Learning in Belief Networks

- **Known structure, fully observable**
- **Known structure, hidden variables**
- **Unknown structure, fully observable**
- **Unknown structure, hidden variables**

Density Estimation

Given: a set of random variables $U = \{V_1, \ldots, V_n\}$

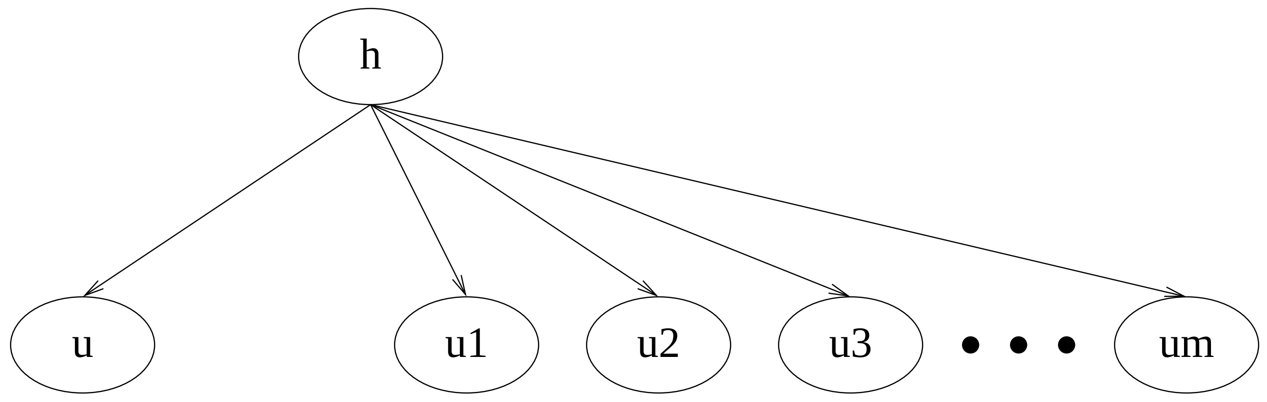
a set S of training examples $u^1, \ldots, u^m$ drawn according to unknown distribution $\mathbf{P}(U)$,

where $u^i = \langle v_1^i, \ldots, v_n^i \rangle$.

a space H of probability models $P(U|\Theta)$ defined by parameters Θ.

Find: Predict the probability $P(u)$ of a new data point u.

This task is known as *density estimation*, because we are trying to estimate a probability density (or probability distribution) $P(U)$.



Bayesian and Quasi-Bayesian Learning Theory

Fundamental Question: Given S and H, how to choose h?

Full Bayesian Answer: Don't Choose!

- **Define a prior probability:** $P(h)$
- **Consider predicting** $P(u)$ **for new example** u:

$$\begin{aligned}
P(u|S) &= \sum_h P(u,h|S) \\
&= \sum_h \frac{P(u,h,S)}{P(S)} \\
&= \sum_h \frac{P(u|h)\cdot P(S|h)\cdot P(h)}{P(S)} \\
&= \sum_h P(u|h)\cdot \textsc{normalize}[P(S|h)\cdot P(h)]
\end{aligned}$$

- **Interpretation: Compute a weighted vote where each hypothesis** h **votes according to its** *posterior probability*, $P(h|S)$. This is called "Bayesian Model Averaging".

Bayesian Learning Theory (2)

Approximating the full Bayesian approach by the single most-likely h: **MAP – Maximum Aposteriori Probability**

Let h^* be the hypothesis with the highest posterior probability:

$$h^* = \operatorname*{argmax}_h \ \textsc{normalize}[P(S|h)P(h)] = \operatorname*{argmax}_h P(S|h)P(h).$$

Approximate the sum over h with just this single hypothesis:

$$\begin{aligned} P(u|S) &= \sum_h P(u|h) \cdot \textsc{normalize}[P(S|h)P(h)] \\ P(u|S) &\approx P(u|h^*) \end{aligned}$$

Bayesian Learning Theory (3)

If $\mathbf{P}(h)$ is the uniform distribution, then we obtain the **Maximum Likelihood Estimate (MLE)**:

$$h^* = \operatorname*{argmax}_h P(S|h)P(h) = \operatorname*{argmax}_h P(S|h)$$

In terms of a hypothesis space H parameterized by Θ, the maximum likelihood estimate $\hat{\Theta}$ is

$$\hat{\Theta} = \operatorname*{argmax}_\theta P(S|\theta).$$

The function $P(S|\Theta)$ is called the *likelihood function*.

