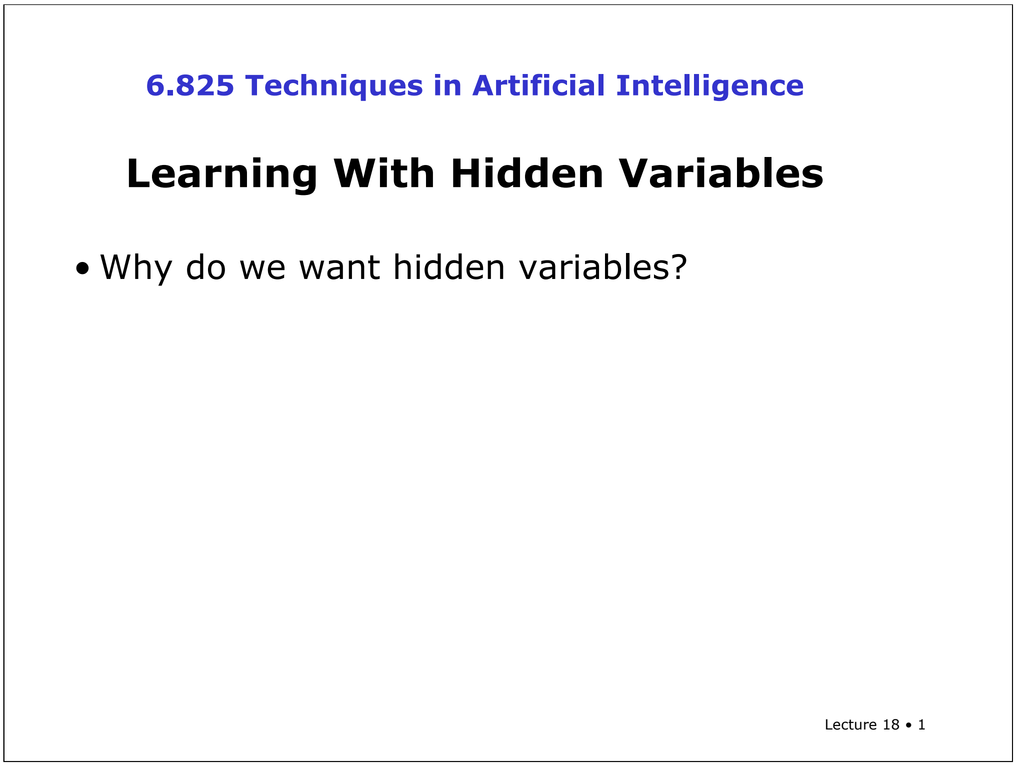
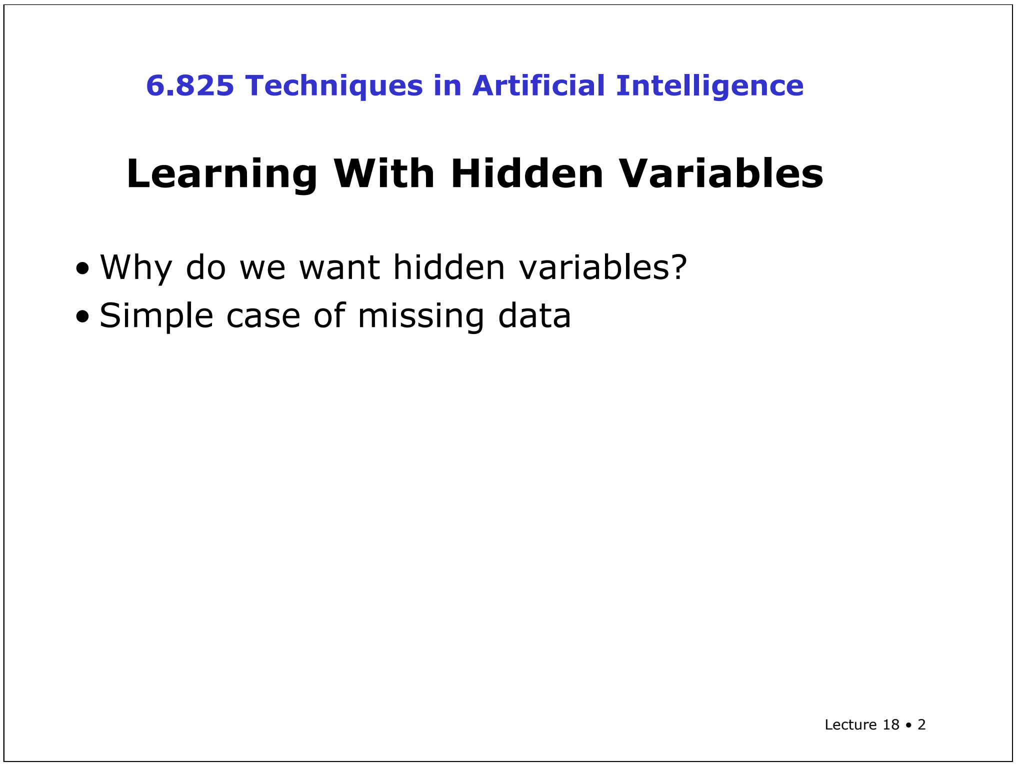


6.825 Techniques in Artificial Intelligence

Learning With Hidden Variables

- Why do we want hidden variables?

In this lecture, we'll think about how to learn Bayes Nets with hidden variables. We'll start out by looking at why you'd want to have models with hidden variables.

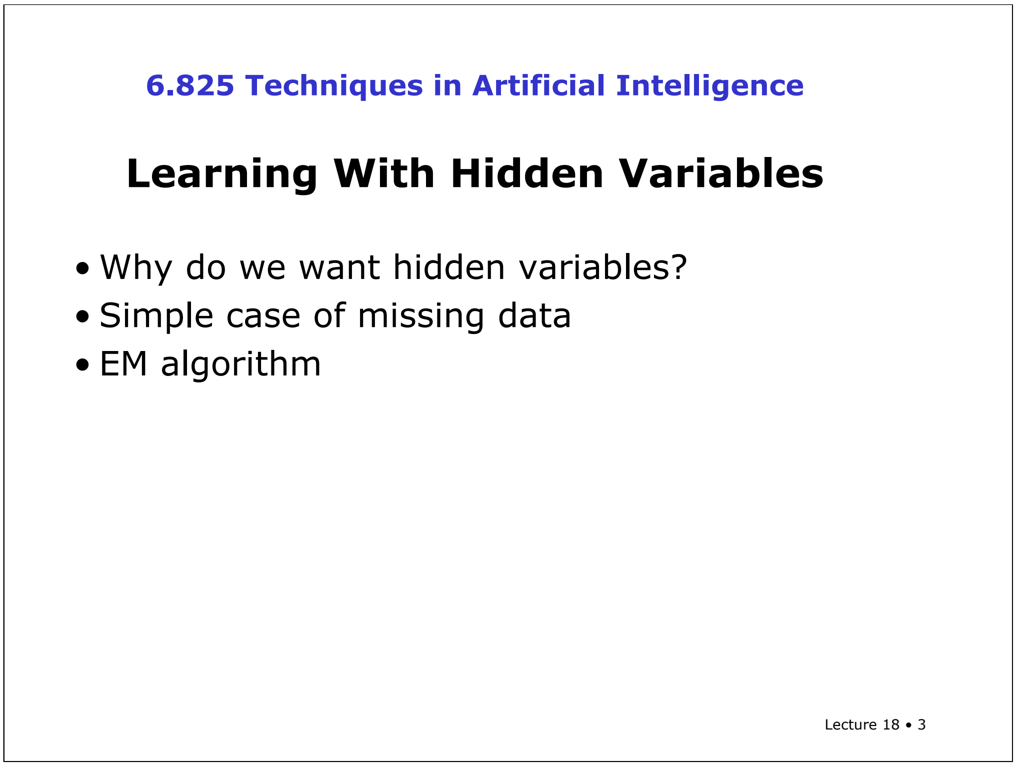


6.825 Techniques in Artificial Intelligence

Learning With Hidden Variables

- Why do we want hidden variables?
- Simple case of missing data

Then, because the technique we’ll use for working with hidden variables is a bit complicated. we’ll start by looking at a simpler problem, of estimating probabilities when some of the data are missing.

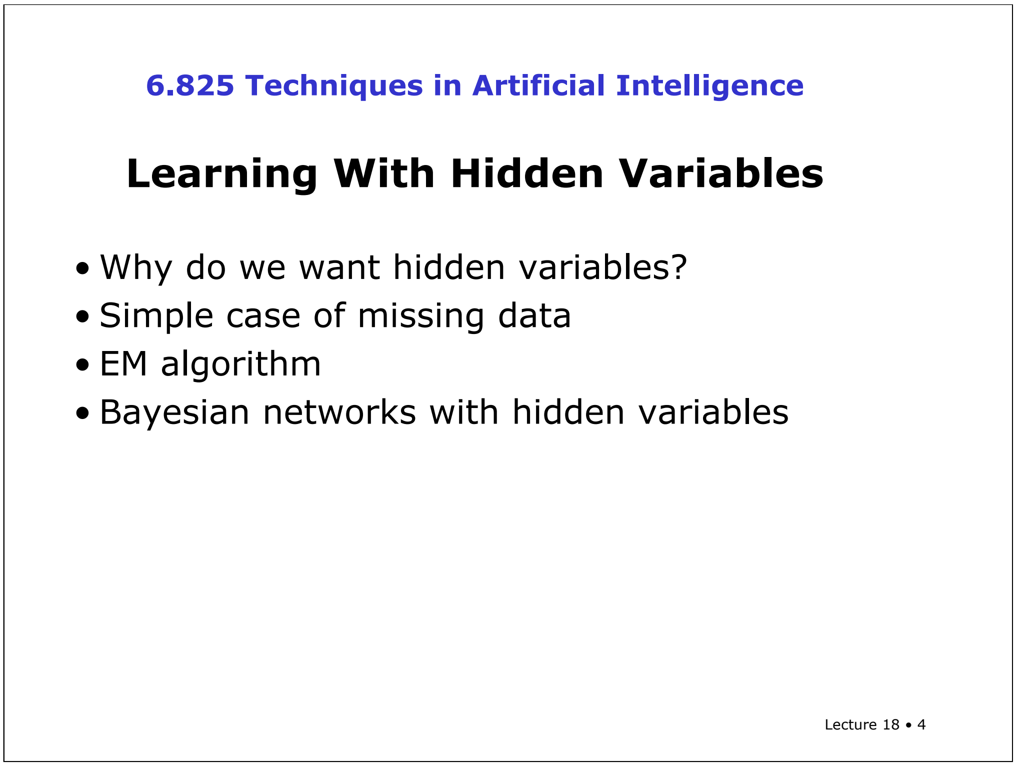


6.825 Techniques in Artificial Intelligence

Learning With Hidden Variables

- Why do we want hidden variables?
- Simple case of missing data
- EM algorithm

That will lead us to the EM algorithm, in general,



6.825 Techniques in Artificial Intelligence

Learning With Hidden Variables

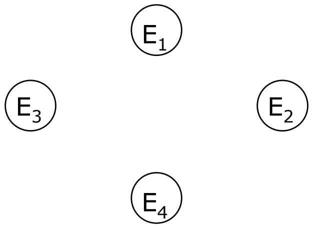
- Why do we want hidden variables?
- Simple case of missing data
- EM algorithm
- Bayesian networks with hidden variables

And we'll finish by seeing how to apply it to bayes nets with hidden nodes, and we'll work a simple example of that in great detail.

