Graphical models, neural networks and variational approximations

Michael Jordan

Center for Biological and Computational Learning
Massachusetts Institute of Technology
E25-229
Cambridge, MA 02139

Thanks and acknowledgments to:

Zoubin Ghahramani, Tommi Jaakkola and Lawrence Saul

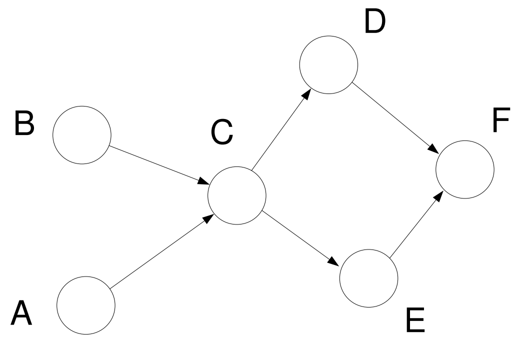
January, 1997

Introduction

- Graphical models unify much of the literature on learning systems
- Examples of graphical models include:
  - mixture models
  - hidden Markov models
  - mixtures of experts models
  - probabilistic decision trees
  - Boltzmann machines
  - association rules (aka loglinear models)
  - neural networks (under a probabilistic interpretation)
  - many "unsupervised learning" systems
- The key calculation for all of these models is the "inference" calculation
  - "inference" is the calculation of the probabilities of unobserved nodes given observed nodes
- A variety of methods are available for inference
  - belief propagation (exact)
  - variational methods (upper and lower bounds)
  - Markov chain Monte Carlo (approximation)

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 can imagine some of these nodes to be "inputs," others to be "outputs," and others to be "hidden"

- We want to associate a 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.,

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

Some ways to use a graphical model

- *Prediction*
  - calculate the probability of *downstream* variables given the values of *upstream* variables
- *Diagnosis*
  - calculate the probability of *upstream* variables given the values of *downstream* variables
- *Value of information*
  - find the node that has the most effect on the value of a probability of interest
- *Cleaning up missing values*
  - fill in a node whose value is unknown with its most probable value
- *Learning*
  - how can we improve a model? how to improve parameter values? what additional nodes or links should be added to get a better model?

Specification of a graphical model

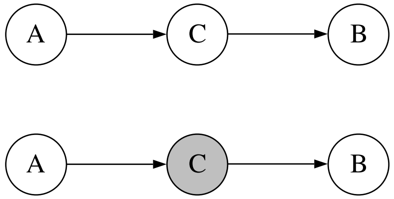
There are two components to any graphical model:

- the *qualitative* specification
- the *quantitative* specification

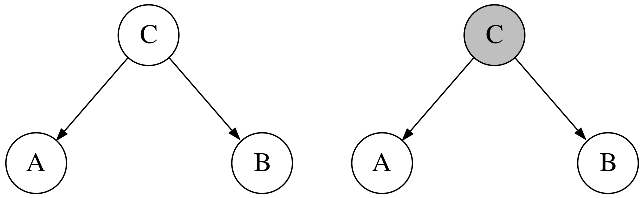
Where do these come from in real life?

- prior knowledge of causal relationships
- prior knowledge of modular relationships
- assessment from experts
- learning from data

Qualitative specification of graphical models

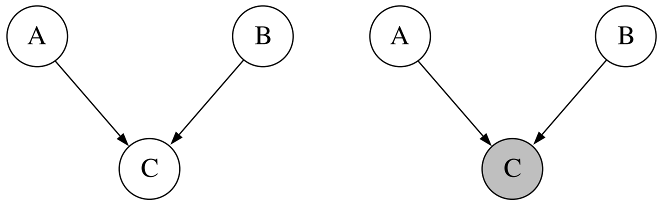


- A and B are marginally *dependent*
- A and B are conditionally *independent*



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

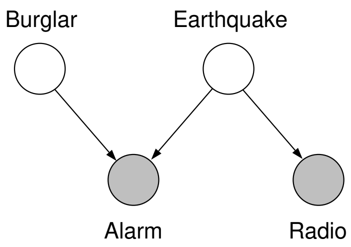
Semantics of graphical models (cont)



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

- This is the interesting case...

"Explaining away" (Pearl)

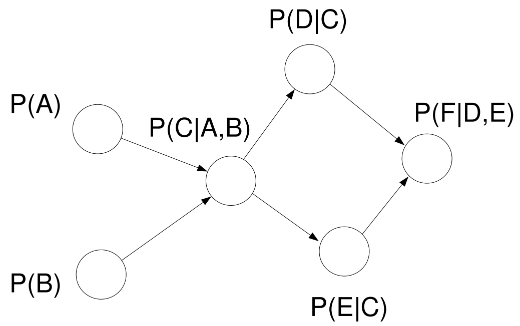


- All connections (in both directions) are "excitatory"
- But an increase in "activation" of Earthquake leads to a decrease in "activation" of Burglar
- Where does the "inhibition" come from?

Quantitative specification of directed graphical models

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

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 $\mathrm{pa}(S_i)$ represent the set of parents of node S_i

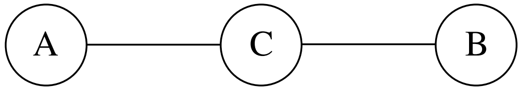
Then

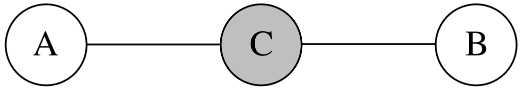
$$\begin{aligned} P(S) &= P(S_1)P(S_2|S_1)\cdots P(S_N|S_{N-1},\ldots,S_1) \\ &= \prod_i P(S_i|S_{i-1},\ldots,S_1) \\ &= \prod_i P(S_i|\mathrm{pa}(S_i)) \end{aligned}$$

where the last line is by assumption

- *We can prove a theorem that states that if arbitrary probability distributions are utilized for* $P(S_i|\mathrm{pa}(S_i))$ *in the formula above, then the family of probability distributions obtained is exactly that set which respects the qualitative specification (the conditional independence relations) described earlier*

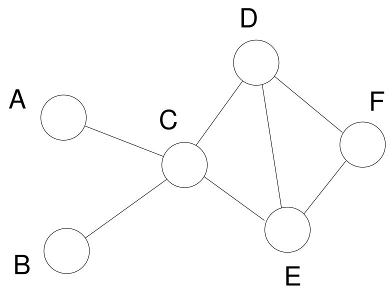
Semantics of undirected graphs



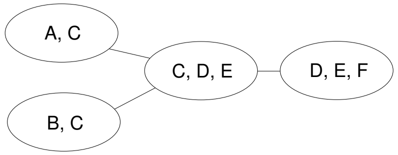


- A and B are marginally *dependent*
- A and B are conditionally *independent*

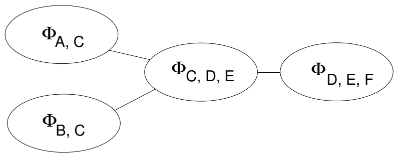
Quantitative specification of undirected models



- 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 real number with each configuration of the clique



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 we assume that the normalization constant is absorbed into one of the potentials

- *We can then prove a theorem that states that if arbitrary potentials are utilized in the product formula for probabilities, then the family of probability distributions obtained is exactly that set which respects the qualitative specification (the conditional independence relations) described earlier*

(This theorem is known as the Hammersley-Clifford theorem)

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*

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

  – *prediction* and *diagnosis* are special cases

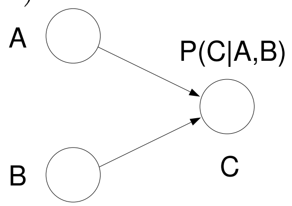
Exact inference algorithms for directed graphs

- 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

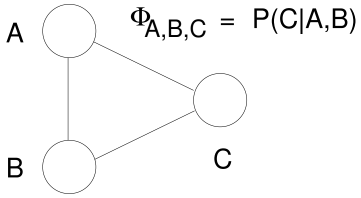
To understand the junction tree algorithm, we need to understand how to "compile" a directed graph into an undirected graph

