Learning in Belief Networks

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

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: an hypothesis $h \in H$ that is a good approximation to $\mathbf{P}(U)$.

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:**

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

Bayesian Learning Theory (2)

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

$$\begin{aligned} P(u|S) &= P(u|h^*) \quad \text{where} \\ h^* &= \operatorname*{argmax}_h P(S|h)P(h) \end{aligned}$$

The prior probability puts a weight on each hypothesis which we can also think of as a penalty. MAP is also sometimes called **penalized likelihood**.

Bayesian Learning Theory (3)

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

$$\begin{aligned} P(u|S) &= \alpha P(u|h^*) \text{ where} \\ h^* &= \operatorname*{argmax}_h P(S|h) \end{aligned}$$

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*.

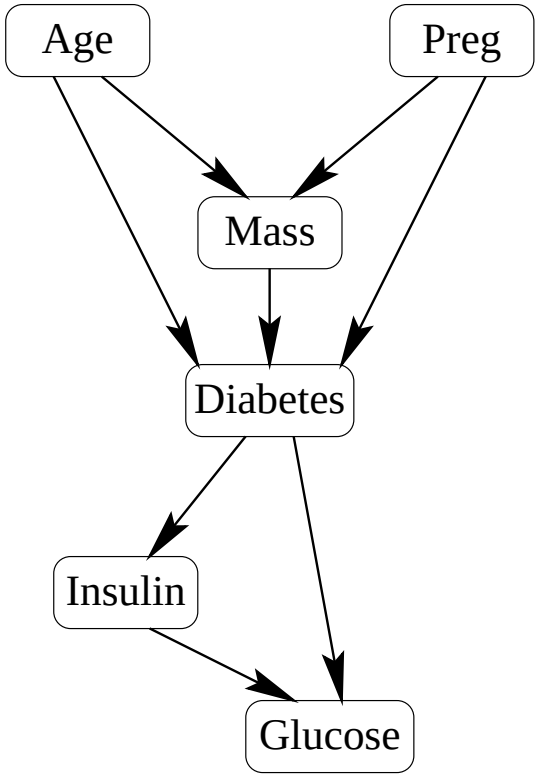
If the training examples are independent, then $P(S|\hat{\Theta}) = \prod_i P(u^i|\hat{\Theta})$.

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**

  $\text{Glucose} < 100 \Rightarrow 0$

  $100 \leq \text{Glucose} < 120 \Rightarrow 1$

  $120 \leq \text{Glucose} < 140 \Rightarrow 2$

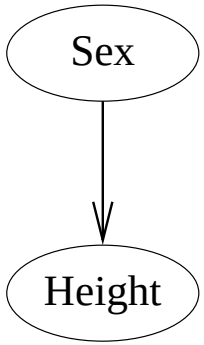
  $140 \leq \text{Glucose} \Rightarrow 3$

- **Count Cases**

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

Belief Networks with Continuous Conditional Probability Distributions

We are not required to represent $P(child|parents)$ as a table. Consider the following model to explain the observed heights of people in the population:



- **Sex:** male, female
- **Height:** continuous.

