (23), we get

$$\begin{aligned}
\frac{\partial \log P(\mathbf{x}[y],t)}{\partial Q(t_0 \mid y_0)} &= \frac{1}{\theta_{t|\mathbf{pa}_t}[y]} \frac{\partial \theta_{t|\mathbf{pa}_t}[y]}{\partial Q(t_0 \mid y_0)} + \sum_{i \in Ch_t} \frac{1}{\theta_{x_i|\mathbf{pa}_i,t}[y]} \frac{\partial \theta_{x_i|\mathbf{pa}_i,t}[y]}{\partial Q(t_0 \mid y_0)} \\
&= Q(y_0) \left[\frac{1\{\mathbf{pa}_t[y_0]=\mathbf{pa}_t[y]\}}{\mathcal{N}(\mathbf{pa}_t)\theta_{t|\mathbf{pa}_t}[y_0]} \right. \\
&\qquad \left. + \sum_{i \in Ch_t} \frac{1\{\mathbf{pa}_i[y_0]=\mathbf{pa}_i[y]\}}{\mathcal{N}(\mathbf{pa}_i,t)^2 \theta_{x_i|\mathbf{pa}_i}[y_0]} \Big(1\{x_i[y] = x_i[y_0]\} \mathcal{N}(\mathbf{pa}_i,t) - \mathcal{N}(x_i,\mathbf{pa}_i,t) \Big) \right] \\
&\equiv Q(y_0) \mathcal{D}(y,t),
\end{aligned} \tag{24}$$

where in the last line we use $\mathcal{D}(y,t)$ to denote the expression in the square brackets.

COMPUTATION OF $\frac{\partial \log Z(y_0,\gamma)}{\partial Q(t_0|y_0)}$

Using Eq. (16) from Appendix A and the above, we can write

$$\frac{\partial \left((1-\gamma) \log Q(t) + \gamma \log P(\mathbf{x}[y],t) \right)}{\partial Q(t_0 \mid y_0)} = Q(y_0) \left[\frac{1-\gamma}{Q(t)} + \gamma \mathcal{D}(y,t) \right]. \tag{25}$$

We can now use Eq. (25) to write the derivative of $Z(y,\gamma)$ since it is a summation over similar expressions

$$\begin{aligned}
\frac{\partial \log Z(y_0,\gamma)}{\partial Q(t_0|y_0)} &= \frac{1}{Z(y_0,\gamma)} \exp^{(1-\gamma)\log Q(t_0) + \gamma \log P(\mathbf{x}[y],t_0)} Q(y_0) \left[\frac{1-\gamma}{Q(t_0)} + \gamma D(y_0,t_0) \right] \\
&= \frac{1}{Z(y_0,\gamma)} Q(y_0) Q(t_0)^{1-\gamma} P(\mathbf{x}[y],t_0)^{\gamma} \left[\frac{1-\gamma}{Q(t_0)} + \gamma D(y_0,t_0) \right] \\
&= Q(y_0) Q(t_0 \mid y_0) \left[\frac{1-\gamma}{Q(t_0)} + \gamma D(y_0,t_0) \right],
\end{aligned} \tag{26}$$

where the last equality follows from Proposition 4.

COMPUTATION OF $\frac{\partial G_{t,y}(Q,\gamma)}{\partial Q(t_0|y_0)}$

We combine Eq. (25) and Eq. (26) to write

$$\frac{\partial G_{t,y}(Q,\gamma)}{\partial Q(t_0 \mid y_0)} = -1\{y = y_0\} + Q(y_0)\left[1 - Q(t_0 \mid y_0)\right] \left[\frac{1-\gamma}{Q(t_0)} + \gamma D(y_0,t_0) \right]. \tag{27}$$

COMPUTATION OF $\frac{\partial \log Z(y,\gamma)}{\partial \gamma}$

The only term that is not immediate is the derivative of $Z(y,\gamma)$ with respect to γ

$$\begin{aligned}
\frac{\partial \log Z(y,\gamma)}{\partial \gamma} &= \frac{1}{Z(y,\gamma)} \sum_{t'} \exp^{(1-\gamma)\log Q(t') + \gamma \log P(\mathbf{x}[y],t')} \left[-\log Q(t') + \log P(\mathbf{x}[y],t') \right] \\
&= \sum_{t'} \frac{1}{Z(y,\gamma)} Q(t')^{1-\gamma} P(\mathbf{x}[y],t')^{\gamma} \left[-\log Q(t') + \log P(\mathbf{x}[y],t') \right] \\
&= \sum_{t'} Q(t'|y) \left[\log P(\mathbf{x}[y],t') - \log Q(t') \right],
\end{aligned}$$

from which follows

$$\frac{\partial G_{t,y}(Q,\gamma)}{\partial \gamma} = \log P(\mathbf{x}[y],t) - \log Q(t) - \sum_{t'} Q(t'|y)\left[\log P(\mathbf{x}[y],t') - \log Q(t')\right]. \tag{28}$$

