GRAPHICAL MODELS AND VARIATIONAL APPROXIMATION

Michael I. Jordan

Department of Electrical Engineering and Computer Science
Department of Statistics
University of California, Berkeley
http://www.ai.mit.edu/projects/jordan.html

Acknowledgments:

Zoubin Ghahramani, Tommi Jaakkola, Marina Meila
Lawrence Saul

June, 1998

GRAPHICAL MODELS

- A wide variety of models are being studied in the machine learning literature, including:
  - neural networks
  - decision trees
  - variants of mixture models
  - variants of hidden Markov models
  - subspace methods
  - naive Bayes models
  - Bayesian networks
  - support vector machines
- Is there any underlying unity in this menagerie?

Graphical models

- Graphical models are a marriage between graph theory and probability theory
- Historically their role in AI has been as probabilistic inference engines
- It turns out that many machine learning models can be viewed as special cases of graphical models

- Some advantages of the graphical model point of view
  - inference and learning are treated together
  - supervised and unsupervised learning are merged seamlessly
  - missing data handled nicely
  - a focus on conditional independence and computational issues
  - prior knowledge and interpretability

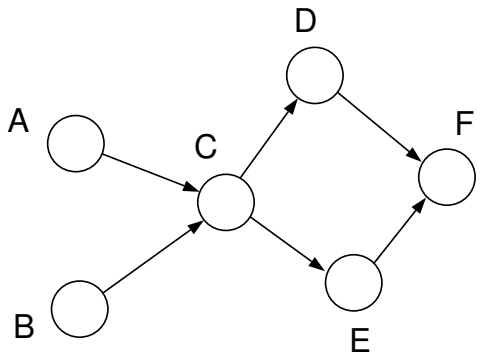
Probability and AI

(Coming soon to a theater near you)

There are a number of reasons why probability theory has been in ascendance in AI, with no end in sight:

- representation
  - modularity
- inference
  - non-modularity
- learning
  - unifying framework
  - close relationships to inference

Graphical models



- representation
  - modularity
- inference
  - non-modularity
- learning
  - unifying framework
  - close relationships to inference

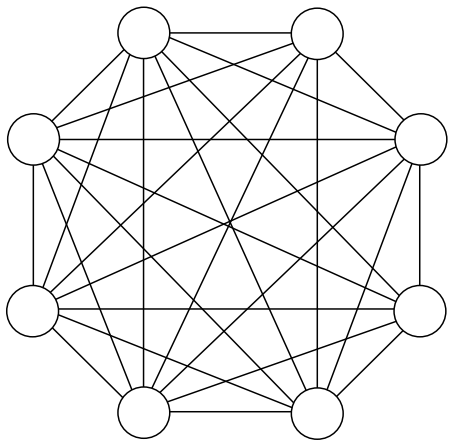
- We'll discuss the *inference* problem in some detail—the problem of calculating the probability of one or more nodes, conditioned on the value of some other set of nodes
  - exact algorithms
  - sampling-based algorithms
  - variational algorithms

Combinatorial complexity

- exact inference algorithms treat probability theory as a combinatorial calculus
  - calculate exactly all of those higher-order statistics (and *only* those higher-order statistics) that could possibly matter
- surprising how far one can go
  - sparse graphs include chains, trees, and "decorated" 1-D graphs
- but, as in any combinatorial calculus, eventually we lose (badly)

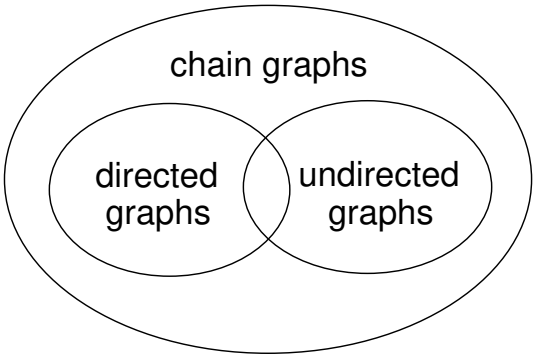
Probability and averaging

- Probability theory has *two* regimes in which things stay simple
  - in sparse graphs, the combinatorics can be handled
  - in dense graphs, probabilities can be simple (despite appearances)



- How can we exploit this virtue of probability theory systematically?
  - i.e., how do we develop a computational calculus that makes use of laws of large numbers?

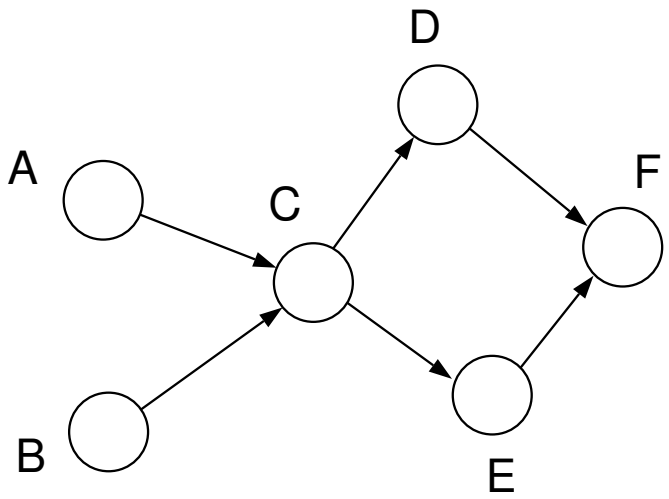
Graphical models



- Alternative names for graphical models: *belief networks, Bayesian networks, probabilistic independence networks, Markov random fields, loglinear models, influence diagrams*

Directed graphical models—basics

- Consider an arbitrary directed (acyclic) graph, where each node in the graph corresponds to a random variable (scalar or vector):



- We want to associate a *joint probability distribution* $P(A, B, C, D, E, F)$ with this graph, and we want all of our calculations to respect this distribution

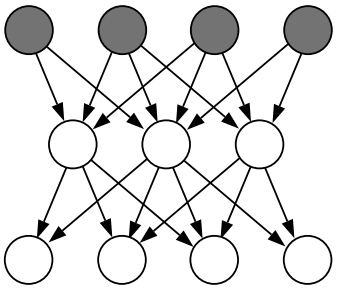
  e.g.,

$$P(F|A, B) = \frac{\Sigma_C \Sigma_D \Sigma_E P(A, B, C, D, E, F)}{\Sigma_C \Sigma_D \Sigma_E \Sigma_F P(A, B, C, D, E, F)}$$

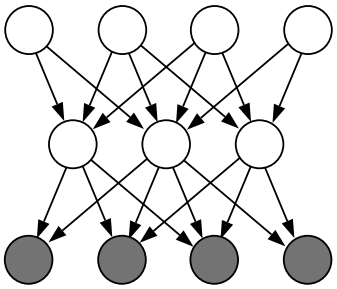
- Inference $\equiv$ bringing the sums "inside"

Some ways to use a graphical model

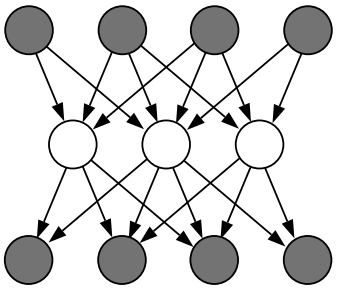
Prediction:



Diagnosis, control, optimization:



Supervised learning:



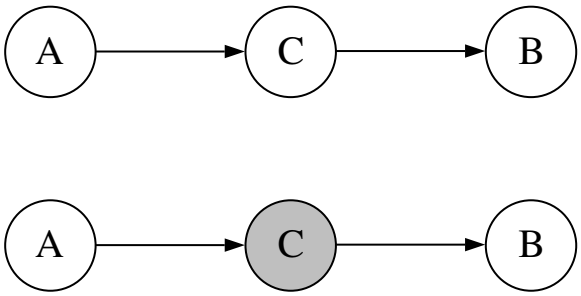
- we want to *marginalize* over the unshaded nodes in each case (i.e., integrate them out from the joint probability)
- "unsupervised learning" is the general case

Specification of a graphical model

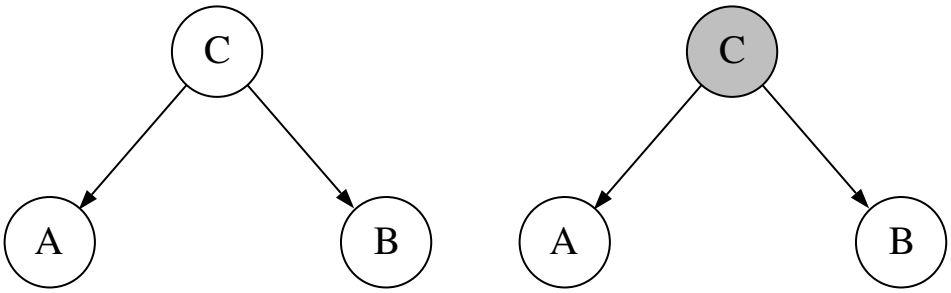
- There are two components to any graphical model:
  - the *qualitative* specification
  - the *quantitative* specification

- Where does the qualitative specification come from?
  - prior knowledge of causal relationships
  - prior knowledge of modular relationships
  - assessment from experts
  - learning from data
  - we simply like a certain architecture (e.g., a layered graph)

Qualitative specification of graphical models

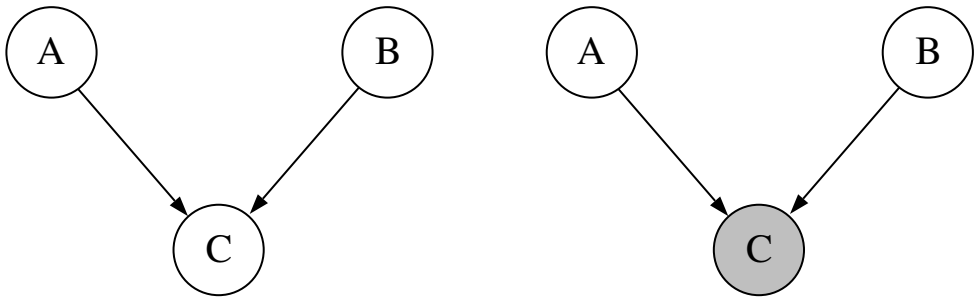


- A and B are marginally *dependent*
- A and B are conditionally *independent*



- A and B are marginally *dependent*
- A and B are conditionally *independent*

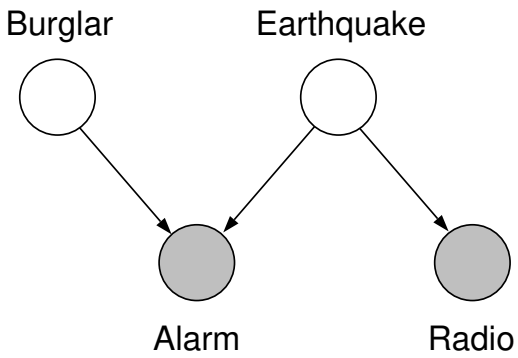
Semantics of graphical models (cont)



- A and B are marginally *independent*
- A and B are conditionally *dependent*

- This is the interesting case...

"Explaining away"



- All connections (in both directions) are "excitatory"
- But an increase in "activation" of Earthquake leads to a decrease in "activation" of Burglar
- Where does the "inhibition" come from?

Qualitative specification (cont.)

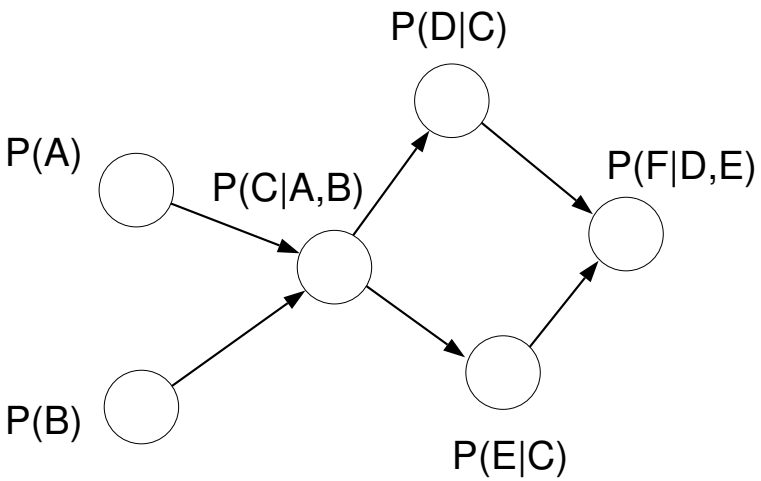
- General question: Is subset A independent of subset C given subset B?
- Use the three schemata recursively
- Yields a list of all of the conditional independencies (CI's) implied by a particular graph

- Consider all possible distributions on the variables in the graph
- Use the list as a filter, retaining a distribution only if it respects all of the CI's
- This yields a *family* of distributions associated with a given graph

Quantitative specification of directed models

- *Question*: how do we specify a joint distribution over the nodes in the graph numerically?

- *Answer*: associate a conditional probability with each node:



and take the product of the *local* probabilities to yield the *global* probabilities

Justification

- In general, let $\{S\}$ represent the set of random variables corresponding to the N nodes of the graph

- For any node S_i, let $\pi(S_i)$ represent the set of parents of node S_i

- Then

$$\begin{aligned} P(S) &= P(S_1)P(S_2|S_1)\cdots P(S_N|S_{N-1},\ldots,S_1) \\ &= \prod_i P(S_i|S_{i-1},\ldots,S_1) \\ &= \prod_i P(S_i|\pi(S_i)) \end{aligned}$$

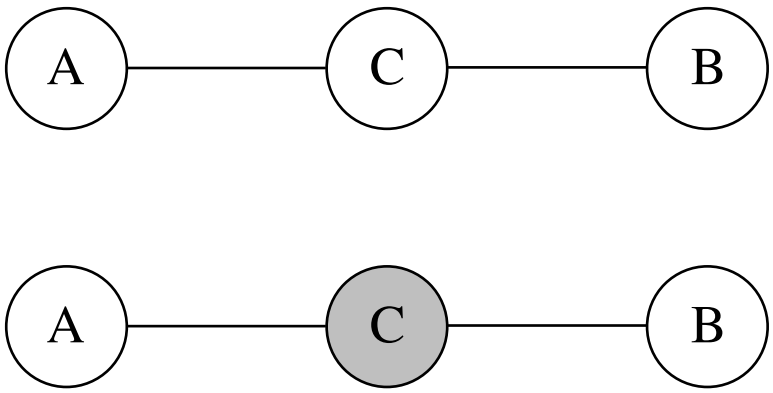
  where the last line is by assumption

- I.e., range over all possible choices of local distributions $P(S_i|\pi(S_i))$
- This yields a second family of distributions associated with a given graph

Justification (cont.)