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:

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

where

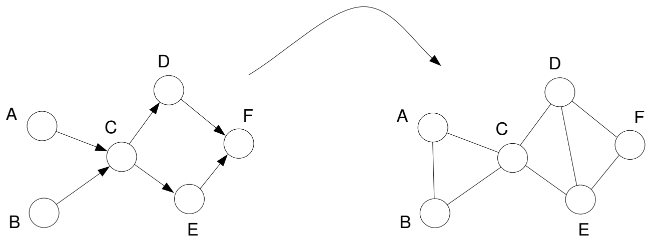
$$\phi_{(A,B,C)} = P(A)P(B)P(C|A,B)$$

and

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

and

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

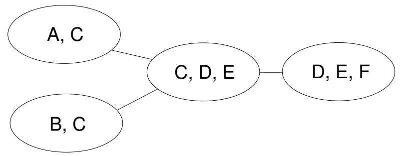


Propagation of probabilities

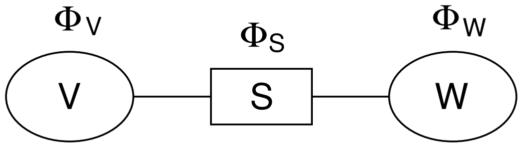
Now suppose that some evidence has been absorbed. How do we propagate this effect to the rest of the graph?

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)).



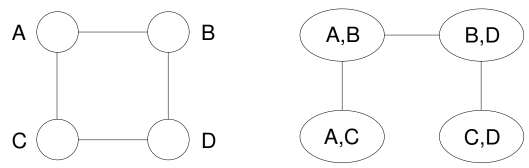
- the cliques need to "agree" on the probability of nodes in the overlap; this is achieved by *marginalizing* and *rescaling*:

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

- this occurs in parallel, distributed fashion throughout the clique tree
- or we can organize a "collect–distribute" protocol for computation on a serial computer

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?

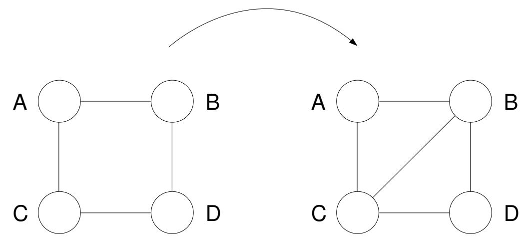
- *Answer*: Nothing. In fact this is a problem with the algorithm as described so far. It is not true that in general *local* consistency implies *global* consistency.

  What else do we need to get such a guarantee?

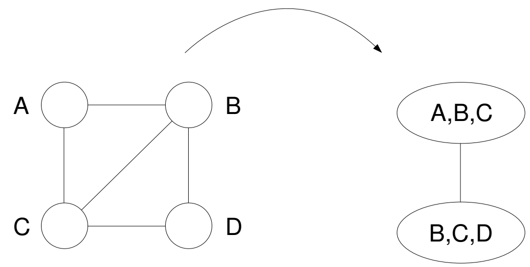
Triangulation

- 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

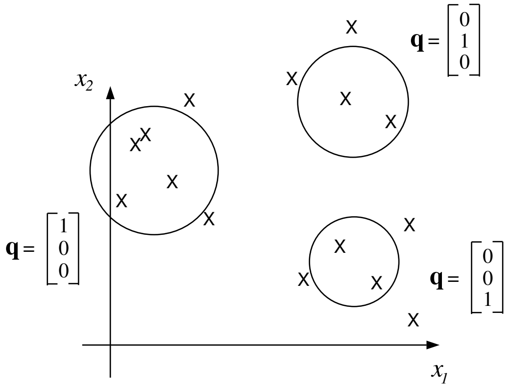
- A clique tree for a triangulated graph is referred to as a *junction tree*
- In junction trees, local consistency implies global consistency
- Thus we can use local message-passing in junction trees to calculate probabilities

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 NP-hard
- in any case, the triangulation is "off-line"; our concern will be with the "on-line" problem of message propagation
  - we will classify the complexity of various architectures by referring to the size of cliques in the "obvious triangulation"

Example: Gaussian mixture models

- A Gaussian mixture model is a popular clustering model



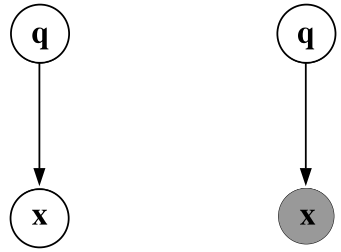
- hidden state $\mathbf{q}$ is a multinomial RV
- output $\mathbf{x}$ is a Gaussian RV
- prior probabilities on hidden states:

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

- 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



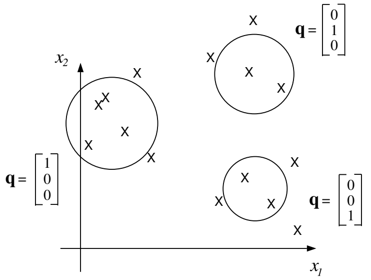
- The inference problem is to calculate the posterior probabilities:

$$\begin{aligned} P(q_i = 1|\mathbf{x}) &= \frac{P(\mathbf{x}|q_i = 1)P(q_i = 1)}{P(\mathbf{x})} \\ &= \frac{\frac{\pi_i}{|\Sigma_i|^{1/2}} \exp\{-\frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma_i^{-1}(\mathbf{x} - \mu_i)\}}{\sum_j \frac{\pi_j}{|\Sigma_j|^{1/2}} \exp\{-\frac{1}{2}(\mathbf{x} - \mu_j)^T \Sigma_j^{-1}(\mathbf{x} - \mu_j)\}} \end{aligned}$$

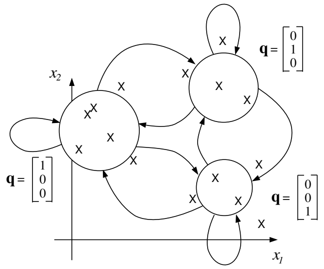
- This can be viewed as a normalized “distance”
- It is the E step of the EM algorithm

Hidden Markov models

- Recall the Gaussian mixture model:



- An HMM is a mixture model with *dynamics*. It jumps from cluster to cluster according to a state transition matrix:



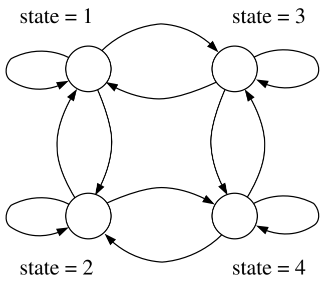
- In each state the HMM generates an output. The sequence of outputs (the X's in the picture) are all that the learner gets to see

The meaning of the "state"

- A hidden Markov model jumps from "state" to "state"
- These "states" do not necessarily have to have a physical meaning. E.g., in speech, we do not necessarily imagine that the speech-producing apparatus moves through a sequence of discrete states
- But the idea is still (very) useful. Essentially we are allowing the data to be analyzed in terms of "modes" or "phases" or "periods" or "seasons"
- This is better than forcing the data to fit the stationarity assumptions characteristic of many time series models

Hidden Markov models (cont.)

- In general, an HMM is defined by a state transition diagram



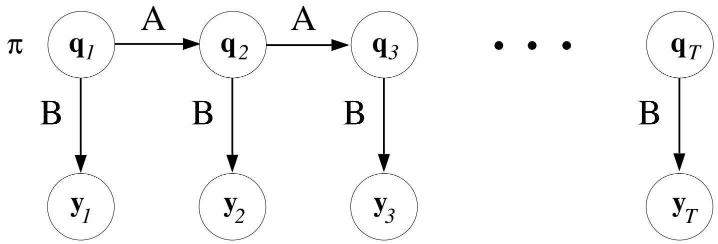
and a class-conditional probability distribution at each node

- Don't get confused! A state transition diagram is *not* a graphical model
  - it's not a directed acyclic graph
  - the arcs don't encode probabilistic independence statements
- Instead, let $\mathbf{q}$ be an indicator vector:

$$\text{e.g.,}\quad \text{state} = 2 \qquad \Longrightarrow \qquad \mathbf{q} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Hidden Markov models (cont.)