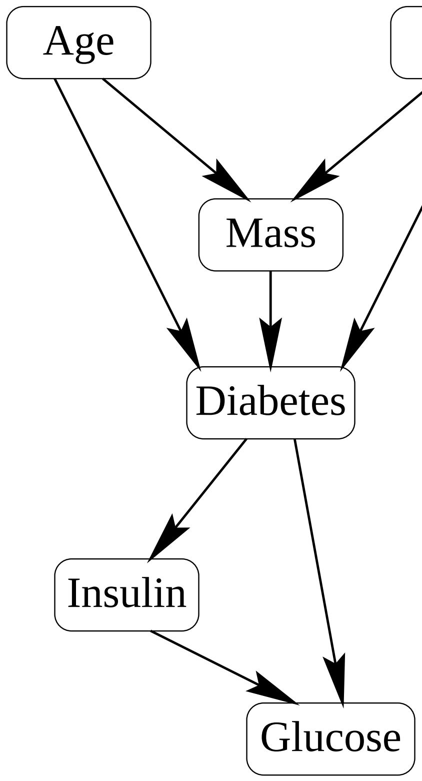
If the training examples are independent, then $P(S|\hat{\Theta}) = \prod_i P(u^i|\hat{\Theta})$.

To optimize $P(S|\hat{\Theta})$ we can optimize $\log P(S|\hat{\Theta}) = \sum_i \log P(u^i|\hat{\Theta})$.

Experimental Methodology

- **Collect Data**
- **Divide data into Training and Testing subsets randomly**
- **Choose $\hat{\Theta}$ using Training data**
- **Evaluate log likelihood on Test data**

Known structure, Fully Observable



Preg	Glucose	Insulin	Mass	Age	Diabetes
5	121	112	26.2	30	0
10	101	180	32.9	63	0
7	137	0	32.0	39	0
12	100	105	30.0	46	0
9	140	0	32.7	45	1
1	102	0	39.5	42	1
2	99	160	36.6	21	0
2	174	120	44.5	24	1
1	111	0	32.8	45	0
5	117	105	39.1	42	0

The parameters $\hat{\Theta}$ are the entries in the CPT's of the 6 nodes of the belief network.

