Uncertainty in Artificial Intelligence, 1995

A few relevant ideas from statistics, neural networks and statistical mechanics

Michael I. Jordan
Tommi Jaakkola
Lawrence Saul

MIT

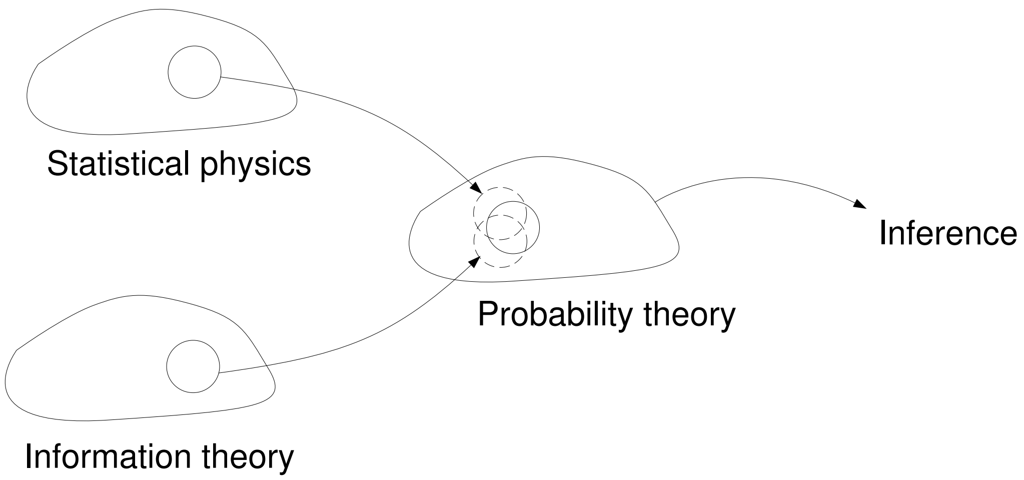
Outline

- The logistic function
- The EM algorithm
- Probabilistic decision trees
- Mean field theory for sigmoid belief networks

Formalisms

There is substantial agreement that probability theory is the best framework available in which to develop learning algorithms.

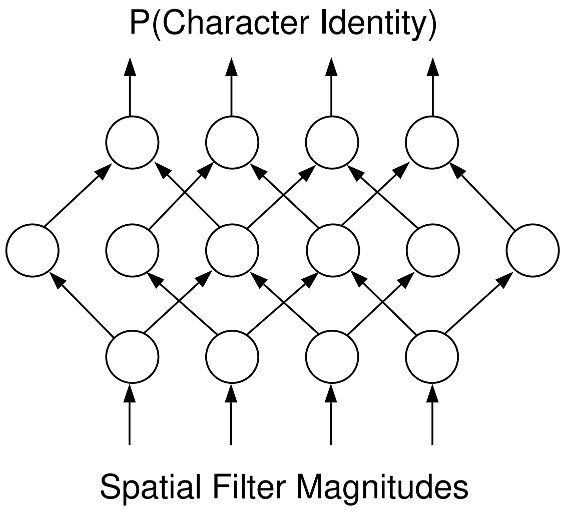
That doesn't mean that we always have to work with probabilities directly. One can work with energies (statistical physics) or codes (information theory) and convert the results to probabilities using standard theorems. Sometimes energies or codes are easier to work with.



(circles: those models that are relatively easy to specify and understand)

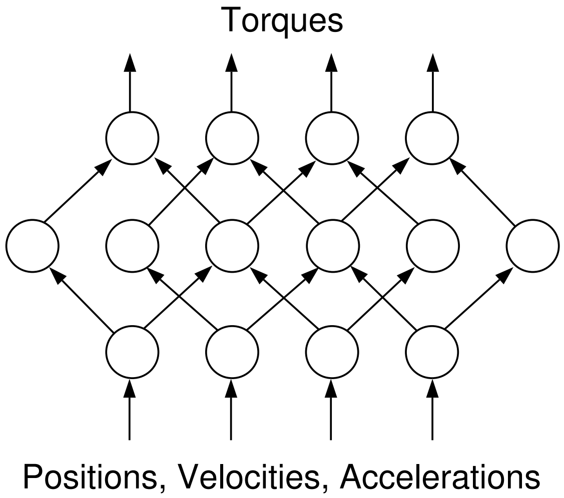
Neural network examples

Character recognition



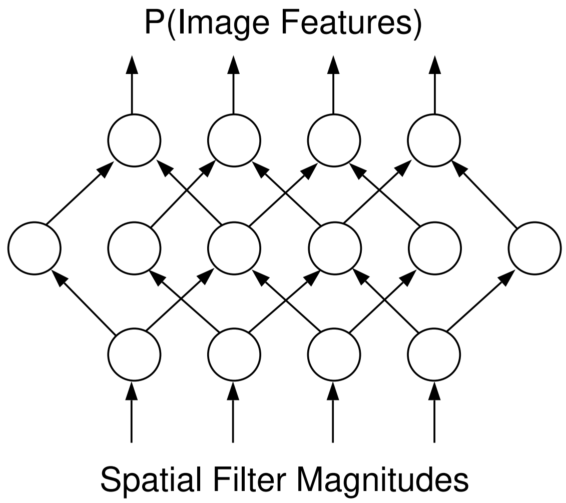
(cf. LeCun, et al., 1989)

Control



(cf. Sandler, et al., 1991)

Anomaly detection



(cf. Tarassenko, 1995)

Generative/Causal/Class-conditional
vs.
Diagnostic/Discriminative/Predictive

- neural nets are typically specified in the diagnostic direction (but cf. some "unsupervised learning")
- belief nets are typically specified in the generative direction (but cf. some "chain graphs")

A Clintonesque view: The diagnostic and generative directions are both important.

The goal of the following discussion is to convince you that diagnostic and generative are two sides of the same coin.

Outline

- $\boxed{\text{The logistic function}}$
- The EM algorithm
- Probabilistic decision trees
- Mean field theory for sigmoid belief networks

Binary classification

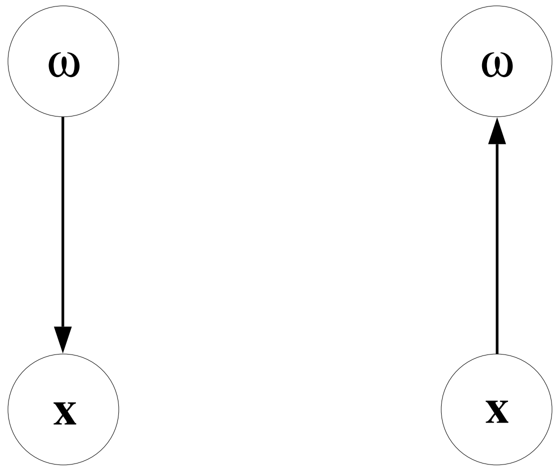
Class label:

$$\omega \in \{\omega_0, \omega_1\}$$

Feature vector:

$$\mathbf{x} = [x_1, x_2, \ldots, x_d]^T$$

Two possible belief network representations:



A B

Parameterizations

In belief network A, we must specify $P(\omega)$ and $P(\mathbf{x}|\omega)$.

In belief network B, we must specify $P(\mathbf{x})$ and $P(\omega|\mathbf{x})$.

- However, since in classifying new examples we will be providing $\mathbf{x}$ as evidence, we will not bother with modeling $P(\mathbf{x})$.

Posterior probabilities

Given a particular vector $\mathbf{x}$, we wish to assign it to one of the two classes.