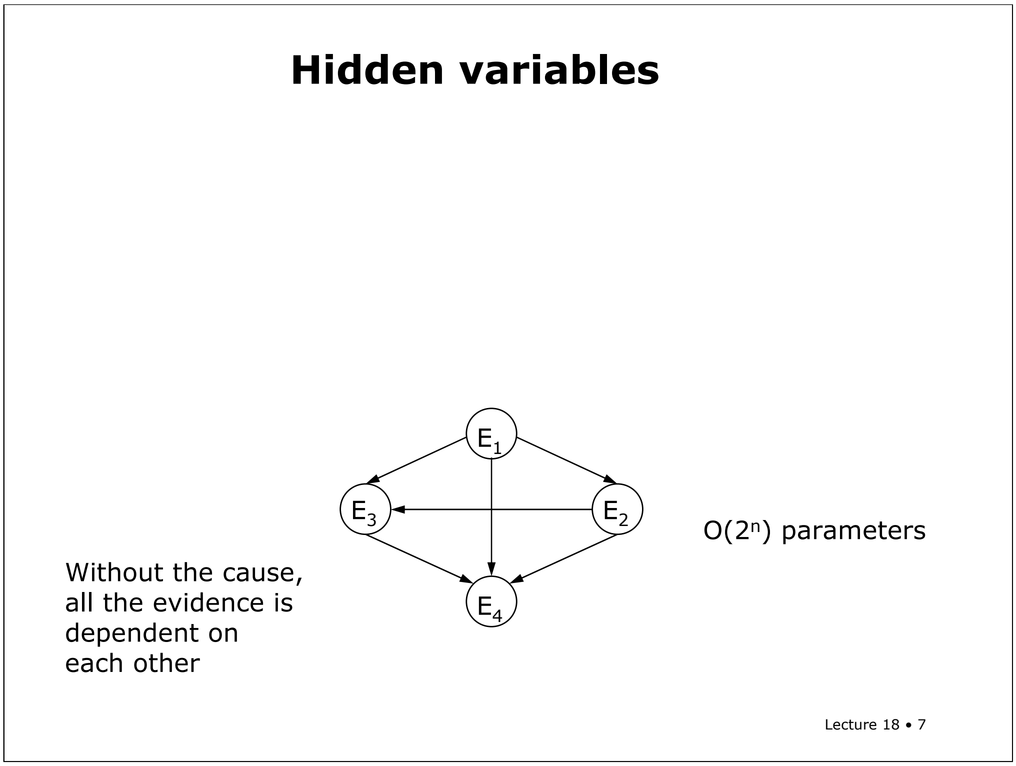
Hidden variables

Why would we ever want to learn a Bayesian network with hidden variables? One answer is: because we might be able to learn lower-complexity networks that way. Another is that sometimes such networks reveal interesting structure in our data.

Hidden variables



Consider a situation in which you can observe a whole bunch of different evidence variables, E1 through En. Maybe they’re all the different symptoms that a patient might have. Or maybe they represent different movies and whether someone likes them.

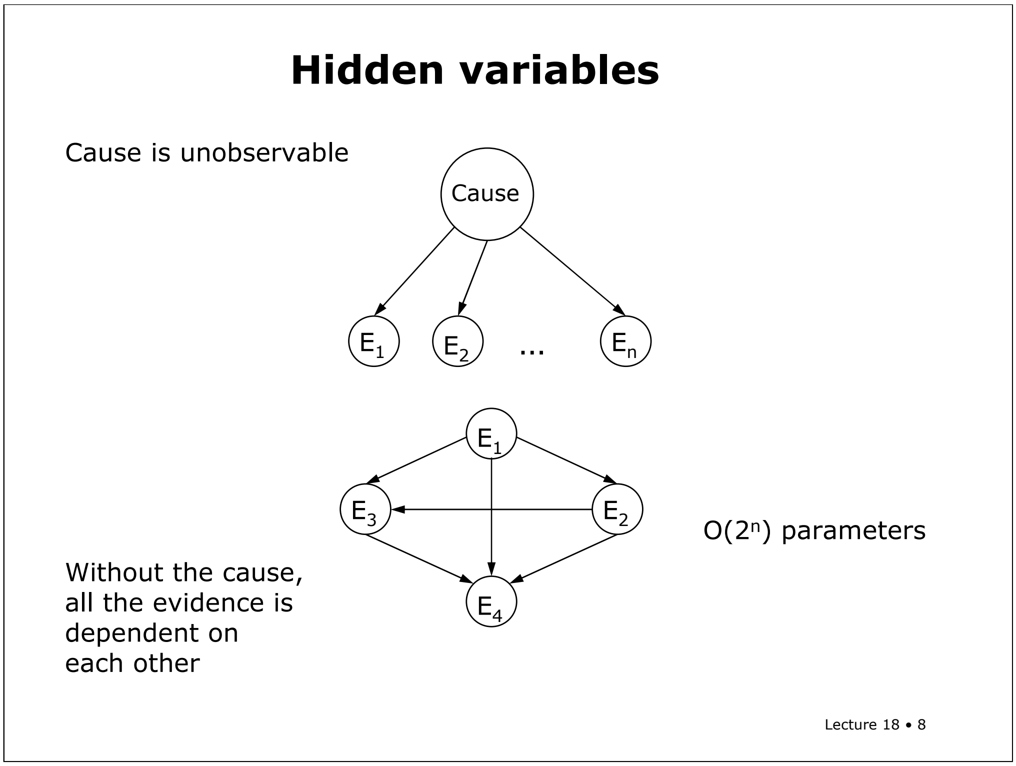


Hidden variables

O(2^n) parameters

Without the cause, all the evidence is dependent on each other

If those variables are all conditionally dependent on one another, then we'd need a highly connected graph that's capable of representing the entire joint distribution between the variables. Because the last node has n-1 parents, it will take on the order of 2^n parameters to specify the conditional probability tables in this network.



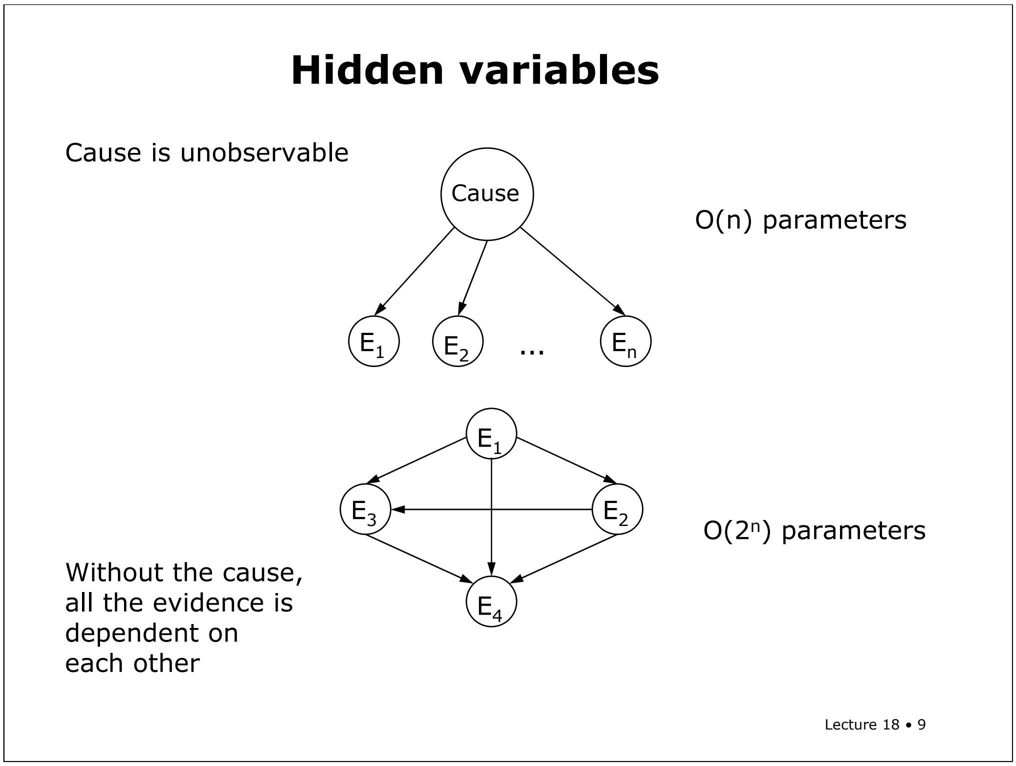
Hidden variables

Cause is unobservable

$O(2^n)$ parameters

Without the cause, all the evidence is dependent on each other

But, in some cases, we can get a considerably simpler model by introducing an additional "cause" node. It might represent the underlying disease state that was causing the patients' symptoms or some division of people into those who like westerns and those who like comedies.



Hidden variables

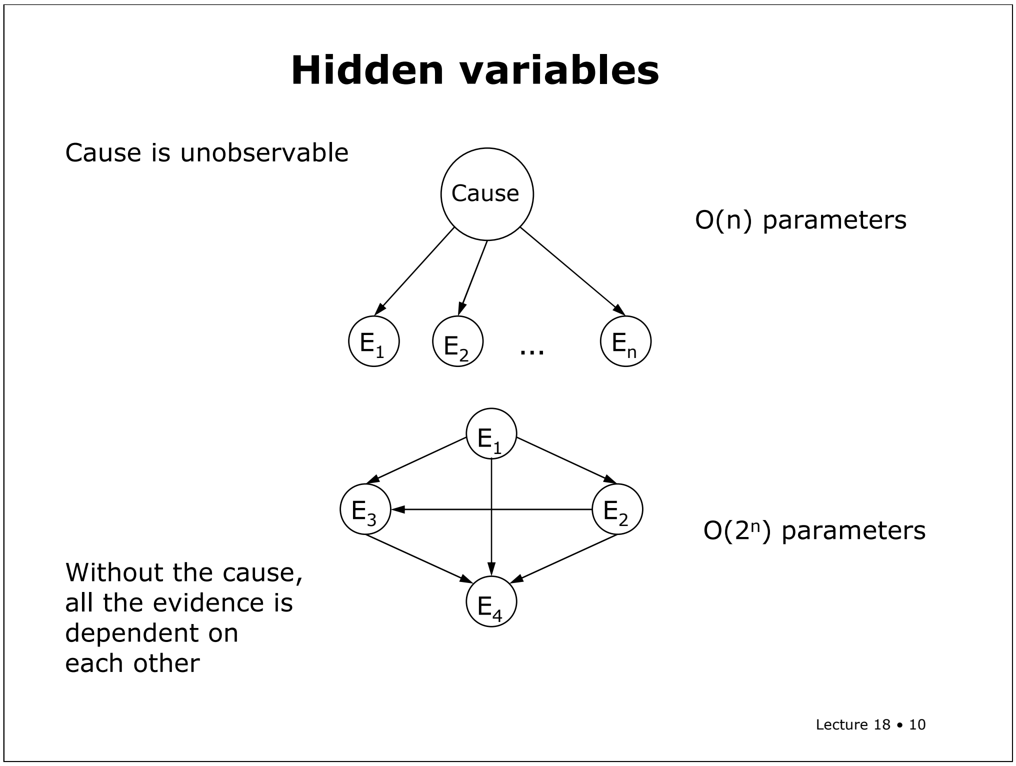
Cause is unobservable

O(n) parameters

O(2^n) parameters

Without the cause, all the evidence is dependent on each other

In the simpler model, the evidence variables are conditionally independent given the causes. That means that it would only require on the order of n parameters to describe all the CPTs in the network, because at each node, we just need a table of size 2 (if the cause is binary; or k if the cause can take on k values), and one (or k-1) parameter to specify the probability of the cause.