Learning Process

- **Discretize the Data**

  Glucose $< 100 \Rightarrow 0$

  $100 \leq$ Glucose $< 120 \Rightarrow 1$

  $120 \leq$ Glucose $< 140 \Rightarrow 2$

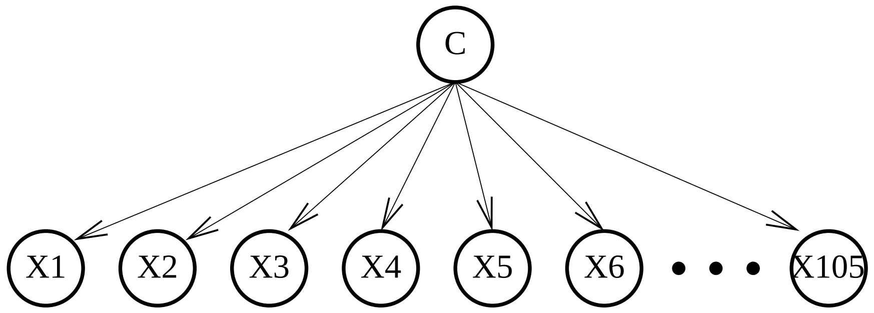
  $140 \leq$ Glucose $\Rightarrow 3$

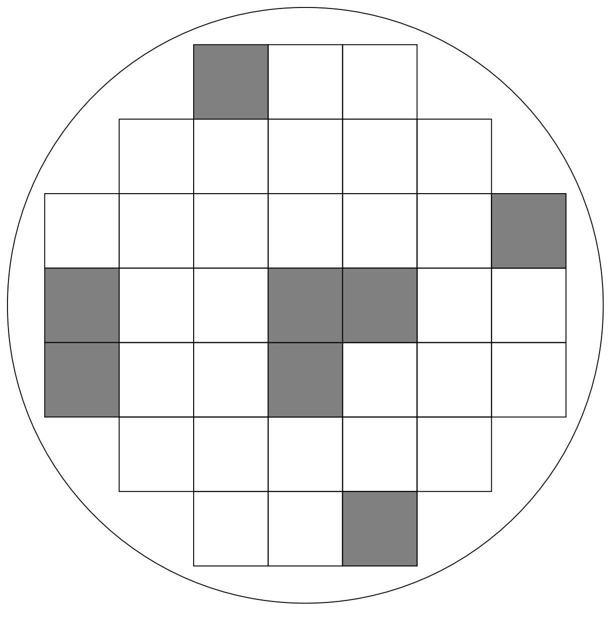
- **Count Cases**

  $$P(Mass = 0|Preg = 1, Age = 2) = \frac{N(Mass = 0, Preg = 1, Age = 2)}{N(Preg = 1, Age = 2)}$$

Known Structure, Hidden Variables

Simplest Case: Finite Mixture Model with unknown mixing proportions. $|C| = 4$.





Complete Data and Incomplete Data

Wafer	$X1$	$X2$	$\cdots$	$X105$	C
1	1	1	$\cdots$	0	?
2	0	1	$\cdots$	1	?
3	0	1	$\cdots$	1	?
4	1	0	$\cdots$	1	?

The given data are incomplete. If we could assign a value for C to each example, then we would have complete data, and learning would be trivial.

Hard EM

Let $W = (X1, X2, \ldots, X105)$ be the observed wafer.

- **Guess initial values for** C (e.g., randomly)
- **Repeat until convergence**
  - **Hard M-Step:** (Compute Maximum Likelihood Estimates from complete data)
    Compute $\mathbf{P}(C)$
    Compute $\mathbf{P}(Xi|C)$ for all i
  - **Hard E-Step:** (Re-estimate the C values.)
    For each wafer, set C to maximize $P(W|C)$

In remote sensing, this is known as the ISODATA clustering algorithm.

$$\max_C P(W|C) = \max_C \log P(W|C) = \max_C \log \prod_i P(Xi|C) = \max_C \sum_i \log P(Xi|C)$$

Hard-EM Example

True Model:

$\mathbf{P}(C)$	
0	0.58
1	0.42

$P(Xi = 1\|C)$	0	1
X1	0.34	0.41
X2	0.19	0.83
X3	0.20	0.15
X4	0.69	0.19
X5	0.57	0.53
X6	0.71	0.93
X7	0.34	0.68
X8	0.43	0.04
X9	0.13	0.65
X10	0.14	0.89

Draw 100 training examples and 100 test examples.

Training Data

C	$X1$	$X2$	$X3$	$X4$	$X5$	$X6$	$X7$	$X8$	$X9$	$X10$
1	1	1	0	1	1	1	1	0	0	1
0	0	1	0	1	1	1	1	1	0	0
0	0	0	1	1	1	1	0	0	0	1
0	1	0	0	0	1	1	0	0	0	0
1	1	1	0	0	1	1	1	0	1	1
0	0	0	1	1	0	1	1	1	0	0
1	1	1	0	1	1	1	1	0	0	1
0	0	0	0	1	1	1	0	1	0	0
0	0	0	0	1	0	0	1	0	0	0
					...					

Note that C is actually hidden in the training data.

The maximum likelihood class of example 7 is actually 0, but its true class is 1. (Probabilities are 0.9725 vs. 0.0275.)