- Theorem: *The two families that we have defined are identical*
- This theorem provides an interesting and important link between graph theory and probability theory

Semantics of undirected graphs

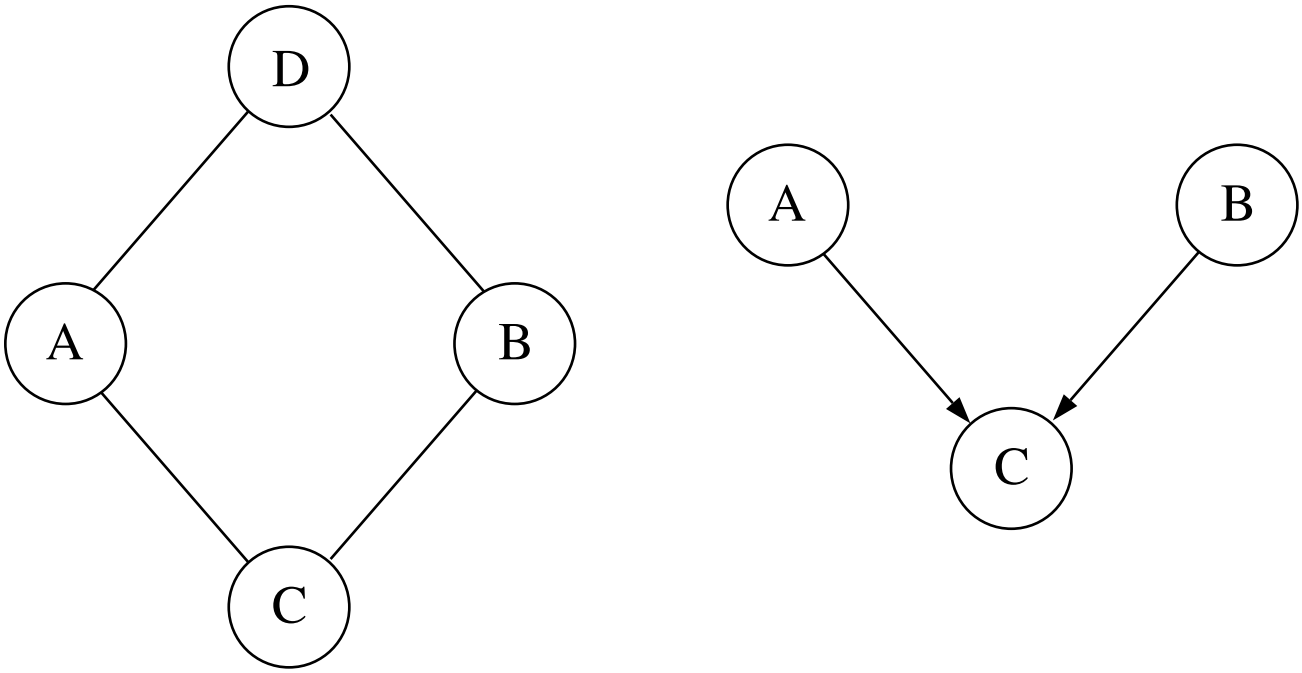


- A and B are marginally *dependent*
- A and B are conditionally *independent*

- General question: Is subset A independent of subset C given subset B?
- Graph separation answers the question
- Yields a list of all of the (conditional) independencies implied by a particular graph

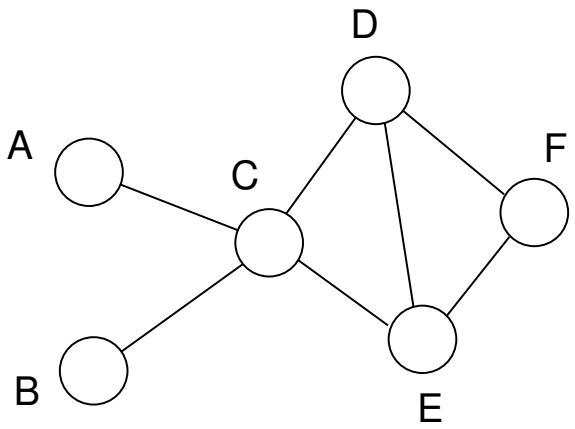
- Consider all possible distributions on the variables in the graph
- Use the list as a filter
- This yields a *family* of distributions associated with a given graph

Comparative semantics

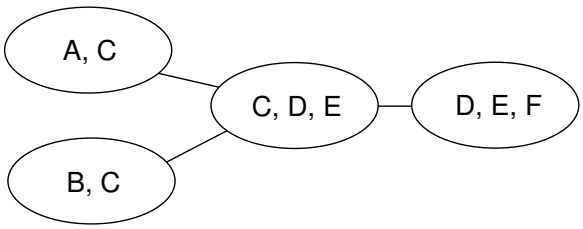


- The graph on the left yields conditional independencies that a directed graph can't represent
- The graph on the right yields marginal independencies that an undirected graph can't represent

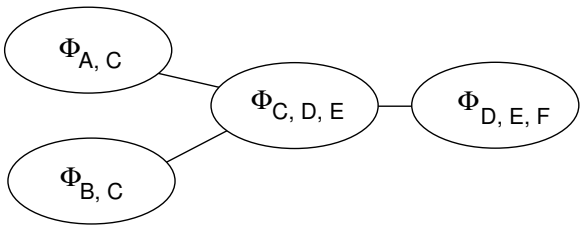
Quantitative specification of undirected models



- identify the *cliques* in the graph:



- define a *configuration* of a clique as a specification of values for each node in the clique
- define a *potential* of a clique as a function that associates a positive real number with each configuration of the clique



Quantitative specification (cont.)

- Consider the example of a graph with binary nodes
- A "potential" is a table with entries for each combination of nodes in a clique

A \ B	0	1
0	1.5	.4
1	.7	1.2

- "Marginalizing" over a potential table simply means collapsing (summing) the table along one or more dimensions

- marginalizing over B

A	
0	1.9
1	1.9

- marginalizing over A

B = 0	B = 1
2.2	1.6

Quantitative specification (cont.)

- Finally, define the probability of a *global* configuration of the nodes as the product of the *local* potentials on the cliques:

$$P(A, B, C, D, E, F) = \phi_{(A,B)}\phi_{(B,C)}\phi_{(C,D,E)}\phi_{(D,E,F)}$$

  where, without loss of generality, we assume that the normalization constant (if any) has been absorbed into one of the potentials

- Ranging over all possible (normalized) sets of potentials, we obtain a second family associated with the graph

- Theorem: *The two families that we have defined are identical*

Evidence and Inference

- "Absorbing evidence" means observing the values of certain of the nodes
- Absorbing evidence divides the units of the network into two groups:

visible units $\{V\}$	those for which we have instantiated values ("evidence nodes").
hidden units $\{H\}$	those for which we do not have instantiated values.

- "Inference" means calculating the *conditional distribution*

$$P(H|V) = \frac{P(H,V)}{\Sigma_{\{H\}}\ P(H,V)}$$

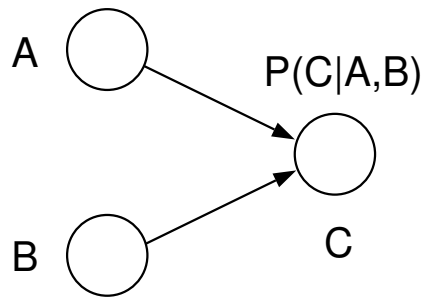
Inference algorithms for directed graphs

- There are several inference algorithms; some of which operate directly on the directed graph
- The most popular inference algorithm, known as the *junction tree* algorithm (which we'll discuss here), operates on an undirected graph
- It also has the advantage of clarifying some of the relationships between the various algorithms

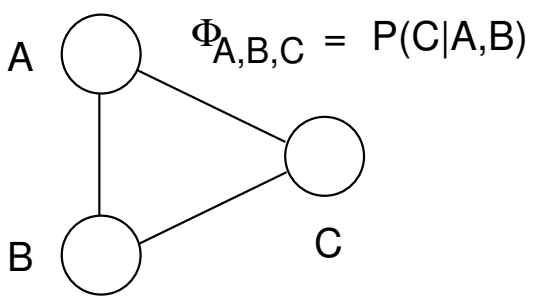
To understand the junction tree algorithm, we need to understand how to "compile" a directed graph into an undirected graph

Moral graphs

- Note that for both directed graphs and undirected graphs, the joint probability is in a product form
- So let's convert local conditional probabilities into potentials; then the products of potentials will give the right answer
- Indeed we can think of a conditional probability, e.g., $P(C|A,B)$ as a function of the three variables A, B, and C (we get a real number for each configuration):



- Problem: A node and its parents are not generally in the same clique
- Solution: Marry the parents to obtain the "moral graph"



Moral graphs (cont.)

- Define the potential on a clique as the product over all conditional probabilities contained within the clique
- *Now* the products of potentials gives the right answer:

$$\begin{aligned} & P(A,B,C,D,E,F) \\ &= P(A)P(B)P(C|A,B)P(D|C)P(E|C)P(F|D,E) \\ &= \phi_{(A,B,C)}\phi_{(C,D,E)}\phi_{(D,E,F)} \end{aligned}$$

where

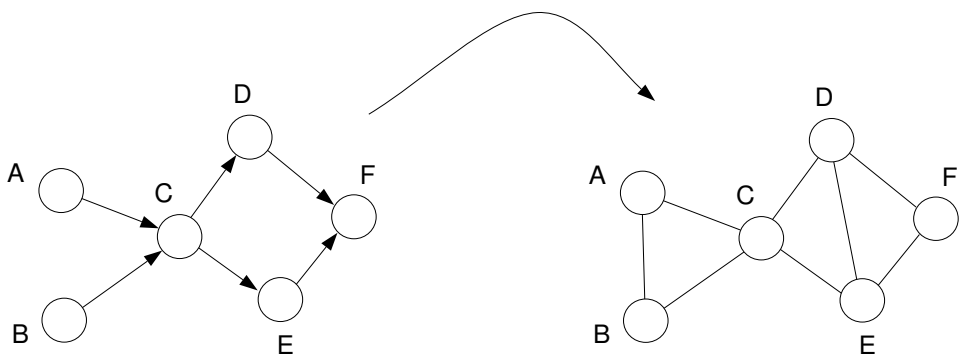
$$\phi_{(A,B,C)} = P(A)P(B)P(C|A,B)$$

and

$$\phi_{(C,D,E)} = P(D|C)P(E|C)$$

and

$$\phi_{(D,E,F)} = P(F|D,E)$$

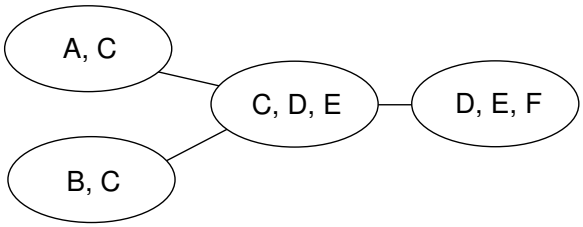


Propagation of probabilities

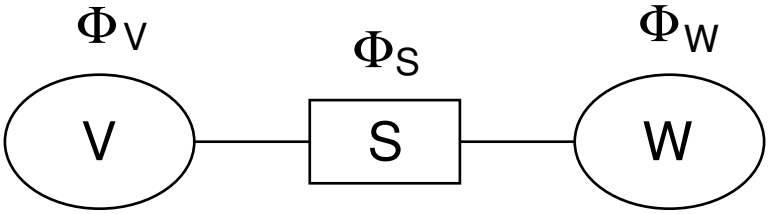
- Now suppose that some evidence has been absorbed.
- This changes the probabilities in one or more of the cliques
- How do we propagate this effect to the rest of the graph?
- *Solution: achieve consistency between the cliques*

Clique trees

- A *clique tree* is an (undirected) tree of cliques



- Consider cases in which two neighboring cliques V and W have an overlap S (e.g., (A, C) overlaps with (C, D, E)).



  - the cliques need to "agree" on the probability of nodes in the overlap; this is achieved by *marginalizing* and *rescaling*:

$$\phi_S^* = \sum_{V \setminus S} \phi_V$$

$$\phi_W^* = \phi_W \frac{\phi_S^*}{\phi_S}$$

  - this occurs in parallel, distributed fashion throughout the clique tree

Clique trees (cont.)

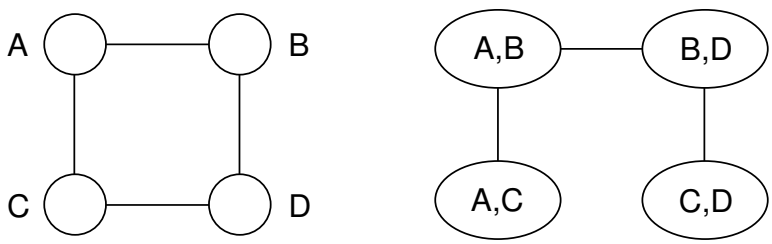
- This simple local message-passing algorithm on a clique tree defines the general probability propagation algorithm for directed graphs!

- Many interesting algorithms are special cases:
  - calculation of posterior probabilities in mixture models
  - Baum-Welch algorithm for hidden Markov models
  - posterior propagation for probabilistic decision trees
  - Kalman filter updates

- The algorithm seems reasonable. Is it correct?

A problem

- Consider the following graph and a corresponding clique tree:



- Note that C appears in two non-neighboring cliques.

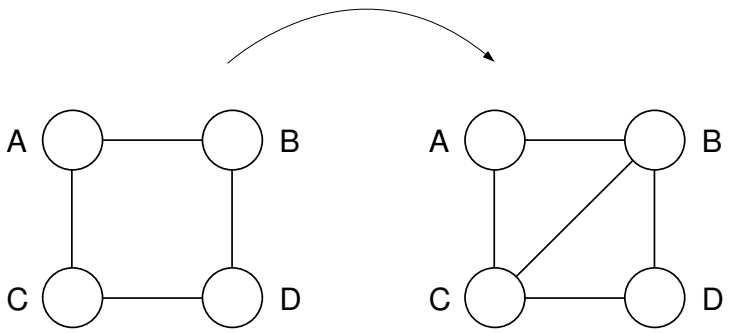
- *Question*: What guarantee do we have that the probability associated with C in these two cliques will be the same?

- *Answer*: Nothing. In fact this is a problem with the algorithm as described so far. It is not true that in general *local* consistency implies *global* consistency.

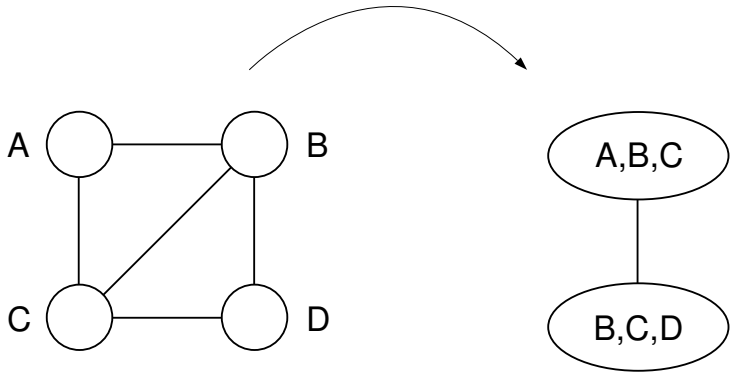
- What else do we need to get such a guarantee?

Triangulation (last idea, hang in there)

- A triangulated graph is one in which *no cycles with four or more nodes* exist in which there is *no chord*
- We *triangulate* a graph by adding chords:



- Now we no longer have our problem:



- A clique tree for a triangulated graph has the *running intersection property*: if a node appears in two cliques, it appears everywhere on the path between the cliques
- Thus *local consistency implies global consistency* for such clique trees

Junction trees

- A clique tree for a triangulated graph is referred to as a *junction tree*

- In junction trees, local consistency implies global consistency. Thus the local message-passing algorithm is (provably) correct.

- It's also possible to show that *only* triangulated graphs have the property that their clique trees are junction trees. Thus, if we want local algorithms, we *must* triangulate.