Bayes rule:

$$P(\omega_0|\mathbf{x}) = \frac{P(\mathbf{x}|\omega_0)P(\omega_0)}{P(\mathbf{x})}$$

Continuing the derivation, we expand the denominator and (with some foresight) introduce an exponential:

$$\begin{aligned} P(\omega_0|\mathbf{x}) &= \frac{P(\mathbf{x}|\omega_0)P(\omega_0)}{P(\mathbf{x}|\omega_0)P(\omega_0) + P(\mathbf{x}|\omega_1)P(\omega_1)} \\ &= \frac{1}{1 + \exp\left\{-\log\left[\frac{P(\mathbf{X}|\omega_0)}{P(\mathbf{X}|\omega_1)}\right] - \log\left[\frac{P(\omega_0)}{P(\omega_1)}\right]\right\}} \end{aligned}$$

We see that the posterior probability can be written in the form of the *logistic function*:

$$y = \frac{1}{1 + e^{-\xi}},$$

where ξ is a function of the likelihood ratio

$$P(\mathbf{x}|\omega_0)/P(\mathbf{x}|\omega_1)$$

and the prior ratio

$$P(\omega_0)/P(\omega_1)$$

Under what conditions will the likelihood ratio be simple?

Class-conditional distributions

In network A, we need the conditional probabilities $P(\mathbf{x}|\omega)$.

Let us assume that they are multivariate Gaussian with identical covariance matrices Σ:

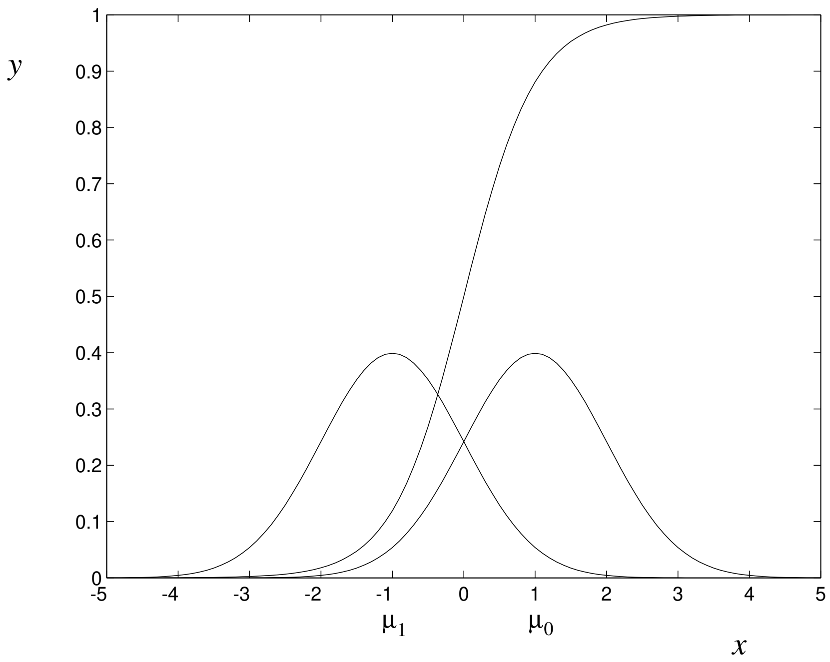
$$P(\mathbf{x}|\omega_i) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^T\Sigma^{-1}(\mathbf{x}-\mu_i)}$$

where μ_i is the mean vector for class i.

Expanding the exponent of the Gaussians, and substituting into the formula for the posterior, we obtain:

$$P(\omega_0|\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}+b)}},$$

where $\mathbf{w}$ and b are simple functions of Σ and μ_i.



A binary classification problem. The class-conditional densities are Gaussians with unit variance. The posterior probability is the logistic function $y = 1/(1+\exp(-2x))$.

Belief network B

The conditional probability $P(\omega|\mathbf{x})$ must be specified.

It should be a logistic-linear function of $\mathbf{x}$ to be consistent with our Gaussian class-conditional assumption.

The network requires $n+1$ numbers on the link from $\mathbf{x}$ to ω. (Belief network B is a neural network).

Advantages of Network A:

- modularity
- causality
- invariance of the conditional to $P(\omega)$

Are there *any* advantages to the $\mathbf{x} \rightarrow \omega$ direction?

The exponential family

Suppose that we investigate class-conditional densities other than Gaussians (e.g., gamma, binomial, Poisson, negative binomial, hypergeometric, etc.). What is the form of the posterior in each of these cases?

All of these distributions are in the *exponential family*:

$$P(\mathbf{x}|\theta, \phi) = \exp\{(\theta^T \mathbf{x} - b(\theta))/a(\phi) + c(\mathbf{x}, \phi)\}$$

Let us assume our two class-conditional densities are in the exponential family with equal dispersion parameters ϕ.

Substituting into the form of the posterior, we find that we obtain the logistic-linear function in all cases.

- i.e., network B doesn't require us to specify a specific class-conditional density
- the conditional distribution is *invariant* across the exponential family
- in statistical terms, network B is more robust than network A

Summary

(cf. ftp://psyche.mit.edu/pub/jordan/uai.ps)

statistical advantage of network A: *efficiency*

statistical advantage of network B: *robustness*

- (cf. McLachlan, 1992, for Fisher discriminant analysis vs. logistic regression)

Each diagnostic architecture will have an accompanying family of generative architectures and vice versa. It is important to have some appreciation of the family that is being assumed on the other side of the coin while working with a particular architecture.

- sensitivity of diagnostic law to generative perturbations
- generality of generative law across diagnostic perturbations

Addendum—logistic functions in hidden layers of neural networks

Can be viewed as "sigmoid belief networks" (Neal, 1992)

- neural networks with belief network semantics

Recent work on learning in sigmoid belief networks:

- Hinton, Dayan, Frey, & Neal (1995)
- Saul, Jaakkola, & Jordan (1995)

Outline

- The logistic function
- $\boxed{\text{The EM algorithm}}$
- Probabilistic decision trees
- Mean field theory for sigmoid belief networks

Learning

- from data and a specification of

$$P(data|model, params)$$

  and

$$P(params|model)$$

  and

$$P(model),$$

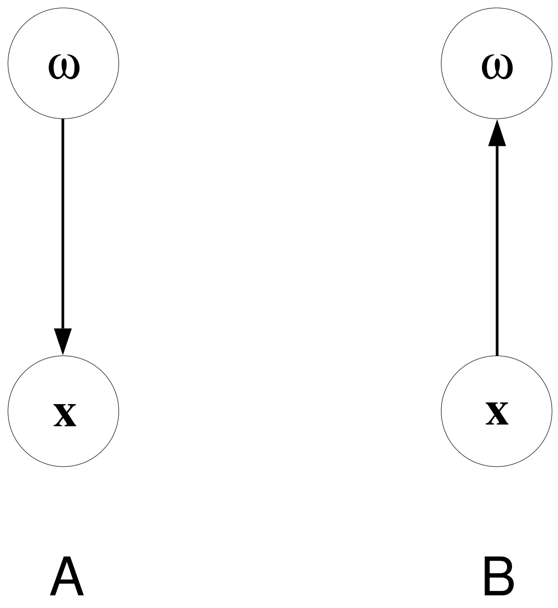
- form various posterior probabilities, e.g.:

$$P(params|data, model) \propto P(data|params, model) \\ P(params|model)$$

- and
  - maximize,
  - integrate via approximating, or
  - integrate via sampling from

  your favorite posterior

Linear parameter estimation example



Suppose we have a *training set* of $\{\mathbf{x}, \omega\}$ pairs:

$$\mathcal{X} = \{\mathbf{x}^{(p)}, \omega^{(p)}\}_{p=1}^{N}$$

We wish to set the parameters of our network to maximize the log likelihood (flat priors):

$$\log P(\mathcal{X}|\Theta) = \Sigma_{p=1}^{N} \log P(\mathbf{x}^{(p)}, \omega^{(p)}).$$

Network A

- Split the data set into those pairs for which $\omega = \omega_0$ and those pairs for which $\omega = \omega_1$
- Estimate the parameters of $P(\mathbf{x}|\omega)$ separately in the two cases
- E.g., for Gaussians, estimate the means and pooled covariance matrix

Network B

- Define:

$$z_p = \begin{cases} 1 & \omega^{(p)} = \omega_0 \\ 0 & \text{otherwise} \end{cases}$$

  and

$$y_p = P(\omega^{(p)} = \omega_0 | \mathbf{x}^{(p)}),$$

- Then the log likelihood is:

$$\begin{aligned} l &= \log \Pi_{p=1}^{N} y_p^{z_p} (1 - y_p)^{(1-z_p)} \\ &= \Sigma_{p=1}^{N} [z_p \log y_p + (1 - z_p) \log(1 - y_p)] \end{aligned}$$

which is the negative *cross entropy* between z_p and y_p.