- And arrange the sequence of state vectors in a chain:



- In this diagram the links correspond to conditional probabilities:
  - state transition probability matrix A:

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

  - emission parameters $B = \{\mu_i, \Sigma_i\}$:

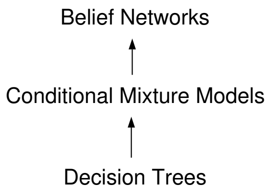
$$P(\mathbf{y}_t|\mathbf{q}_t = i, \mu_i, \Sigma_i)$$

    is a Gaussian distribution with mean μ_i and covariance Σ_i.
  - initial state probabilities π:

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

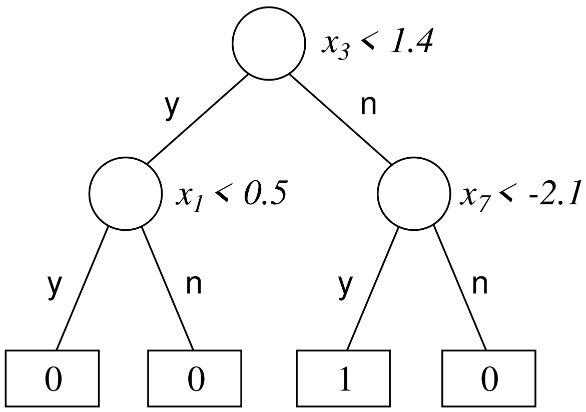
- And the graph makes conditional independence statements

Example: Probabilistic decision trees



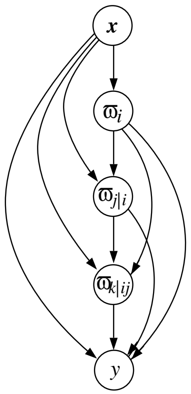
- Trees should also yield efficient algorithms
- Convert decision tree problems into (conditional) mixture problems
- Mixtures are instances of general graphical models

Classical decision trees
(CART, C4.5)

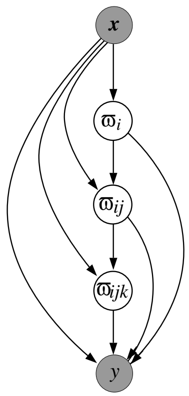


- drop the data set down the tree
- at each node, try to find a split of the input space (a half-plane) that yields the largest gain in "purity" on left and right
- build a large tree and prune backward to create a nested sequence of trees
- pick the best tree from the sequence using cross-validation

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

Posterior splitting probabilities

$$
\begin{aligned}
h_i &\equiv P(\omega_i|\mathbf{x},\mathbf{y}) \\
h_{j|i} &\equiv P(\omega_{ij}|\mathbf{x},\mathbf{y},\omega_i) \\
h_{k|ij} &\equiv P(\omega_{ijk}|\mathbf{x},\mathbf{y},\omega_i,\omega_{ij}) \\
& \ldots
\end{aligned}
$$

(cf. *prior* splitting probabilities)

$$
\begin{aligned}
g_i &\equiv P(\omega_i|\mathbf{x}) \\
g_{j|i} &\equiv P(\omega_{ij}|\mathbf{x},\omega_i) \\
g_{k|ij} &\equiv P(\omega_{ijk}|\mathbf{x},\omega_i,\omega_{ij}) \\
& \ldots
\end{aligned}
$$

Bayes' rule yields:

$$h_i = \frac{g_i \sum_j g_{j|i} \sum_k g_{k|ij} P_{ijk}(\mathbf{y}|\mathbf{x})}{\sum_i g_i \sum_j \sum_j g_{j|i} \sum_k g_{k|ij} P_{ijk}(\mathbf{y}|\mathbf{x})}$$

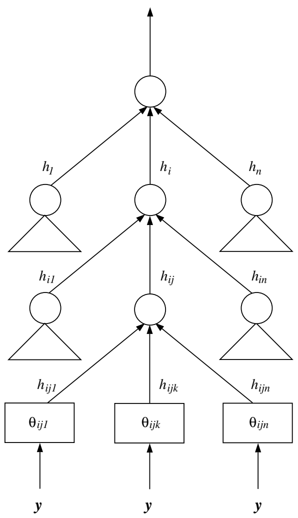
$$h_{j|i} = \frac{g_{j|i} \sum_k g_{k|ij} P_{ijk}(\mathbf{y}|\mathbf{x})}{\sum_j g_{j|i} \sum_k g_{k|ij} P_{ijk}(\mathbf{y}|\mathbf{x})}$$

$$h_{k|ij} = \frac{g_{k|ij} P_{ijk}(\mathbf{y}|\mathbf{x})}{\sum_k g_{k|ij} P_{ijk}(\mathbf{y}|\mathbf{x})}$$

$$\ldots$$

Note that the denominator at each level appears in the numerator at the next level up. We can organize the computation as a recursion.

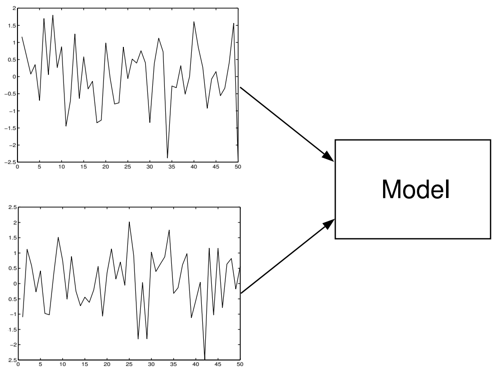
Posterior propagation



- This algorithm is efficient, because of the tree structure.
- Once the posteriors are calculated (the E step), each node has a local parameter estimation problem to solve (the M step)
  - the internal nodes solve classification problems
  - the leaf nodes solve weighted classification or regression problems
- The algorithm is guaranteed to increase the likelihood on each iteration

Coupled hidden Markov models

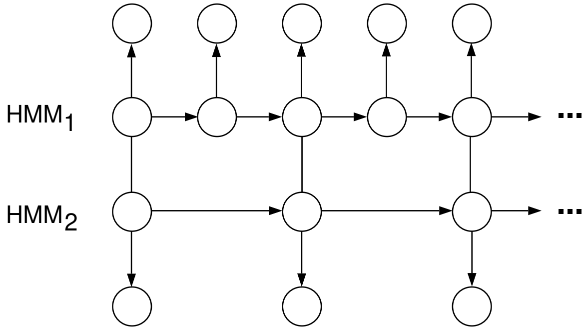
- Consider the problem of *fusing* a pair of signals. For example, one signal could be video and one signal audio. Or one signal could be one stock exchange and the other a different stock exchange.



- How can we do this using hidden Markov models? (How can we do this, period)?
- Suppose that we have an HMM for each of the two signals? How to fuse the HMMs?