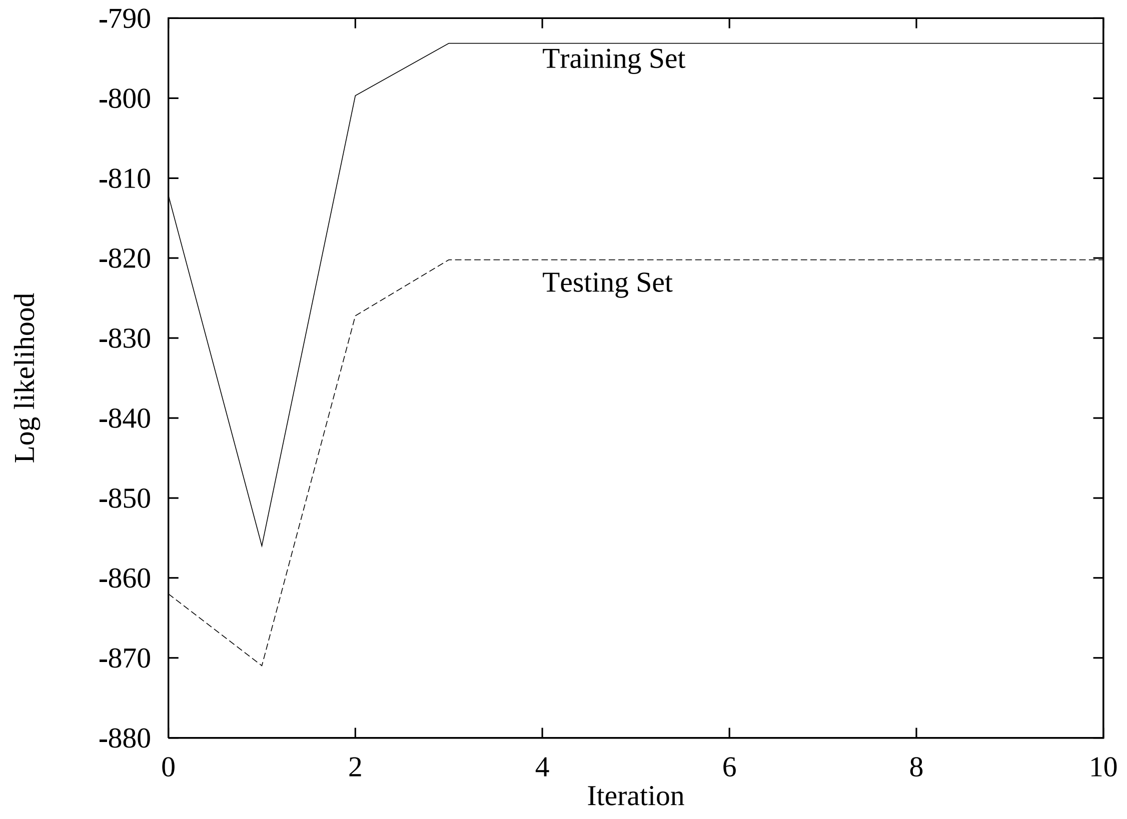
Hard-EM: Fit of Model to Training Data (including true C)

$\mathbf{P}(C)$	
0	0.61
1	0.39

$P(Xi = 1 \mid C)$	0	1
$X1$	0.28	0.41
$X2$	0.15	0.85
$X3$	0.15	0.13
$X4$	0.67	0.23
$X5$	0.49	0.51
$X6$	0.74	0.97
$X7$	0.39	0.69
$X8$	0.34	0.03
$X9$	0.10	0.67
$X10$	0.16	0.87

Hard-EM could achieve this fit if it correctly guessed the identity of the underlying classes.

Hard-EM Training Curve



Hard-EM Fitted Model

$\mathbf{P}(C)$	
0	0.43
1	0.57

$P(Xi = 1\|C)$	0	1
$X1$	0.35	0.32
$X2$	0.81	0.12
$X3$	0.09	0.18
$X4$	0.26	0.68
$X5$	0.60	0.42
$X6$	0.95	0.74
$X7$	0.65	0.40
$X8$	0.02	0.37
$X9$	0.67	0.05
$X10$	0.86	0.12

Note that the classes are "reversed", in that the learned class 0 corresponds to the true class 1. But since the true class is never observed, this is irrelevant. The likelihoods are the same.

Search Can Get Stuck in Local Minima

The problem here is that some parameters went to zero.