Hidden variables

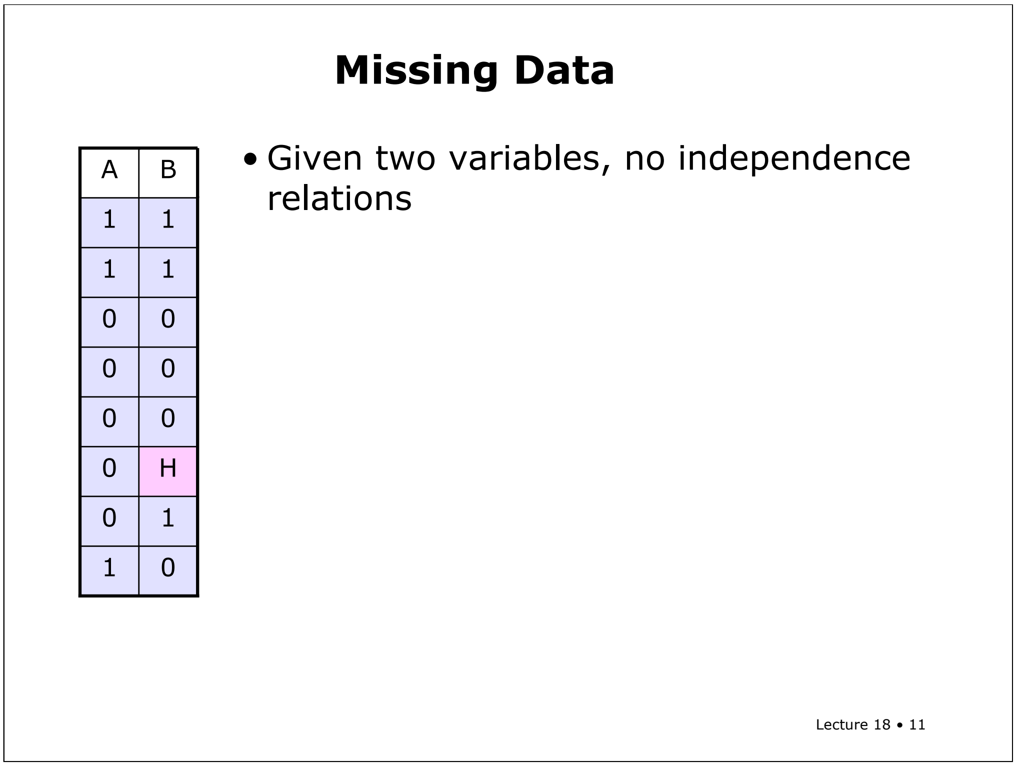
Cause is unobservable

O(n) parameters

$O(2^n)$ parameters

Without the cause, all the evidence is dependent on each other

So, what if you think there's a hidden cause? How can you learn a network with unobservable variables?

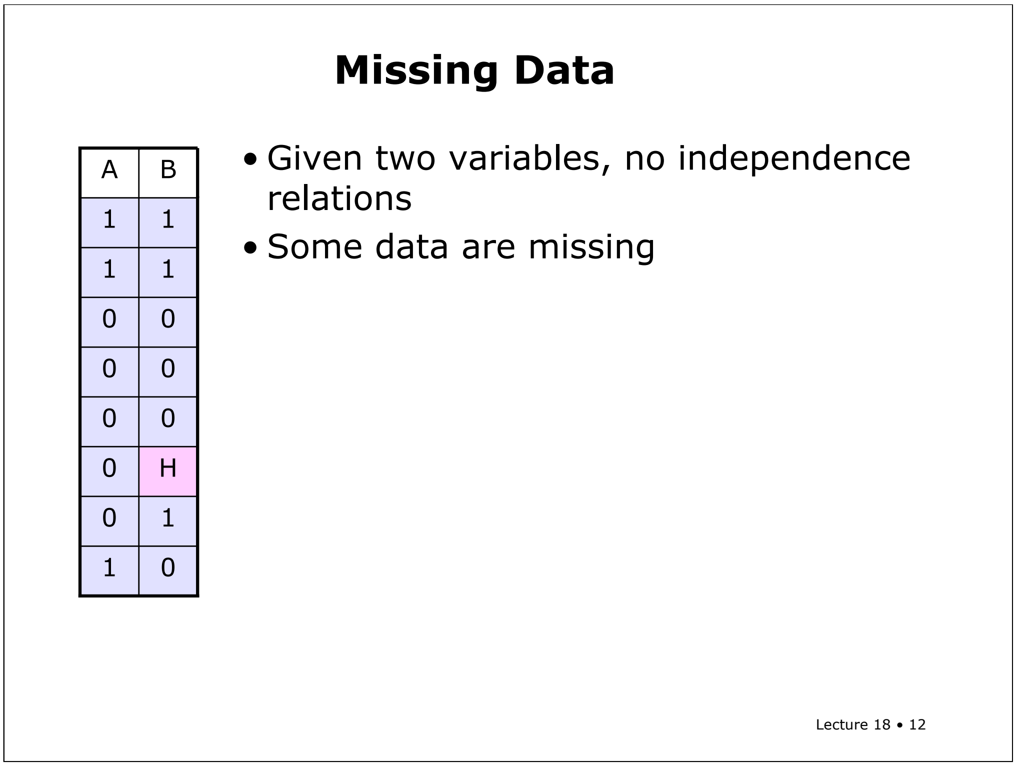


Missing Data

A	B
1	1
1	1
0	0
0	0
0	0
0	H
0	1
1	0

- Given two variables, no independence relations

We'll start out by looking at a very simple case. Imagine that you have two binary variables A and B, and you know they're not independent. So you're just trying to estimate their joint distribution. Ordinarily, you'd just count up how many were true, true; how many were false, false; and so on, and divide by the total number of data cases to get your maximum likelihood probability estimates.

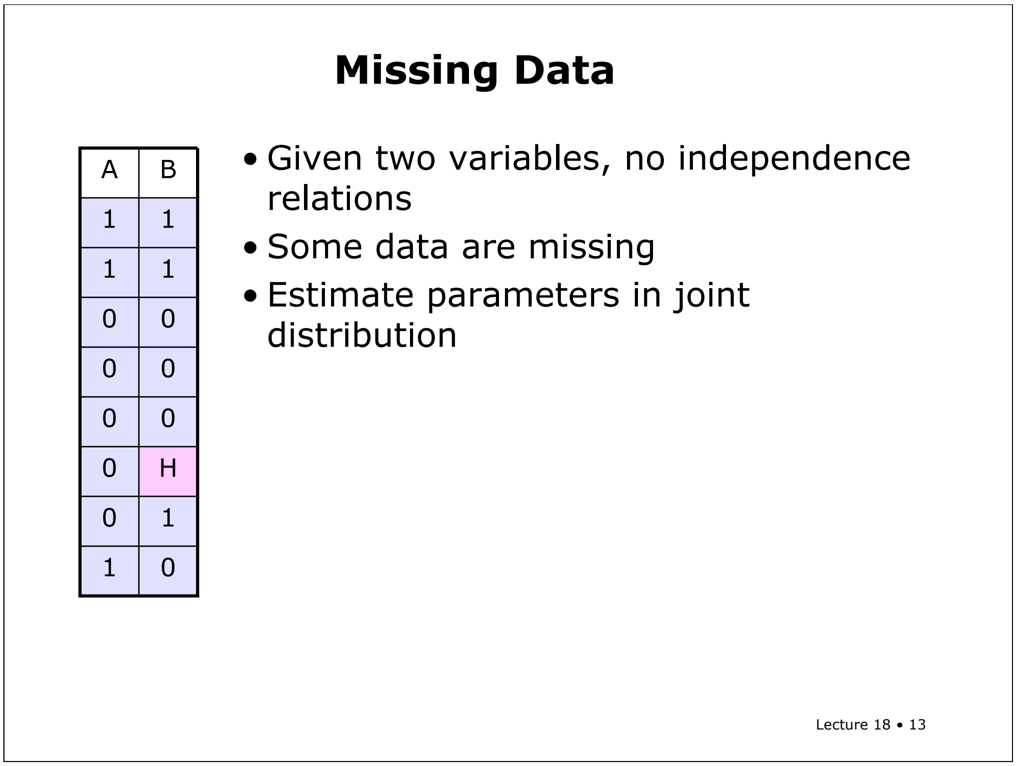


Missing Data

A	B
1	1
1	1
0	0
0	0
0	0
0	H
0	1
1	0

- Given two variables, no independence relations
- Some data are missing

But in our case, some of the data are missing. If a whole case were missing, there wouldn’t be much we could do about it; there’s no real way to guess what it might have been that will help us in our estimation process. But if some variables in a case are filled in, and others are missing, then we’ll see how to make use of the variables that are filled in and how to get a probability distribution on the missing data.