$\mathbf{P}(S)$

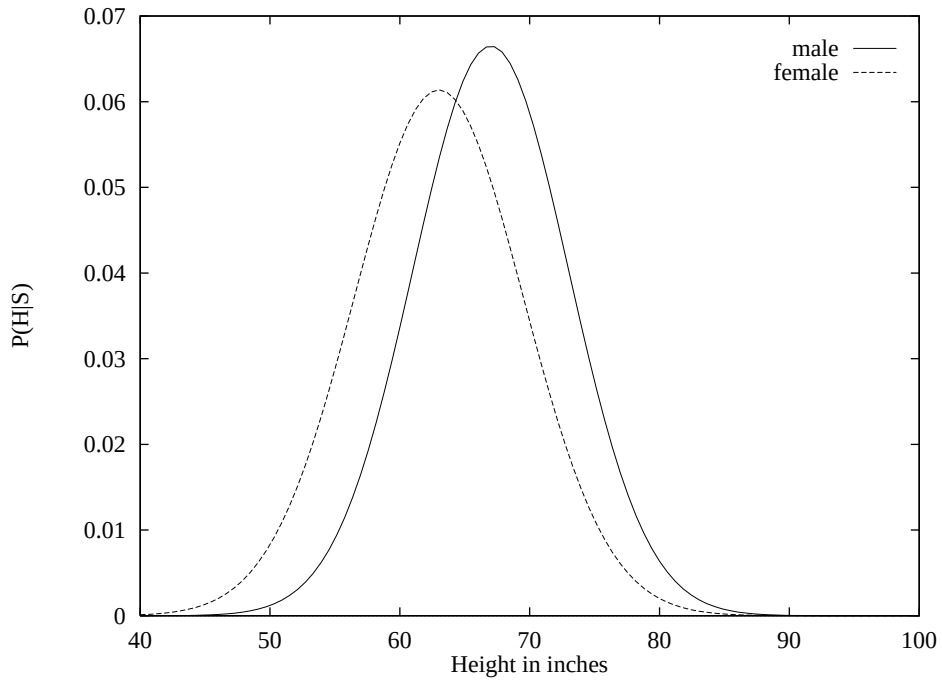
male	.49
female	.51

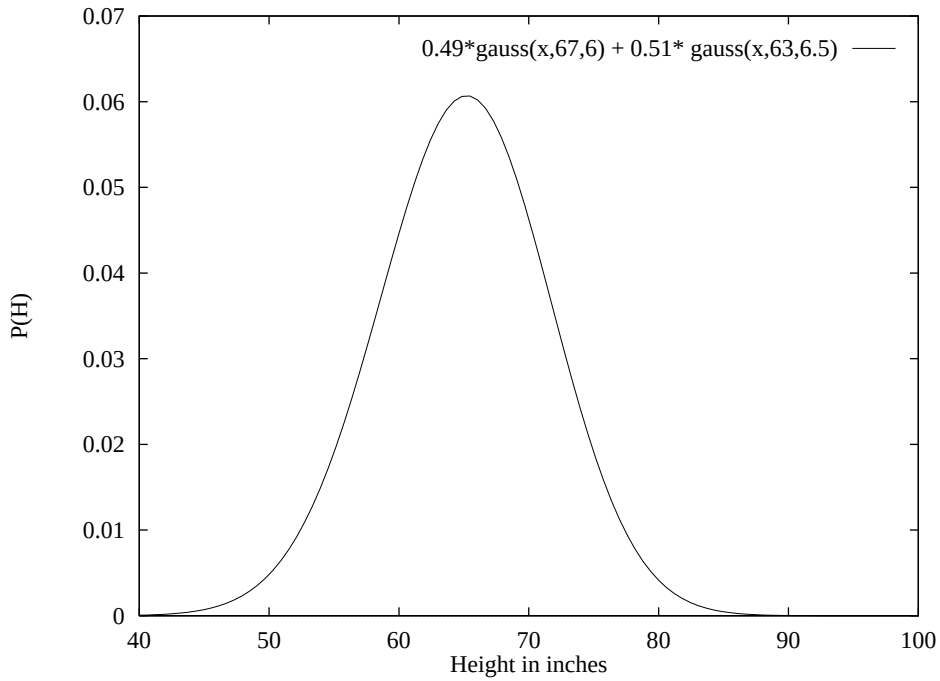
$\mathbf{P}(H\|S)$	male	female
	$\mathcal{N}(67, 6)$	$\mathcal{N}(63, 6.5)$

where $\mathcal{N}(\mu, \sigma)$ denotes the normal distribution.

Gaussian Mixture Models

This is called a Gaussian mixture model, because $P(Height)$ is a mixture of two gaussians. The probabilities $P(Sex)$ (0.49 and 0.51) are called the *mixing proportions*.