Coupled HMMs

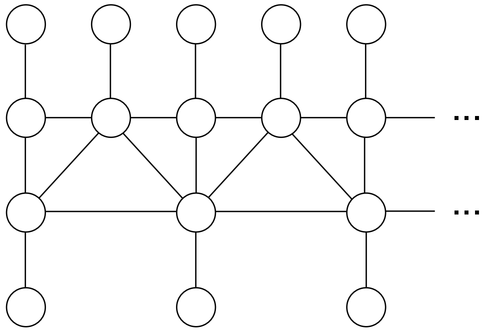
- We could define a "product HMM" by taking the product of the two state spaces. I.e., the states of the product HMM would be pairs of states from the underlying HMMs
  - but this will produce a huge transition matrix, probably leading to overfitting
  - and perhaps the underlying sequences are on different time scales. We don't want to have to oversample or undersample
- A better solution: draw the graphical model that we have in mind:



and ask whether the junction tree algorithm will be efficient for this structure

Triangulated coupled HMM

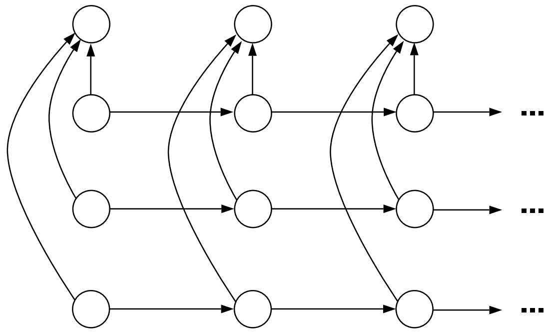
- This figure shows the triangulation of the coupled HMM:



- The largest clique size is N^3, and there are $O(T)$ such cliques, so the algorithm is still (reasonably) efficient
- A research group at Ricoh has successfully used this architecture to fuse video and audio signals for automatic lip-reading

Factorial hidden Markov models

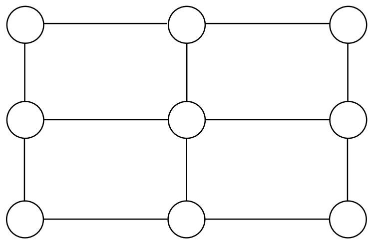
- Imagine that a time series is created from a set of M loosely-coupled underlying mechanisms
  - real speech is better characterized this way
  - many economic systems are better characterized this way
- Each of these mechanisms may have their own particular dynamic laws, and we want to avoid merging them into a single "meta-state," with a single transition matrix
- Here is the graphical model that we would like to use:



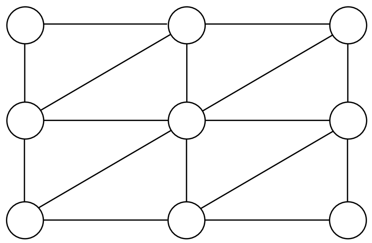
- When we triangulate do we get an efficient structure?

Triangulation?

- Here is the basic structure that we need to triangulate:



- Unfortunately, the following graph is not triangulated:



- To see this, "pick up" the central node. The "base" is a chordless cycle of length six!
- When we triangulate this kind of lattice structure we essentially get a fully connected structure. The junction tree algorithm is not efficient for factorial HMMs.

Variational algorithms save the day

- Later we will show how variational algorithms can be used to calculate *bounds* on the exact values of the probabilities
- The bounds provide a usable algorithm:

Modeling J. S. Bach's chorales

Discrete event sequences:

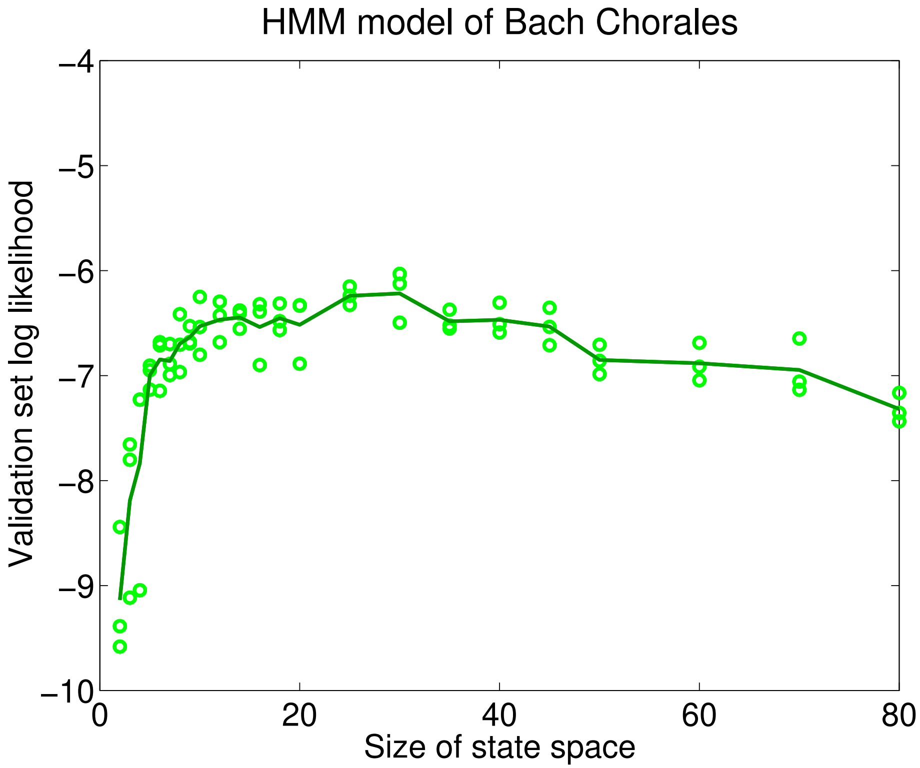
Attribute	Description	Representation
pitch	pitch of the event	int $[0, 127]$
keysig	key signature of the chorale (num of sharps and flats)	int $[7, 7]$
timesig	time signature of the chorale	int (1/16 notes)
fermata	event under fermata?	binary
st	start time of event	int (1/16 notes)
dur	duration of event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

First 40 events of 66 chorale melodies:

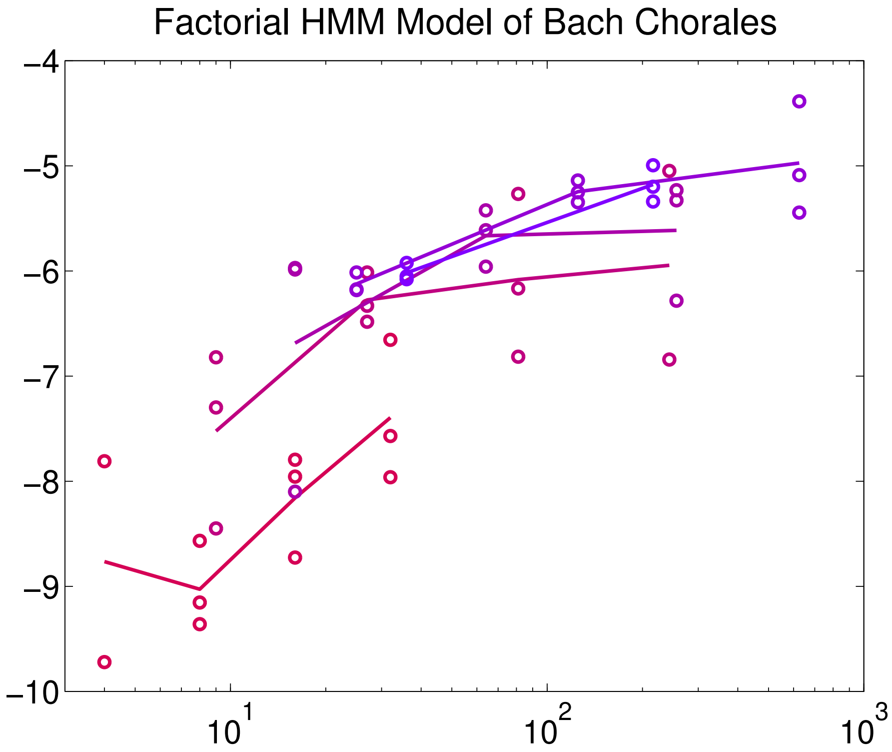
- training: 30 melodies
- test: 36 melodies

See Conklin and Witten (1995).