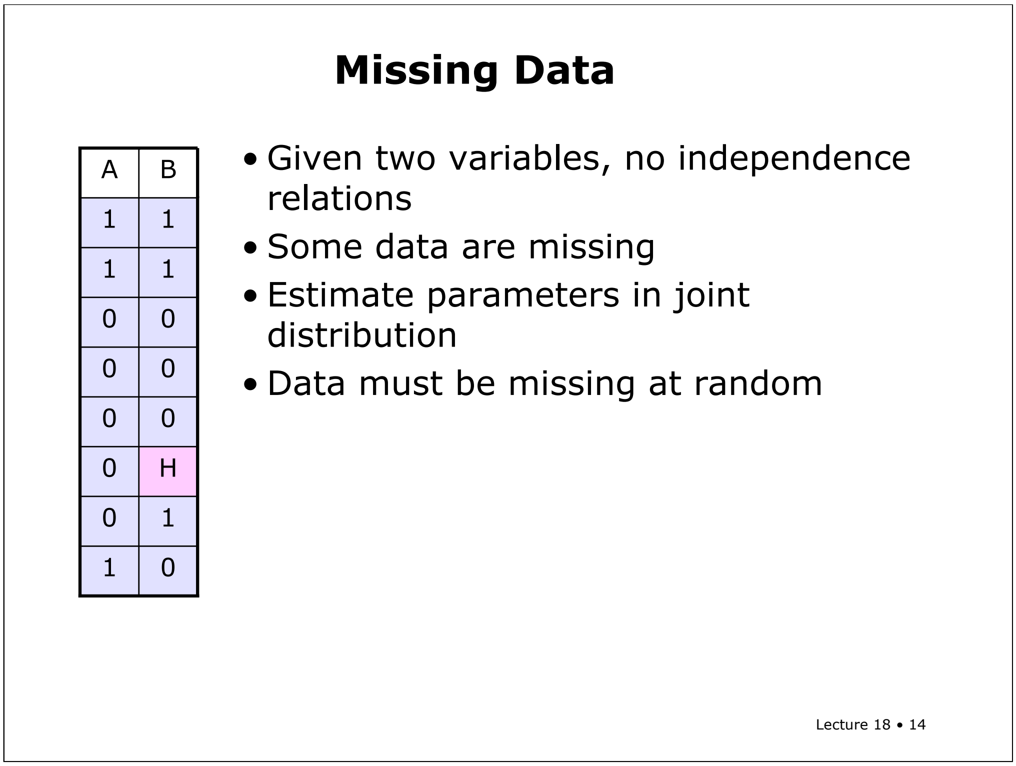


Missing Data

A	B
1	1
1	1
0	0
0	0
0	0
0	H
0	1
1	0

- Given two variables, no independence relations
- Some data are missing
- Estimate parameters in joint distribution

Here, in our example, we have 8 data points, but one of them is missing a value for B.

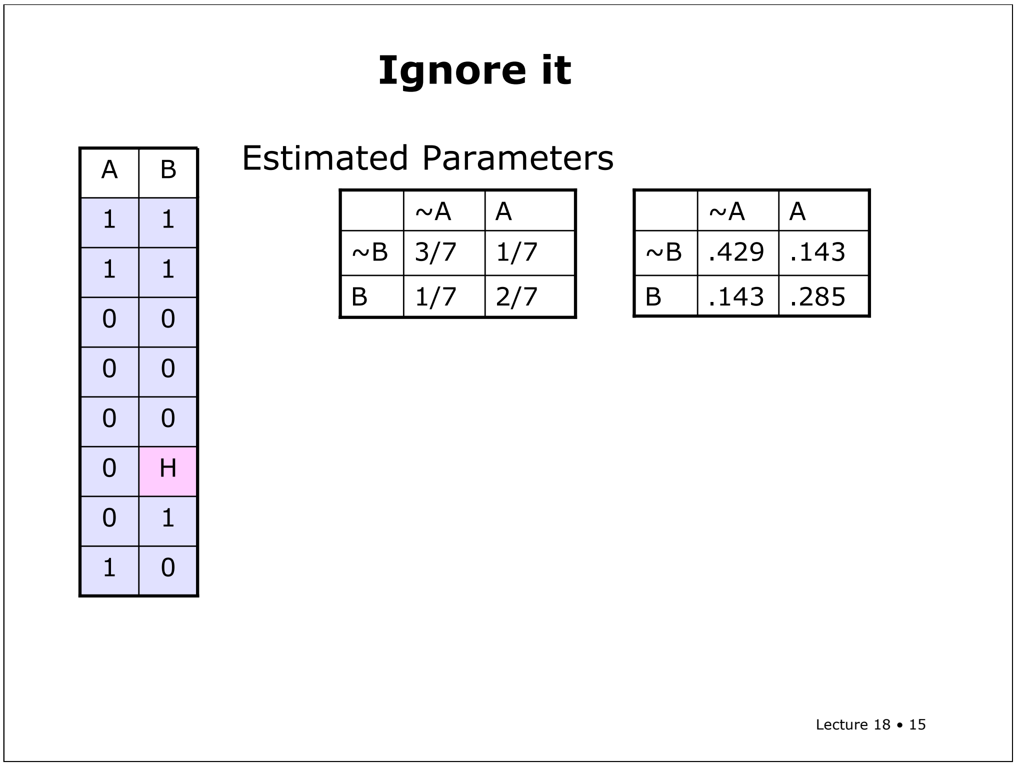


Missing Data

A	B
1	1
1	1
0	0
0	0
0	0
0	H
0	1
1	0

- Given two variables, no independence relations
- Some data are missing
- Estimate parameters in joint distribution
- Data must be missing at random

In order for the methods we'll talk about here to be of use, the data have to be missing at random. That means that the fact that a data item is missing is independent of the value it would have had. So, for instance, if you didn't take somebody's blood pressure because he was already dead, then that reading would not be missing at random! But if the blood-pressure instrument had random failures, unrelated to the actual blood pressure, then that data would be missing at random.



Ignore it

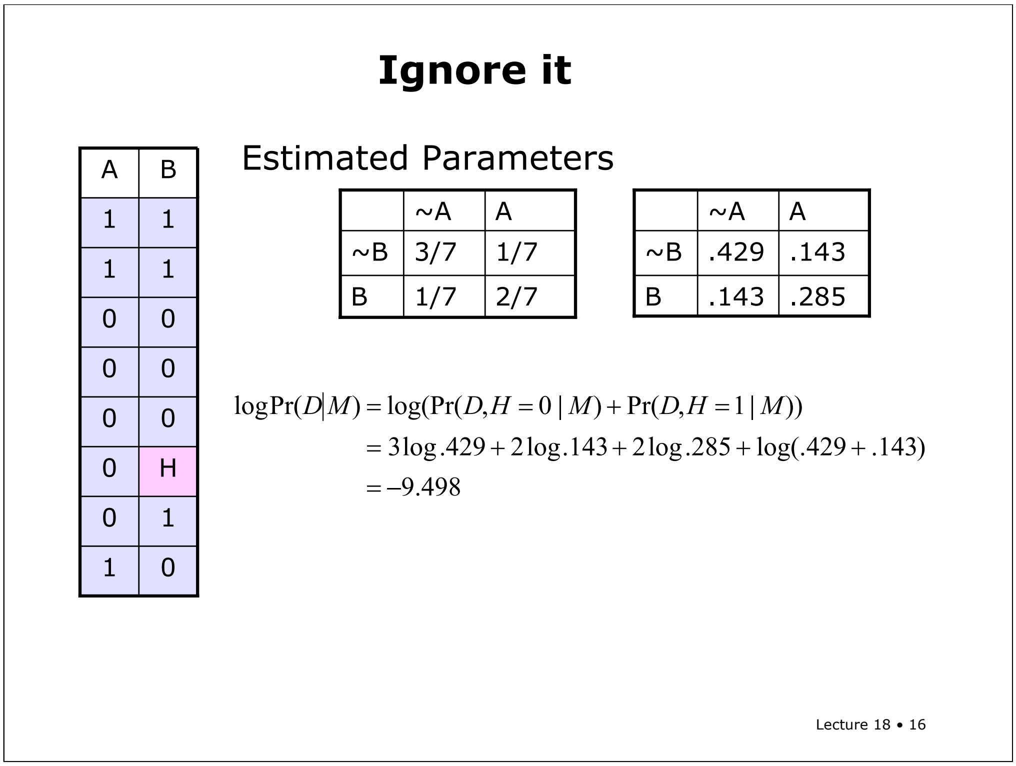
A	B
1	1
1	1
0	0
0	0
0	0
0	H
0	1
1	0

Estimated Parameters

	~A	A
~B	3/7	1/7
B	1/7	2/7

	~A	A
~B	.429	.143
B	.143	.285

The simplest strategy of all is to just ignore any cases that have missing values. In our example, we'd count the number of cases in each bin and divide by 7 (the number of complete cases).



Ignore it

A	B
1	1
1	1
0	0
0	0
0	0
0	H
0	1
1	0

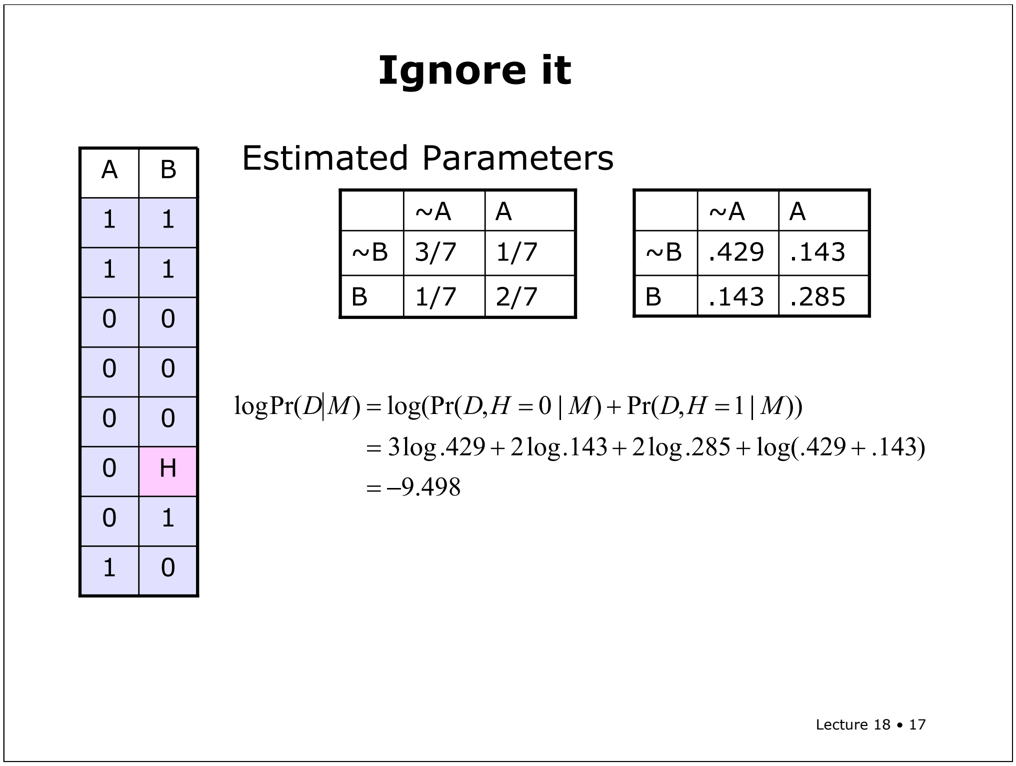
Estimated Parameters

	~A	A
~B	3/7	1/7
B	1/7	2/7

	~A	A
~B	.429	.143
B	.143	.285

$$\begin{aligned}\log \Pr(D|M) &= \log(\Pr(D,H=0\mid M)+\Pr(D,H=1\mid M))\\ &= 3\log .429+2\log .143+2\log .285+\log(.429+.143)\\ &= -9.498\end{aligned}$$

It’s easy, and it gives us a log likelihood score of –9.498. Whether that’s good or not remains to be seen. We’ll have to see what results we get with other methods.