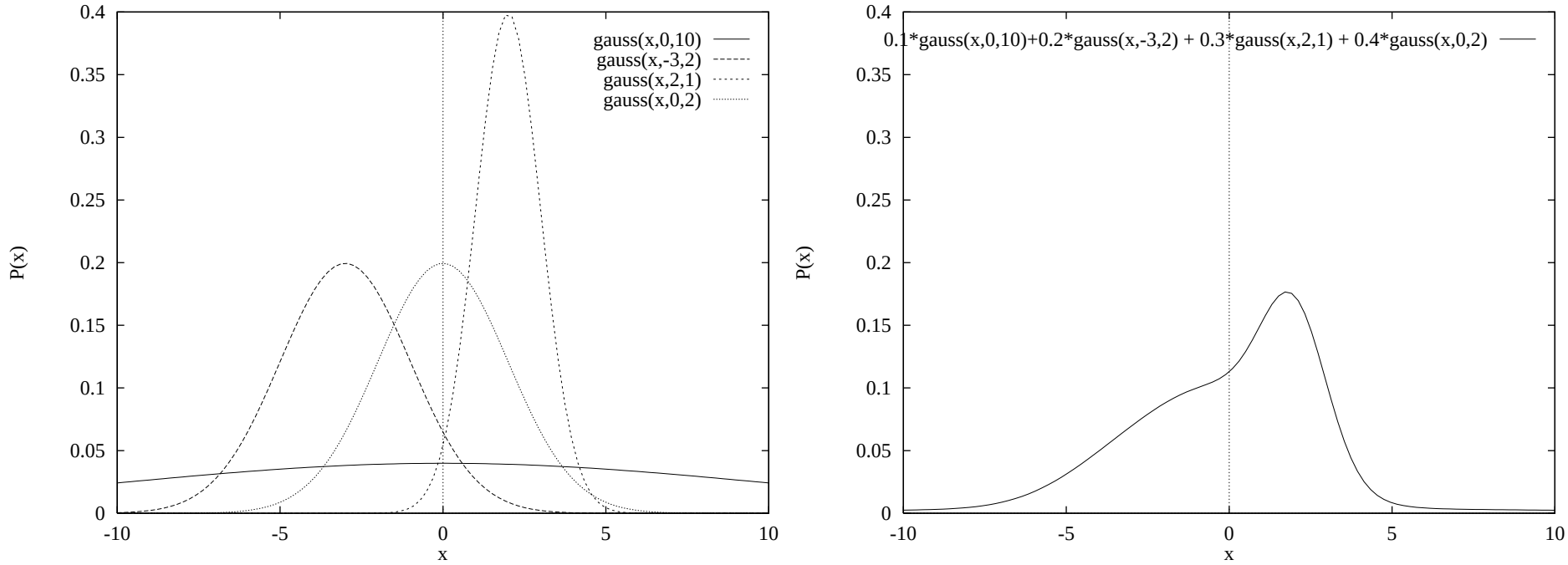




$$\begin{aligned} P(Height) &= \sum_{Sex} P(Sex) \cdot P(Height|Sex) \\ &= 0.49 \cdot P(Height|male) + 0.51 \cdot P(Heither|female) \end{aligned}$$

Gaussian Mixture Models are Very Flexible

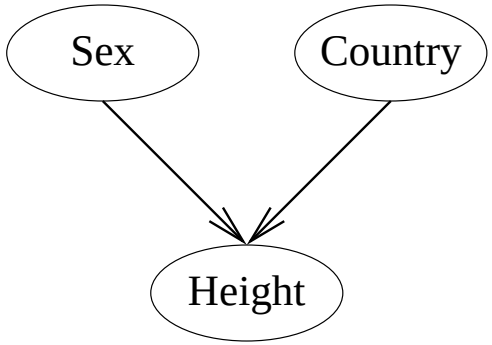
Here is a mixture of 4 Gaussians with mixing proportions of 0.1, 0.2, 0.3, and 0.4.



Many complex distributions can be modeled as mixtures of gaussians.

More Complex Mixture Models

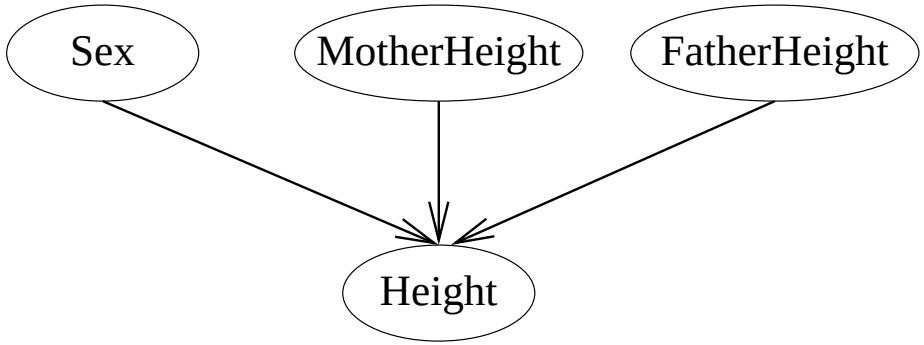
The mixing proportions can be products of simpler distributions.



$$P(Height) = \sum_{Country} \sum_{Sex} P(Country) \cdot P(Sex) \cdot P(Height|Country, Sex)$$

Continuous Variables as Inputs

Suppose height of child depends on sex of child and average height of parents. The mean of each Height distribution depends on the mean of the parents' heights.



$\mathbf{P}(S)$

male	.49
female	.51

$\mathbf{P}(MH)$

	$\mathcal{N}(63, 6.5)$

$\mathbf{P}(FH)$

	$\mathcal{N}(67, 6)$

$\mathbf{P}(H\|S, MH, FH)$	male	female
	$\mathcal{N}((MH + FH)/2 + 2.5, 6)$	$\mathcal{N}((MH + FH)/2 - 2, 6.5)$

Learning Algorithms For Mixture Models

Variable	Constraint	Depends on
Sex	Bernoulli	none
MotherHeight	Gaussian	none
FatherHeight	Gaussian	none
Height	Gaussian; mean	MH, FH, S
	std. deviation	S