We wish to minimize the cross entropy with respect to $\mathbf{w}$.

Two methods for minimizing the cross entropy:

- *gradient descent*

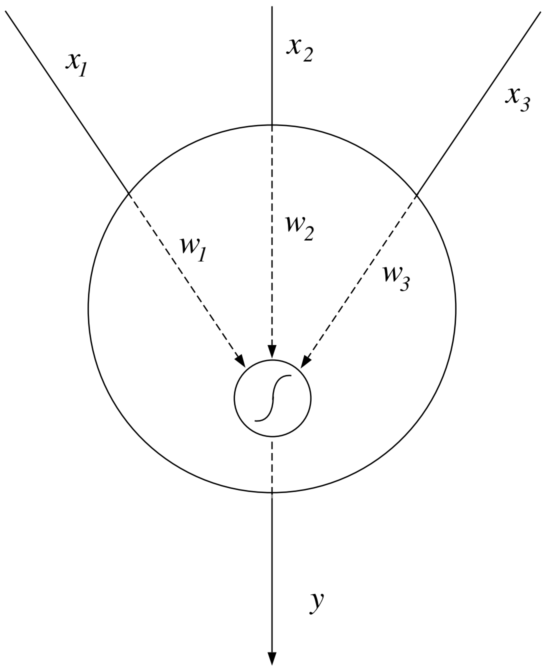
$$\Delta \mathbf{w} = \mu \Sigma_{p=1}^{N} (z_p - y_p) \mathbf{x}^{(p)}$$

- *Newton-Raphson*

$$\Delta \mathbf{w} = (X^T V_r X)^{-1} X^T V_r \mathbf{z}^*$$

The latter is known as Iteratively Reweighted Least Squares (IRLS).

Summary—a logistic-linear node



Logistic model of the posterior probability:

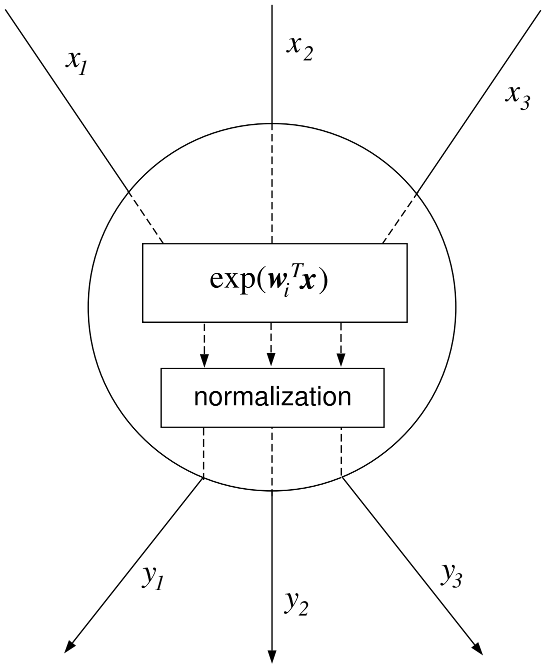
$$y = \frac{1}{1 + e^{-\xi}}$$

$$\xi = \mathbf{w}^T \mathbf{x}$$

IRLS algorithm to minimize cross entropy:

$$\Delta \mathbf{w} = (X^T V_r X)^{-1} X^T V_r \mathbf{z}^*$$

Generalization to multinomial model



Softmax model of the posterior probabilities:

$$y_i = \frac{e^{\xi_i}}{\sum_j e^{\xi_j}}$$

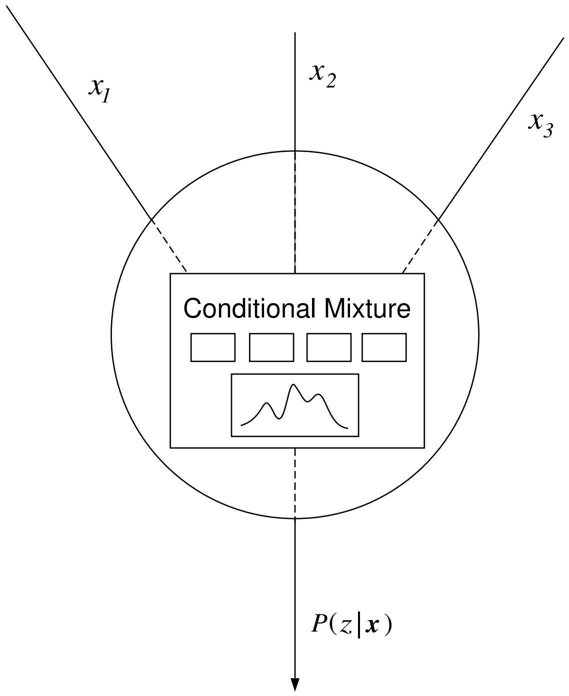
$$\xi_i = \mathbf{w}_i^T \mathbf{x}$$

IRLS algorithm to minimize cross entropy:

$$\Delta \mathbf{w}_i = (X^T V_{i,r} X)^{-1} X^T V_{i,r} \mathbf{z}_i^*$$

Generalization to mixture model

("Mixture of experts", Jacobs, et al., 1991)



Conditional mixture model for posterior:

$$P(z|\mathbf{x}) = \sum_i g_i(\mathbf{x}; \mathbf{v}_i) P(z_i|\mathbf{x}; \mathbf{w}_i)$$

EM algorithm to minimize mixture likelihood:

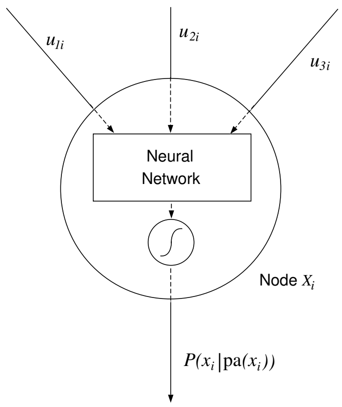
$$\Delta \mathbf{w}_i = (X^T \Gamma_{i,r} X)^{-1} X^T \Gamma_{i,r} \mathbf{z}_i^*$$

(cf. probabilistic decision trees)

Belief network/neural network hybrids: completely observed case

(cf. loglinear/Gaussian models, e.g., Whittaker, 1990)

- Replace the probability table at each node in a belief network with a neural network (or GAM, probabilistic decision tree, etc.).



  Smooths the probability table (cf. noisy-OR).

- A set of N complete observations of the belief network yields a training set for each node:

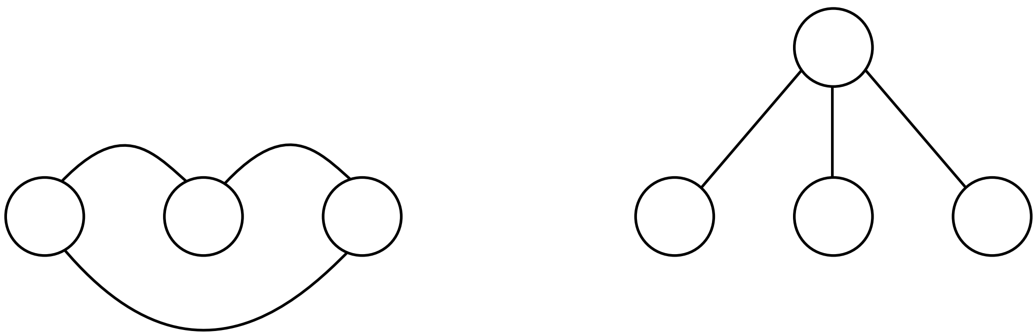
$$\mathcal{X}_i = \{\mathrm{pa}(X_i^{(p)}), X_i^{(p)}\}_{p=1}^{N}$$

- Utilize IRLS or gradient or EM method at each node to adjust its parameters

Hidden variables—the EM algorithm

(Dempster, Laird, & Rubin, 1977; Lauritzen, 1992)

- The goal is to maximize the likelihood (or posterior probability) of the parameters given the evidence. But suppose that the likelihood is complicated, and suppose further that introducing a set of hidden variables can simplify the likelihood.