Ignore it

A	B
1	1
1	1
0	0
0	0
0	0
0	H
0	1
1	0

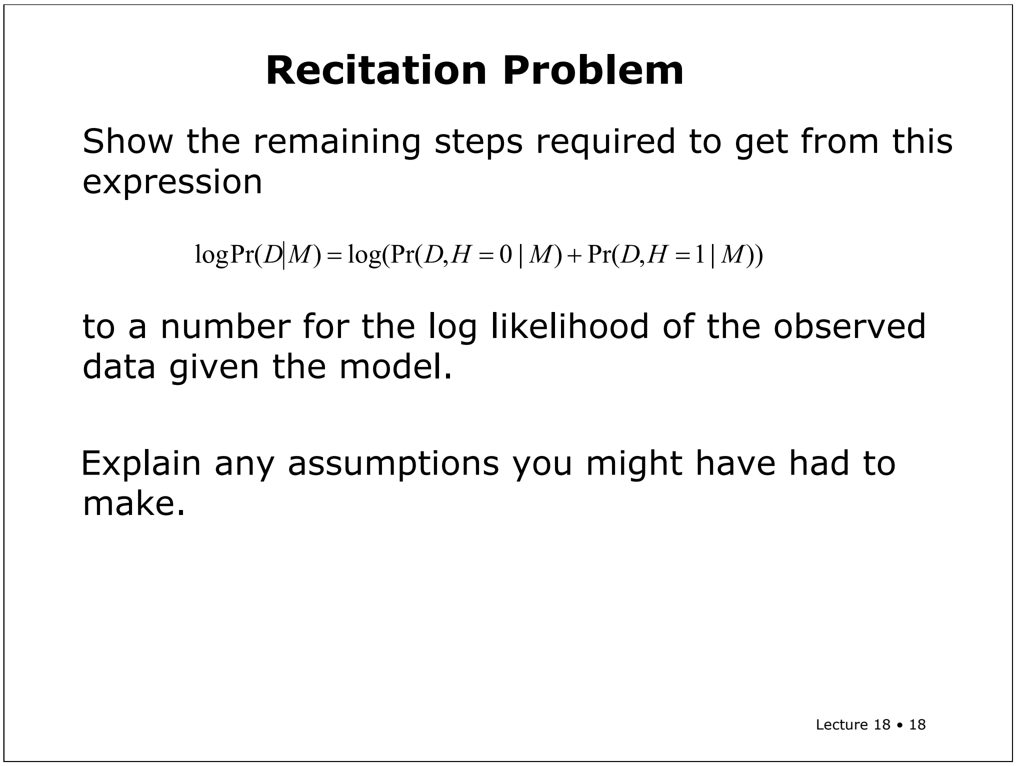
Estimated Parameters

	~A	A
~B	3/7	1/7
B	1/7	2/7

	~A	A
~B	.429	.143
B	.143	.285

$$\begin{aligned}\log \Pr(D|M) &= \log(\Pr(D,H=0 \mid M) + \Pr(D,H=1 \mid M)) \\ &= 3\log .429 + 2\log .143 + 2\log .285 + \log(.429 + .143) \\ &= -9.498\end{aligned}$$

Note that, in order to compute the log likelihood of the actual data (which is what we’re trying to maximize), we’ll need to marginalize out the hidden variable H. We accomplish that by summing over both of its values.



Recitation Problem

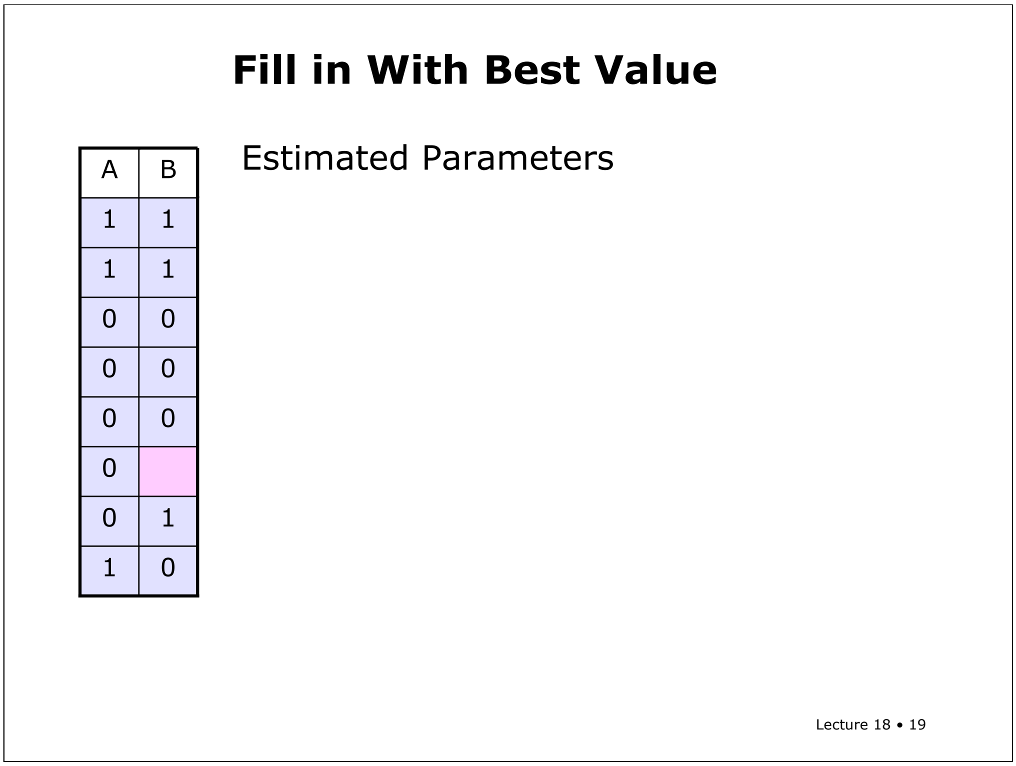
Show the remaining steps required to get from this expression

$$\log \Pr(D|M) = \log(\Pr(D,H=0 \mid M) + \Pr(D,H=1 \mid M))$$

to a number for the log likelihood of the observed data given the model.

Explain any assumptions you might have had to make.

I skipped a couple of steps in showing you my computation of the log likelihood on the previous slide. Please fill them in and show what assumptions have to be made along the way.

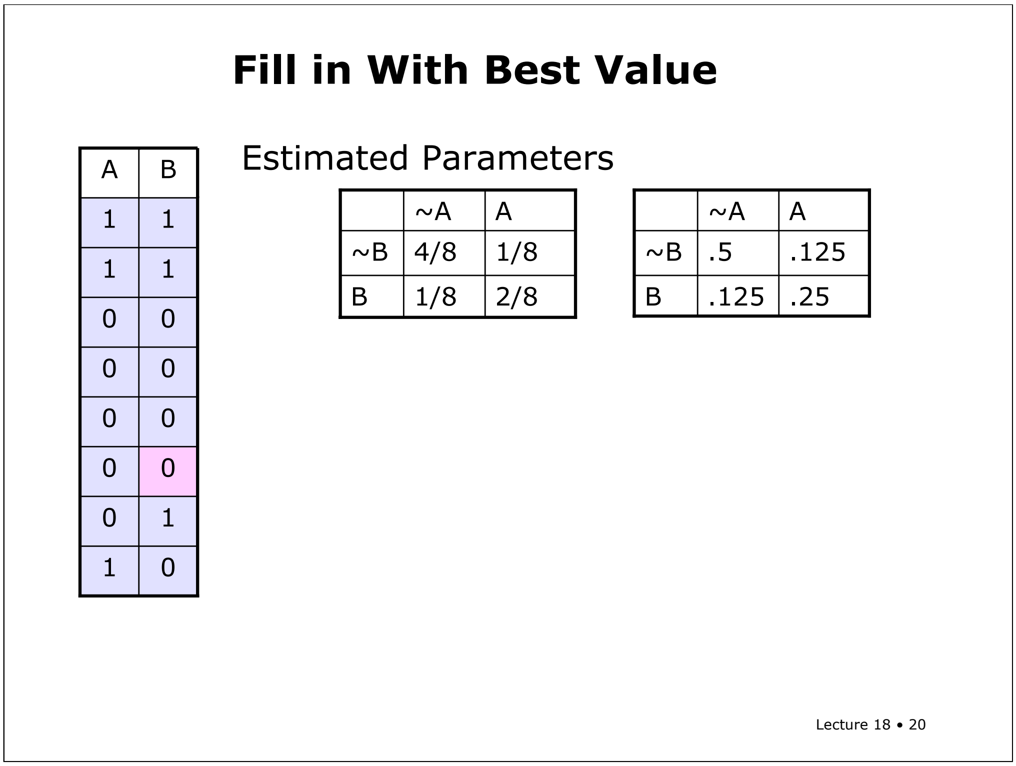


Fill in With Best Value

A	B
1	1
1	1
0	0
0	0
0	0
0	
0	1
1	0

Estimated Parameters

Another strategy would be to fill in the missing value with the value that makes the log likelihood (of the actual data) biggest.



Fill in With Best Value

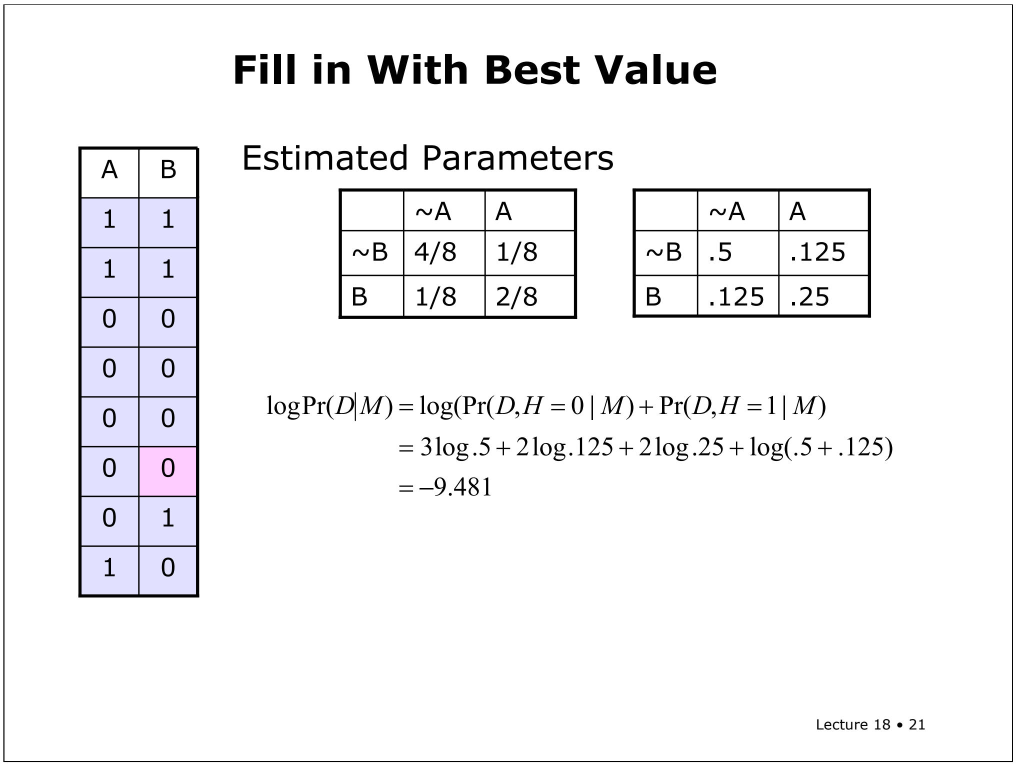
A	B
1	1
1	1
0	0
0	0
0	0
0	0
0	1
1	0

Estimated Parameters

	~A	A
~B	4/8	1/8
B	1/8	2/8

	~A	A
~B	.5	.125
B	.125	.25

In this case, that value is 0. Once you fill in the missing value, you can estimate the probabilities using the standard counting procedure.