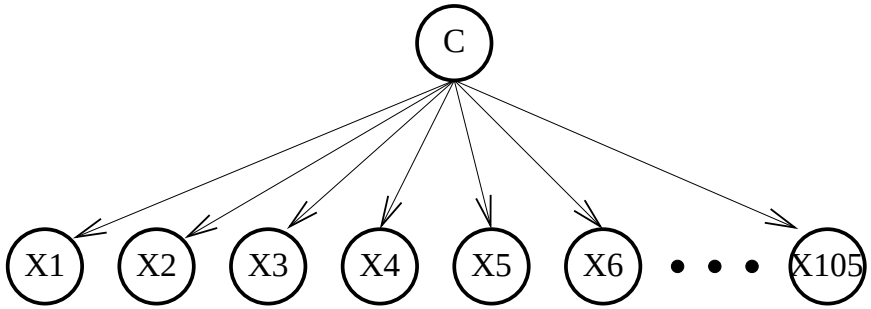
Collect data. We have a separate supervised learning problem for each node.

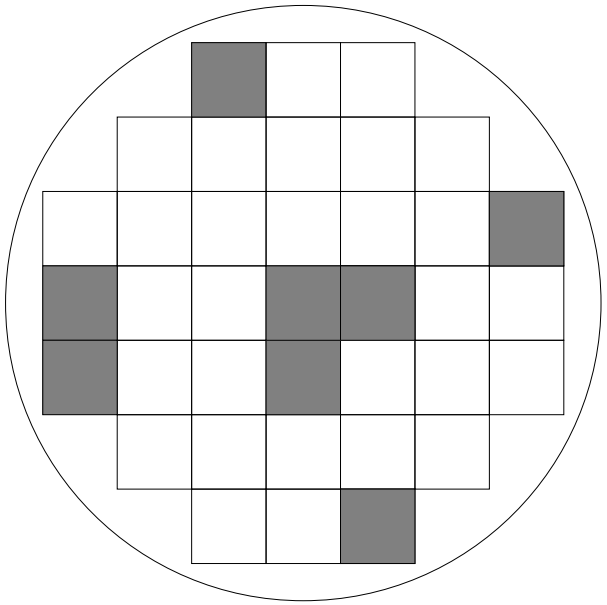
Example 1: FatherHeight: Compute mean and standard deviation of the heights of the observed fathers.

Example 2: Height: We need to impose some model on the mean of Height. For example, we might use a simple linear model or a neural network to predict μ in terms of MH and FH, separately for each S.

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

$P(Xi = 1 \mid C)$	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.)

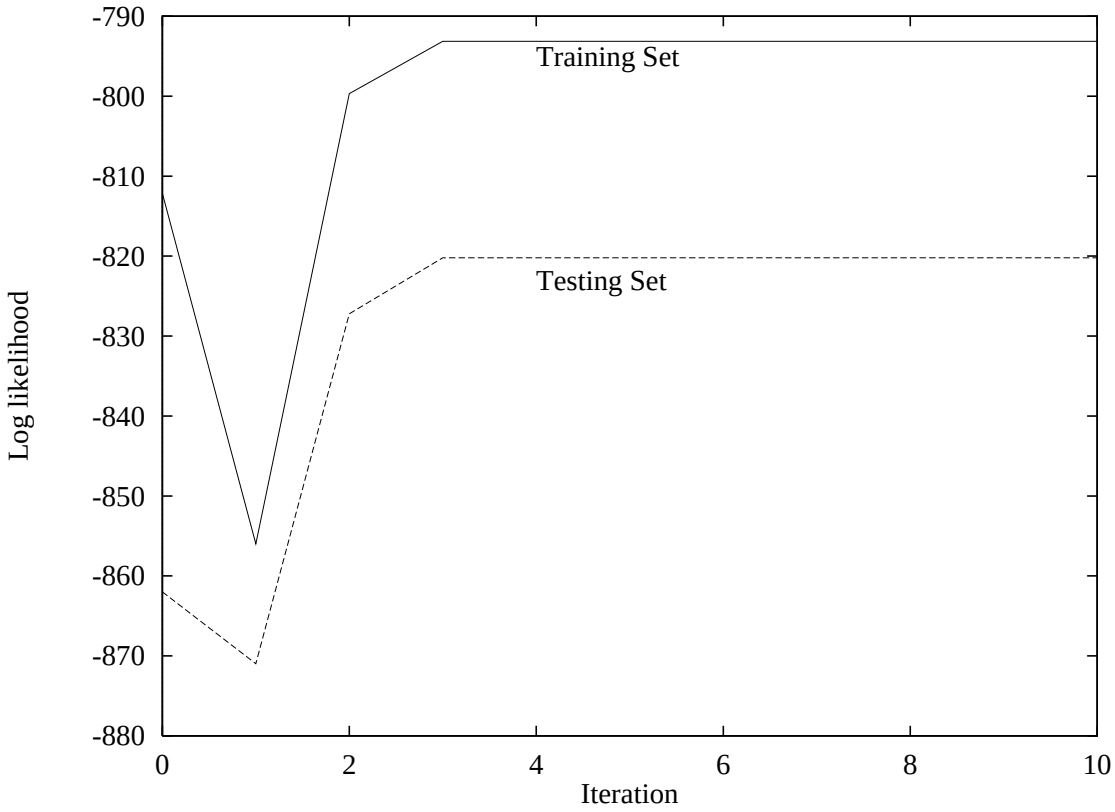
Hard-EM: Fit of Model to Training Data (including true C)