Summary of the junction tree algorithm

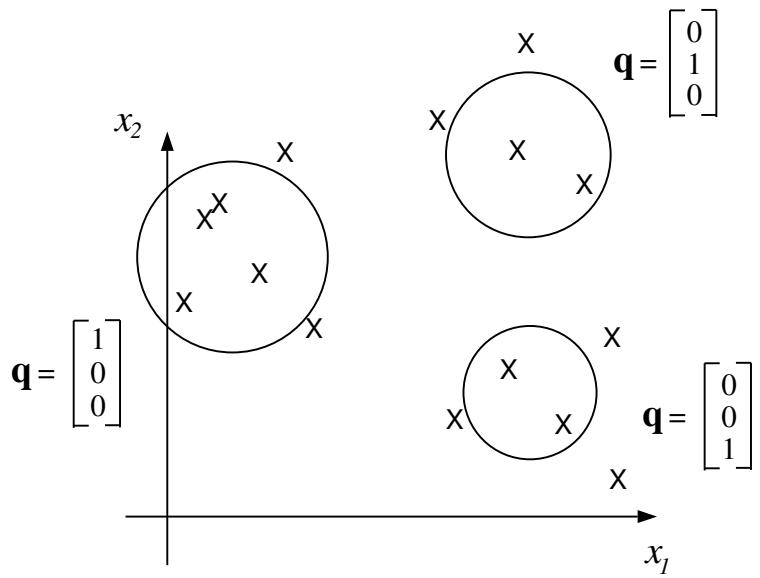
---

1. *Moralize* the graph
2. *Triangulate* the graph
3. *Propagate* by local message-passing in the junction tree

- Note that the first two steps are "off-line"
- Note also that these steps provide a bound of the complexity of the propagation step
  - marginalizing and rescaling scale exponentially in the size of a clique (the number of nodes in the clique)

Example: Gaussian mixture models

- A Gaussian mixture model is a popular clustering model



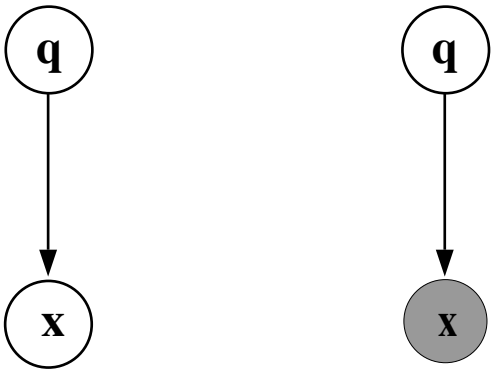
- hidden state $\mathbf{q}$ is a multinomial RV
- output $\mathbf{x}$ is a Gaussian RV
- prior probabilities on hidden states:

$$\pi_i = P(q_i = 1)$$

- class-conditional probabilities:

$$P(\mathbf{x}|q_i = 1) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \exp\{-\frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma_i^{-1}(\mathbf{x} - \mu_i)\}$$

Gaussian mixture models as a graphical model



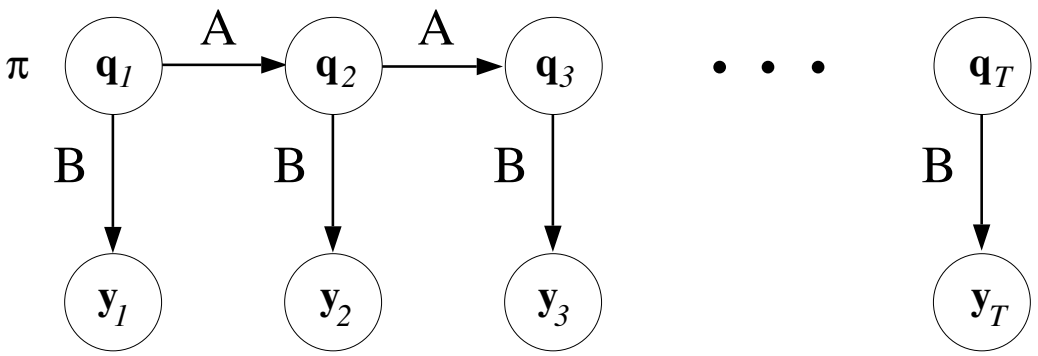
- The inference problem is to calculate the posterior probabilities:

$$
\begin{aligned}
P(q_i = 1|\mathbf{x}) &= \frac{P(\mathbf{x}|q_i = 1)P(q_i = 1)}{P(\mathbf{x})} \\
&= \frac{\frac{\pi_\mathrm{i}}{|\Sigma_\mathrm{i}|^{1=2}} \exp\{-\frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma_i^{-1}(\mathbf{x} - \mu_i)\}}{\Sigma_j \frac{\pi_\mathrm{j}}{|\Sigma_\mathrm{j}|^{1=2}} \exp\{-\frac{1}{2}(\mathbf{x} - \mu_j)^T \Sigma_j^{-1}(\mathbf{x} - \mu_j)\}}
\end{aligned}
$$

- This is a (trivial) example of the rescaling operation on a clique potential

Example: Hidden Markov models

- A hidden Markov model is a popular time series model
- It is a "mixture model with dynamics"



- T time steps
- M states ($\mathbf{q}_t$ is a multinomial RV)
- N outputs ($\mathbf{y}_t$ is a multinomial RV)
- state transition probability matrix A:

$$A = P(\mathbf{q}_{t+1}|\mathbf{q}_t)$$

- emission matrix B:

$$B = P(\mathbf{y}_t|\mathbf{q}_t)$$

- initial state probabilities π:

$$\pi = P(\mathbf{q}_1)$$

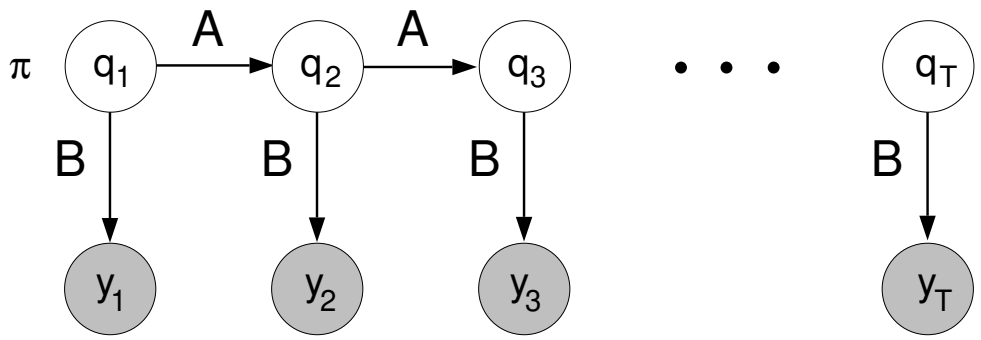
HMM as a graphical model

- Each node has a probability distribution associated with it.
- The graph of the HMM makes conditional independence statements.
- For example,

$$P(\mathbf{q}_{t+1}|\mathbf{q}_t, \mathbf{q}_{t-1}) = P(\mathbf{q}_{t+1}|\mathbf{q}_t)$$

  can be read off the graph as a separation property.

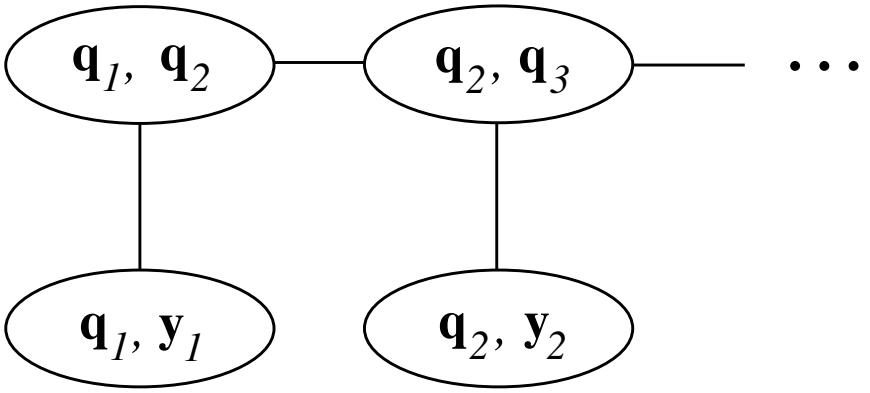
HMM probability calculations



- The time series of $\mathbf{y}_t$ values is the *evidence*
- The *inference* calculation involves calculating the probabilities of the hidden states $\mathbf{q}_t$ given the evidence
- The classical algorithm for doing this calculation is the *forward-backward* algorithm
- The forward-backward algorithm involves:
  - multiplication (conditioning)
  - summation (marginalization) in the lattice.
- It is a special case of the junction tree algorithm (cf. Smyth, et al., 1997)

Is the algorithm efficient?

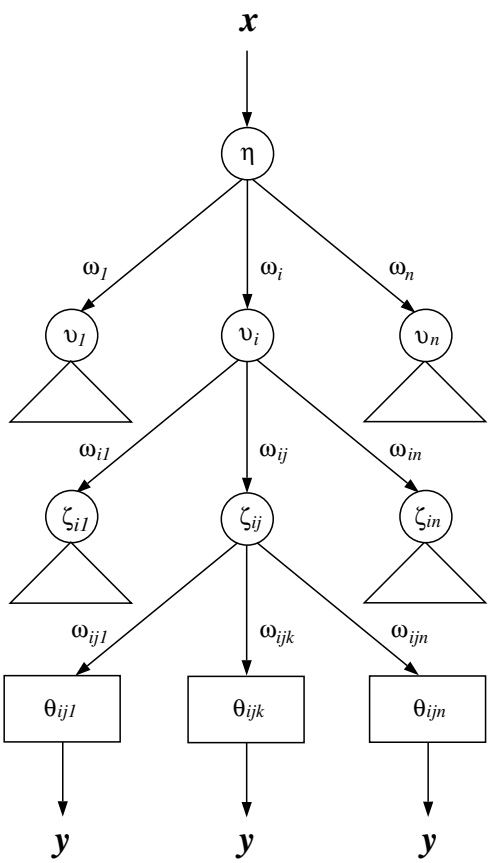
- To answer this question, let's consider the junction tree
- Note that the moralization and triangulation steps are trivial, and we obtain the following junction tree:



- The cliques are no bigger than N^2, thus the marginalization and rescaling required by the junction tree algorithm runs in time $O(N^2)$ per clique
- There are T such cliques, thus the algorithm is $O(N^2T)$ overall

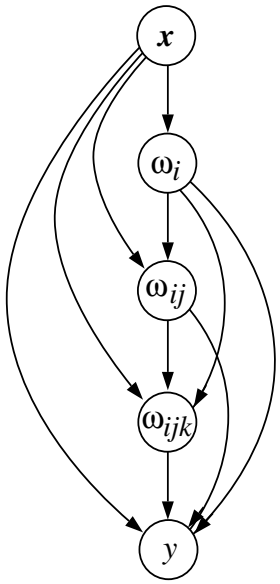
Hierarchical mixture of experts

- The HME is a "soft" decision tree
- The input vector **x** drops through a tree of "gating networks," each of which computes a probability for the branches below it

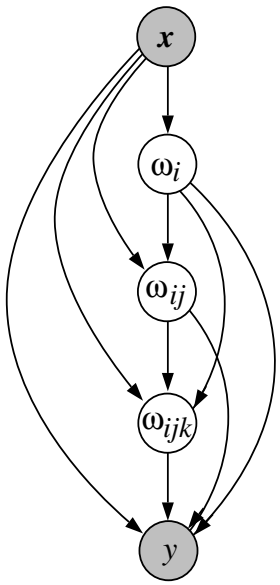


- At the leaves, the "experts" compute a probability for the output **y**
- The overall probability model is a conditional mixture model

Representation as a graphical model



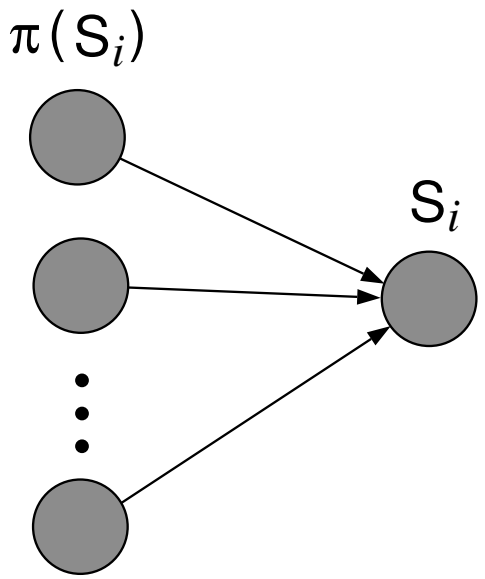
- For learning and inference, the nodes for $\mathbf{x}$ and $\mathbf{y}$ are observed (shaded)



- We need to calculate probabilities of unshaded nodes (E step of EM)

Learning (parameter estimation)

- Consider a single (observed) node S_i with (observed) parents $\pi(S_i)$, with probability $P(S_i|\pi(S_i), \theta_i)$:



- Estimating parameters in this case is a classical statistical problem; we can use maximum likelihood
- I.e., we maximize $\log P(P(S_i|\pi(S_i), \theta_i)$ with respect to θ_i
- In machine learning, this is called "supervised learning"

Learning (cont.)

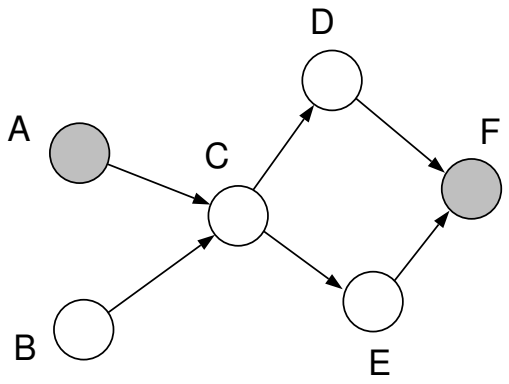
- Consider now a network in which each node is observed
- Let θ represent the vector of all of the parameters θ_i
- The joint probability is a product, thus the log likelihood is a sum:

$$\begin{aligned} \log P(S|\theta) &= \log \prod_i P(S_i|\pi(S_i), \theta) \\ &= \sum_i \log P(S_i|\pi(S_i), \theta_i) \end{aligned}$$

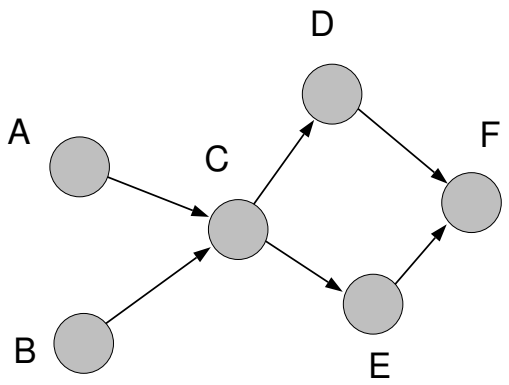
- I.e., the maximization problem decouples into a set of separate maximization problems
- I.e., for completely observed networks, the parameter estimation problem for the network is no more difficult than that for a single node

Learning (cont.)

- When there are hidden variables we use the Expectation-Maximization (EM) algorithm
- The E step of the EM algorithm involves calculating the probabilities of hidden variables given visible variables
  - this is exactly the inference problem!

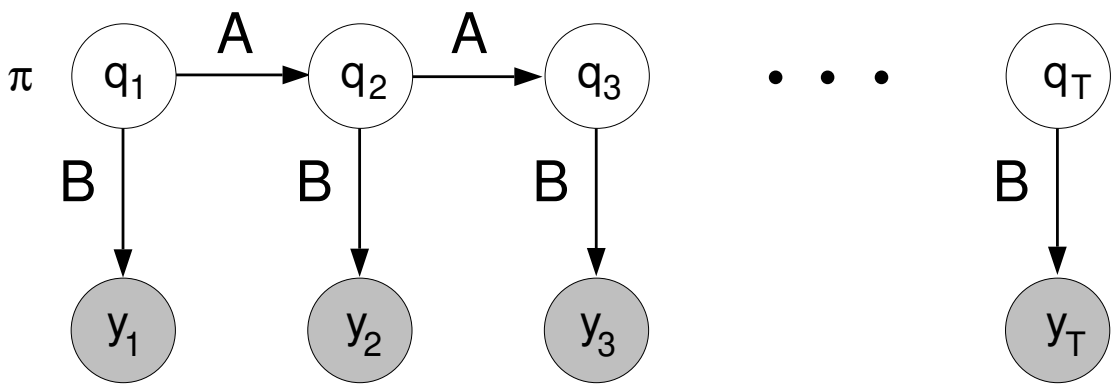


- The M step involves parameter estimation for a fully observed graph



- EM provides a neat link between inference and learning

Example—Hidden Markov models



Problem: Given a sequence of outputs

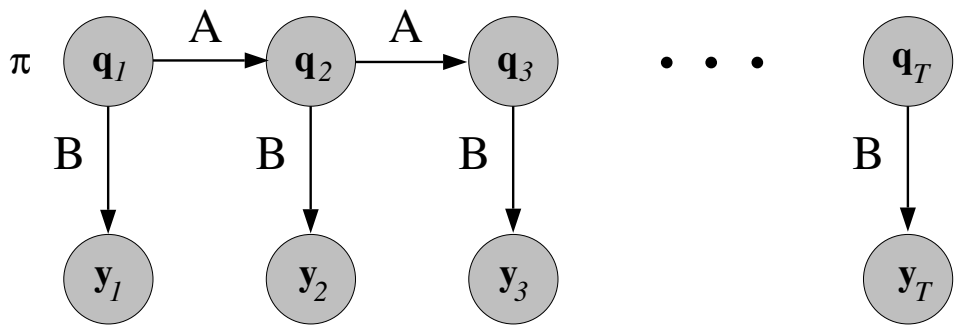
$$\mathbf{y}_{\overline{1,T}} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T\}$$

infer the parameters A, B and π.