The EM algorithm

- Introduce the hidden variables into the probability model, which from $P(V|\theta)$ becomes $P(V, H|\theta)$.
- (E step): Using the conditional distribution $P(H|V, \theta^{(t)})$, take the expected value of the hidden variables given the evidence $V = v$ and the current parameter values $\theta^{(t)}$.
- (M step): Pretend that the expected values of the hidden variables calculated in the E step were actually observed. Update the parameters using the algorithm for the completely observed case. The new model is $P(H, V|\theta^{(t+1)})$.
- Iterate.

This procedure guarantees *monotone convergence* in likelihood.

Application to belief networks

- Suppose evidence v is observed on nodes $V \subseteq S$. Define $H = S \setminus V$.
- Use, e.g., Hugin to compute $P(H|V = v)$.
- Use, e.g., IRLS to update the probability model at each node.
- Iterate.

Gradient methods are also applicable (Russell, Binder, & Koller, 1994), but EM has some advantages: it is usually faster, it provides estimates of uncertainty, it doesn't require a step size, it guarantees monotone convergence (see, e.g., Meng & Rubin, 1991; Xu & Jordan, 1995).

Outline

- The logistic function
- The EM algorithm
- $\boxed{\text{Probabilistic decision trees}}$
- Mean field theory for sigmoid belief networks

EM and probabilistic decision trees

Belief Networks

$\uparrow$

Conditional Mixture Models

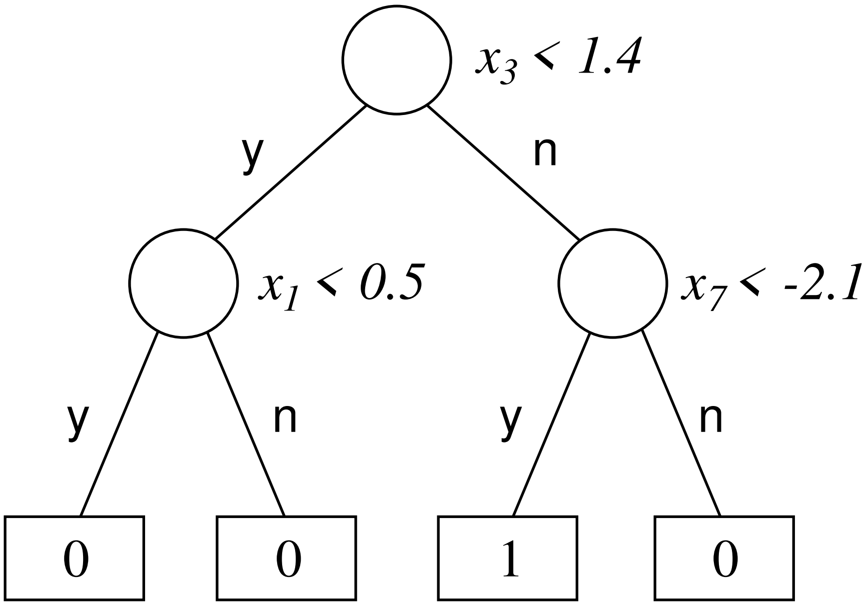
$\uparrow$

Decision Trees

- Decision trees are generally developed without probabilistic considerations (but see Buntine, 1991).
- I will describe a method for probabilizing decision trees that yields an efficient EM algorithm.
- The algorithm is to trees what Baum-Welch (HMM's) is to chains.
- We convert decision tree problems into (conditional) mixture problems. Can also convert mixtures into (decomposable) belief networks.

Classical decision trees

(CART, C4.5)



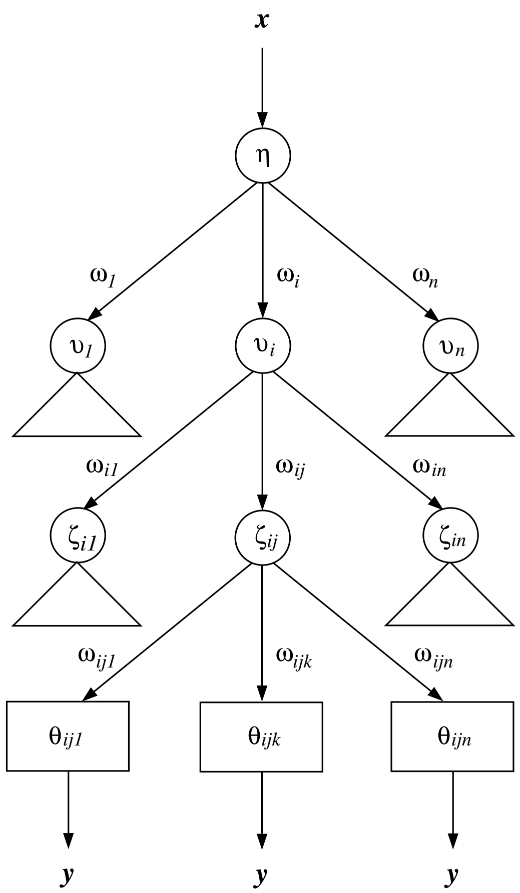
- drop the data set down the tree
- at each node, try to find a split of the input space (a half-plane) that yields the largest gain in "purity" on left and right
- build a large tree and prune backward to create a nested sequence of trees
- pick the best tree from the sequence using cross-validation

Probabilistic Decision Trees

(Jordan & Jacobs, 1994)

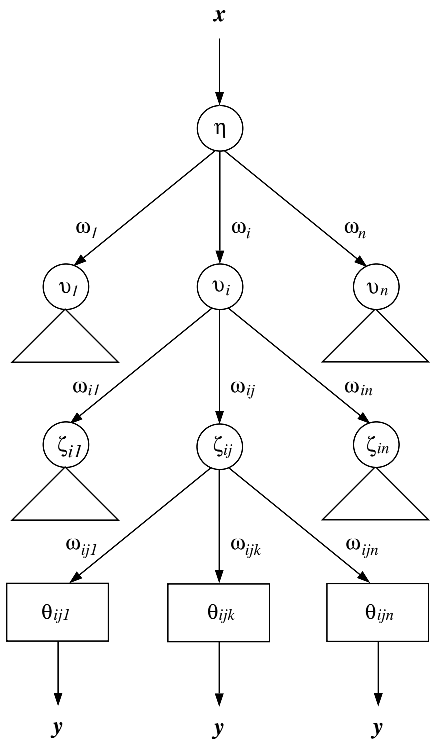
- drop inputs down the tree and use probabilistic models for decisions (splits)
- at leaves of trees use probabilistic models to generate outputs from inputs
- use a Bayes' rule recursion to compute posterior credit for nonterminals in the tree

The basic idea is to convert the decision tree into a conditional mixture model



Model the splits in the decision tree using categorical probability models

- let $\omega_i, \omega_{ij}, \omega_{ijk}, \ldots$ represent multinomial split variables at the nonterminals
- these variables will be treated as "hidden" variables (cf. states of an HMM)
- each path down the tree defines a component of a conditional mixture



Splitting models at the nonterminals:

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

Output models at the leaves:

$$
P(\mathbf{y}|\mathbf{x}, \omega_i, \omega_{ij}, \omega_{ijk} \ldots)
$$

Likelihood of the tree is a mixture

The total probability of an output $\mathbf{y}$ given an input $\mathbf{x}$ is given by the sum across all paths from the root to the leaves:

$$
\begin{aligned}
P(\mathbf{y}|\mathbf{x},\Theta) \;=\; & \Sigma_i P(\omega_i|\mathbf{x}) \Sigma_j P(\omega_{ij}|\mathbf{x},\omega_i) \\
& \Sigma_k P(\omega_{ijk}|\mathbf{x},\omega_i,\omega_{ij}) \cdots \\
& P(\mathbf{y}|\mathbf{x},\omega_i,\omega_{ij},\omega_{ijk}\ldots)
\end{aligned}
$$

This is a (conditional) mixture model:

$$
P(\mathbf{y}|\mathbf{x},\Theta) \;=\; \Sigma_{ijk\ldots} P(\Omega_{ijk\ldots}|\mathbf{x}) P(\mathbf{y}|\mathbf{x},\Omega_{ijk\ldots})
$$

Component Models

Split models (nonterminals)

- $P(\omega_i|\mathbf{x}, \eta)$ is a classification model
- any parametric classification model is appropriate—we use a multinomial logit ("softmax") model

  this yields "soft" linear discriminants—soft version of a CART/C4.5 tree

Leaf models

- we use simple generalized linear models:

  Regression—linear regression

  Binary classification—logistic regression

  Multiway classification—multinomial logit model

  (can also handle count estimates, failure estimates, etc.)

Applying EM to the HME decision tree

- The hidden variables are the unknown values of the decisions (splits) in the decision tree.

  Define indicator variables z_i, $z_{j|i}$, $z_{k|ij}$, ...

Complete likelihood:

$$l_c(\Theta, \mathcal{Y}) = \Sigma_p \Sigma_i z_i^{(p)} \Sigma_j z_{j|i}^{(p)} \cdots \log[g_i^{(p)} g_{j|i}^{(p)} \cdots P_{ijk\cdots}(\mathbf{y}^{(p)}|\mathbf{x}^{(p)})$$

Incomplete likelihood:

$$l(\Theta, \mathcal{X}) = \Sigma_p \log[\Sigma_i g_i^{(p)} \Sigma_j g_{j|i}^{(p)} \cdots P_{ijk\cdots}(\mathbf{y}^{(p)}|\mathbf{x}^{(p)})]$$

- We need to compute the expected values of the hidden indicator variables, e.g.:

$$E(z_i^{(p)}|\mathbf{x}^{(p)}, \mathbf{y}^{(p)}) = P(\omega_i^{(p)}|\mathbf{x}^{(p)}, \mathbf{y}^{(p)})$$

Posterior splitting probabilities

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

(cf. *prior* splitting probabilities)

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

Bayes' rule yields:

$$h_i = \frac{g_i \Sigma_j g_{j|i} \Sigma_k g_{k|ij} P_{ijk}(\mathbf{y}|\mathbf{x})}{\Sigma_i g_i \Sigma_j \Sigma_j g_{j|i} \Sigma_k g_{k|ij} P_{ijk}(\mathbf{y}|\mathbf{x})}$$

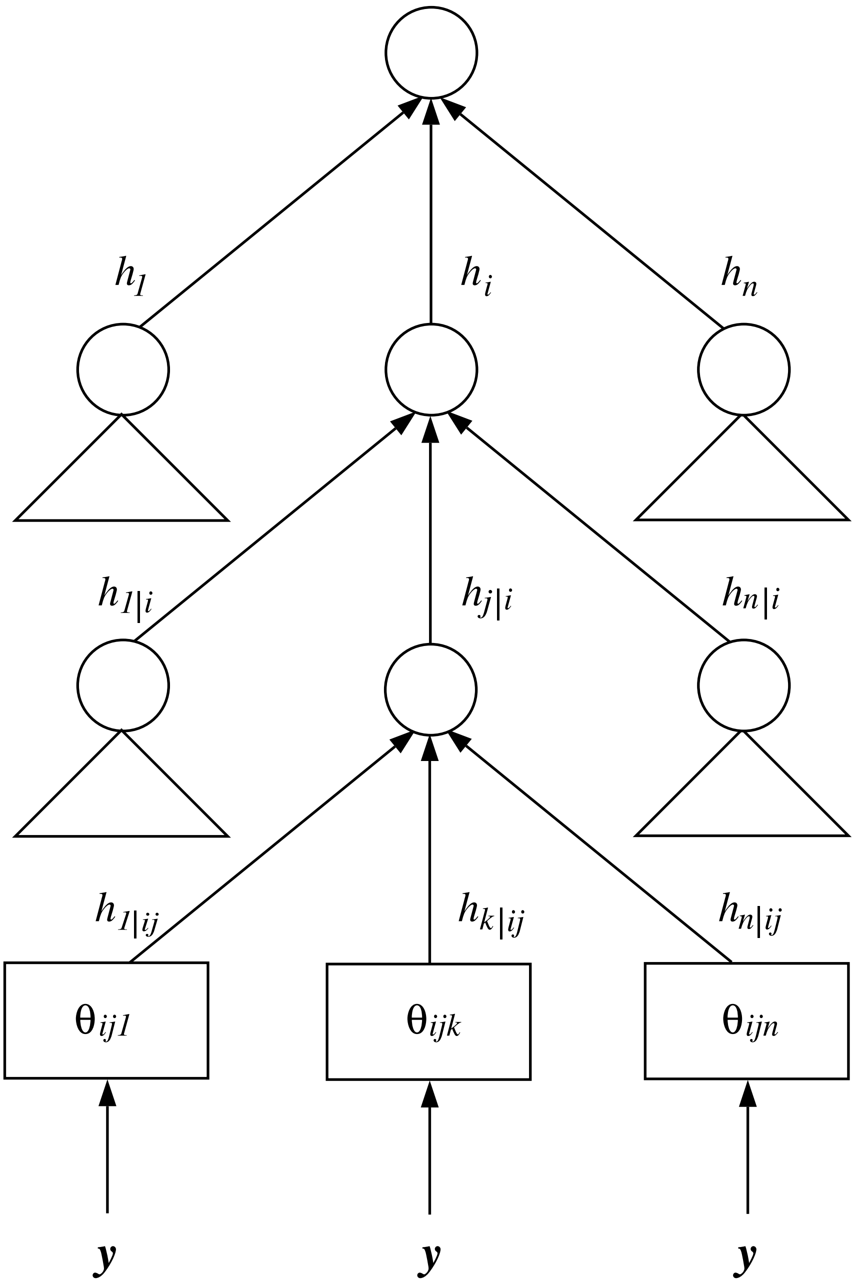
$$h_{j|i} = \frac{g_{j|i} \Sigma_k g_{k|ij} P_{ijk}(\mathbf{y}|\mathbf{x})}{\Sigma_j g_{j|i} \Sigma_k g_{k|ij} P_{ijk}(\mathbf{y}|\mathbf{x})}$$

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

$$\cdots$$

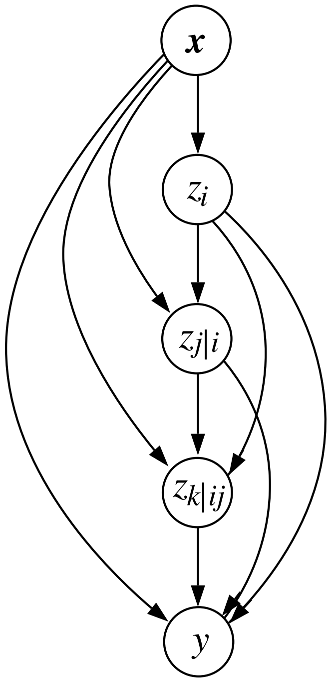
Note that the denominator at each level appears in the numerator at the next level up. We can organize the computation as a recursion.