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

$P(Xi = 1\|C)$	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

$P(Xi = 1\|C)$	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 Θ randomly.

repeat until convergence

E-Step: For each wafer, compute $\tilde{\mathbf{P}}(C|W)$

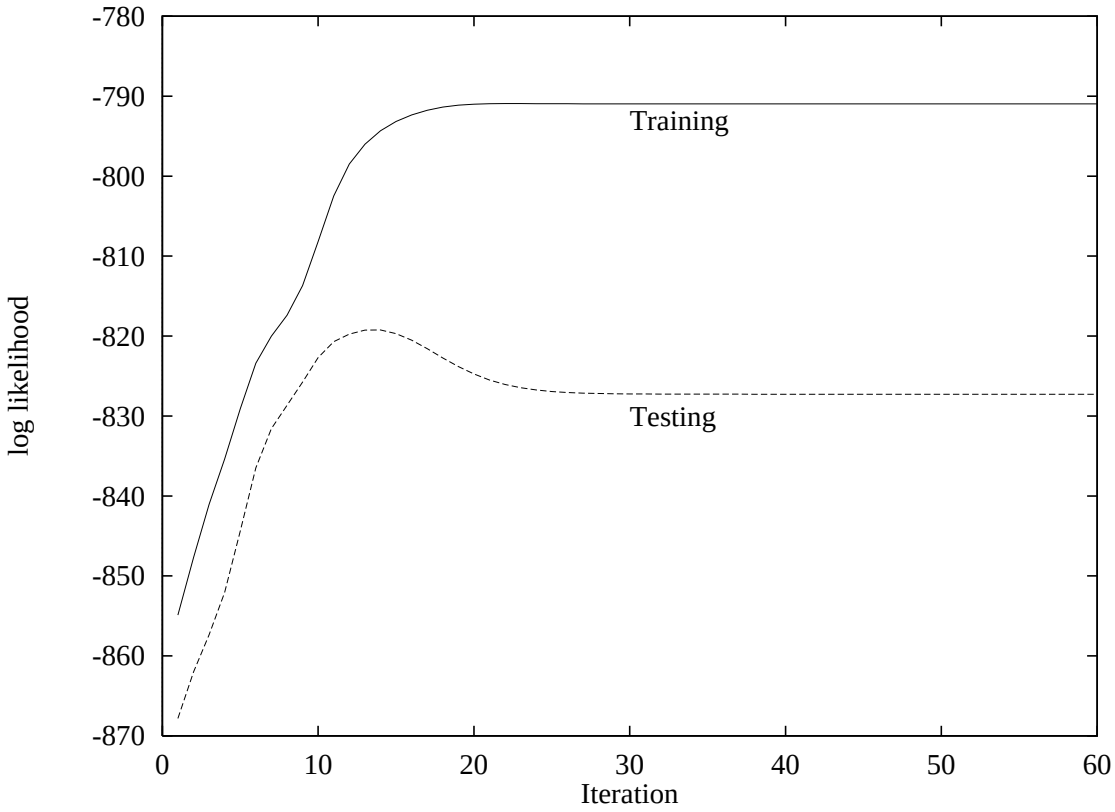
M-Step: Compute Maximum (or Penalized) Likelihood Estimates from weighted data.

$$P(C) = \frac{\Sigma_i \tilde{P}(C|W_i)}{|S|}$$

$$P(X = x|C = c) = \frac{\Sigma_{\{i|X_i=x\}} \tilde{P}(C = c|W_i)}{\Sigma_i \tilde{P}(C = c|W_i)}$$