- To see how to solve this problem, let's consider a simpler problem

Fully observed Markov models

- Suppose that at each moment in time we know what state the system is in:



- Parameter estimation is *easy* in this case:
  - to estimate the state transition matrix elements, simply keep a running count of the number n_{ij} of times the chain jumps from state i to state j. Then estimate a_{ij} as:

$$\hat{a}_{ij} = \frac{n_{ij}}{\Sigma_j \, n_{ij}}$$

  - to estimate the Gaussian output probabilities, simply record which data points occurred in which states and compute sample means and covariances
  - as for the initial state probabilities, we need multiple output sequences (which we usually have in practice)

HMM parameter estimation

- When the hidden states are not in fact observed (the case we're interested in), we first *estimate* the probabilities of the hidden states
  - this is a straightforward application of the junction tree algorithm (i.e., the forward-backward algorithm)
- We then use the probability estimates instead of the counts in the parameter estimation formulas to get *update* formulas
- This gives us a better model, so we run the junction tree algorithm again to get *better estimates* of the hidden state probabilities
- And we *iterate* this procedure

EM proof of convergence

- Divide the nodes S into visible nodes V and hidden nodes H
- We want to maximize $\log P(V|\theta)$ with respect to θ, but it's too hard
- If the hidden nodes were observed, we could work with $\log P(V, H|\theta)$ instead
- We can't do that, but we *can* work with an *averaged* version of $\log P(V, H|\theta)$

EM proof (cont.)

- We will introduce an averaging function $Q(H|V)$.
- For any such function we have the following bound:

$$\begin{aligned}
\log P(V|\theta) &= \log \Sigma_H P(H,V|\theta) \\
&= \log \Sigma_H Q(H|V) \cdot \frac{P(H,V|\theta)}{Q(H|V)} \\
&\geq \Sigma_H Q(H|V) \log \left[\frac{P(H,V|\theta)}{Q(H|V)} \right] \\
&= \Sigma_H Q(H|V) \log P(H,V|\theta) \\
&\quad -\Sigma_H Q(H|V) \log Q(H|V)
\end{aligned}$$

- The right hand side depends on the choice of $Q(H|V)$ and the choice of θ
- We can maximize by *coordinate ascent*

EM proof (cont.)

- Maximizing with respect to $Q(H|V)$:
  - letting $Q(H|V) = P(H|V, \theta)$ gives $P(V|\theta)$ on the right hand side
  - i.e., we maximize the bound wrt Q by doing inference!
  - this is the E step
- Maximizing with respect to θ:
  - only the first term in the bound depends on θ
  - so we're maximizing the averaged likelihood, where the inference algorithm does the averaging
  - this is the M step

CONCLUSIONS (PART I)

- Graphical models provide a general formalism for putting together graphs and probabilities
- The framework allows a rapprochement between inference and learning
- An elegant general inference algorithm is available
- But we'd better be concerned about combinatorial explosion...

INTRACTABLE GRAPHICAL MODELS

- There are a number of examples of graphical models in which exact inference is efficient:
  - chain-like graphs
  - tree-like graphs
- However, there are also a number of examples of graphical models in which exact inference can be (hopelessly) inefficient:
  - dense graphs
  - layered graphs
  - coupled graphs
- A variety of methods are available for *approximate* inference in such situations:
  - Markov chain Monte Carlo (stochastic)
  - variational methods (deterministic)

Computational complexity of exact computation

- passing messages requires marginalizing and scaling the clique potentials
- thus the time required is exponential in the number of variables in the largest clique
- good triangulations yield small cliques
  - but the problem of finding an optimal triangulation is hard (#P in fact)
- in any case, the triangulation is "off-line"; our concern is generally with the "on-line" problem of message propagation

Probability theory and computation

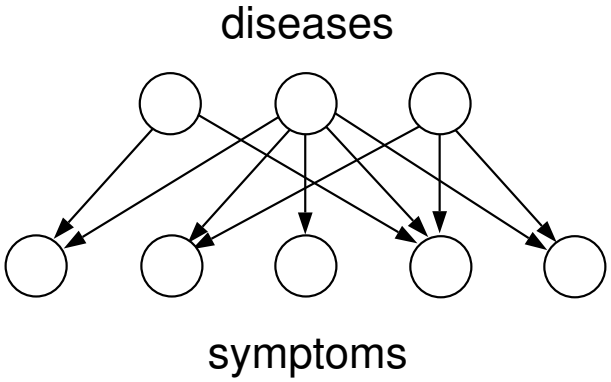
- For sparse graphs, probabilistic computation is often quite feasible (cf. Pearl)
  - a message-passing algorithm can do the job
- For dense graphs, probabilistic computation is once again feasible (cf. statistical mechanics, information theory)
  - laws of large numbers come to our rescue
- *Complex graphs can be probabilistically simple*
  - how can we exploit this fact?

First, let's consider some intractable architectures

Quick Medical Reference (QMR)

(University of Pittsburgh)

- 600 diseases, 4000 symptoms
- arranged as a bipartite graph

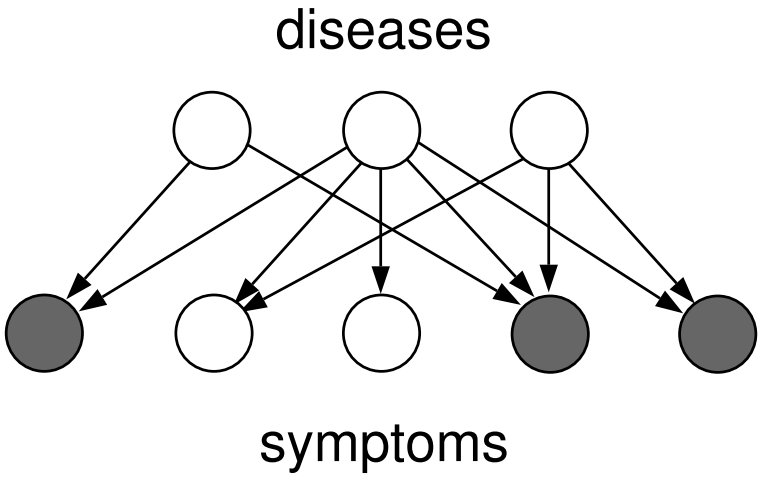


- Node probabilities $P(symptom_i|diseases)$ were obtained from an expert, under a noisy-OR model
- Want to do diagnostic calculations:

$$P(diseases|findings)$$

- Current methods (exact and Monte Carlo) are infeasible

QMR (cont.)



- "noisy-OR" parameterization:

$$P(f_i = 0|d) = (1 - q_{i0}) \prod_{j \in pa_i} (1 - q_{ij})^{d_j}$$

- rewrite in an exponential form:

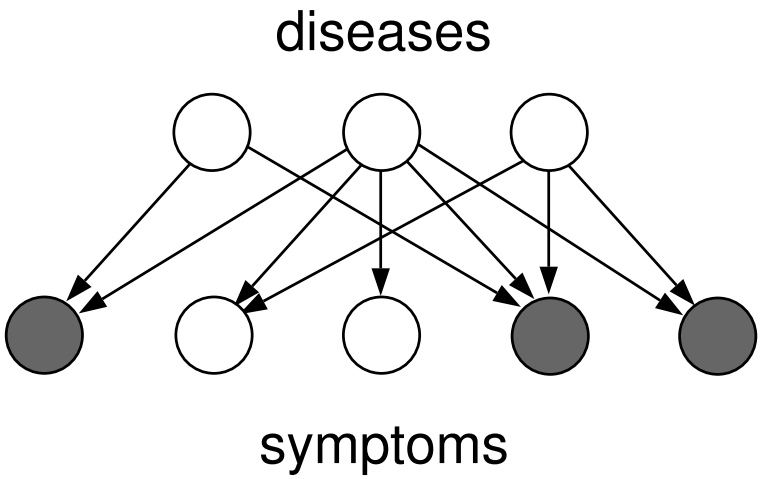
$$P(f_i = 0|d) = e^{-\theta_{i0} - \sum_{j \in pa_i} \theta_{ij} d_j}$$

  where $\theta_{ij} \equiv -\log(1 - q_{ij})$

- probability of positive finding:

$$P(f_i = 1|d) = 1 - e^{-\theta_{i0} - \sum_{j \in pa_i} \theta_{ij} d_j}$$

QMR (cont.)



- Joint probability:

$$\begin{aligned} P(f,d) &= P(f|d)P(d) \\ &= \left[\prod_i P(f_i|d)\right]\left[\prod_j P(d_j)\right] \\ &= \left[\left(1-e^{-\theta_{10}-\sum_{j\in pa_1}\theta_{1j}d_j}\right)\left(1-e^{-\theta_{20}-\sum_{j\in pa_2}\theta_{2j}d_j}\right)\right. \\ &\quad \left.\cdots\left(1-e^{-\theta_{k0}-\sum_{j\in pa_k}\theta_{kj}d_j}\right)\right]\left[\prod_j P(d_j)\right] \end{aligned}$$

- Positive findings couple the disease nodes
- Median size of maximal clique is 151 nodes

Multilayer neural networks as graphical models

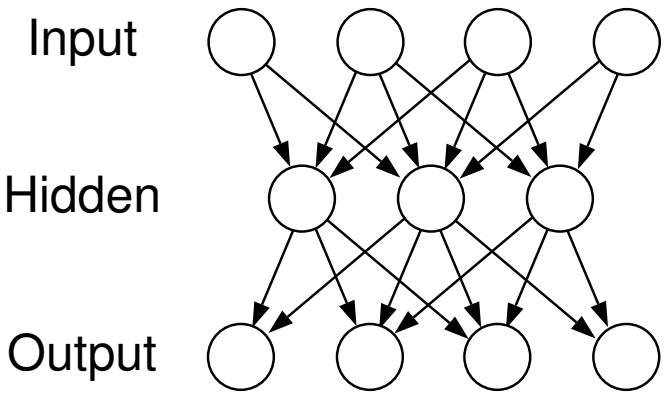
(cf. Neal, 1992; Saul, Jaakkola, & Jordan, 1996)

- Associate with node i a latent binary variable whose conditional probability is given by:

$$P(S_i = 1 | S_{pa_i}) = \frac{1}{1 + e^{-\sum_{j \in pa_i} \theta_{ij} S_j - \theta_{i0}}}$$

  where pa_i indexes the parents of node i

- A multilayer neural network with logistic hidden units:

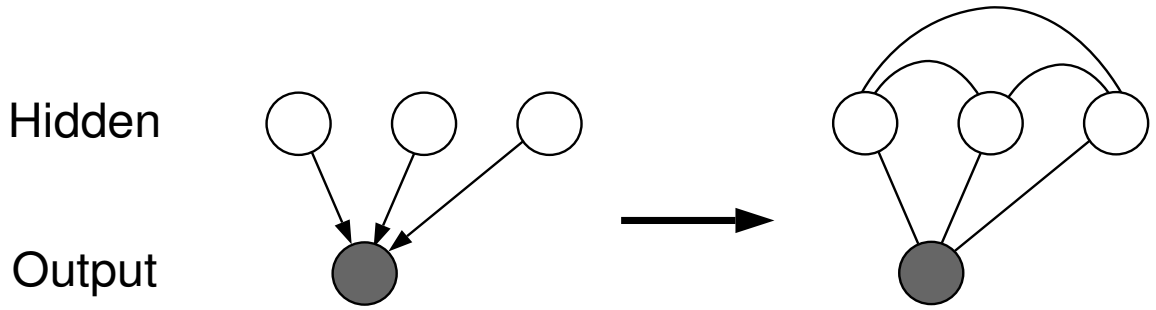


- has a joint distribution that is a product of logistic functions:

$$P(S) = \prod_i \left[\frac{e^{\left(\sum_{j \in pa_i} \theta_{ij} S_j + \theta_{i0}\right) S_i}}{1 + e^{\sum_{j \in pa_i} \theta_{ij} S_j + \theta_{i0}}} \cdot \right]$$

Complexity of neural network inference

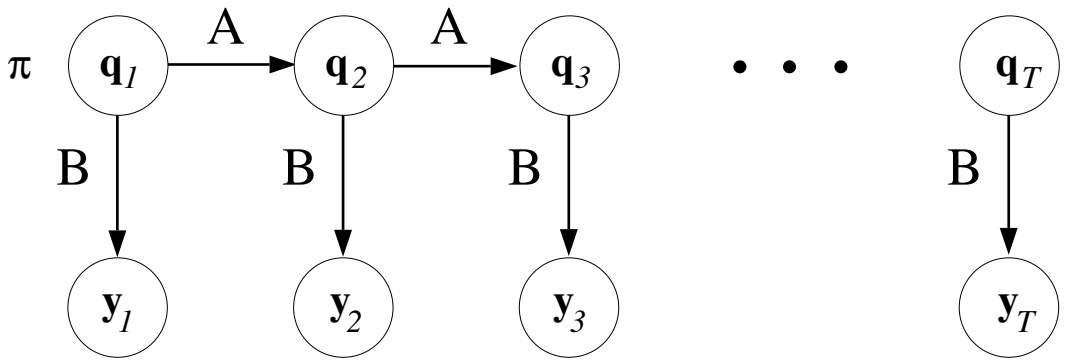
- When an output node is known (as it is during learning), moralization links the hidden units:



- So inference scales at least as badly as $O(2^N)$
- And triangulation adds even more links

Hidden Markov models

- Recall the hidden Markov model:



- T time steps
- M states ($\mathbf{q}_t$ is a multinomial RV)
- N outputs ($\mathbf{y}_t$ is a multinomial RV)
- state transition probability matrix A:

$$A = P(\mathbf{q}_{t+1}|\mathbf{q}_t)$$

- emission matrix B:

$$B = P(\mathbf{y}_t|\mathbf{q}_t)$$

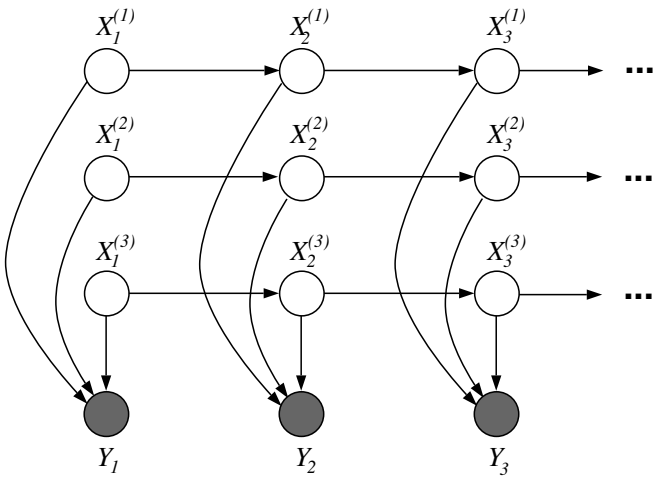
- initial state probabilities π:

$$\pi = P(\mathbf{q}_1)$$

Factorial hidden Markov models

(cf. Williams & Hinton, 1991; Ghahramani & Jordan, 1997)

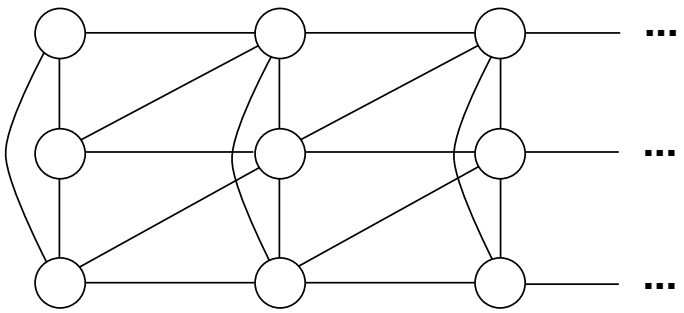
- Imagine that a time series is created from a set of M loosely-coupled underlying mechanisms
- Each of these mechanisms may have their own particular dynamic laws, and we want to avoid merging them into a single "meta-state," with a single transition matrix
  - avoid choosing a single time scale
  - avoid over-parameterization
- Here is the graphical model that we would like to use:



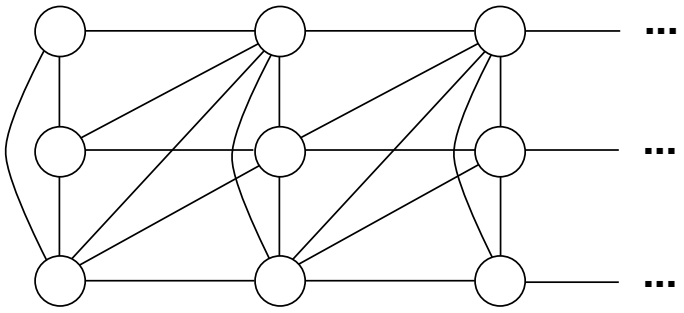
- When we triangulate do we get an efficient structure?

Triangulation?