Fitting a single HMM to the Bach chorale data

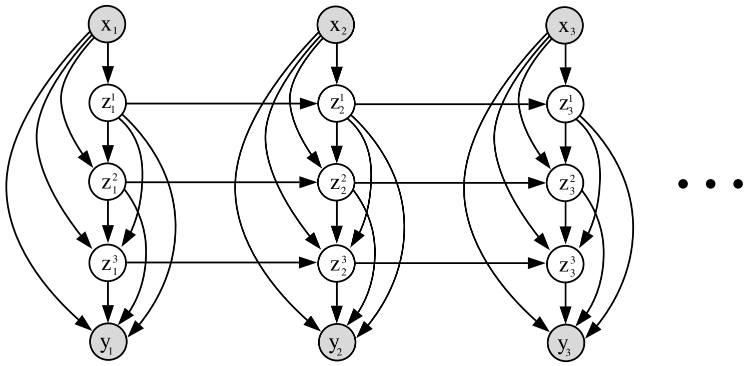


Fitting a factorial HMM to the Bach chorale data



Hidden Markov decision trees

- 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 use variational methods

Multilayer neural networks as graphical models

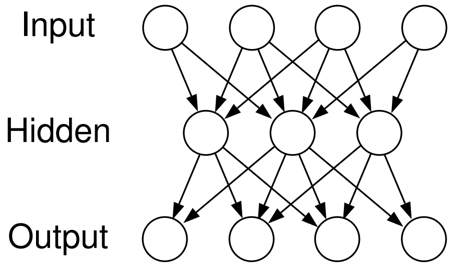
(cf. Neal, 1992; Saul, Jaakkola, & Jordan, 1995)

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

$$P(S_i = 1|\mathbf{x}_i) = \frac{1}{1 + e^{-\mathbf{w}_i^T \mathbf{x}_i}}$$

  where $\mathbf{x}_i$ are the parents of node i

- A multilayer neural network with logistic hidden units:



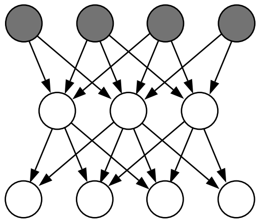
- has a joint distribution that is a product of logistic functions:

$$P(\{S_1, S_2, \ldots, S_N\}) = \prod_i \frac{1}{1 + e^{-\mathbf{w}_i^T \mathbf{x}_i}}$$

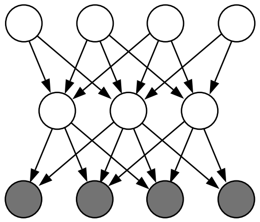
Multilayer neural networks (cont.)

- Many neural network operations can be viewed as graphical model inference problems:

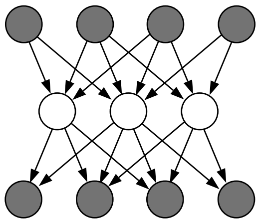
Prediction:



Diagnosis, control, optimization:



Learning:



- Clearly, however, the cliques in the junction tree will be intractably large

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 $P(S)$

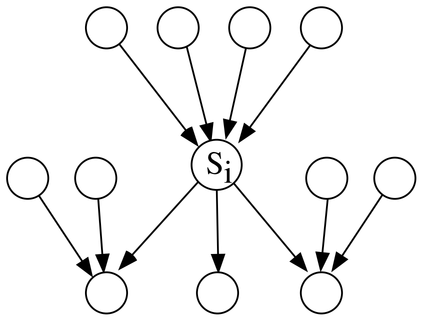
Gibbs sampling

- 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

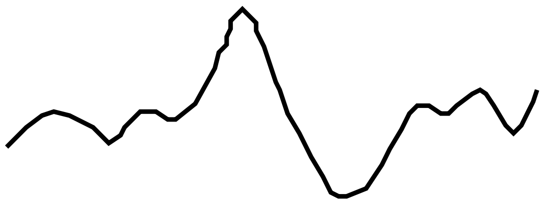
- Variational methods are deterministic approximation methods
- They have several advantages compared to MCMC methods
  - they can be much faster
  - they yield upper and lower bounds on probabilities
- And they have several disadvantages
  - they are not as simple and widely applicable as MCMC methods
  - 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)

Learning via variational methods

- 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 posterior at each iteration
- Bayesian methods for graphical models:
  - the variational approximation to the posterior may be easier to integrate than the true posterior
  - the variational approximation to the posterior may be easier to sample from than the true posterior
  - see Jaakkola and Jordan (AI-stats '97)

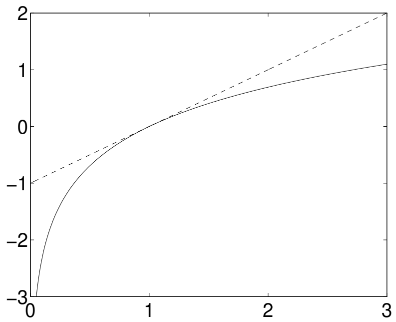
Introduction to variational methods

- Commonly used in quantum mechanics:
  - we have a complicated wave function



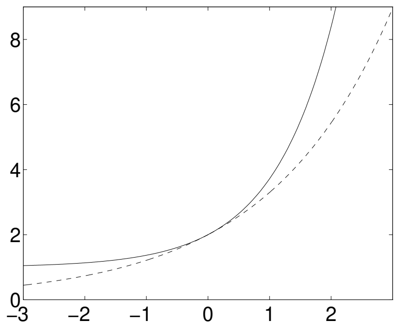
  - we only care about a scalar value, the energy
  - consider a parameterized family of wave functions
  - minimize the energy over the parameters
- Can guarantee upper or lower bounds on the true energy (suffices for many physical applications)

Examples of variational transformations



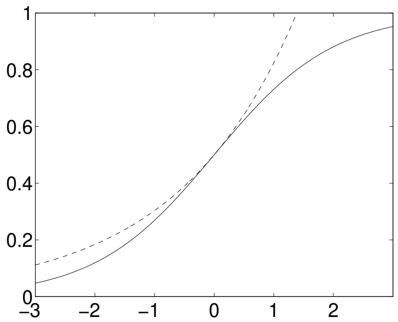
- $$\begin{aligned} \log(x) &= \min_\lambda \{\lambda x - \log \lambda - 1\} \\ &\leq \lambda x - \log \lambda - 1 \end{aligned}$$
  - λ is a *variational parameter*
  - transforms a nonlinearity into a linearity

Examples of variational transformations



- $$\begin{aligned} e^x + e^y &= \max_\lambda \{e^{\lambda\, x + (1-\lambda)\, y + H(\lambda)}\} \\ &\geq e^{\lambda\, x + (1-\lambda)\, y + H(\lambda)} \end{aligned}$$
  - where $H(\cdot)$ is the binary entropy
  - products of $e^x + e^y$ terms are complicated; products of exponentials are simple

Examples of variational transformations



- $$g(x) = \frac{1}{1+e^{-x}} = \min_{\lambda}\{e^{\lambda x - H(\lambda)}\}$$
  $$\leq e^{\lambda x - H(\lambda)}$$
  - where $H(\cdot)$ is the binary entropy
  - a nonlinear function now a simple exponential
- (see Jaakkola and Jordan (NIPS '97) for a general formalism for obtaining such variational transformations)

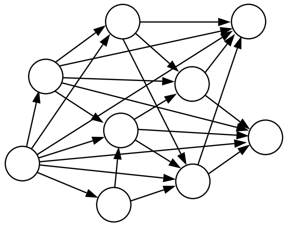
Variational methods and graphical models

- Exact methods
  - key idea is *decomposability*
  - this is achieved via triangulation
- Variational methods (upper and lower bounds)
  - key idea is *recursive elimination*
  - this is achieved via variational transformations
  - easy to establish lower bounds variationally via the Kullback-Leibler (KL) divergence

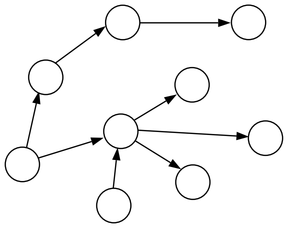
Lower bounds and KL divergence

(see, e.g., Saul & Jordan, NIPS '95)

- Consider a dense graph characterized by a joint distribution $P(H|V)$



- We remove links to obtain a simpler graph (amenable to exact methods), characterized by a joint distribution $Q(H|V)$



- $Q(H|V)$ has extra degrees of freedom, given by *variational parameters* μ_i
- Choose μ_i so as to minimize

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

Lower bounds and KL divergence (cont.)