COMPUTATION OF THE CONTINUATION DIRECTION

We can now compute all the elements of the derivative matrix of Eq. (9)

$$H_{t,y}(Q,\gamma) = \left(\frac{\partial G_{t,y}(Q,\gamma))}{\partial Q(t|y)} \middle| \frac{\partial G_{t,y}(Q,\gamma)}{\partial \gamma}\right).$$

To compute the orthogonal direction to the derivative, we solve Eq. (10)

$$H(Q,\gamma)\Delta = 0.$$

As noted in Section 4, this can be prohibitively expensive and we resort to $H(Q,\gamma)$ with a diagonal approximation for elements of $\frac{\partial G_{t,y}(Q,\gamma)}{\partial Q(t|y)}$ computed in Eq. (27). We denote by $h_{y,t}$ the diagonal entry for $Y = y$ and $T = t$ and $h^{\gamma}_{y,t}$ the corresponding derivative with respect to γ. We then have to solve a set of equations of the form

$$d_{t,y}h_{y,t} + d_{\gamma}h^{\gamma}_{y,t} = 0,$$

where $d_{t,y}$ and d_{γ} are the elements of Δ. Setting $d_{\gamma} = 1$ (an equivalent solution up to scaling) we get the unique solution

$$d_{t,y} = -\frac{h^{\gamma}_{y,t}}{h_{y,t}}.$$

Normalizing Δ using the derivative of $\boldsymbol{I}_Q(T;Y)$ as described in Eq. (11) can now be easily computed given the Lemma 10 in Appendix A.

B.2 Multiple Hidden Variables

When computing the derivative with respect to the parameters associated with a specific hidden variable t_i, the only change in $G_{t,y}(Q,\gamma)$ is that $\log P(\mathbf{x}[y],t)$ is replaced by $\boldsymbol{E}_{Q(\mathbf{T}|t_i,y)}[\log P(\mathbf{x}[y],\mathbf{T})]$. In this case we simply compute the expectation of Eq. (24) over the T's that are in the Markov blanket of t_i. The rest of the details remain the same.

References

J. Adachi and M. Hasegawa. Molphy version 2.3, programs for molecular phylogenetics based on maximum likelihood. Technical report, The Institute of Statistical Mathematics, Tokyo, Japan, 1996.

S. Becker, S. Thrun, and K. Obermayer, editors. *Advances in Neural Information Processing Systems 15*. MIT Press, Cambridge, Mass., 2002.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, United Kingdom, 1995.

X. Boyen, N. Friedman, and D. Koller. Discovering the hidden structure of complex dynamic systems. In K. Laskey and H. Prade, editors, *Proc. Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI '99)*, pages 91–100, San Francisco, 1999. Morgan Kaufmann.

J.S. Breese and D. Koller, editors. *Proc. Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI '01)*. Morgan Kaufmann, San Francisco, 2001.

K. Chang and R. Fung. Refinement and coarsening of bayesian networks. In P. P. Bonissone, M. Henrion, L. N. Kanal, and J. F. Lemmer, editors, *Proc. Sixth Annual Conference on Uncertainty Artificial Intelligence (UAI '90)*, pages 475–482, San Francisco, 1990. Morgan Kaufmann.

P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. Autoclass: a Bayesian classification system. In *Proc. Fifth International Workshop on Machine Learning*, pages 54–64. Morgan Kaufmann, San Francisco, 1988.

D. M. Chickering. Learning equivalence classes of Bayesian network structures. In E. Horvitz and F. Jensen, editors, *Proc. Twelfth Conference on Uncertainty in Artificial Intelligence (UAI '96)*, pages 150–157, San Francisco, 1996. Morgan Kaufmann.

D. M. Chickering and D. Heckerman. Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, 29:181–212, 1997.

A. Corduneanu and T. Jaakkola. Continuation methods for mixing heterogeneous sources. In A. Darwich and N. Friedman, editors, *Proc. Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI '02)*, pages 111–118, San Francisco, 2002. Morgan Kaufmann.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, 1991.

M. H. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill, New York, 1970.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B 39:1–39, 1977.

T. El-Hay and N. Friedman. Incorporating expressive graphical models in variational approximations: Chain-graphs and hidden variables. In Breese and Koller (2001), pages 136–143.

G. Elidan and N. Friedman. Learning the dimensionality of hidden variables. In Breese and Koller (2001), pages 144–151.

G. Elidan, N. Lotner, N. Friedman, and D. Koller. Discovering hidden variables: A structure-based approach. In Leen et al. (2001), pages 479–485.