- Unfortunately, the following graph is not triangulated:



- Here is a triangulation:

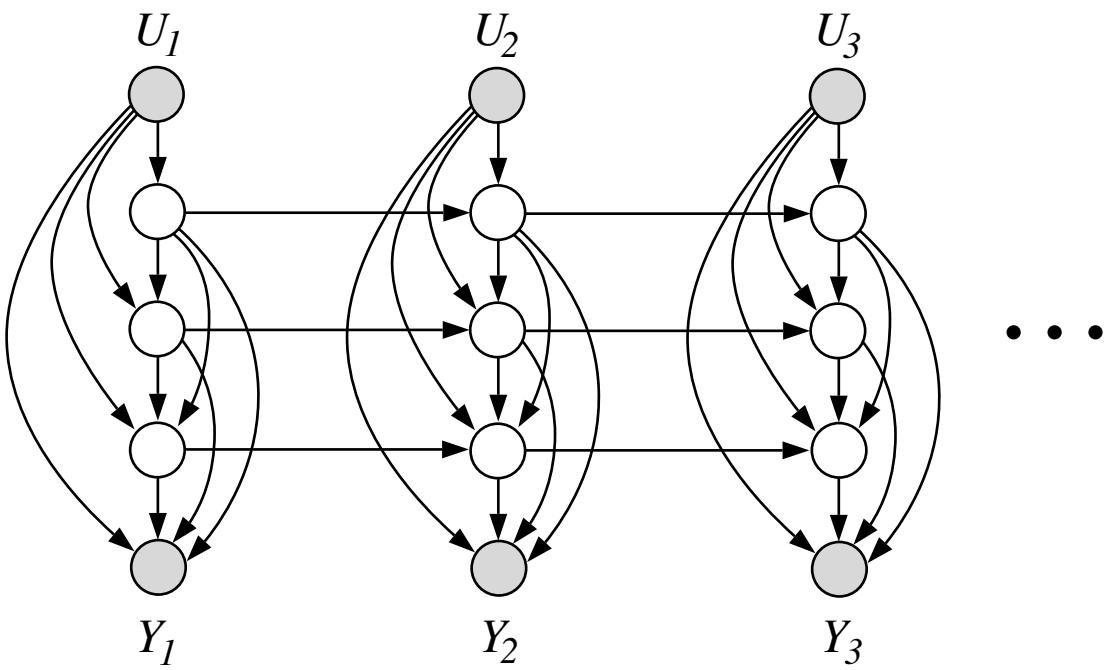


- We have created cliques of size N^4. The junction tree algorithm is not efficient for factorial HMMs.

Hidden Markov decision trees

(Jordan, Ghahramani, & Saul, 1997)

- We can combine decision trees with factorial HMMs
- This gives a "command structure" to the factorial representation



- Appropriate for multiresolution time series
- Again, the exact calculation is intractable and we must use variational methods

Markov chain Monte Carlo (MCMC)

- Consider a set of variables $S = \{S_1, S_2, \ldots, S_N\}$
- Consider a joint probability density $P(S)$
- We would like to calculate statistics associated with $P(S)$:
  - marginal probabilities, e.g., $P(S_i)$, or $P(S_i, S_j)$
  - conditional probabilities, e.g., $P(H|V)$
  - likelihoods, i.e., $P(V)$
- One way to do this is to generate samples from $P(S)$ and compute empirical statistics
  - but generally it is hard to see how to sample from $P(S)$
- We set up a simple Markov chain whose *equilibrium distribution* is $P(S)$

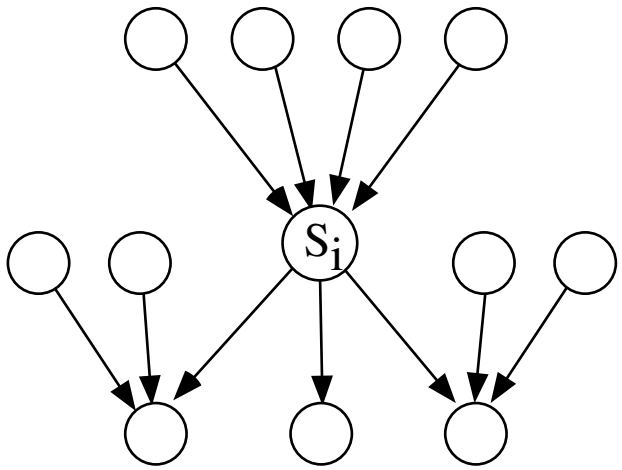
Gibbs sampling

- Gibbs sampling is a widely-used MCMC method
- Recall that we have a set of variables $S = \{S_1, S_2, \ldots, S_N\}$
- We set up a Markov chain as follows:
  - initialize the S_i to arbitrary values
  - choose i randomly
  - sample from $P(S_i|S \backslash S_i)$
  - iterate
- It is easy to prove that this scheme has $P(S)$ as its equilibrium distribution

- How to do Gibbs sampling in graphical models?

Markov blankets

- The *Markov blanket* of node S_i is the minimal set of nodes that renders S_i conditionally independent of all other nodes
- For undirected graphs, the Markov blanket is just the set of neighbors
- For directed graphs, the Markov blanket is the set of parents, children and co-parents:



- The conditional $P(S_i|S\backslash S_i)$ needed for Gibbs sampling is formed from the product of the conditional probabilities associated with S_i and each of its children
- This implies that the conditioning set needed to form $P(S_i|S\backslash S_i)$ is the Markov blanket of S_i (which is usually much smaller than $S\backslash S_i$)

Issues in Gibbs sampling

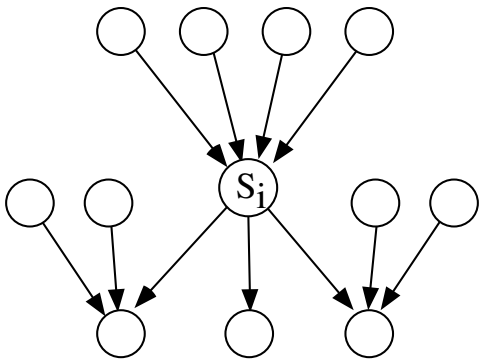
- Gibbs sampling is generally applicable
- It can be very slow
  - i.e., the time to converge can be long
- Moreover it can be hard to diagnose convergence
  - but reasonable heuristics are available
- It is often possible to combine Gibbs sampling with exact methods

Variational methods

- Variational methods are deterministic approximation methods
  - perhaps unique among deterministic methods in that they tend to work best for *dense* graphs
- They have some advantages compared to MCMC methods
  - they can be much faster
  - they yield upper and lower bounds on probabilities
- And they have several disadvantages
  - they are not as simple and widely applicable as MCMC methods
  - they require more art (thought on the part of the user) than MCMC methods
- But both variational approaches and MCMC approaches are evolving rapidly
- Note also that they can be combined (and can be combined with exact methods)

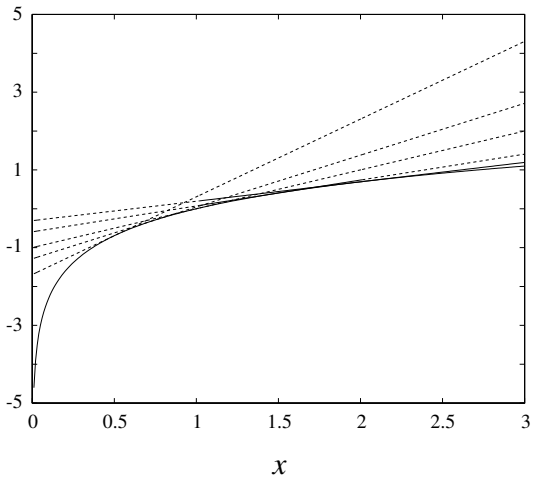
Introduction to variational methods

- Intuition—in a dense graph, each node is subject to many stochastic influences from nodes in its Markov blanket
  - laws of large numbers



- We want to exploit such averaging where applicable and where needed, while using exact inference algorithms on tractable "backbones"

Example of a variational transformation

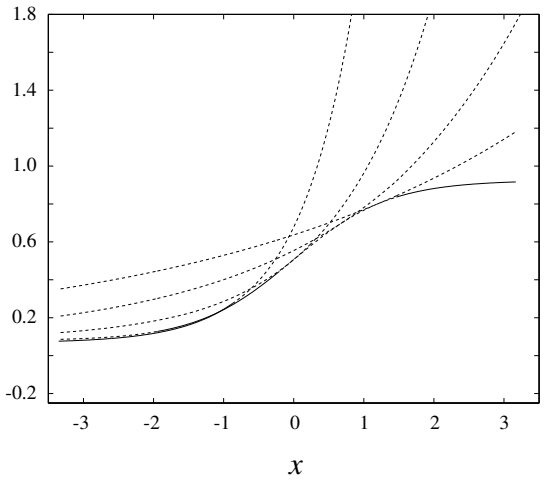


- $$\begin{aligned} \log(x) &= \min_{\lambda}\{\lambda x - \log\lambda - 1\} \\ &\leq \lambda x - \log\lambda - 1 \end{aligned}$$

  – λ is a *variational parameter*

  – transforms a nonlinearity into a linearity

Example of a variational transformation



- $$g(x) = \frac{1}{1+e^{-x}} = \min_{\lambda}\{e^{\lambda x - H(\lambda)}\} \leq e^{\lambda x - H(\lambda)}$$

  – where $H(\cdot)$ is the binary entropy

  – a nonlinear function now a simple exponential

  – cf. "tilted distributions"

Convex duality approach

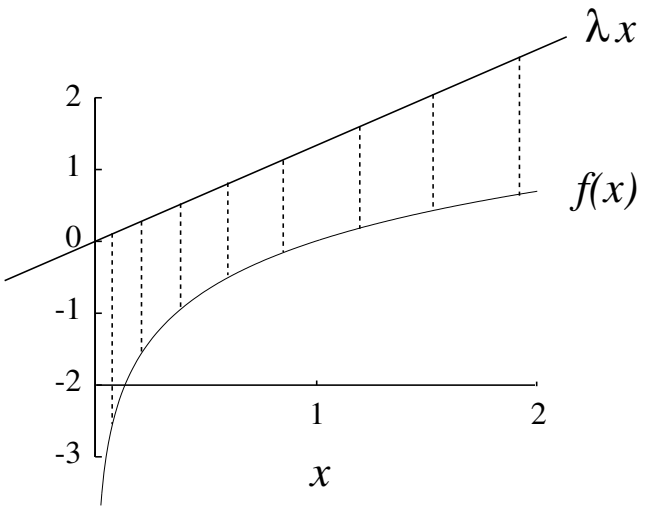
(Jaakkola & Jordan, NIPS '97)

- for concave $f(x)$:

$$f(x) = \min_{\lambda} \left\{ \lambda^T x - f^*(\lambda) \right\}$$
$$f^*(\lambda) = \min_{x} \left\{ \lambda^T x - f(x) \right\}$$

- yields bounds:

$$f(x) \leq \lambda^T x - f^*(\lambda)$$
$$f^*(\lambda) \leq \lambda^T x - f(x)$$



- (lower bounds obtained from convex $f(x)$)

Variational transformations and inference

- Two basic approaches—*sequential* and *block*
  - we discuss the sequential approach first and return to the block approach later

- In the *sequential* approach, we introduce variational transformations sequentially, node by node
  - this yields a sequence of increasingly simple graphical models
  - every transformation introduces a new variational parameter, which we (lazily) put off evaluating until the end
  - eventually we obtain a model that can be handled by exact techniques, at which point we stop introducing transformations

Variational QMR

(Jaakkola & Jordan, 1997)

- Recall the noisy-OR probability of a positive finding:

$$P(f_i = 1|d) = 1 - e^{-\theta_{i0} - \Sigma_{j \in pa_i} \theta_{ij} d_j}$$

- The logarithm of this function is concave, thus we can utilize convex duality
  - evaluating the conjugate function, we obtain:

$$f^*(\lambda) = -\lambda \ln \lambda + (\lambda + 1) \ln(\lambda + 1)$$

- Thus we upper bound the probability of a positive finding:

$$\begin{aligned} P(f_i = 1|d) &\leq e^{\lambda_i(\theta_{i0} + \Sigma_{j \in pa_i} \theta_{ij} d_j) - f^*(\lambda_i)} \\ &= e^{\lambda_i \theta_{i0} - f^*(\lambda_i)} \prod_{j \in pa_i} \left[e^{\lambda_i \theta_{ij}} \right]^{d_j} \end{aligned}$$

- This is a factorized form; it effectively changes the "priors" $P(d_j)$ by multiplying them by $e^{\lambda_i \theta_{ij}}$ and delinking the i^{th} node from the graph

Variational QMR (cont.)

- Recall the joint distribution:

$$P(f,d) = \left[\left(1 - e^{-\theta_{10} - \sum_{j \in pa_1} \theta_{1j} d_j}\right)\left(1 - e^{-\theta_{20} - \sum_{j \in pa_2} \theta_{2j} d_j}\right) \cdots \left(1 - e^{-\theta_{k0} - \sum_{j \in pa_k} \theta_{kj} d_j}\right)\right] \left[\prod_j P(d_j)\right]$$

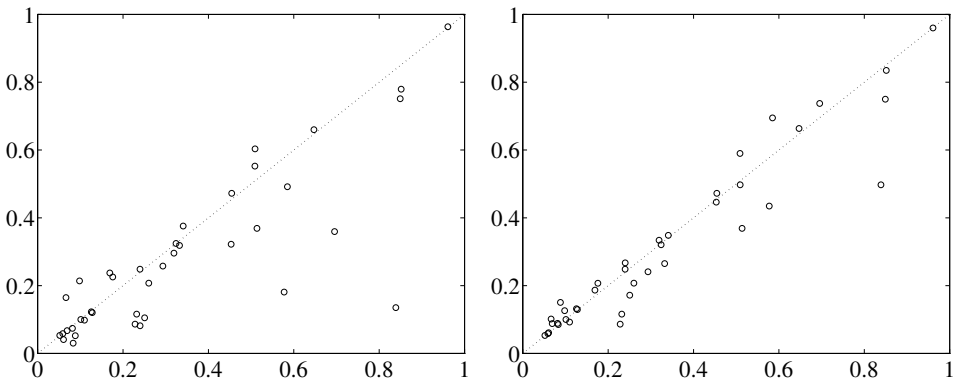
- After a variational transformation:

$$P(f,d) \leq \left[\left(1 - e^{-\theta_{10} - \sum_{j \in pa_1} \theta_{1j} d_j}\right)\left(1 - e^{-\theta_{20} - \sum_{j \in pa_2} \theta_{2j} d_j}\right) \cdots \left(e^{\lambda_k \theta_{k0} - f^*(\lambda_k)} \prod_{j \in pa_k} \left[e^{\lambda_k \theta_{kj}}\right]^{d_j}\right)\right] \left[\prod_j P(d_j)\right]$$

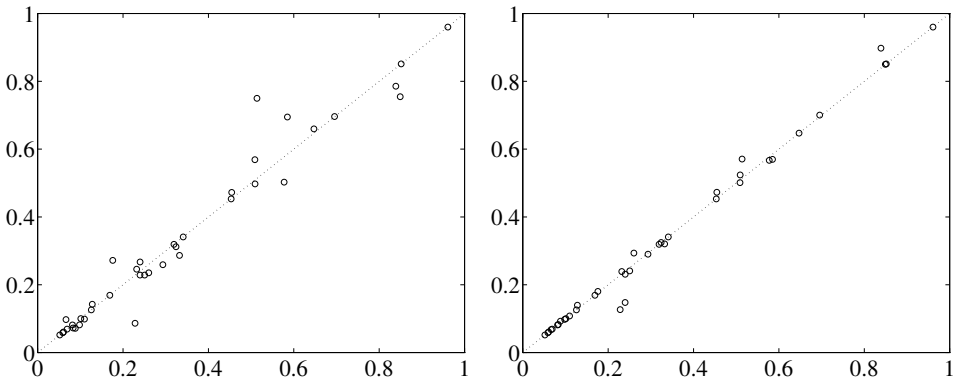
- Use a greedy method to introduce the variational transformations
  - choose the node that is estimated to yield the most accurate transformed posterior
- Optimize across the variational parameters λ_i
  - this turns out to be a convex optimization problem

QMR results

- We studied 48 clinicopathologic (CPC) cases
- 4 of these CPC cases had less than 20 positive findings; for these we calculated the disease posteriors exactly
- Scatterplots of exact vs. variational posteriors for (a) 4 positive findings treated exactly, (b) 8 positive findings treated exactly:

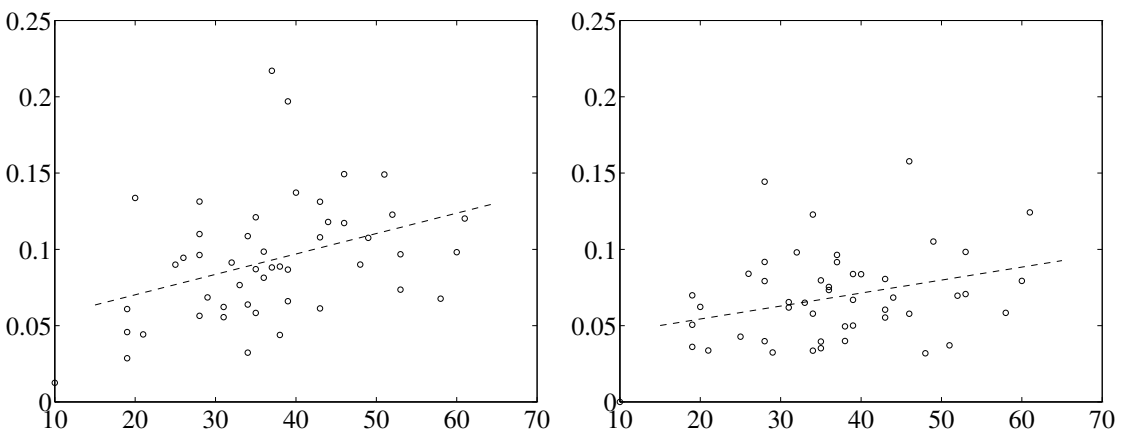


- Scatterplots of exact vs. variational posteriors for (a) 12 positive findings treated exactly, (b) 16 positive findings treated exactly:



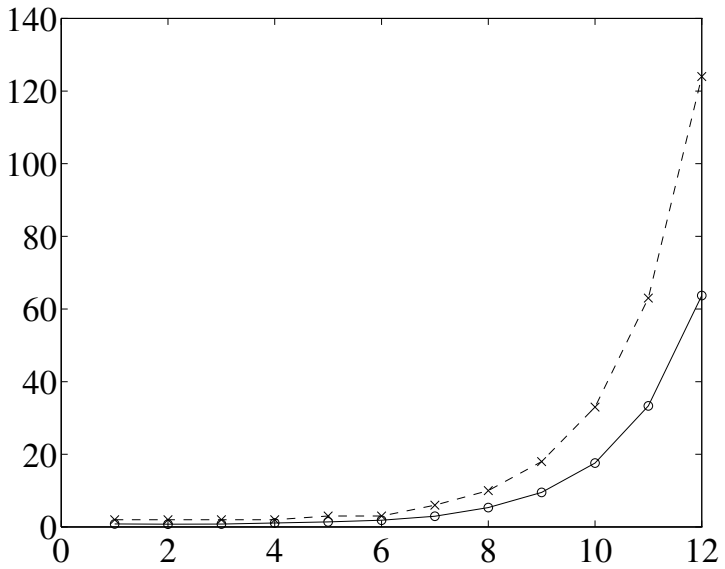
QMR results (cont.)

- For the remaining 44 CPC cases, we had no way to calculate the gold standard
- Thus we assessed the variational accuracy via a sensitivity estimate
  - we calculated the squared difference in posteriors when a particular node was treated exactly or variationally transformed, and averaged across nodes
  - a small value of this average suggests that we have the correct posteriors
  - (we validated this surrogate on the 4 cases for which we could do the exact calculation)
- Sensitivity estimates vs. number of positive findings when (a) 8 positive findings treated exactly, (b) 12 positive findings treated exactly:



QMR results (cont.)

- Timing results in seconds (Sparc 10) as a function of the number of positive findings treated exactly (solid line—average across CPC cases; dashed line—maximum across CPC cases):

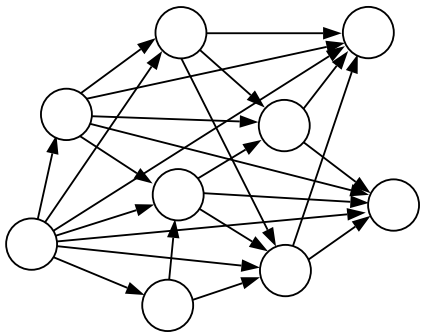


A cautionary note

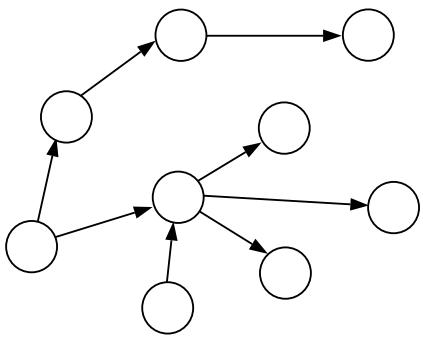
- these disease marginals are based on the upper variational distribution, which appears tight
- this distribution is guaranteed to upper bound the *likelihood*
- to obtain direct upper and lower bounds on the disease *marginals*, which are conditional probabilities, we need upper *and* lower bounds on the likelihood
- the lower bounds we obtain in our current implementation, however, are not tight enough
- thus, although the marginals we report appear to yield good approximations empirically, we cannot guarantee that they bound the true marginals

Variational transformations and inference

- Two basic approaches—*sequential* and *block*
- The block approach treats the approximation problem as a global optimization problem
- Consider a dense graph characterized by a joint distribution $P(H, V|\theta)$



- We remove links to obtain a simpler graph characterized by a conditional distribution $Q(H|V, \theta, \mu)$



- (Examples will be provided below...)

Variational inference (cont.)

(Dayan, et al., 1995; Hinton, et al., 1995; Saul & Jordan, 1996)

- $Q(H|V,\theta,\mu)$ has extra degrees of freedom, given by *variational parameters* μ_i
  - we can think of these as being obtained by a sequence of variational transformations applied to the nodes
  - but we now want to take a more global view

- Choose μ_i so as to minimize

$$KL(Q\|P) = \sum_H Q(H|V,\theta,\mu) \log \frac{Q(H|V,\theta,\mu)}{P(H|V,\theta)}$$

- We will show that this yields a lower bound on the probability of the evidence (the likelihood)

- Minimizing the KL divergence requires us to compute averages under the Q distribution; we must choose Q so that this is possible
  - i.e., we choose our simplified graph so that it is amenable to exact methods

Variational inference (cont.)

- The fact that this is a lower bound follows from Jensen's inequality

$$\begin{aligned} \log P(V) &= \log \Sigma_H P(H,V) \\ &= \log \Sigma_H Q(H|V) \cdot \frac{P(H,V)}{Q(H|V)} \\ &\geq \Sigma_H Q(H|V) \log \left[\frac{P(H,V)}{Q(H|V)}\right] \end{aligned}$$

- The difference between the left and right hand side is the KL divergence:

$$KL(Q||P) = \Sigma_H Q(H|V) \log \left[\frac{Q(H|V)}{P(H|V)}\right]$$

  which is positive; thus we have a lower bound

Linking the two approaches

(Jaakkola, 1997)

- The block approach can be derived within the convex duality framework

$$f(x) \geq \lambda^T x - f^*(\lambda)$$

  - treat the distribution $Q(H|V, \theta, \mu)$ as λ; a vector-valued variational parameter (one value for each configuration H)
  - the argument x becomes $\log P(H, V|\theta)$; also a vector-valued variable (one value for each configuration H)
  - the function $f(x)$ becomes $\log P(V|\theta)$
  - it turns out that the conjugate function $f^*(x)$ is the negative entropy function

- Thus convex duality yields:

$$\log P(V) \geq \sum_H Q(H|V) \log P(H, V) - \sum_H Q(H|V) \log Q(H|V)$$

  which is the bound derived earlier from Jensen's inequality

Learning via variational methods

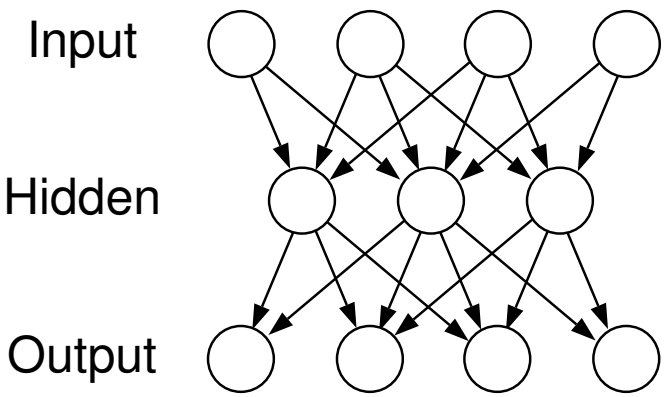
(Neal & Hinton, 1998)

- MAP parameter estimation for graphical models:
  - the EM algorithm is a general method for MAP estimation
  - "inference" is the E step of EM for graphical models (calculate $P(H|V)$ to "fill in" the hidden values)
  - variational methods provide an approximate E step
  - more specifically we increase the lower bound on the likelihood at each iteration

Neural networks and variational approximations

(Saul, Jaakkola, & Jordan, 1996)

- A multilayer neural network with logistic hidden units:



has a joint distribution that is a product of logistic functions:

$$P(H, V|\theta) = \prod_i \left[\frac{e^{\left(\sum_{j \in pa_i} \theta_{ij} S_j + \theta_{i0}\right) S_i}}{1 + e^{\sum_{j \in pa_i} \theta_{ij} S_j + \theta_{i0}}} . \right]$$

- The simplest variational approximation, which we will refer to as a *mean field* approximation considers the factorized approximation:

$$Q(H|V, \mu) = \Pi_{i \in H} \ \mu_i^{S_i} (1 - \mu_i)^{1 - S_i}.$$

Division of labor

- The KL bound has two basic components:

$$\begin{pmatrix} variational \\ entropy \end{pmatrix} = -\Sigma_H \; Q(H|V) \log Q(H|V)$$

$$\begin{pmatrix} variational \\ energy \end{pmatrix} = -\Sigma_H \; Q(H|V) \log P(H,V)$$

- What we need is the difference:

$$\log P(V|\theta) \geq \begin{pmatrix} variational \\ entropy \end{pmatrix} - \begin{pmatrix} variational \\ energy \end{pmatrix}$$

Maximizing the lower bound

- We find the best approximation by varying $\{\mu_i\}$ to minimize $KL(Q||P)$.

- This amounts to maximizing the lower bound:

$$\log P(V|\theta) \geq \begin{pmatrix} variational \\ entropy \end{pmatrix} - \begin{pmatrix} variational \\ energy \end{pmatrix}.$$

- Abuse of notation: we define *clamped* values (0 or 1) for the instantiated nodes,

$$\mu_i = S_i \text{ for } i \in V.$$

Variational entropy

- The variational entropy is:

$$-\Sigma_H \; Q(H|V) \log Q(H|V)$$

- Our *factorized approximation* is:

$$Q(H|V) = \Pi_i \; \mu_i^{S_i} (1 - \mu_i)^{1 - S_i},$$

- The *joint entropy* is the sum of the individual unit entropies:

$$-\Sigma_i \; \left[\mu_i \log \mu_i + (1 - \mu_i) \log (1 - \mu_i) \right].$$

Variational energy

- The variational energy is:

$$-\Sigma_H \; Q(H|V) \log P(H,V)$$

- In sigmoid networks:

$$\begin{aligned} \log P(H,V) \; = \; & -\sum_{ij} \theta_{ij} S_i S_j \\ & + \sum_i \log \left[1 + e^{\Sigma_j \, \theta_{ij} S_j} \right] . \end{aligned}$$

- The first term are common to undirected networks. *The last term is not.*

- Averaging over $Q(H|V)$ gives:

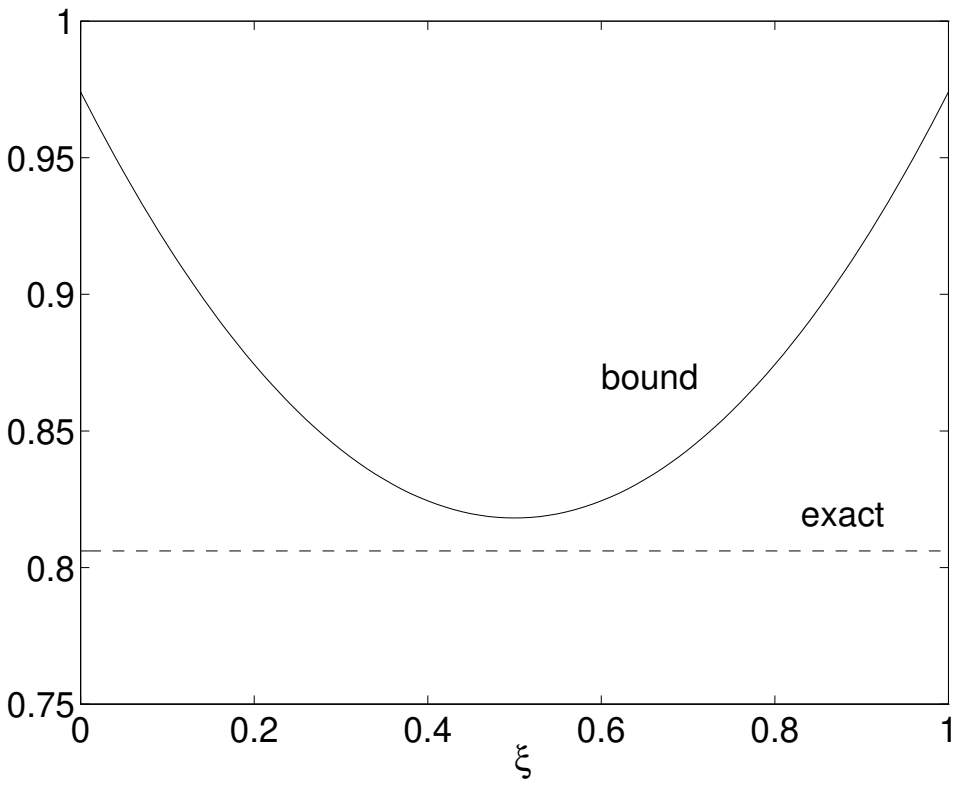
$$-\sum_{ij} \theta_{ij} \mu_i \mu_j \sum_i \left\langle \log \left[1 + e^{\Sigma_j \, \theta_{ij} S_j} \right] \right\rangle$$

- The last term is intractable, so (again) we use Jensen's inequality to obtain an upper bound...

- **Lemma** (Seung): for any random variable z, and any real number ξ:

$$\langle \log[1 + e^z] \rangle \leq \xi \langle z \rangle + \log \left\langle e^{-\xi z} + e^{(1-\xi)z} \right\rangle .$$

- **Ex**: z is Gaussian with zero mean and unit variance.



- **Ex**: z is the sum of weighted inputs to S_i;

$$z = \sum_j \theta_{ij} S_j + h_i.$$

The lemma provides an upper bound on the intractable terms in the variational energy:

$$\sum_i \left\langle \log \left[1 + e^{\sum_j \theta_{ij} S_j} \right] \right\rangle$$

Variational (mean field) equations

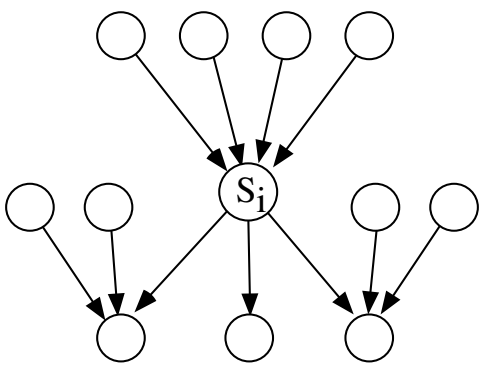
- The bound on the log-likelihood,

$$\log P(V|\theta) \geq \left(\begin{array}{c} variational \\ entropy \end{array} \right) - \left(\begin{array}{c} variational \\ energy \end{array} \right),$$

  is valid for any setting of the variational parameters, $\{\mu_i\}$

- The optimal $\{\mu_i\}$ are found by solving the variational equations:

$$\mu_i = \sigma \left(\sum_j [\theta_{ij}\mu_j + \theta_{ji}(\mu_j - \xi_j) - K_{ji}] \right)$$

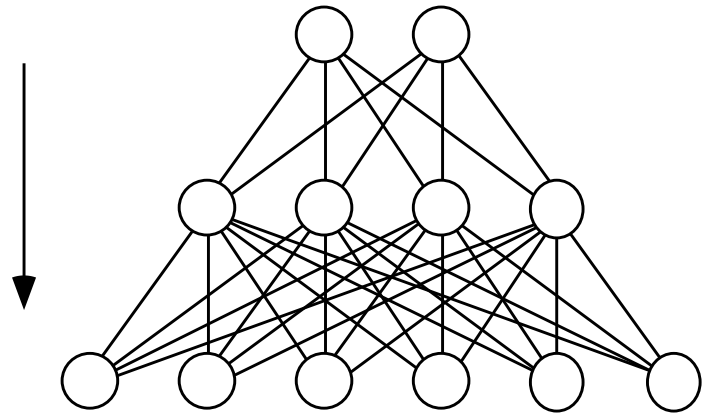


- The effective input to S_i is composed of terms from its *Markov blanket*

Numerical experiments

- For small networks, the variational bound can be compared to the true likelihood obtained by exact enumeration.