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

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

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

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

which is positive; thus we have a lower bound

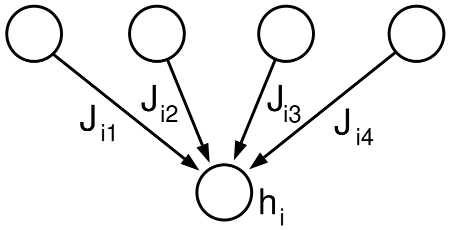
Neural networks and variational approximations

(Saul, Jaakkola, & Jordan, JAIR '96)

- Nodes represent binary random variables:

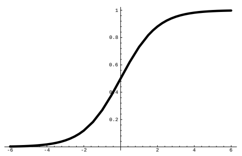
$$\{S_i\}_{i=1}^N = \{0,1\}^N$$

- Weights J_{ij} are attached to each link in the directed graph:



- A logistic function is used to represent conditional probabilities:

$$P(S_i = 1|\Pi_{S_i}) = \sigma\left(\sum_j J_{ij} S_j + h_i\right)$$



$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Evidence and inference

- 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

- Inferences are made under the **conditional distribution**

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

where Z_V is the **likelihood** of the observed evidence:

$$Z_V = P(V) = \Sigma_H P(H,V)$$

Division of labor

- The KL bound has two basic components:

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

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

- What we need is the difference:

$$\ln Z \geq \begin{pmatrix}\textbf{variational}\\ \textbf{entropy}\end{pmatrix} - \begin{pmatrix}\textbf{variational}\\ \textbf{energy}\end{pmatrix}$$

Completely factorized (mean field) approximation

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

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

- The parameters μ_i are the mean values of S_i under $Q(H|V)$.

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

- This amounts to maximizing the lower bound:

$$\ln Z \geq \begin{pmatrix}\textbf{variational}\\ \textbf{entropy}\end{pmatrix} - \begin{pmatrix}\textbf{variational}\\ \textbf{energy}\end{pmatrix}.$$

- 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:

$$-\Sigma_H Q(H|V) \ln Q(H|V)$$

- 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:

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

Variational energy

- The variational energy is:

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

- In sigmoid networks:

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

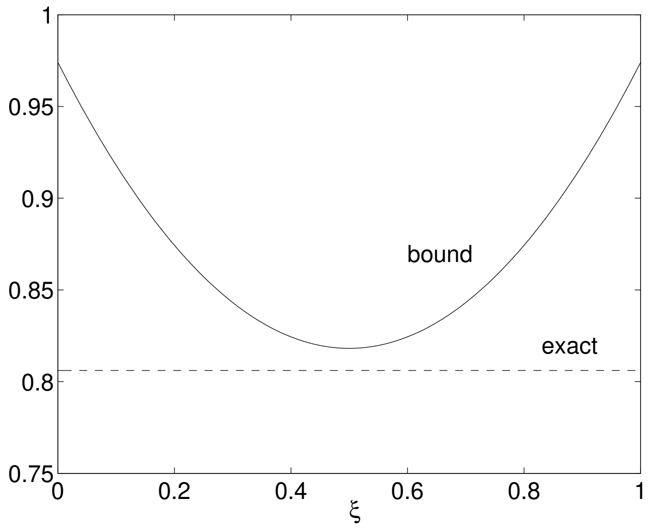
- The first two terms are common to undirected networks. **The last term is not**.

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

$$-\sum_{ij} J_{ij}\mu_i\mu_j - \sum_i h_i \mu_i + \sum_i \left\langle \ln\left[1 + e^{\Sigma_j J_{ij} S_j + h_i}\right]\right\rangle$$

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

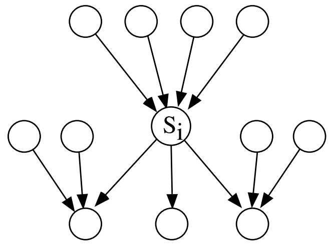
- **Lemma** (Seung): for any random variable z, and
any real number ξ:
$$\langle \ln[1+e^z] \rangle \leq \xi\langle z\rangle + \ln\left\langle e^{-\xi z} + e^{(1-\xi)z}\right\rangle.$$
- **Ex**: z is Gaussian with zero mean and unit variance.



- **Ex**: z is the sum of weighted inputs to S_i;
$$z = \sum_j J_{ij}S_j + h_i.$$
The lemma provides an upper bound on the intractable terms in the variational energy:
$$\sum_i \left\langle \ln\left[1+e^{\sum_j J_{ij}S_j + h_i}\right]\right\rangle$$

Variational (mean field) equations

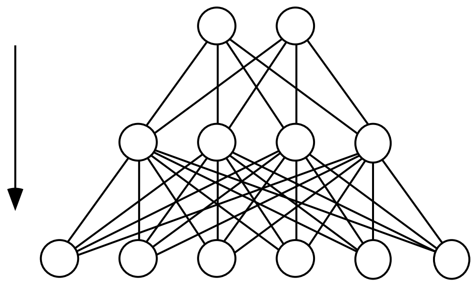
- The bound on the log-likelihood,
$$\ln Z \geq \begin{pmatrix}\textbf{variational}\\ \textbf{entropy}\end{pmatrix} - \begin{pmatrix}\textbf{variational}\\ \textbf{energy}\end{pmatrix},$$
is valid for any setting of the variational parameters, $\{\mu_i\}$
- The optimal $\{\mu_i\}$ are found by solving the variational equations:
$$\mu_i = \sigma\left(h_i + \sum_j\left[J_{ij}\mu_j + J_{ji}(\mu_j - \xi_j) - K_{ji}\right]\right)$$



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

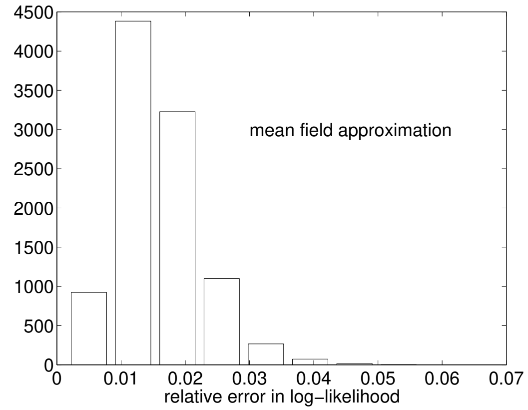
Numerical experiments

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

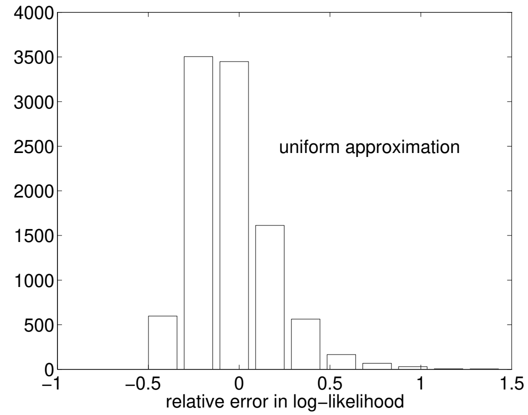


- We considered the event that all the units in the bottom layer were inactive.
- This was done for 10000 random networks whose weights and biases were uniformly distributed between -1 and 1.