Posterior propagation



This is the E step of the EM algorithm

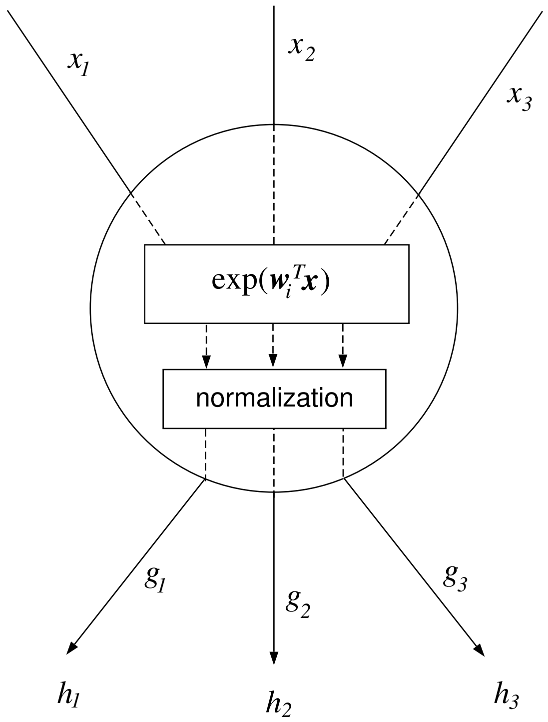
Belief network representation of the E step



Dropping the arrows, we see that the network is already moralized and triangulated, i.e., the conditional mixture is decomposable

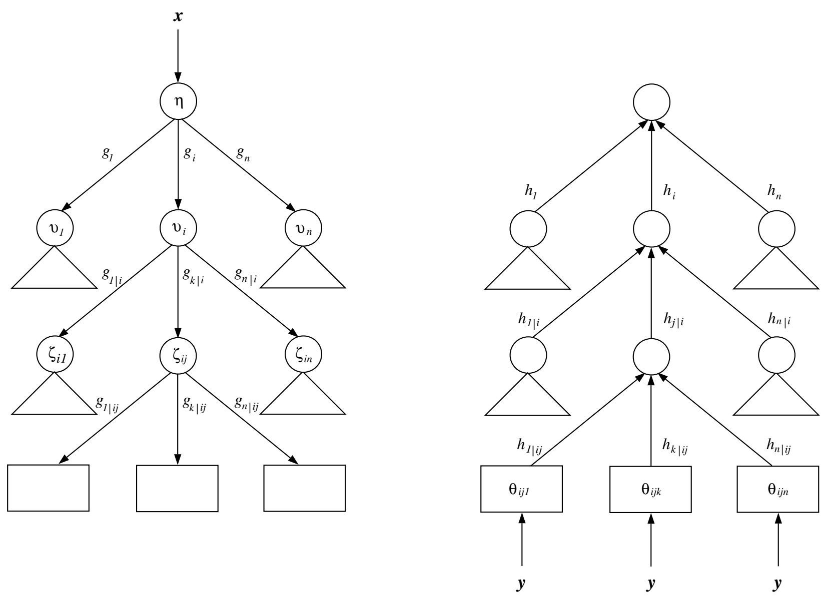
The posterior propagation is decimation of this decomposable distribution

The M step



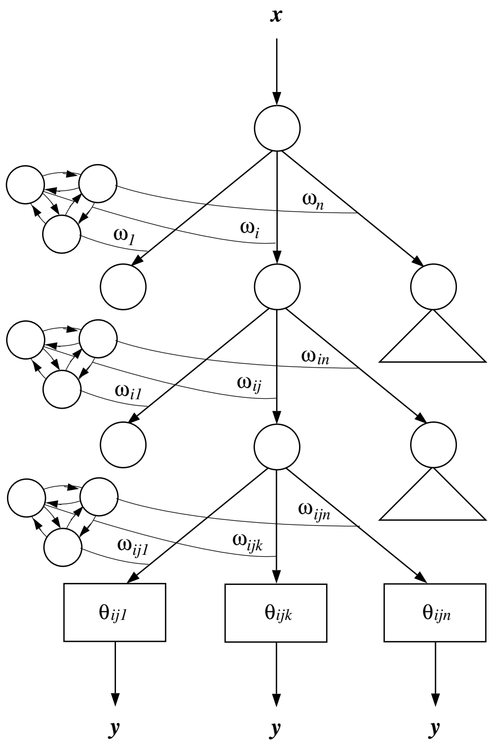
- The M step decouples into a set of separate maximum likelihood problems
- At each nonterminal, use IRLS, with the posteriors $h_i^{(t)}$, $h_{j|i}^{(t)}$, etc., serving as the targets
- At the leaves, obtain weighted likelihoods where the weights are the product of the posteriors from root to leaf

Summary—HME Parameter Estimation



- drop the data set down the tree
- for each data point, compute the posterior probabilities for every branch of the tree
- at each nonterminal, use the posterior probabilities as (soft) classification targets
- at each leaf, fit a local model, where each data point is weighted by the product of the posterior probabilities from the root to that leaf

Addendum—Hidden Markov Decision Trees



- Each decision at a node is dependent on the decision at the previous moment at that node
  - This yields a Markov model at each node

- An EM algorithm can be derived, treating the Markov states as hidden variables
  - It combines a forward-backward pass with an up-down pass

Outline

- The logistic function
- The EM algorithm
- Probabilistic decision trees
- Mean field theory for sigmoid belief networks

Mean field theory for sigmoid belief networks

(Saul, Jaakkola, & Jordan, 1995)

- A methodology for approximating joint and conditional probabilities in belief networks.
- Provides a lower bound on the log likelihood.

One page primer on statistical mechanics

- N random variables: $\{S_i\}$
- *configuration* α: $\{S_i = s_i^\alpha\}$
- *energy* of configuration α: $E_\alpha \equiv E(\{s_i^\alpha\})$
- *Boltzmann factor*: $\exp\{-E_\alpha\}$
- *partition function*: $Z \equiv \Sigma_\alpha \exp\{-E_\alpha\}$
- *probability* of configuration α:

$$P_\alpha = \frac{\exp\{-E_\alpha\}}{Z}$$

- *free energy*:

$$F \equiv \Sigma_\alpha P_\alpha E_\alpha + \Sigma_\alpha P_\alpha \log P_\alpha$$

- Fundamental Theorem:

$$F = -\log Z$$

probabilities $\rightarrow$ energies

In physics one usually starts with energies and goes to probabilities (via $P_\alpha = \exp\{-E_\alpha\}/Z$).

In the application to belief nets, however, we are given a probability distribution from the dag's conditional probabilities:

$$P_\alpha = \Pi_i P(S_i^\alpha | \mathrm{pa}(S_i^\alpha))$$

We will compile these into energies as follows:

$$E_\alpha = -\log P_\alpha$$

This implies that initially our partition function Z is unity:

$$Z = \Sigma_\alpha \exp\{-E_\alpha\} = \Sigma_\alpha P_\alpha = 1$$

Moreover, these energies are defined on the moral graph:

$$\begin{aligned} E_\alpha &\equiv -\log P_\alpha \\ &= -\log \left[\Pi_i P(S_i^\alpha|\mathrm{pa}(S_i^\alpha))\right] \\ &= \Sigma_i \left[-\log P(S_i^\alpha|\mathrm{pa}(S_i^\alpha))\right] \end{aligned}$$

which is a sum across terms defined on the cliques of the moral graph.

I.e., we don't triangulate (we're only after an approximation to P_α)

Evidence and likelihoods

Suppose evidence v is observed on nodes $V \subseteq S$. Define $H = S \setminus V$. Denote the likelihood as Z_e.