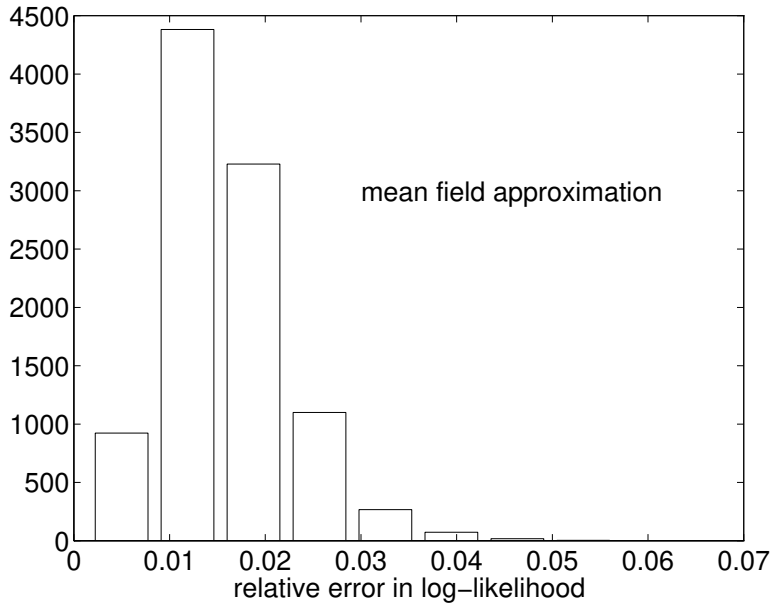
.



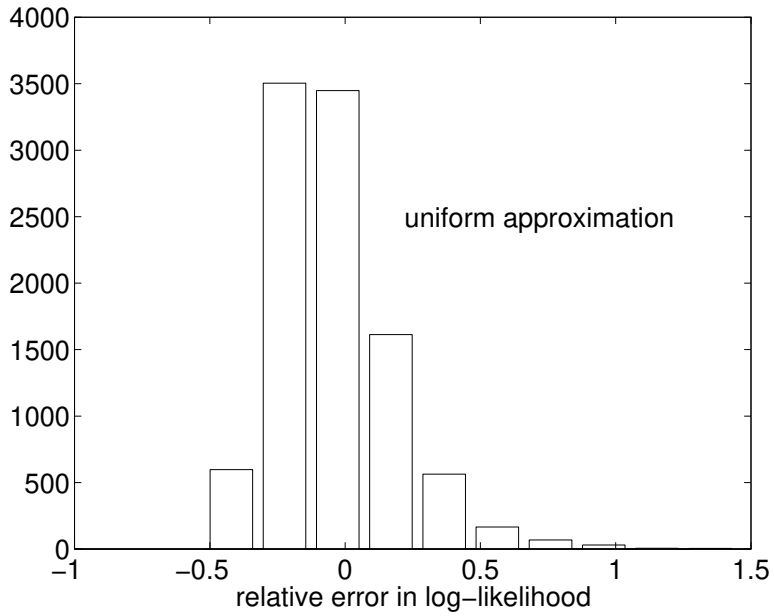
- We considered the event that all the units in the bottom layer were inactive.

- This was done for 10000 random networks whose weights and biases were uniformly distributed between -1 and 1.

- *Mean field approximation*:

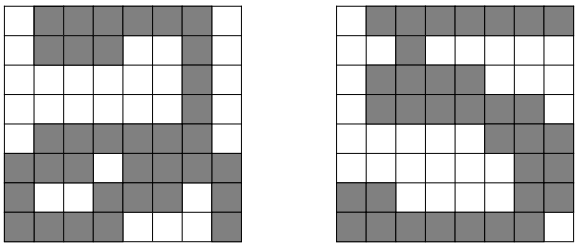


- *Uniform approximation*:



Digit recognition

- Images:



- Confusion matrix:

	0	1	2	3	4	5	6	7	8	9
0	388	2	2	0	1	3	0	0	4	0
1	0	393	0	0	0	1	0	0	6	0
2	1	2	376	1	3	0	4	0	13	0
3	0	2	4	373	0	12	0	0	6	3
4	0	0	2	0	383	0	1	2	2	10
5	0	2	1	13	0	377	2	0	4	1
6	1	4	2	0	1	6	386	0	0	0
7	0	1	0	0	0	0	0	388	3	8
8	1	9	1	7	0	7	1	1	369	4
9	0	4	0	0	0	0	0	8	5	383

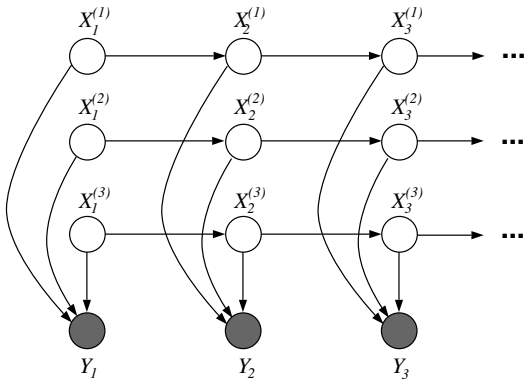
- Comparative results:

algorithm	test error
nearest neighbor	6.7
back propagation	5.6
Helmholtz machine	4.8
variational	4.6

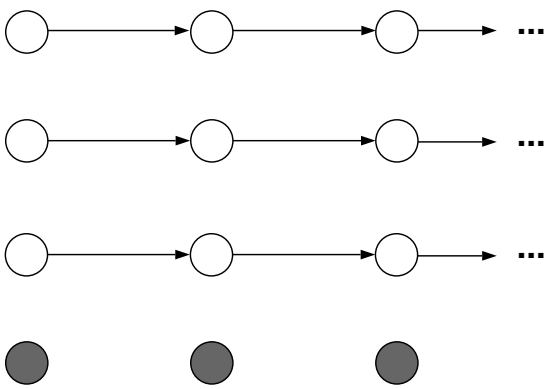
Example—Factorial HMM

(Ghahramani & Jordan, 1997)

- Recall the factorial hidden Markov model, which yielded intractably large cliques when triangulated:

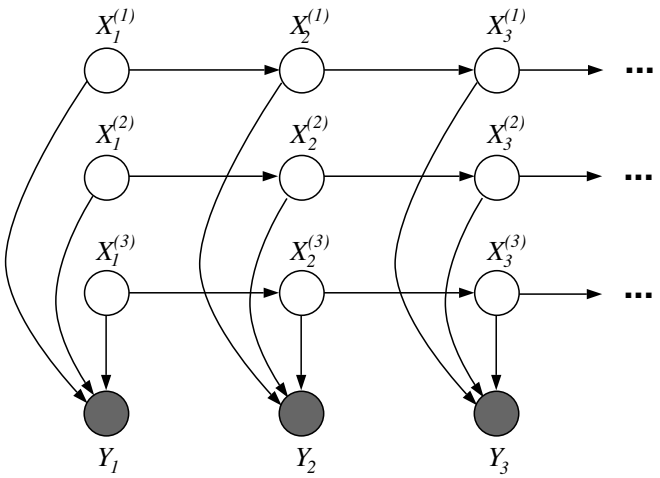


- We can variationally transform this model into:



where we see that we have to solve separate simple HMM problems on each iteration. The variational parameters couple the chains.

Example—Factorial HMM (cont.)



- M independent Markov chains given the observations

$$Q(\{X_t\}|\theta) = \prod_{m=1}^{M} Q(X_1^{(m)}|\theta) \prod_{t=2}^{T} Q(X_t^{(m)}|X_{t-1}^{(m)}, \theta),$$

where

$$\begin{aligned} Q(X_1^{(m)}|\theta) &= \pi^{(m)} h_1^{(m)} \\ Q(X_t^{(m)}|X_{t-1}^{(m)}, \theta) &= P^{(m)} h_t^{(m)}. \end{aligned}$$

- The parameters of this approximation are the $h_t^{(m)}$, which play the role of observation log probabilities for the HMMs.

Example—Factorial HMM (cont.)

- Minimizing the KL divergence results in the following fixed point equation:

$$h_t^{(m)} \propto \exp\{W^{(m)'} C^{-1} \left(Y_t - \hat{Y}_t\right) + W^{(m)'} C^{-1} W^{(m)} \langle X_t^{(m)} \rangle - \frac{1}{2}\Delta^{(m)}\},$$

  where

$$\hat{Y}^t = \sum_{\ell=1}^{M} W^{(\ell)} \langle X_t^{(\ell)} \rangle,$$

  and $\Delta^{(m)}$ is the vector of diagonal elements of $W^{(m)'} C^{-1} W^{(m)}$.

- Repeat until convergence of $KL(Q\|P)$:
  1. Compute $h_t^{(m)}$ using fixed-point equation, which depends on $\langle X_t^{(m)} \rangle$
  2. Compute $\langle X_t^{(m)} \rangle$ using forward-backward algorithm on HMMs with observation log probabilities given by $h_t^{(m)}$

Results

- Fitting the FHMM to the Bach Chorale dataset: (Ghahramani & Jordan, 1997):

Modeling J. S. Bach's chorales

Discrete event sequences:

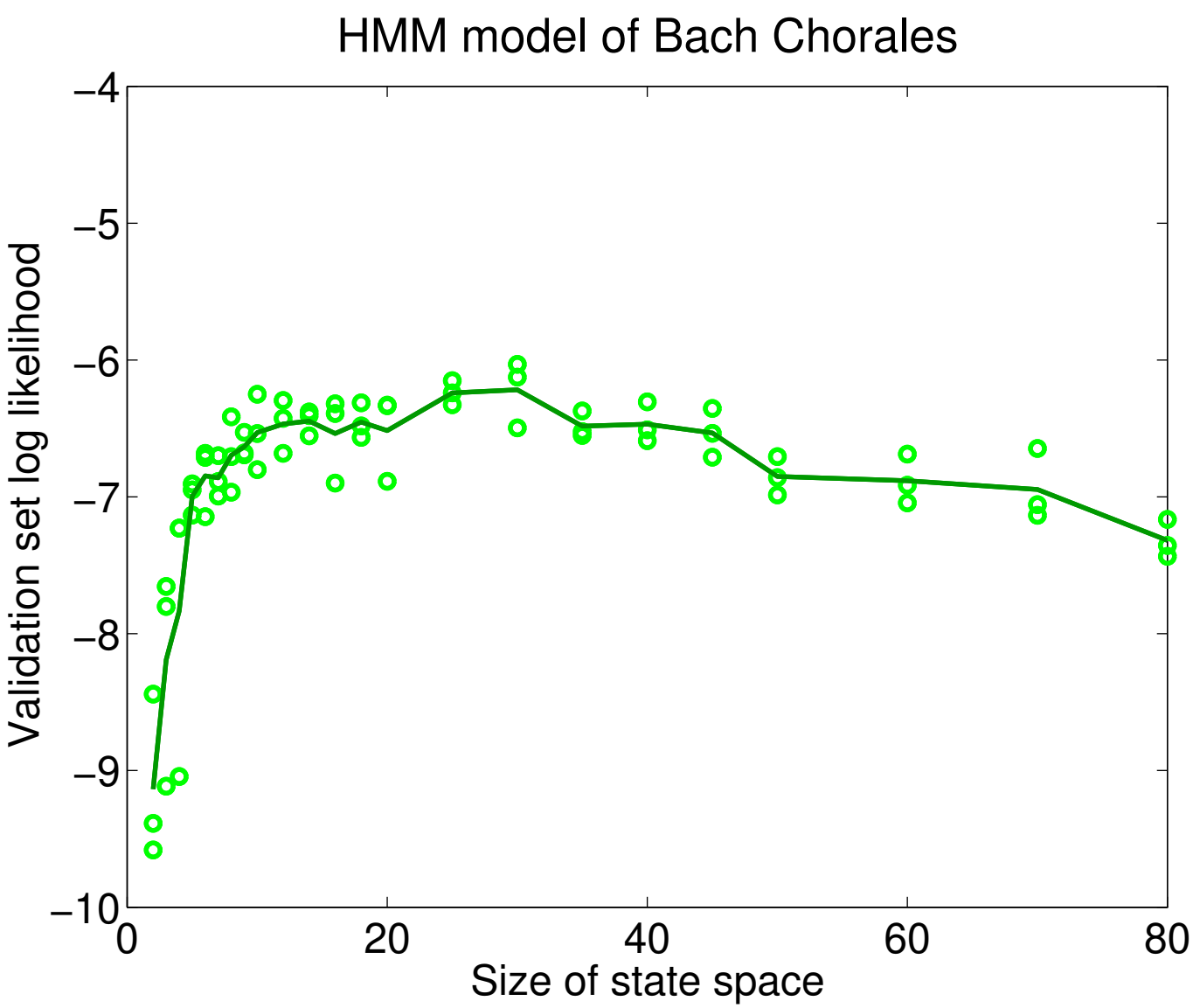
Attribute	Description	Representation
pitch	pitch of the event	int $[0, 127]$
keysig	key signature of the chorale (num of sharps and flats)	int $[7, 7]$
timesig	time signature of the chorale	int (1/16 notes)
fermata	event under fermata?	binary
st	start time of event	int (1/16 notes)
dur	duration of event	int (1/16 notes)

First 40 events of 66 chorale melodies:

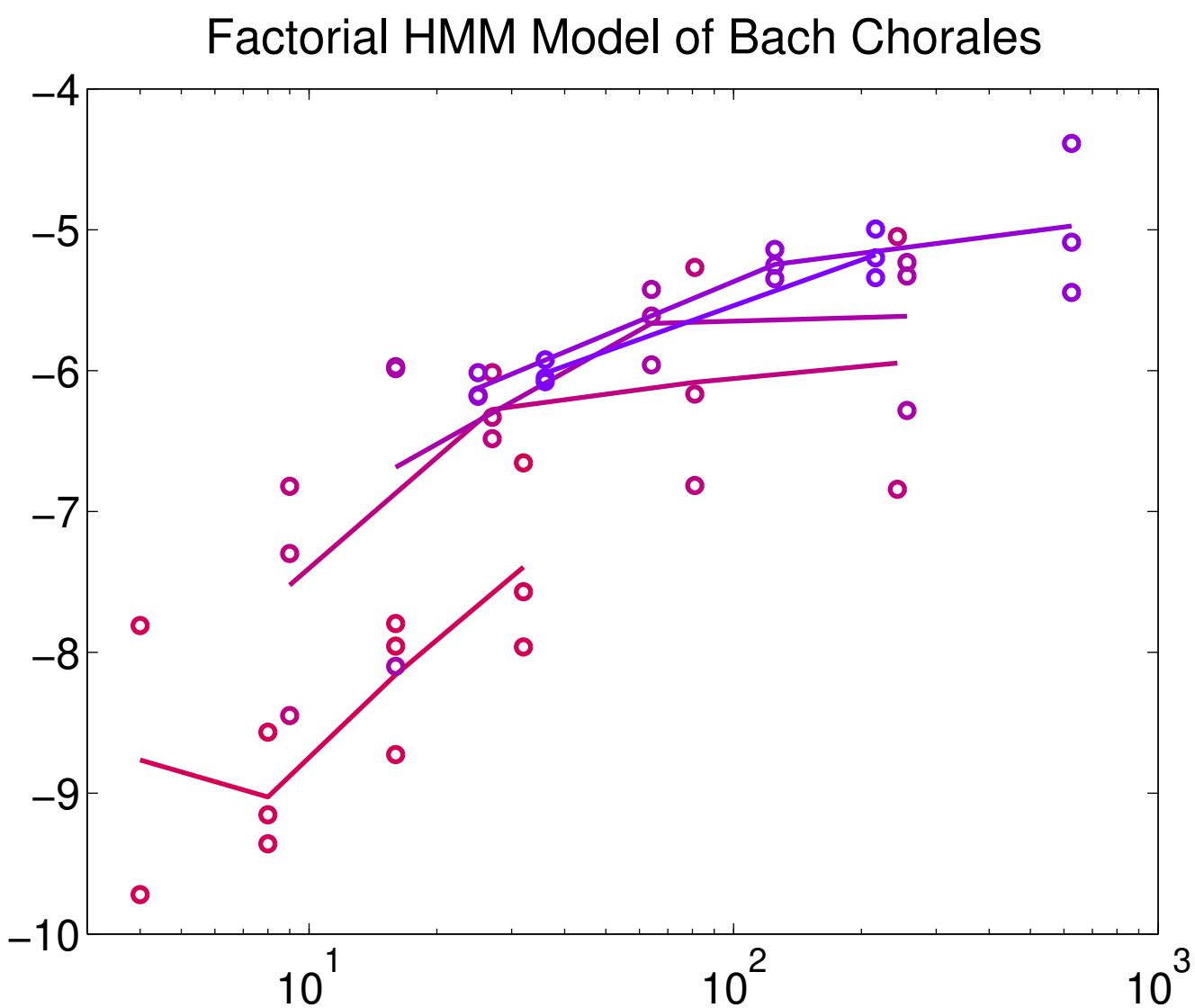
- training: 30 melodies
- test: 36 melodies

See Conklin and Witten (1995).