- **Mean field approximation**:

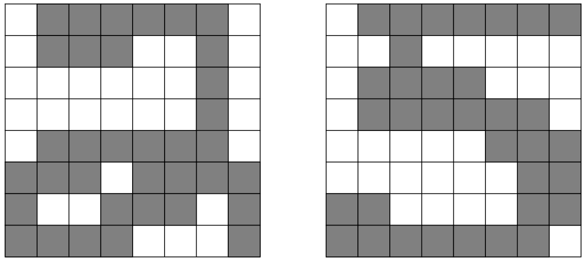


- **Uniform approximation**:



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

classification error: less than 5%

Extensions

- discrete-valued units

$$S_i \in \{1, 2, \ldots, n\}$$

- other activation functions

Ex: $P(S_i = 1|z) = \dfrac{1}{\sqrt{\pi}} \int^z dx\; e^{-x^2}$

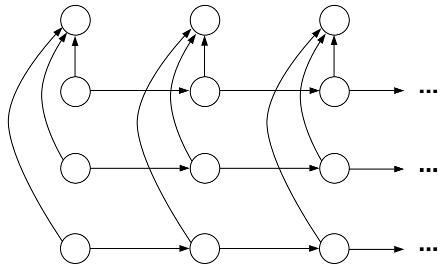
- continuous variables

Ex: $P\left(z_i | \{z_{j<i}\}\right)$ is non-gaussian

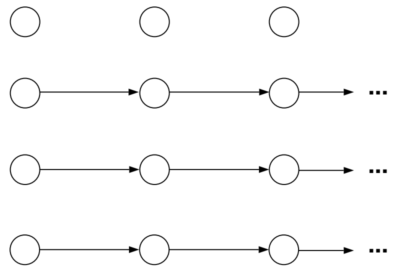
Example—Factorial HMM

(Ghahramani & Jordan, 1996)

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

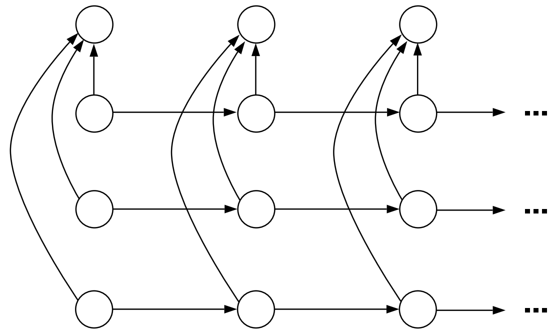


We can variationally transform this model into:



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

$$Q(\{S_t\}|\theta) = \prod_{m=1}^{M} Q(S_1^{(m)}|\theta) \prod_{t=2}^{T} Q(S_t^{(m)}|S_{t-1}^{(m)}, \theta),$$

where

$$\begin{aligned} Q(S_1^{(m)}|\theta) &= \pi^{(m)} h_1^{(m)} \\ Q(S_t^{(m)}|S_{t-1}^{(m)}, \theta) &= P^{(m)} h_t^{(m)}. \end{aligned}$$

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:

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

where

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

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

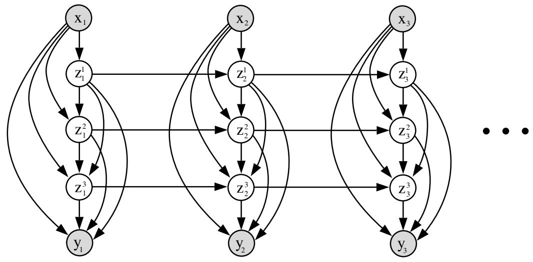
Repeat until convergence of $KL(Q\|P)$:

1. Compute $h_t^{(m)}$ using fixed-point equation, which depends on $\langle S_t^{(m)} \rangle$
2. Compute $\langle S_t^{(m)} \rangle$ using forward-backward algorithm on HMMs with observation log probabilities given by $h_t^{(m)}$

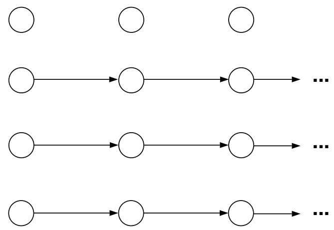
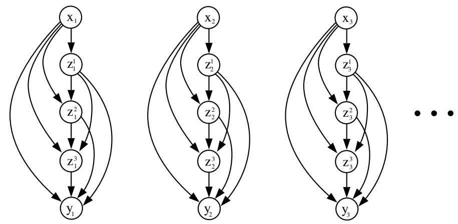
Example—hidden Markov decision tree

(Jordan, Ghahramani, & Saul, 1996)

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

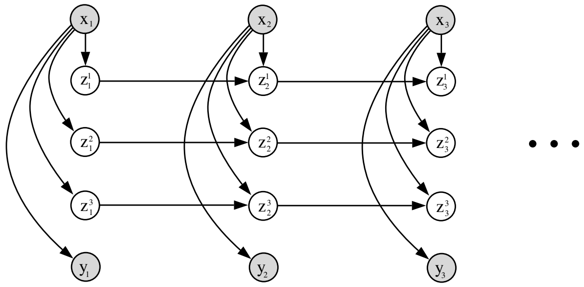


We can variationally transform this model into one of two simplified models:


or


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:

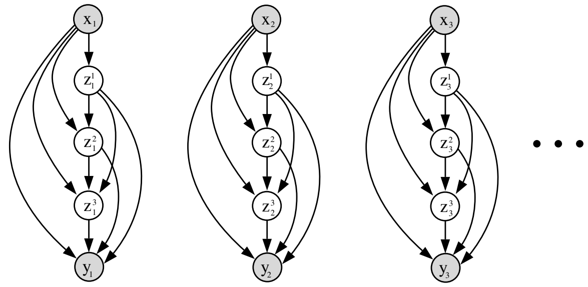
$$Q(\{\mathbf{z}_t^1, \mathbf{z}_t^2, \mathbf{z}_t^3\} \mid \{\mathbf{y}_t\}, \{\mathbf{x}_t\}) =$$
$$\frac{1}{Z_Q} \prod_{t=2}^{T} \tilde{a}_t^1(\mathbf{z}_t^1|\mathbf{z}_{t-1}^1)\tilde{a}_t^2(\mathbf{z}_t^2|\mathbf{z}_{t-1}^2)\tilde{a}_t^3(\mathbf{z}_t^3|\mathbf{z}_{t-1}^3)$$
$$\prod_{t=1}^{T} \tilde{q}_t^1(\mathbf{z}_t^1)\tilde{q}_t^2(\mathbf{z}_t^2)\tilde{q}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

  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

Recursive elimination framework

(Jaakkola & Jordan, 1996)

Exact	*Approximate*
1) Moralization	1) Variational transformation
2) Triangulation	2)
3) Junction tree	3)
4) Marginalization	4) Recursive marginalization

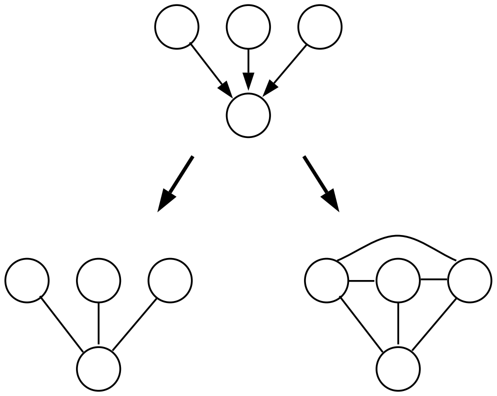
Recursive elimination has two steps:

- variational transformation
- recursive marginalization

Variational transformation

- Conditional probabilities are transformed variationally yielding upper or lower bounds:

$$P(S_i|\,\mathrm{pa}_i\,) \gtrless C(\lambda_i)P(S_i|\,\mathrm{pa}_i, \lambda_i\,)$$

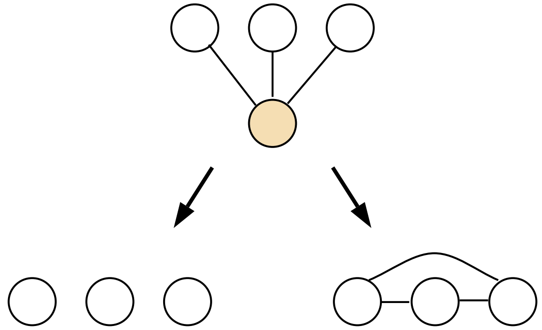


- cf. moralization
- The transformation retains compact representation of dependencies