The likelihood is the probability of the evidence:

$$\begin{aligned} Z_e &= P(V = v) \\ &= \Sigma_H P(V = v, H) \end{aligned}$$

We now show that the likelihood is a partition function and therefore the log likelihood is a negative free energy.

Consider the conditional probability of the hidden nodes given the evidence:

$$
\begin{aligned}
P(H|V=v) &= \frac{P(V=v,H)}{P(V=v)} \\
&= \frac{\exp\{-E_{\alpha,V=v}\}}{Z_e}
\end{aligned}
$$

where $E_{\alpha,V=v}$ is the energy of the α^{th} hidden state for fixed evidence $V=v$:

$$
\begin{aligned}
E_{\alpha,V=v} &= \Sigma_{i\in H}\left[-\log P(S_i^{\alpha}|\mathrm{pa}(S_i^{\alpha}))\right] \\
&\quad +\Sigma_{i\in V}\left[-\log P(S_i=v_i|\mathrm{pa}(S_i))\right]
\end{aligned}
$$

Because $E_{\alpha,V=v}$ has "absorbed evidence," the partition function Z_e:

$$Z_e = \Sigma_\alpha \exp\{-E_{\alpha,V=v}\}$$

is no longer unity.

We could try to calculate Z_e or $\log Z_e$ directly by doing the sum (cf. Hugin), or approximate the sum in some way.

Our approach is to use the fundamental theorem:

$$-\log Z_e = \Sigma_\alpha P_\alpha E_{\alpha,V=v} + \Sigma_\alpha P_\alpha \log P_\alpha$$

and calculate separately the approximations in energy and entropy.

Sigmoid belief networks

A sigmoid belief network is a binary dag, with logistic-linear probabilities:

$$P(S_i = 0|\mathrm{pa}(S_i)) = \frac{1}{1 + e^{\Sigma_{j \in \mathrm{pa}(S_i)} J_{ij} S_j}}$$

where the J_{ij} are the parameters. (Henceforth $J_{ij} = 0$ for $j \notin \mathrm{pa}(S_i)$.)

Given that

$$P(S_i = 1|\mathrm{pa}(S_i)) = \frac{e^{\Sigma_j J_{ij} S_j}}{1 + e^{\Sigma_j J_{ij} S_j}},$$

we can write both probabilities at once as:

$$P(S_i|\mathrm{pa}(S_i)) = \frac{e^{S_i \Sigma_j J_{ij} S_j}}{1 + e^{\Sigma_j \in J_{ij} S_j}}.$$

Energy for sigmoid belief networks

The joint probability for the belief network:

$$
\begin{aligned}
P(\{S_i\}) &= \Pi_i \frac{e^{S_i \Sigma_j J_{ij} S_j}}{1 + e^{\Sigma_j J_{ij} S_j}} \\
&= \frac{\exp\{\Sigma_{ij} J_{ij} S_i S_j\}}{\Pi_i \left[1 + e^{\Sigma_j J_{ij} S_j}\right]}
\end{aligned}
$$

Recalling the initialization for the energies:

$$E_\alpha = -\log P_\alpha,$$

we obtain the energy for a sigmoid belief network:

$$E_\alpha = -\Sigma_{ij} J_{ij} S_i^\alpha S_j^\alpha - \Sigma_i \log\left[1 + \exp\{\Sigma_j J_{ij} S_j^\alpha\}\right]$$

(cf. Markov random fields, where the first term usually appears, but the second term doesn't).

Approximations

Given that P_α is intractable (by assumption), we look for an approximation Q_α that is tractable.

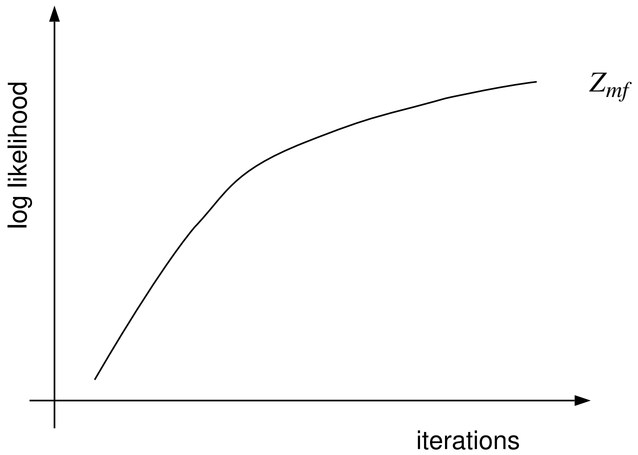
We will insist that within our approximation, we obtain an upper bound on the true free energy (i.e., a lower bound on the log likelihood).

Why?

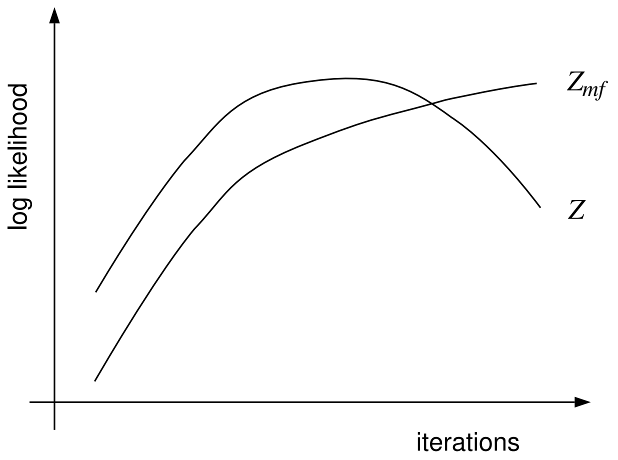
Approximations

Recall that the EM algorithm guarantees monotone convergence of the log likelihood.

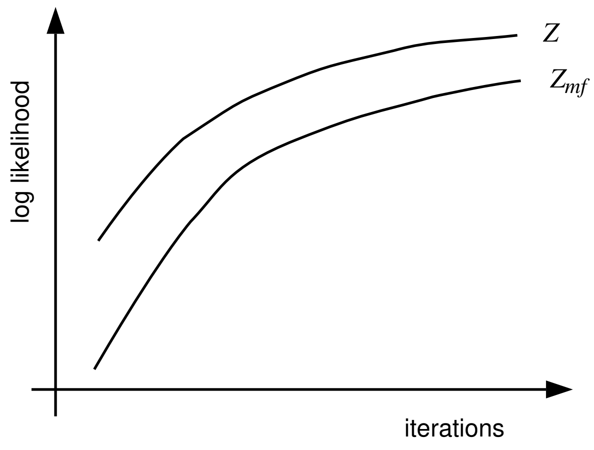
Thus, using Q_α in the EM algorithm, we can guarantee that an approximate log likelihood increases as we iterate.

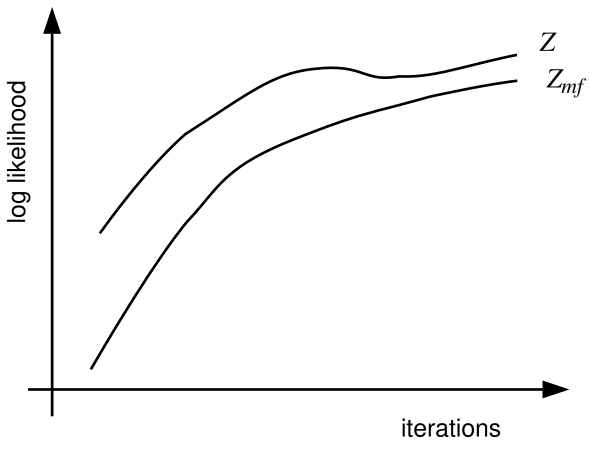


Without a guarantee of a bound, the true log likelihood could do anything:



With the guarantee of a bound, the true log likelihood may either always increase, or decrease towards the approximate log likelihood:





This latter is actually not a bad thing, as we see by calculating the difference between the true log likelihood and the approximate log likelihood:

$$
\begin{aligned}
\log Z - \log Z_{mf} &= F_{mf} - F \\
&= \Sigma_\alpha Q_\alpha E_\alpha + \Sigma_\alpha Q_\alpha \log Q_\alpha \\
&\quad -\Sigma_\alpha P_\alpha E_\alpha - \Sigma_\alpha P_\alpha \log P_\alpha \\
&= \Sigma_\alpha Q_\alpha \log \frac{Q_\alpha}{P_\alpha} \\
&= KL(Q_\alpha || P_\alpha) \\
&> 0 \qquad \text{(proof of lower bound).}
\end{aligned}
$$

Given that Q_α is "simple," this implies that a decrease in the true log likelihood is a form of regularization (Geoff Hinton, p.c.).