Fitting a single HMM to the Bach chorale data



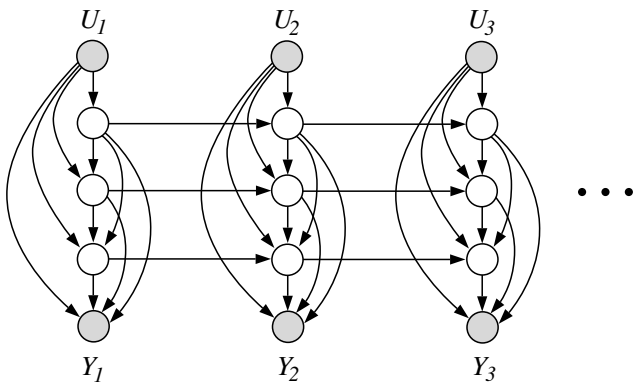
Fitting a factorial HMM to the Bach chorale data



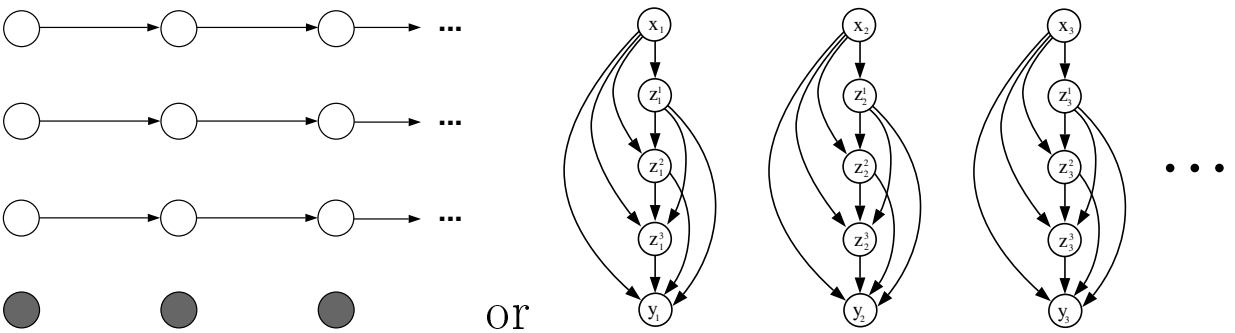
Example—hidden Markov decision tree

(Jordan, Ghahramani, & Saul, 1997)

- Recall the hidden Markov decision tree, which also yielded intractably large cliques when triangulated:

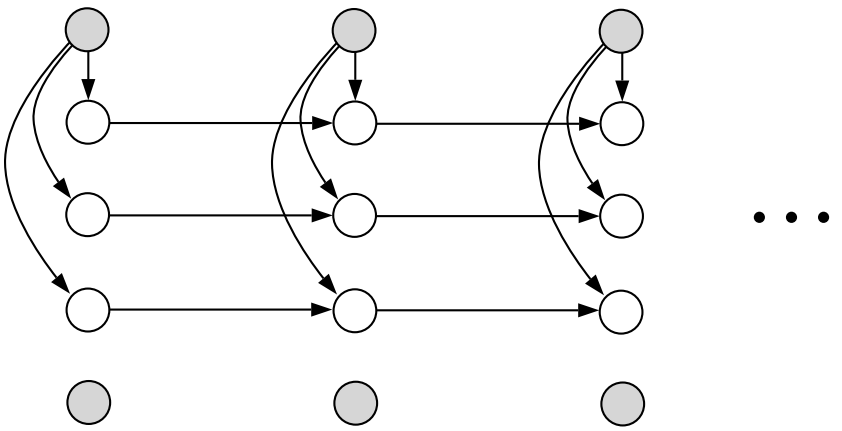


- We can variationally transform this model into one of two simplified models:



Forest of chains approximation

- Eliminating the vertical links that couple the states yields an approximating graph that is a forest of chains:



- The Q distribution is given by:

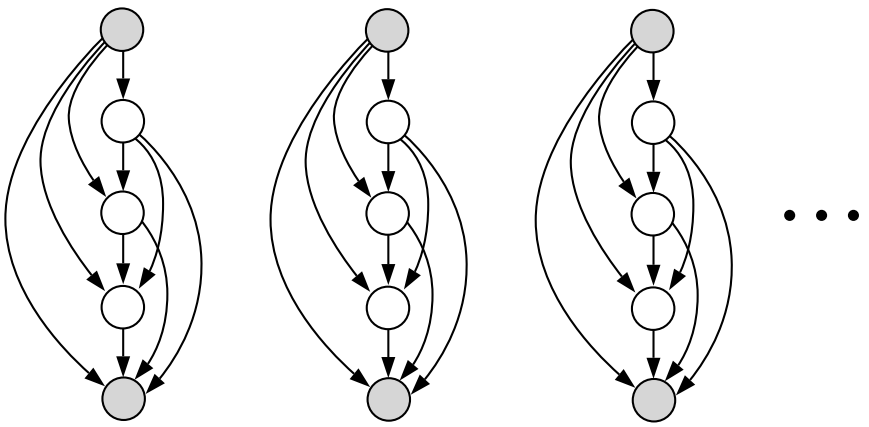
$$Q(\{\mathbf{z}_t^1, \mathbf{z}_t^2, \mathbf{z}_t^3\} \mid \{\mathbf{y}_t\}, \{\mathbf{x}_t\}) =$$
$$\frac{1}{Z_Q} \prod_{t=2}^{T} \tilde{a}_t^1(\mathbf{z}_t^1|\mathbf{z}_{t-1}^1)\tilde{a}_t^2(\mathbf{z}_t^2|\mathbf{z}_{t-1}^2)\tilde{a}_t^3(\mathbf{z}_t^3|\mathbf{z}_{t-1}^3)$$
$$\prod_{t=1}^{T} \tilde{q}_t^1(\mathbf{z}_t^1)\tilde{q}_t^2(\mathbf{z}_t^2)\tilde{q}_t^3(\mathbf{z}_t^3)$$

  where $\tilde{a}_t^i(\mathbf{z}_t^i|\mathbf{z}_{t-1}^i)$ and $\tilde{q}_t^i(\mathbf{z}_t^i)$ are potentials that provide the variational parameterization

- The resulting algorithm is efficient because we know an efficient subroutine for single chains (the forward-backward algorithm)

Forest of trees approximation

- Eliminating the horizontal links that couple the states yields an approximating graph that is a forest of trees:



- The Q distribution is given by:

$$Q(\{\mathbf{z}_t^1, \mathbf{z}_t^2, \mathbf{z}_t^3\} \mid \{\mathbf{y}_t\}, \{\mathbf{x}_t\}) = \prod_{t=1}^{T} \tilde{r}_t^1(\mathbf{z}_1^1)\tilde{r}_t^2(\mathbf{z}_1^2|\mathbf{z}_1^1)\tilde{r}_t^3(\mathbf{z}_1^3|\mathbf{z}_1^1, \mathbf{z}_1^2)$$

- The resulting algorithm is efficient because we know an efficient subroutine for decision trees (the upward recursion from Jordan and Jacobs)

A Viterbi-like approximation

- We can develop a Viterbi-like algorithm by utilizing an approximation Q that assigns probability one to a single path $\{\bar{\mathbf{z}}_t^1, \bar{\mathbf{z}}_t^2, \bar{\mathbf{z}}_t^3\}$:

$$Q(\{\mathbf{z}_t^1, \mathbf{z}_t^2, \mathbf{z}_t^3\} \mid \{\mathbf{y}_t\}, \{\mathbf{x}_t\}) = \begin{cases} 1 \text{ if } \mathbf{z}_t^i = \bar{\mathbf{z}}_t^i, \quad \forall t, i \\ 0 \text{ otherwise} \end{cases}$$

- Note that the entropy $Q \ln Q$ is zero
- The evaluation of the energy $Q \ln P$ reduces to substituting $\bar{\mathbf{z}}_t^i$ for $\mathbf{z}_t^i$ in P
- The resulting algorithm involves a subroutine in which a standard Viterbi algorithm is run on a single chain, with the other (fixed) chains providing field terms

CONCLUSIONS (PART II)

- General frameworks for probabilistic computation in graphical models:
  - exact computation
  - stochastic approximation
  - deterministic approximation
- Variational methods are deterministic approximation methods
  - they aim to take advantage of law-of-large-numbers phenomena in dense networks
  - they yield upper and lower bounds on desired probabilities
- The techniques can be combined to yield hybrid techniques which may well turn out to be better than any single technique

ADDITIONAL TOPICS

- There are many other topics that we have not covered, including:
  - learning structure (see Heckerman tutorial)
  - causality (see UAI homepage)
  - qualitative graphical models (see UAI homepage)
  - relationships to planning (see UAI homepage)
  - influence diagrams (see UAI homepage)
  - relationships to error control coding (see Frey thesis)

- The Uncertainty in Artificial Intelligence (UAI) homepage:

  `http://www.auai.org`

REFERENCES

Jensen, F. (1996). *An Introduction to Bayesian Networks*. London: UCL Press (also published by Springer-Verlag).

Lauritzen, S. L., & Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *Journal of the Royal Statistical Society B, 50*, 157-224.

Jensen, F. V., Lauritzen, S. L. and Olesen, K. G., (1990) Bayesian updating in recursive graphical models by local computations. *Computational Statistical Quarterly. 4*, 269–282.

Dawid, A. P. (1992). Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing, 2*, 25-36.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufman.

Whittaker, J. (1990). *Graphical Models in Applied Multivariate Statistics*. New York: John Wiley.

Lauritzen, S. (1996). *Graphical Models*. Oxford: Oxford University Press.

Spiegelhalter, D. J., Dawid, A. P., Lauritzen, S. L., & Cowell, R. G. (1993). Bayesian Analysis in Expert Systems, *Statistical Science, 8*, 219-283.

Heckerman, D. (1995). A tutorial on learning Bayesian networks. [available through http://www.auai.org].

Buntine, W. (1994). Operations for learning with graphical models. *Journal of Artificial Intelligence Research, 2*, 159-225. [available through http://www.auai.org].

Jordan, M. I. (ed) (1998). *Learning in Graphical Models*. Kluwer Academic Press.

Titterington, D., Smith, A. and Makov, U. (1985) Statistical analysis of finite mixture models. Wiley, Chichester, UK.

Little, R. J. A., & Rubin, D. B. (1987). *Statistical Analysis with Missing Data*. New York: John Wiley.

Rabiner, L., (1989) A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE, 77*, 257-285.

Smyth, P., Heckerman, D., and Jordan, M. I. (1997). Probabilistic independence networks for hidden Markov probability models. *Neural Computation*, *9*, 227–270.

Jordan, M. and Jacobs, R. (1994) Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214.

Saul, L. K., and Jordan, M. I. (1995) Boltzmann chains and hidden Markov models. In G. Tesauro, D. S. Touretzky & T. K. Leen, (Eds.), *Advances in Neural Information Processing Systems 7*, MIT Press, Cambridge MA.

Neal, R. (1995) Probabilistic inference using Markov Chain Monte Carlo methods. [`http://www.cs.toronto.edu/~radford/`]

BUGS [`http://www.mrc-bsu.cam.ac.uk/bugs`]

Jordan, M. I., & Bishop, C. (1997) Neural networks. In Tucker, A. B. (Ed.), *CRC*

Handbook of Computer Science, Boca Raton, FL: CRC Press.

Neal, R. (1992) Connectionist learning of belief networks. *Artificial Intelligence*, **56**, pp 71-113.

Neal, R. & Hinton, G. E. (1998) A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, (Ed.), *Learning in Graphical Models*. Kluwer Academic Press.

Hinton, G. E., Dayan, P., Frey, B., and Neal, R. (1995) The wake-sleep algorithm for unsupervised neural networks. *Science*, 268:1158–1161.

Dayan, P., Hinton, G. E., Neal, R., and Zemel, R. S. (1995) Helmholtz Machines. *Neural Computation*, bf 7, 1022-1037.

Saul, L. K., & Jordan, M. I. (1996) Exploiting tractable substructures in intractable networks. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems 8*, MIT Press, Cambridge MA.

Saul, L., Jaakkola, T. and Jordan, M. (1996) Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, *4*, 61–76.

Williams, C. K. I., & Hinton, G. E. (1991) Mean field networks that learn to discriminate temporally distorted strings. In Touretzky, D. S., Elman, J., Sejnowski, T., & Hinton, G. E., (Eds.), *Proceedings of the 1990 Connectionist Models Summer School*. San Mateo, CA: Morgan Kaufmann.

Ghahramani, Z., and Jordan, M. I. (1996) Factorial Hidden Markov models. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems 8*, MIT Press, Cambridge MA.

Jaakkola, T., & Jordan, M. I. (1996). Computing upper and lower bounds on likelihoods in intractable networks. In E. Horvitz (Ed.), *Workshop on Uncertainty in Artificial Intelligence*, Portland, Oregon.

Jaakkola, T., & Jordan, M. I. (in press). Bayesian logistic regression: a variational approach. In D. Madigan & P. Smyth (Eds.), Proceedings of the 1997 Conference on Artificial Intelligence and Statistics, Ft. Lauderdale, FL.

Saul, L. K., & Jordan, M. I. (in press). Mixed memory Markov models. In D. Madigan & P. Smyth (Eds.), Proceedings of the 1997 Conference on Artificial Intelligence and Statistics, Ft. Lauderdale, FL.

Frey, B., Hinton, G. E., & Dayan, P. (1996). Does the wake-sleep algorithm produce good density estimators? In D. S. Touretzky, M. C. Mozer, M. E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems 8*, MIT Press, Cambridge MA.

Jordan, M. I., Ghahramani, Z., & Saul, L. K. (1997). Hidden Markov decision trees. In M. C. Mozer, M. I. Jordan, & T. Petsche (Eds.), *Advances in Neural Information Processing Systems 9*, MIT Press, Cambridge MA.

Jaakkola, T. S. (1997). *Variational methods for inference and estimation in graphical models*. Unpublished doctoral dissertation, Massachusetts Institute of Technology.

Jaakkola, T., & Jordan, M. I. (1997). Recursive algorithms for approximating probabilities in graphical models. In M. C. Mozer, M. I. Jordan, & T. Petsche (Eds.), *Advances in*

Neural Information Processing Systems 9, MIT Press, Cambridge MA.

Frey, B. (1997). *Bayesian networks for pattern classification, data compression, and channel coding.* Unpublished doctoral dissertation, University of Toronto.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., & Saul, L. K. (1998). An introduction to variational methods for graphical models. In M. I. Jordan (Ed.), *Learning in Graphical Models.* Dordrecht: Kluwer Academic Press.