Fill in With Best Value

A	B
1	1
1	1
0	0
0	0
0	0
0	0
0	1
1	0

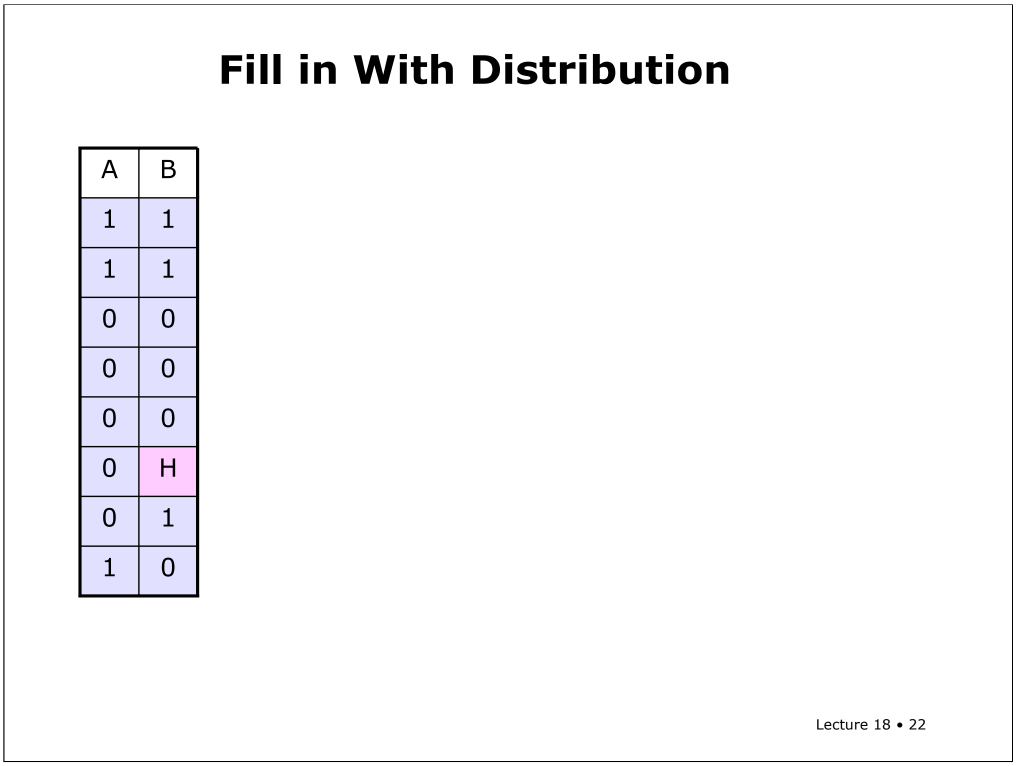
Estimated Parameters

	~A	A
~B	4/8	1/8
B	1/8	2/8

	~A	A
~B	.5	.125
B	.125	.25

$$\begin{aligned}\log\Pr(D|M) &= \log(\Pr(D,H=0\mid M)+\Pr(D,H=1\mid M)\\ &= 3\log.5+2\log.125+2\log.25+\log(.5+.125)\\ &= -9.481\end{aligned}$$

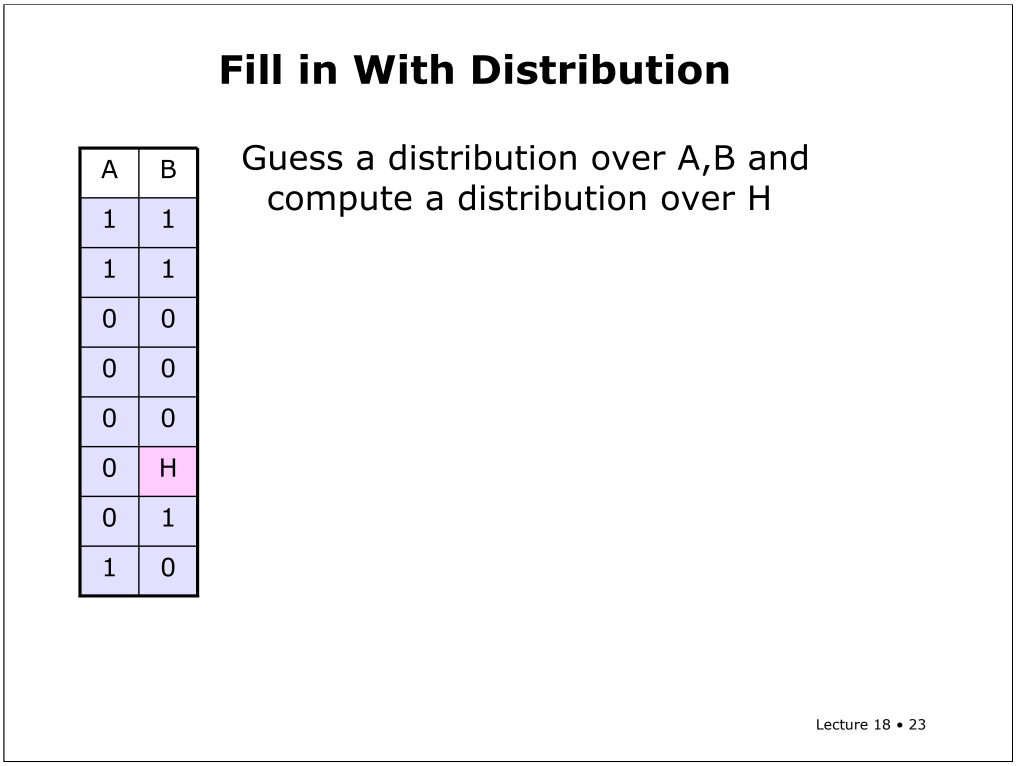
That gives us a model with a log likelihood of –9.481, which is an improvement over –9.498, which was the value of the previous model.



Fill in With Distribution

A	B
1	1
1	1
0	0
0	0
0	0
0	H
0	1
1	0

Filling in the missing data point with a particular value might be a bit too extreme. After all, we can’t usually tell from the data exactly what that value should be, so it makes sense to fill in a “soft” assignment for that value, somehow.

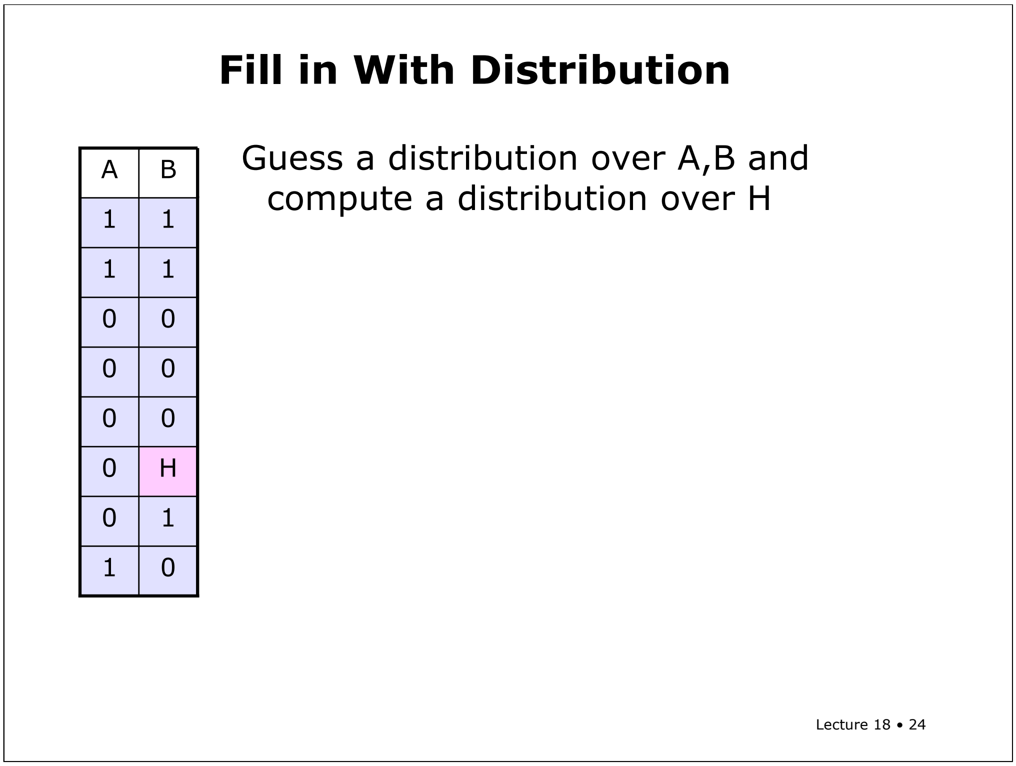


Fill in With Distribution

A	B
1	1
1	1
0	0
0	0
0	0
0	H
0	1
1	0

Guess a distribution over A,B and compute a distribution over H

Ideally, we’d like to fill in that value using our knowledge of the joint distribution of the variables. But we were hoping to use the filled-in value to compute the joint distribution! So what do we do?

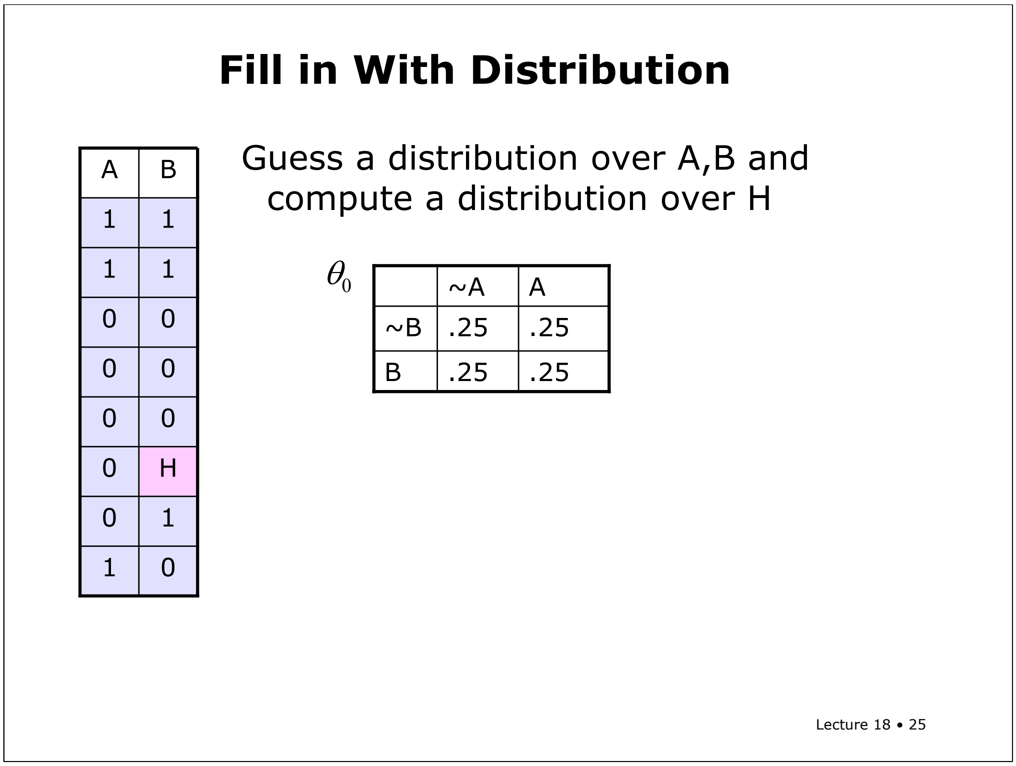


Fill in With Distribution

A	B
1	1
1	1
0	0
0	0
0	0
0	H
0	1
1	0

Guess a distribution over A,B and compute a distribution over H

We’ll look at an iterative procedure that alternates between filling in the missing data with a distribution and estimating a new joint probability distribution.