Naive mean field approximation

Consider the following approximation:

$$Q\left(\{S_i\}_{i\in H}|V=v\right)=\prod_{i\in H}\mu_i^{S_i}(1-\mu_i)^{1-S_i},$$

where the μ_i are parameters that are at our disposal to make the approximation as tight as possible.

We minimize the approximation error $KL(Q_\alpha||P_\alpha)$ and obtain the following *mean field equations*:

$$\mu_i = g(\Sigma_{j\in H}\left[J_{ij}+J_{ji}\right]\mu_j + h_i^t - \Sigma_j\left[\xi_j J_{ji} + K_{ij}\lambda_j\right]),$$

where $g(\cdot)$ is the logistic function. (See Saul, Jaakkola, & Jordan, 1995 for the derivation).

Structured mean field approximation

Suppose the true distribution has the form

$$P_\alpha = \frac{1}{Z} \exp \left\{ \Sigma_{ij} J_{ij} S_i S_j + \Sigma_i h_i S_i + \Sigma_{ij} K_{ij} S_i S_j \right\},$$

where the weights J_{ij} define a tractable skeleton network and the extra weights K_{ij} spoil this tractability.

Consider the following approximation:

$$Q_\alpha = \frac{1}{Z_Q} \exp \left\{ \Sigma_{ij} J_{ij} S_i S_j + \Sigma_i \hat{h}_i S_i \right\},$$

where we choose the parameters $\hat{h}_i$ to minimize the approximation error $KL(Q_\alpha || P_\alpha)$.

If for $K_{ij} \to 0$, the network factorizes into independent tractable substructures, then we obtain the mean field equations:

$$\hat{h}_i = h_i + \Sigma_{ij} K_{ij} \langle S_j \rangle_Q.$$

Mixture mean field approximation

Consider a mixture of mean field approximations (cf. DNF):

$$Q(\{S_i\}_{i\in H}|V=v) = \Sigma_m q_m \prod_{i\in H} \mu_{mi}^{S_i}(1-\mu_{mi})^{1-S_i}$$

where q_m are the mixture coefficents. This approximation allows us to capture correlations between the variables.

Minimizing $KL(Q_\alpha||P_\alpha)$ yields "mean field" equations:

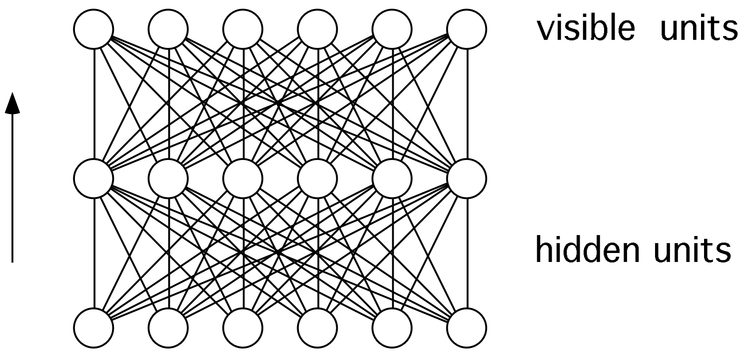
$$\mu_{mi} = g\left(\cdots + \log\frac{r_{mi}}{1-r_{mi}} + \Sigma_n \lambda_n A_{nm}(i)(2r_{mi}-1)\right)$$

$$q_m = \frac{\exp\{-\tilde{F}_{mf}(m)\}}{\Sigma_n \exp\{-\tilde{F}_{mf}(n)\}}$$

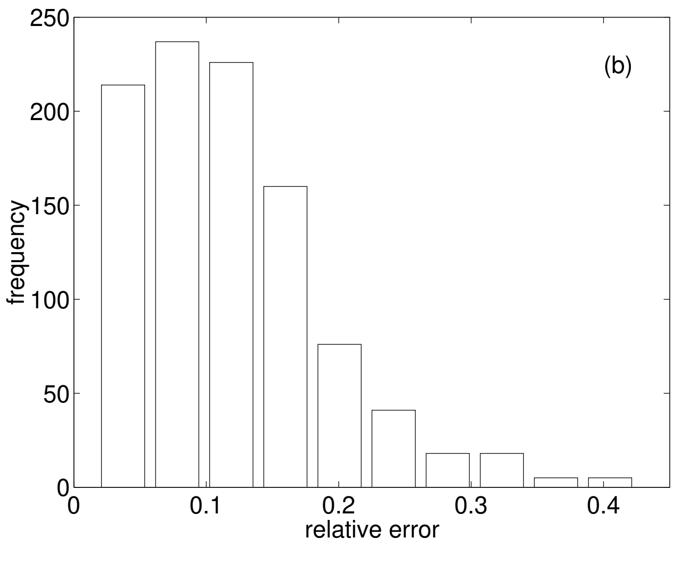
where the dots denote the usual mean field terms and $\tilde{F}_{mf}$ is a corrected mean field free energy.

Empirical results—random networks

1250 random networks of the following structure were generated:

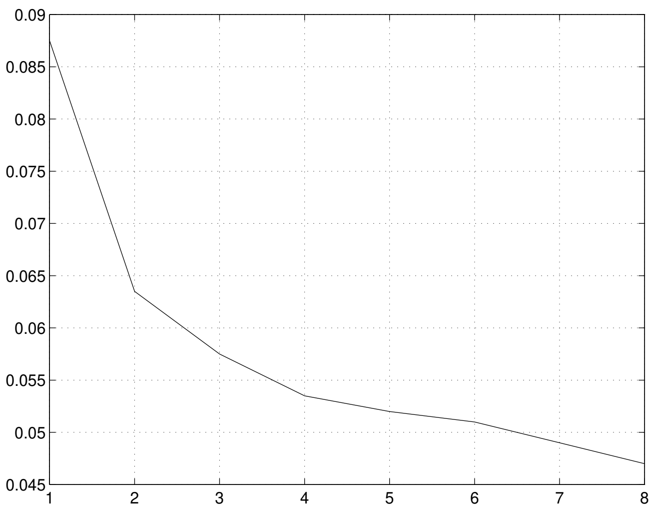


Histogram of relative error (ground truth was calculated by exact enumeration):



Empirical results—digit recognition

- 8x8 images of handwritten digits
- 7000 images were used for training
- 4000 images were used for testing
- separate generative model for each digit, using three layer architectures



Error rate compares favorably to standard learning algorithms

Conclusions

- belief networks with neural networks or decision trees in the internal nodes
  - flexible smoothing of probability tables
  - limited role, interpretability not hurt?
- neural networks or decision trees with belief network semantics
  - supervised/unsupervised learning
  - new applications for belief networks
  - new methods for approximations
- EM as a unifying framework
  - exact methods
  - approximate methods
- causal BN's and discriminative NN's as two sides of same coin

ftp and http

handout on logistic function:

ftp://psyche.mit.edu/pub/jordan/uai.ps

probabilistic decision trees:

ftp://psyche.mit.edu/pub/jordan/hierarchies.ps

mean field theory:

ftp://psyche.mit.edu/pub/lksaul/belief.ps

Home Page:

$\boxed{\text{http://www.ai.mit.edu/projects/jordan.homepage}}$