$\mathbf{P}(C)$	
0	0.93
1	0.07

$P(Xi = 1 \mid C)$	0	1
$X1$	0.35	0.00
$X2$	0.42	0.43
$X3$	0.12	0.43
$X4$	0.47	0.86
$X5$	0.53	0.14
$X6$	0.83	0.86
$X7$	0.51	0.57
$X8$	0.16	1.00
$X9$	0.34	0.00
$X10$	0.47	0.00

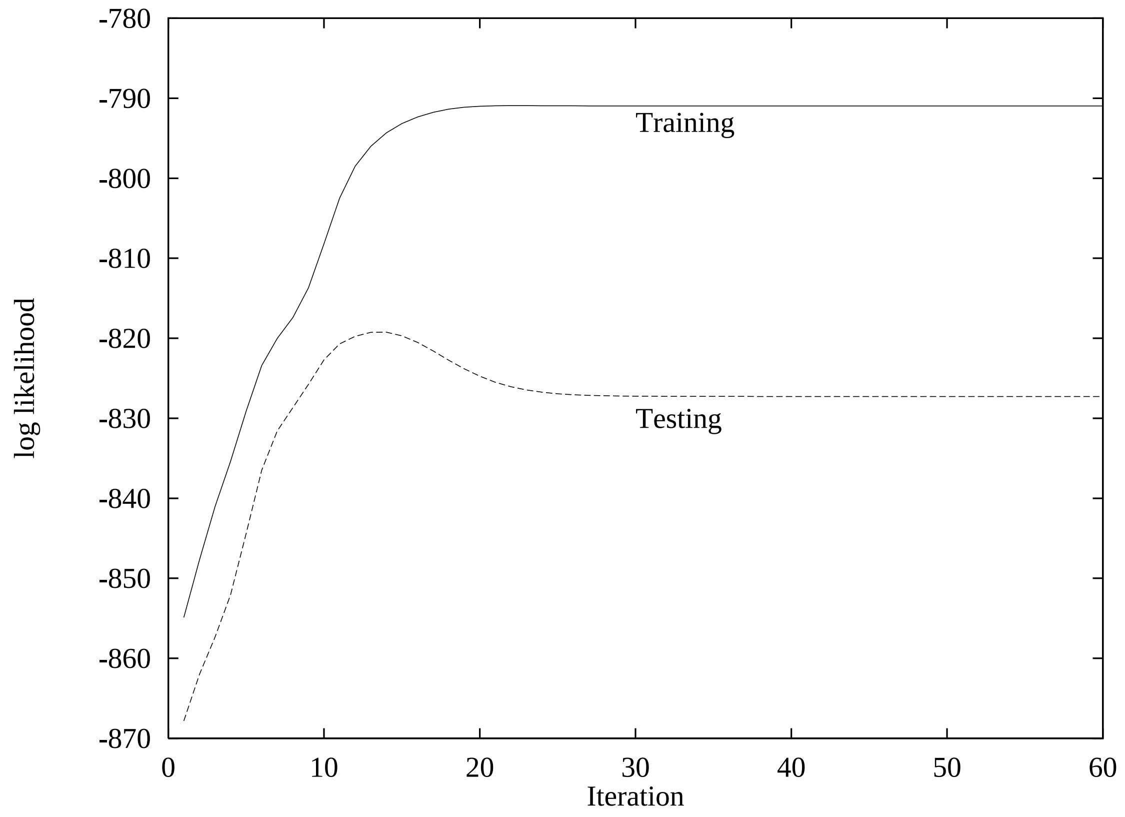
The Expectation-Maximization (EM) Algorithm

For Maximum Likelihood or Penalized Maximum Likelihood.

```
Initialize $\Theta$ randomly.
**repeat** until convergence
    **E-Step:** For each wafer, compute $\tilde{\mathbf{P}}(C|W)$
    **M-Step:** Compute Maximum (or Penalized) Likelihood Estimates from weighted data.
        $P(C) = \frac{\Sigma_i \tilde{P}(C|W_i)}{|S|}$
        $P(X = x|C = c) = \frac{\Sigma_{\{i|X_i=x\}} \tilde{P}(C = c|W_i)}{\Sigma_i \tilde{P}(C = c|W_i)}$
```

We treat $\tilde{P}(C|W)$ as fractional "counts". Each wafer W_i belongs to class C fractionally. These are sometimes called the "responsibilities" (i.e., class C is responsible for wafer W according to $\tilde{P}(C|W)$.

EM Training Curve



Each iteration of EM is guaranteed to improve the likelihood of the data. Hence, EM is guaranteed to converge to a local maximum of the (penalized) likelihood function.