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

$P(Xi = 1\|C)$	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

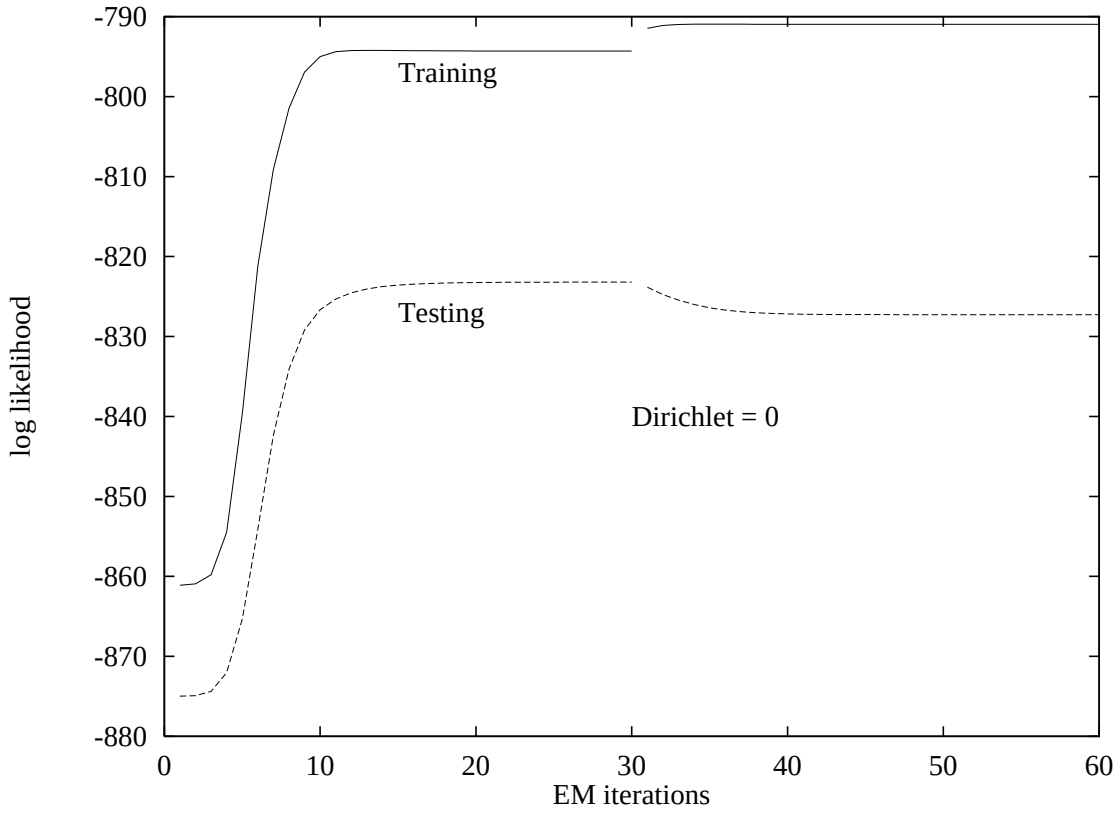
Avoiding Overfitting

- **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. $k = 1/2$ 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.

Approximating Full Bayes via Markov Chain Monte Carlo Methods

One way to approximate Full Bayes is to draw samples from the joint distribution $P(h|S) = P(\Theta|S)$. Suppose we could draw L examples of h in this way, then we could compute $P(u|S)$ as follows:

$$\begin{aligned} P(u|S) &= \sum_h P(u|h)P(h|S) \\ &\approx \frac{1}{L}\sum_\ell P(u|h_\ell) \end{aligned}$$

It is possible to construct a Markov chain that will randomly jump from one Θ to another in such a way that it obeys $P(\Theta|S)$. We can simulate this Markov chain, draw random samples from it, and average them.

Sampling From a Joint Distribution

- **Forward Simulation**
- **Gibbs Sampling**

Forward Simulation

We can run any belief network in the "forward direction" as follows:

While all variables do not have values do

Find a variable X all of whose parents have values

Draw a value x according to $P(X|parents(X))$

Output the values of all variables.

This works for any network as long as there is no evidence.

Suppose we have evidence $V = 0$. Then when we come to variable V, we generate v according to $P(V|parents(V))$ and then check whether $v = 0$. If it does, then this sample is ok, otherwise we must *reject* this sample and start the simulation over again. This is called *rejection sampling*, and it can be very inefficient.