G. Elidan, M. Ninio, N. Friedman, and D. Schuurmans. Data perturbation for escaping local maxima in learning. In *Proc. National Conference on Artificial Intelligence (AAAI '02)*, pages 132–139. AAAI Press, Menlo Park, CA, 2002.

N. Friedman. Learning belief networks in the presence of missing values and hidden variables. In D. Fisher, editor, *Proc. Fourteenth International Conference on Machine Learning*, pages 125–133. Morgan Kaufmann, San Francisco, 1997.

N. Friedman, O. Mosenzon, N. Slonim, and N. Tishby. Multivariate information bottleneck. In Breese and Koller (2001), pages 152–161.

A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, and P. O. Brown. Genomic expression program in the response of yeast cells to environmental changes. *Molecular Biology of the Cell*, 11:4241–4257, 2000.

F. Glover and M. Laguna. Tabu search. In C. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, Oxford, England, 1993. Blackwell Scientific Publishing.

D. Heckerman. A tutorial on learning with Bayesian networks. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, Dordrecht, Netherlands, 1998.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. In R. López de Mantarás and D. Poole, editors, *Proc. Tenth Conference on Uncertainty in Artificial Intelligence (UAI '94)*, pages 293–301, San Francisco, 1994. Morgan Kaufmann.

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

E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106:620–630, 1957.

M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An introduction to variational approximations methods for graphical models. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, Dordrecht, Netherlands, 1998.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220 (4598):671–680, 1983.

W. Lam and F. Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10:269–293, 1994.

S. L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201, 1995.

T. K. Leen, T. G. Dietterich, and V. Tresp, editors. *Advances in Neural Information Processing Systems 13*. MIT Press, Cambridge, Mass., 2001.

J. Martin and K. VanLehn. Discrete factor analysis: Learning hidden variables in Bayesian networks. Technical report, Department of Computer Science, University of Pittsburgh, 1995.

M. Meila and M. I. Jordan. Estimating dependency structure as a hidden variable. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 584–590, Cambridge, Mass., 1998. MIT Press.

R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, Dordrecht, Netherlands, 1998.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

F. Pereira, N. Tishby, and L. Lee. Distributional clustering of English words. In *31st Annual Meeting of the ACL*, pages 183–190, 1993.

K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proc. IEEE*, 86:2210–2239, 1998.

N. Slonim, N.Friedman, and T.Tishby. Agglomerative multivariate information bottleneck. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 929–936, Cambridge, Mass., 2002. MIT Press.

N. Slonim and N. Tishby. Agglomerative information bottleneck. In S. A. Solla, T. K. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 617–623, Cambridge, Mass., 2000. MIT Press.

N. Slonim and N. Tishby. Data clustering by markovian relaxation and the information bottleneck method. In Leen et al. (2001), pages 640–646.

N. Slonim and Y. Weiss. Maximum likelihood and the information bottleneck. In Becker et al. (2002), pages 351–358.

N. A. Smith and J. Eisner. Annealing techniques for unsupervised statistical language learning. In *Proc. 42nd Annual Meeting of the Association for Computational Linguistics*, 2004.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. Number 81 in Lecture Notes in Statistics. Springer-Verlag, New York, 1993.

A. Stolcke and S. Omohundro. Hidden Markov Model induction by bayesian model merging. In Stephen José Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 11–18. Morgan Kaufmann, San Mateo, CA, 1993.

M. Szummer and T. Jaakkola. Information regularization with partially labeled data. In Becker et al. (2002), pages 640–646.

B. Thiesson. Score and information for recursive exponential models with incomplete data. In D. Geiger and P. Shanoy, editors, *Proc. Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI '97)*, San Francisco, 1997. Morgan Kaufmann.

B. Thiesson, C. Meek, D. M. Chickering, and D. Heckerman. Learning mixtures of Bayesian networks. In G. F. Cooper and S. Moral, editors, *Proc. Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI '98)*, pages 504–513, San Francisco, 1998. Morgan Kaufmann.

N. Tishby, F. Pereira, and W. Bialek. The information bottleneck method. In B. Hajek and R. S. Sreenivas, editors, *Proc. 37th Allerton Conference on Communication, Control and Computation*, pages 368–377. University of Illinois, 1999.

N. Ueda and R. Nakano. Deterministic annealing EM algorithm. *Neural Networks*, 11(2):271–282, 1998.

L. T. Watson. Theory of globally convergent probability-one homotopies for non-linear programming. Technical Report TR-00-04, Department of Computer Science, Virginia Tech, 2000.

M. Whiley and D. M. Titterington. Applying the deterministic annealing expectation maximization algorithm to Naive Bayes networks. Technical Report 02-5, Department of Statistics, University of Glasgow, 2002.

N. L. Zhang. Hierarchical latent class models for cluster analysis. *Journal of Machine Learning Research*, 5:697–723, 2004.