It is a local search algorithm, so initialization is important. It is important to break symmetries.

EM Resulting Model

$\mathbf{P}(C)$	
0	0.35
1	0.65

$P(Xi = 1 \mid C)$	0	1
$X1$	0.41	0.28
$X2$	0.81	0.21
$X3$	0.11	0.15
$X4$	0.26	0.63
$X5$	0.56	0.47
$X6$	0.97	0.75
$X7$	0.74	0.38
$X8$	0.00	0.34
$X9$	0.76	0.08
$X10$	0.96	0.16

Avoiding Overfitting

- **Early Stopping**. Hold out some of the data and measure log likelihood during training. When it reaches a local maximum, stop training.

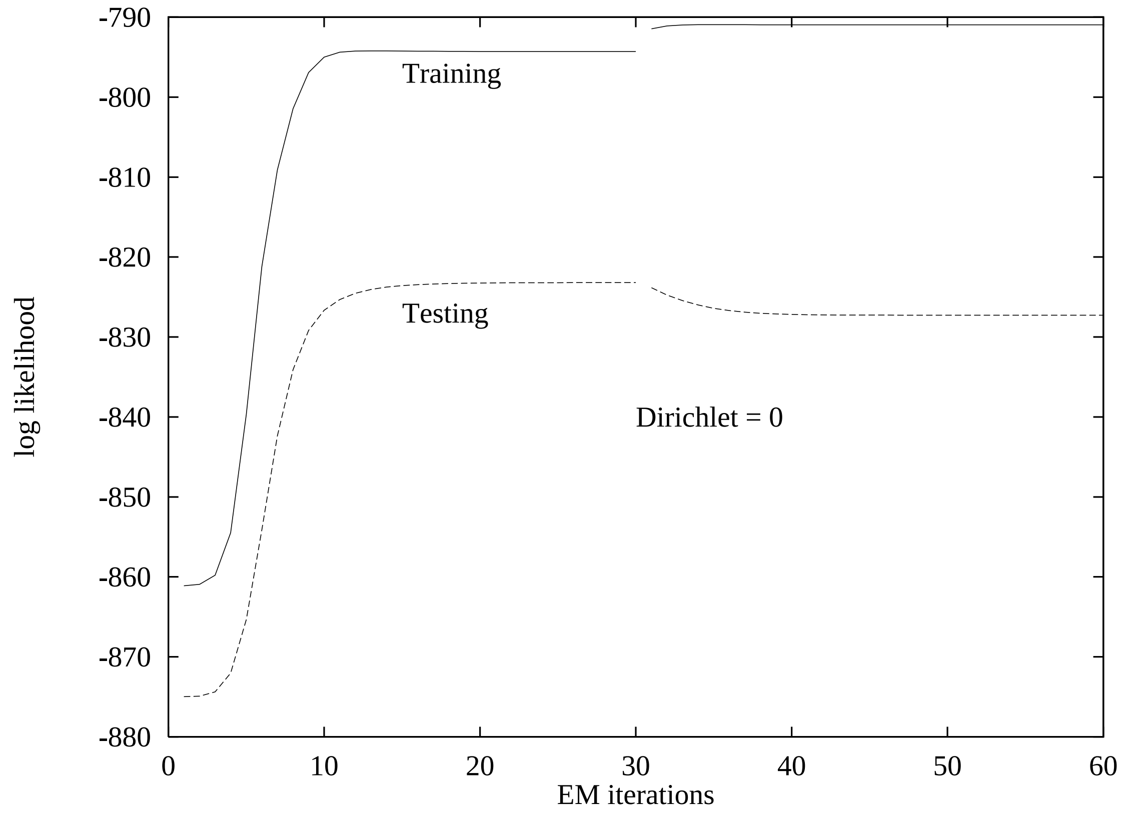
- **Penalized Likelihood**. Add a penalty for probabilities close to zero and one. A standard way to do this is to add "fake counts" to the data. For example, when counting up the number of times $X5$ is true, we add k; and the same when counting up the number false. Then we divide through by the total plus $2k$:

  $$P(X5 = true | C = 1) = \frac{\#\{X5 = true\} + k}{\#\{C = 1\} + 2k}$$

  This is called a Dirichlet Prior with parameter k. $k = 1$ (known as the Laplace Correction) gives a uniform distribution, and larger values of k bias the distribution toward 0.5.