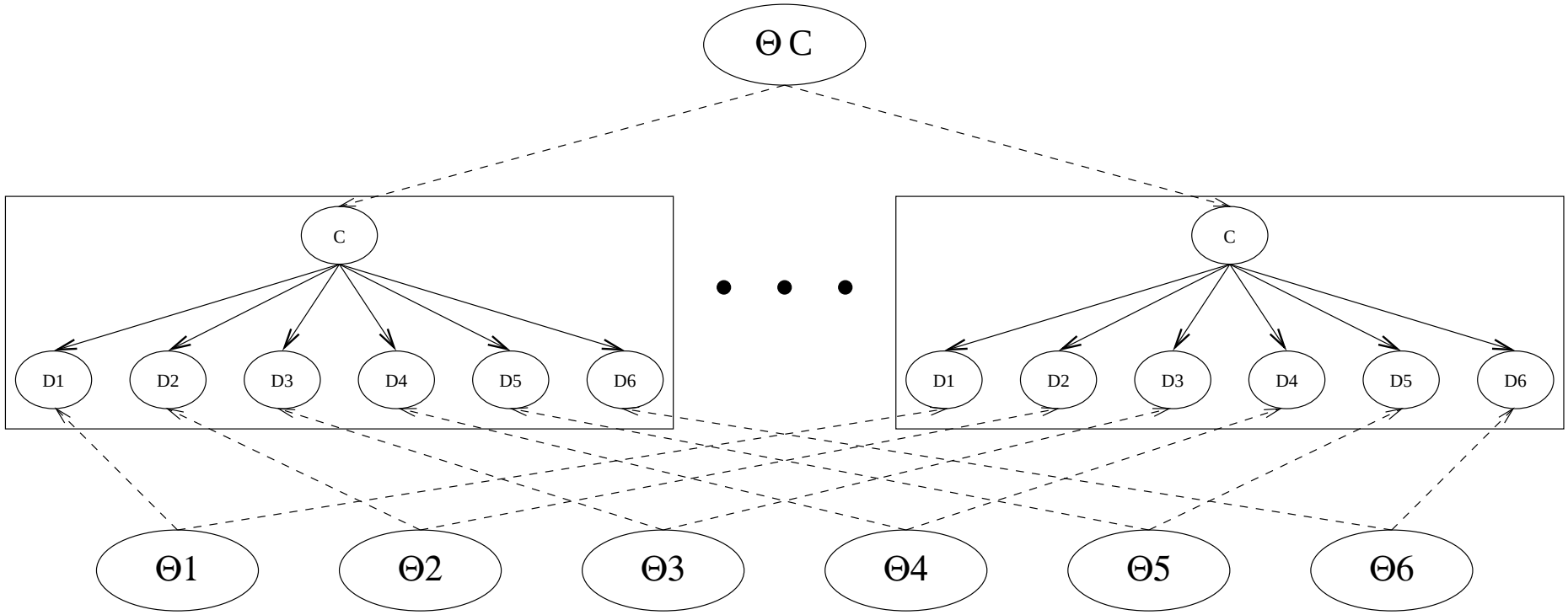
Gibbs Sampling

Suppose we can compute $\mathbf{P}(X|U - X)$ for all variables X in the belief network. Then the following gives a Markov chain whose stationary distribution is $P(U)$.

Choose an (arbitrary) ordering for the (unobserved) variables: $X_1, \ldots, X_n$
Assign values at random to all variables
Repeat forever
    For i from 1 to n do
        Draw x_i according to $\mathbf{P}(X_i|U - X)$

Applying Gibbs Sampling During Learning (Naive Bayes)

We need to treat the (unknown) parameters of the belief network as if they were additional nodes. Let Θ_{Xi} be the parameters in the $\mathbf{P}(Xi|C)$ tables, and Θ_C be the parameters in the $\mathbf{P}(C)$ table. The original network is copied once for each training example.



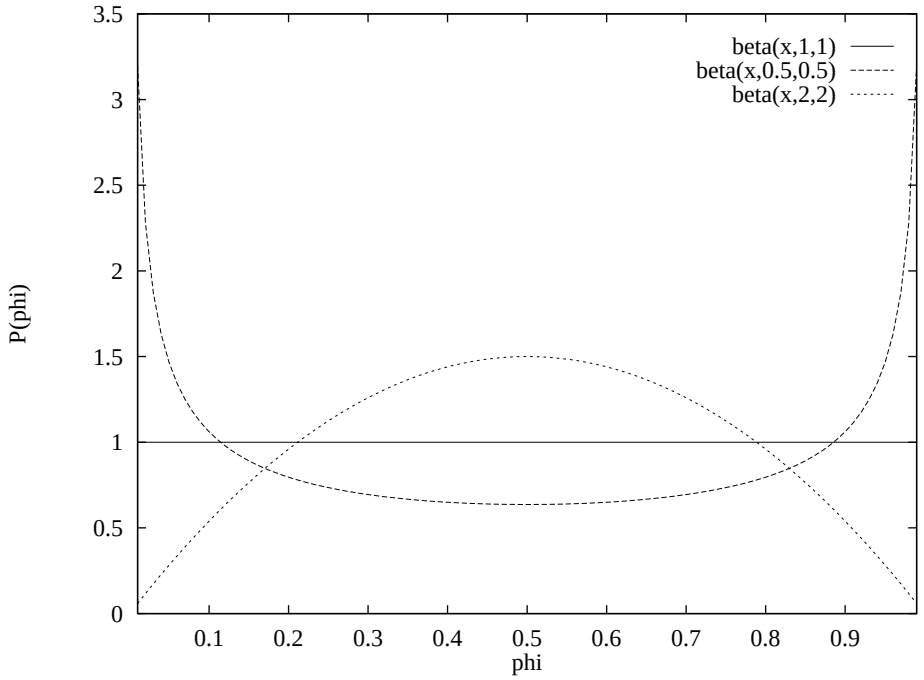
Let the probability distributions for the Θ nodes be $P(\Theta_{Xi})$ and $P(\Theta_C)$.