Fill in With Distribution

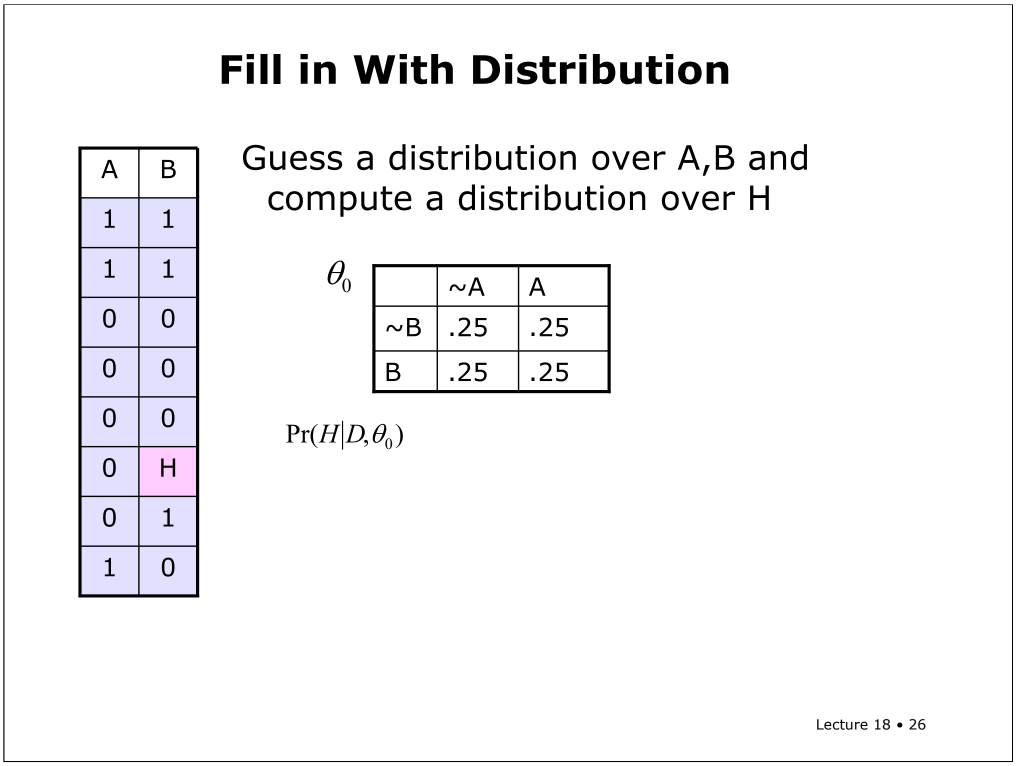
A	B
1	1
1	1
0	0
0	0
0	0
0	H
0	1
1	0

Guess a distribution over A,B and compute a distribution over H

θ_0

	~A	A
~B	.25	.25
B	.25	.25

So, let’s just start by initializing our joint to the uniform 0.25 distribution.



Fill in With Distribution

A	B
1	1
1	1
0	0
0	0
0	0
0	H
0	1
1	0

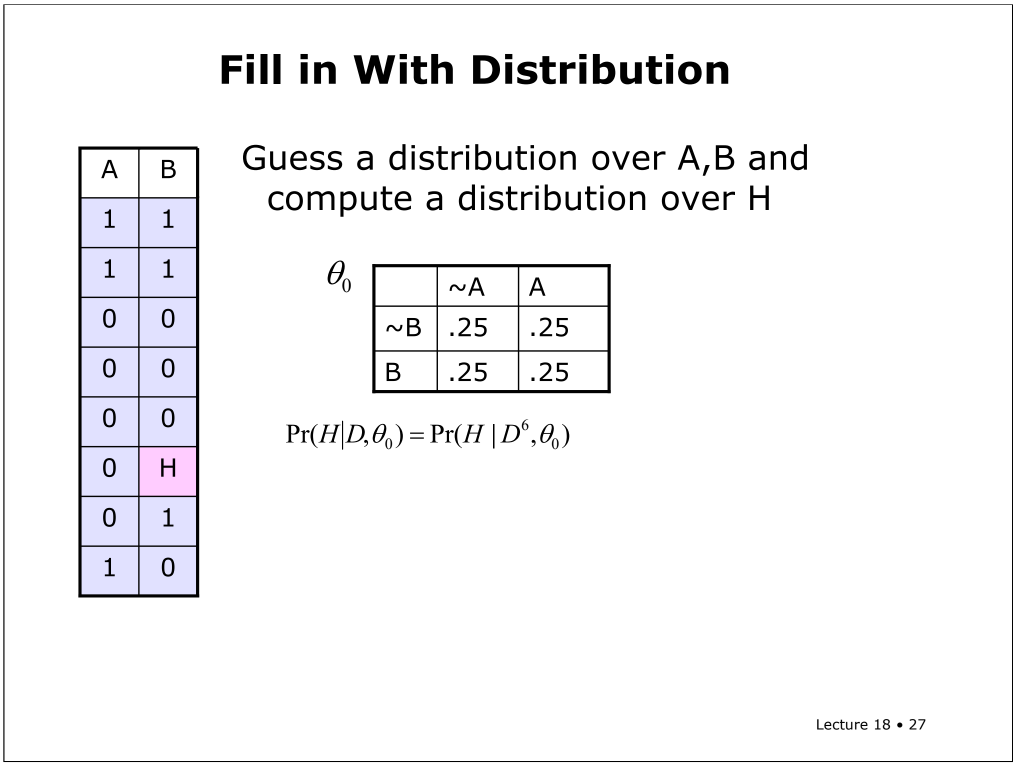
Guess a distribution over A,B and compute a distribution over H

θ_0

	~A	A
~B	.25	.25
B	.25	.25

$\Pr(H|D,\theta_0)$

Then, we can compute a probability distribution over the missing variable H.



Fill in With Distribution

A	B
1	1
1	1
0	0
0	0
0	0
0	H
0	1
1	0

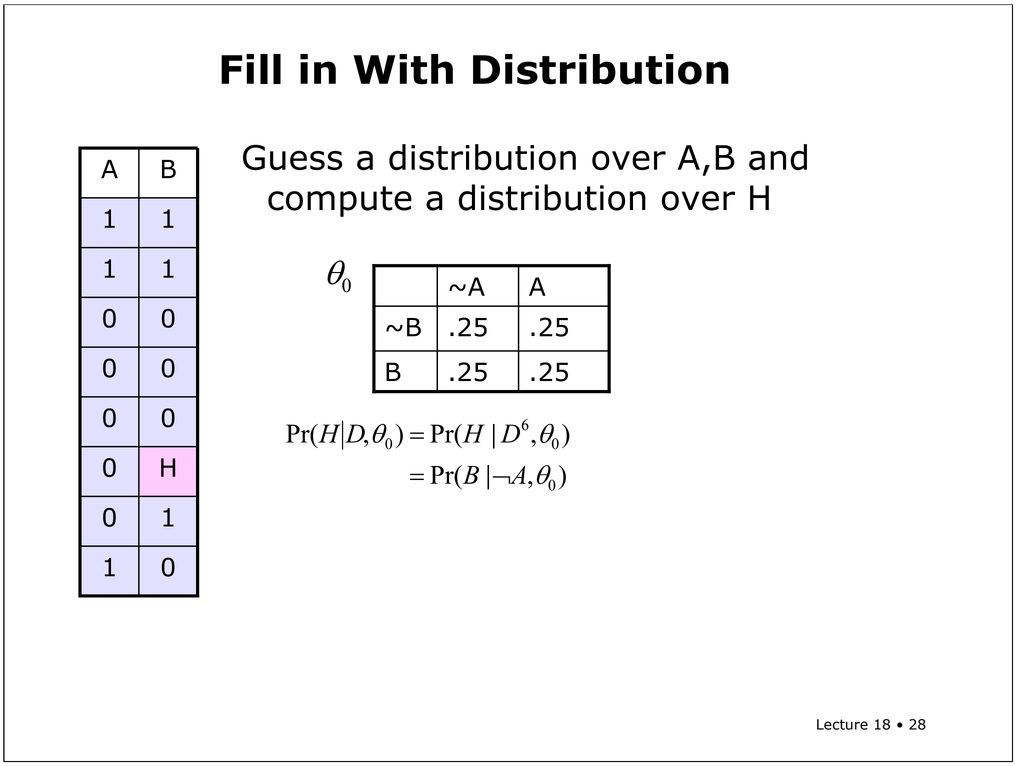
Guess a distribution over A,B and compute a distribution over H

θ_0

	~A	A
~B	.25	.25
B	.25	.25

$$\Pr(H|D,\theta_0) = \Pr(H \mid D^6,\theta_0)$$

First, we note that, under the assumption that the data cases are independent given the model, the value of a missing variable can only depend on observed data in the same case, case 6.