- **Full Bayes**. Guaranteed to avoid overfitting, but usually impractical.

EM with Dirichlet Prior



When k is reset to 0, EM overfits immediately.

A good value for k can be chosen by using part of the training data as a holdout set and trying different values.

Comparison of Results

Method	Training Set	Test Set
true model	-802.8547	-816.3976
hard-M true data	-794.0431	-824.1350
hard-EM:	-791.6860	-826.9377
soft-M true data	-792.1115	-821.3197
soft-EM:	-790.9655	-827.2721
soft-M true data $\alpha = 1$	-794.9489	-821.1579
soft-EM $\alpha = 1$	-794.3053	-823.1899

- **true model:** likelihood of training and test data computed using the true model.
- **hard-M true data:** this is the model fit by the hard-M step using the true data (i.e., true values of C). It is the best that hard-EM could do if it guessed all of the C's correctly.
- **soft-M true data:** this is the model fit by one iteration of EM starting with the true model. It is the best that EM could do on this data set if it guessed $\tilde{\mathbf{P}}(C|W)$ correctly.
- **soft-M true data $\alpha = 1$:** this is the model fit by one iteration of EM starting with the true model; Dirichlet prior of 1. This is the best that could be achieved by penalized EM.

Unknown Structure, Fully Observable: Chou and Liu Method

Hypothesis space: Space of all directed tree-structured belief networks

```
ChouLiu
    for all pairs $(X_i, X_j)$ of variables do
        compute $I(X_i; X_j) = \sum_{x_i,x_j} P(x_i, x_j) \log \frac{P(x_i, x_j)}{P(x_i)P(x_j)}$ (mutual information)
    end for
    Construct a complete graph $G$ over the variables such that
        the weight on the edge from $X_i$ to $X_j$ is $I(X_i; X_j)$
    Compute the maximum weight spanning tree $T$ of $G$
    Choose a root node arbitrarily, and direct all edges away from it, recursively.
    Fit the probability tables of the resulting tree-structured belief network to the training data.
end ChouLiu
```

Greedy Algorithm for Growing A Belief Network

Cooper and Herskovits (1992) showed that for a given network structure h^s where all variables are discrete, multinomial random variables with Dirichlet priors, the likelihood of the data S is:

$$P(S|h^s) = \prod_{i=1}^{m} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \cdot \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}$$

In this formula:

S is the training data

h^s is a network structure

i indexes the random variables

j indexes the possible configurations of the parent random variables

k indexes the possible values of variable X_i

N_{ijk} is the number of training examples where $X_i = k$ when the parents of X_i are in configuration j.

$N_{ij} = \Sigma_k N_{ijk}$

α_{ijk} is our "pseudo-counts" for N_{ijk} $\alpha_{ij} = \Sigma_k \alpha_{ijk}$

Greedy Algorithm K2

Cooper and Herskovitz start with a belief network with no edges and hill-climb to maximize $\log P(S|h^s)$. To keep the search feasible, they ask the user to give an ordering of the variables X_i such that if X_i appears after X_j, then there cannot be an arc from X_i to X_j. In effect, the user is asked to choose one possible direction of causality for each pair of variables.

The algorithm then considers the nodes in order and greedily chooses parents of that node until adding a parent does not improve $\log P(S|h^s)$.

Optimal Reinsertion

Moore et al. (2003) developed a new algorithm for learning belief net structure called Optimal Reinsertion. It works by starting with an initial network (e.g., from Chou & Liu) and then iteratively deleting a node (and all of its edges) and then *optimally* re-inserting that node (with new edges) into the network. It gives much better results than greedy algorithms.

Unknown Structure, Hidden Variables

Structural EM Algorithm (Nir Friedman, 1997). Repeat:

- **E-Step**: Compute "complete data" from the incomplete data and the current Bayesian network structure and parameters.
- **Structural M-Step:** Apply structure learning algorithm to the "complete data". This results in a new network structure.
- **Standard M-Step:** Find MAP estimates of the parameters of the new network structure.