Prior Probability of a Probability

Suppose X is a discrete random variable with K possible values: $\{1, \ldots, K\}$. Then the probability distribution $\mathbf{P}(X)$ is a *multinomial* distribution with parameters $\Theta = \{\theta_1, \ldots, \theta_K\}$ such that $\Sigma_k \, \theta_k = 1$. The standard *prior* probability distribution for Θ is the Dirichlet distribution:

$$Dir(\Theta|\alpha_1, \ldots, \alpha_K) = \frac{\Gamma(\alpha)}{\Pi_k \, \Gamma(\alpha_k)} \prod_k \theta_k^{\alpha_k - 1},$$

where the α_k are the parameters of this distribution, and $\alpha = \Sigma_k \, \alpha_k$. Normally, we set $\alpha_k = 1$, which corresponds to a uniform prior distribution.

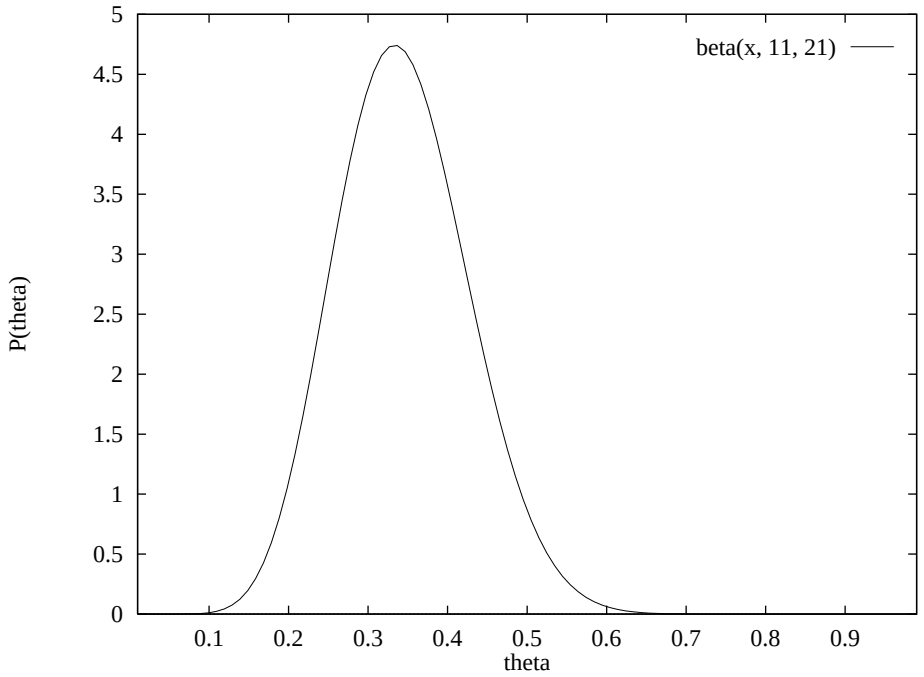


Posterior Probability of a Probability

After observing $(n_1, \ldots, n_k)$ examples of the values of X, the *posterior* distribution of Θ is

$$Dir(\Theta|\alpha_1 + n_1, \alpha_2 + n_2, \ldots, \alpha_k + n_k)$$

For example, after observing 10 successes and 20 failures, the $P(\theta_{success})$ is the following:



There exist algorithms for sampling from a Dirichlet distribution.

Gibbs Sampling for Naive Bayes Model

Because the X values are known, we do not include them in the Gibbs sampling. Let S stand for all of the known X values.

repeat forever

Generate C according to $P(C|S, \Theta_C)$

Generate Θ_W according to $P(\Theta_W|C, \Theta_C)$

Generate Θ_C according to $P(\Theta_C|S, \Theta_W)$

After some point (e.g., 1000 steps), collect Q of Θ_W and Θ_C: $(\Theta_W^1, \Theta_C^1) \ldots, (\Theta_W^Q, \Theta_C^Q)$.

To compute $P(u|S)$:

$$P(u|S) = \frac{1}{Q} \sum_q P(u|\Theta_W^q, \Theta_C^q).$$

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 $\log P(S|h^s)$.