Recursive marginalization

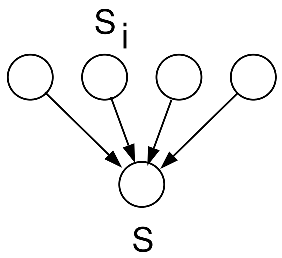
- Recursively eliminate variables from the network
- Each elimination consists of a variational transformation and consistently yields upper or lower bounds:

$$\sum_{S_n} P(S_1, \ldots, S_n) \gtrless C(\lambda_n)\; P(S_1, \ldots, S_{n-1}|\lambda_n)$$



- cf. triangulation

Upper bounds: example



- Dependency model from logistic regression:

$$P(S=1|\,\text{pa}\,) = g(\textstyle\sum_i W_i S_i)$$

where $g(x)$ is the logistic function

$$g(x) = \frac{1}{1+\exp\{-x\}}$$

- Problem: compute

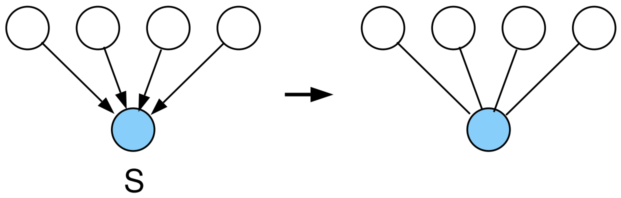
$$\begin{aligned} P(S=1) &= \sum_{S_i \in \text{pa}} P(S=1|\,\text{pa}\,) \prod_i P(S_i) \\ &= \sum_{S_i \in \text{pa}} g(\textstyle\sum_i W_i S_i) \prod_i P(S_i) \end{aligned}$$

Upper bounds: example

- Cannot compute exactly but we can obtain an upper bound using a variational transformation:

$$\begin{aligned} P(S=1) &= \sum_{S_i \in \text{pa}} g(\textstyle\sum_i W_i S_i) \prod_i P(S_i) \\ &\leq \sum_{S_i \in \text{pa}} e^{\lambda \sum_i W_i S_i - H(\lambda)} \prod_i P(S_i) \\ &= \sum_{S_i \in \text{pa}} e^{-H(\lambda)} \prod_i \left[e^{\lambda\, W_i S_i} P(S_i) \right] \end{aligned}$$

- The graphical consequence of this transformation is

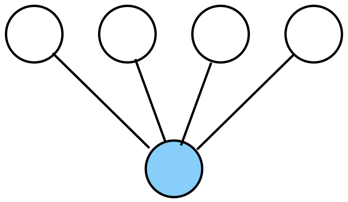


Upper bounds: example

- The marginalization step

$$P(S=1) \leq \sum_{S_i \in \mathrm{pa}} e^{-H(\lambda)} \prod_i \left[e^{\lambda W_i S_i} P(S_i) \right]$$
$$= e^{-H(\lambda)} \prod_i \left[P(S_i{=}1) e^{\lambda(W_i)} + P(S_i{=}0) \right]$$

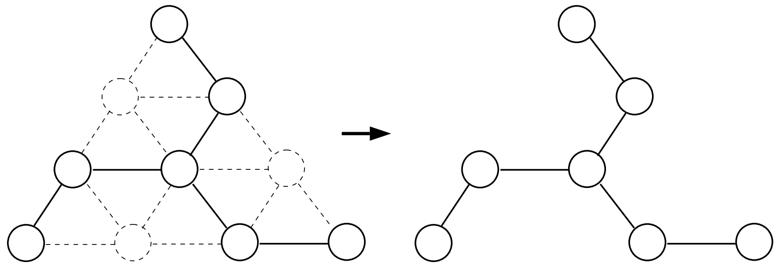
- This is simple because the variables are independent (conditionally on $S=1$)



Exact and approximate inference

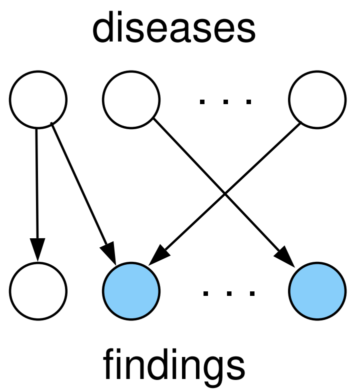
Exact	*Approximate*
1) Moralization	1) Variational transformation
2) Triangulation	2)
3) Junction tree	3)
4) Marginalization	4) Recursive marginalization

- These frameworks can be combined whenever the approximations yield a graph for which the exact methods are feasible



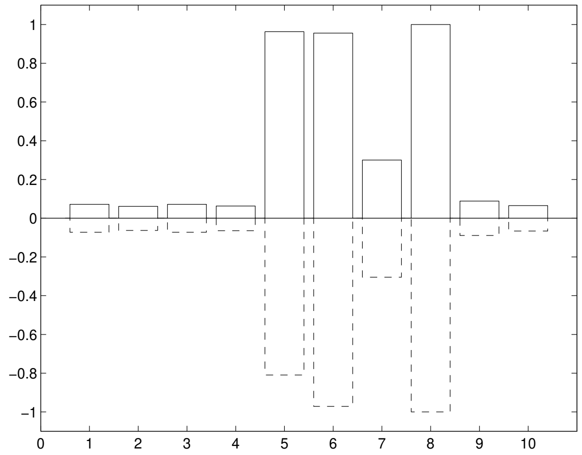
Preliminary results on QMR

- We want to compute the probabilities of diseases given some findings



- The dependency model is noisy-OR

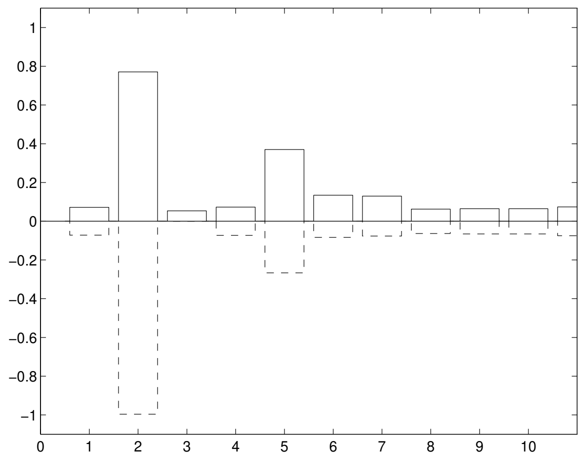
Preliminary results on QMR



The solid bars are exact disease marginals and the dashed ones are computed using a lower bound variational approximation

- To get rigorous bounds on the disease marginals we need an upper bound as well
- The simplest upper bounds are not accurate enough for this purpose

Preliminary results on QMR



Conclusions

- Three frameworks for probabilistic computation in graphical models:
  - exact computation
  - variational approximation
  - sampling-based approximation
- An advantage of variational approximations is that they yield upper and lower bounds on the desired probabilities
- The techniques can be combined to yield hybrid techniques which may well turn out to be better than any single technique

http

Home Page:
http://www.ai.mit.edu/projects/jordan.html

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].

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.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., and Freeman, D. (1988). Autoclass: A Bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, MI.

Poritz, A. M. (1988) Hidden Markov models: a guided tour. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing.* New York: IEEE Press. vol.1, 7-13.

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. (in press). Probabilistic independence networks for hidden Markov probability models. *Neural Computation.*

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984) *Classification and Regression Trees.* Wadsworth International Group, Belmont, CA.

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. (in press) Neural networks. *CRC Handbook of Computer Science.* CRC Press. [available from ftp://psyche.mit.edu/pub/jordan/crc.ps].

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

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

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

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.

Jordan, M. I., Ghahramani, Z., & Saul, L. K. (in press). 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., & 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.

Jaakkola, T., & Jordan, M. I. (in press). 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.