Fill in With Distribution

A	B
1	1
1	1
0	0
0	0
0	0
0	H
0	1
1	0

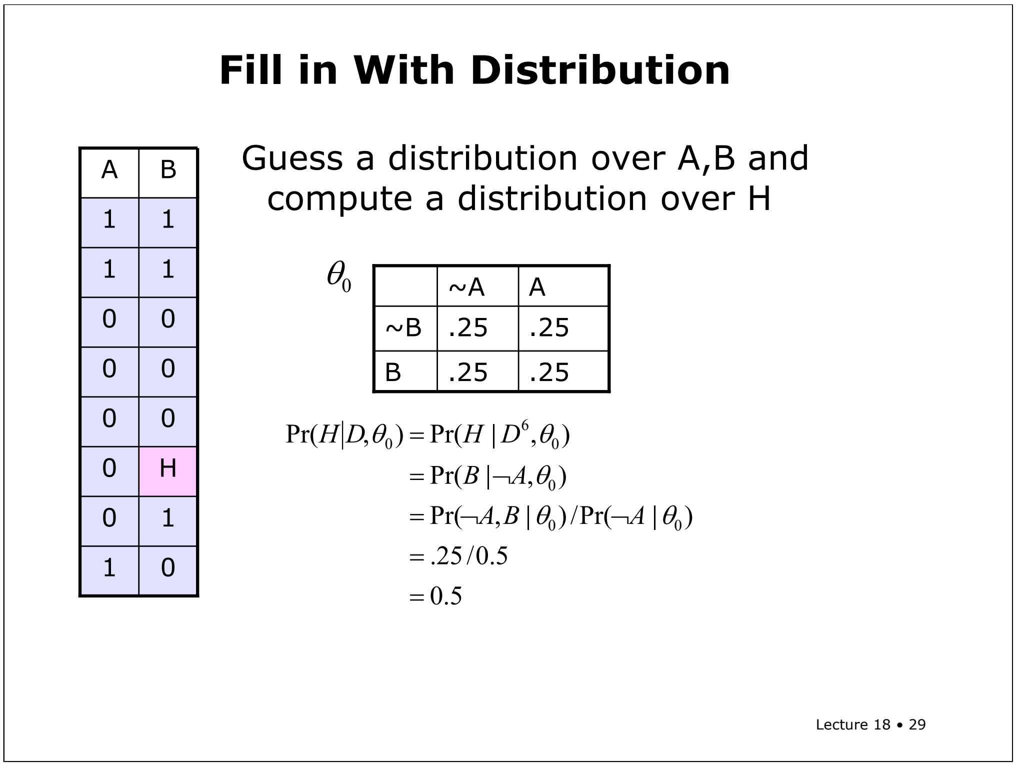
Guess a distribution over A,B and compute a distribution over H

θ_0

	~A	A
~B	.25	.25
B	.25	.25

$$\Pr(H|D,\theta_0) = \Pr(H \mid D^6,\theta_0)$$
$$= \Pr(B \mid \neg A,\theta_0)$$

Since the missing variable is B and the observed one is not A, we just need the probability of B given not A,



Fill in With Distribution

A	B
1	1
1	1
0	0
0	0
0	0
0	H
0	1
1	0

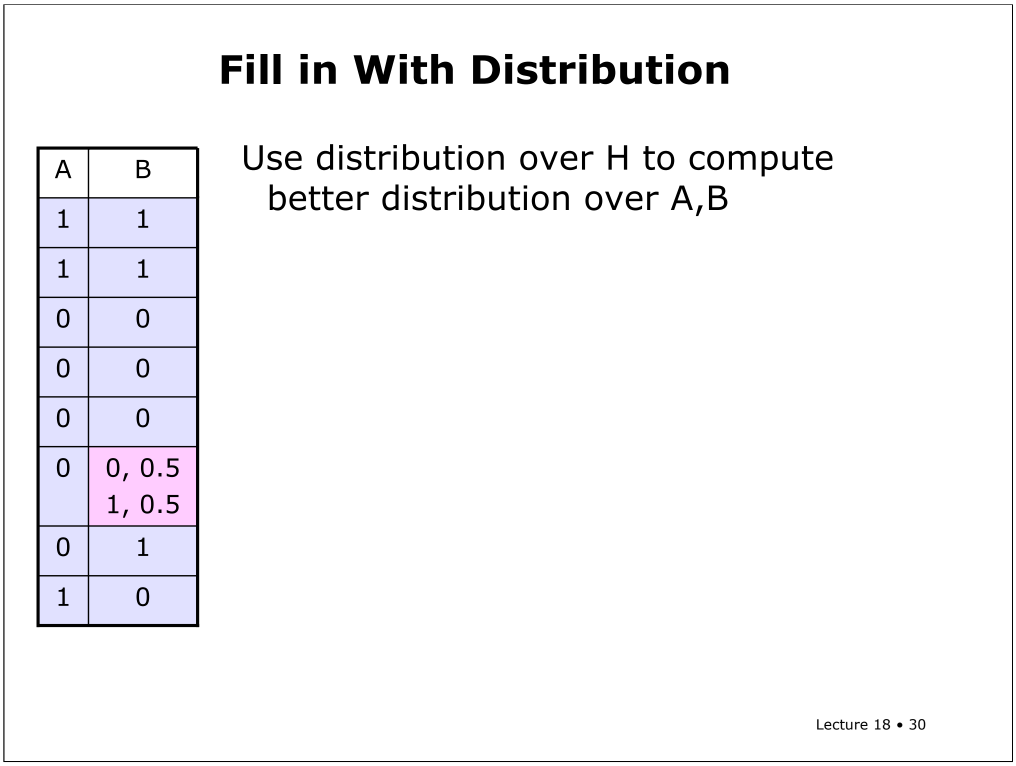
Guess a distribution over A,B and compute a distribution over H

θ_0

	~A	A
~B	.25	.25
B	.25	.25

$$\begin{aligned}\Pr(H|D,\theta_0) &= \Pr(H \mid D^6, \theta_0)\\ &= \Pr(B \mid \neg A, \theta_0)\\ &= \Pr(\neg A, B \mid \theta_0) / \Pr(\neg A \mid \theta_0)\\ &= .25 / 0.5\\ &= 0.5\end{aligned}$$

which we can calculate easily from the distribution.

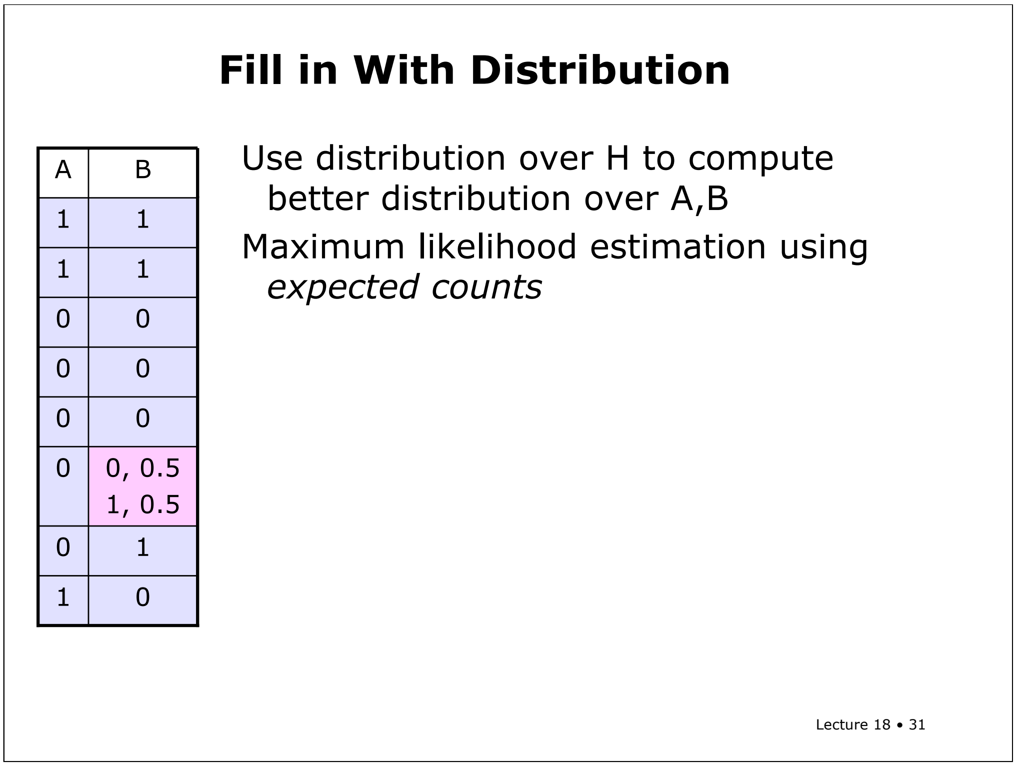


Fill in With Distribution

A	B
1	1
1	1
0	0
0	0
0	0
0	0, 0.5 1, 0.5
0	1
1	0

Use distribution over H to compute better distribution over A,B

Now we can fill in our missing data with a distribution: it has value 0 with probability 0.5 and value 1 with probability 0.5.



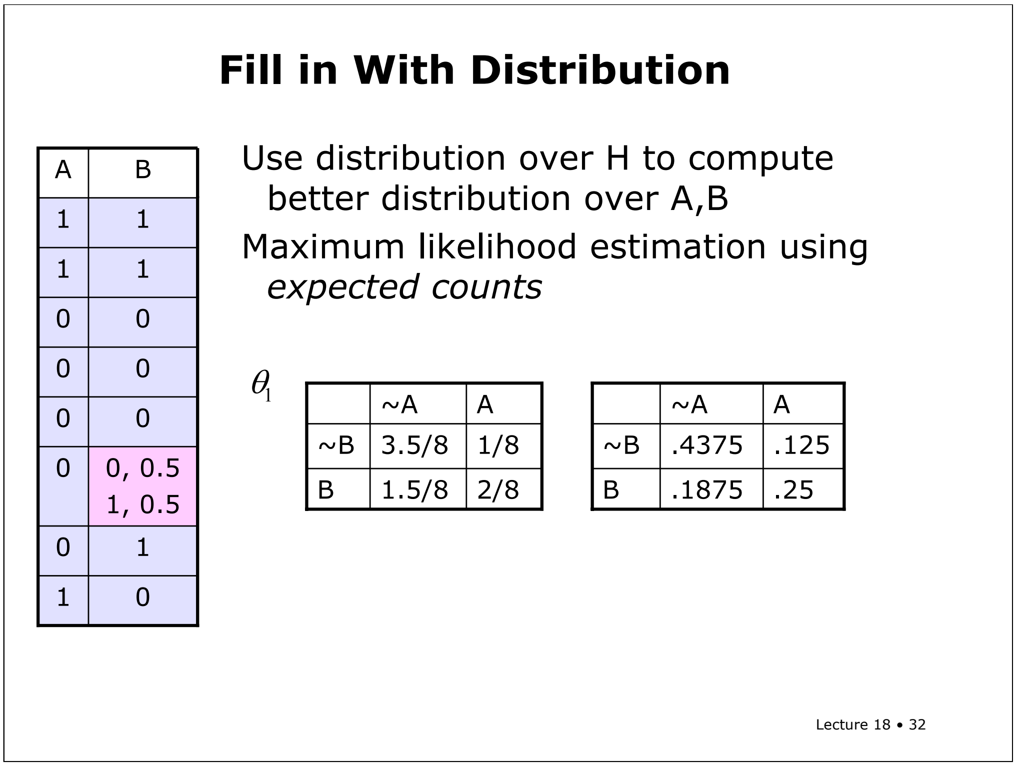
Fill in With Distribution

A	B
1	1
1	1
0	0
0	0
0	0
0	0, 0.5 1, 0.5
0	1
1	0

Use distribution over H to compute better distribution over A,B

Maximum likelihood estimation using *expected counts*

Given those values we can re-estimate the parameters in our model. We'll do counting, as before, but this time, the 6th data case will be counted as 1/2 an instance of 00 and 1/2 an instance of 01. You can think of these counts as expected values of the true count, based on the uncertainty in the actual value of H.



Fill in With Distribution

A	B
1	1
1	1
0	0
0	0
0	0
0	0, 0.5 1, 0.5
0	1
1	0

Use distribution over H to compute better distribution over A,B

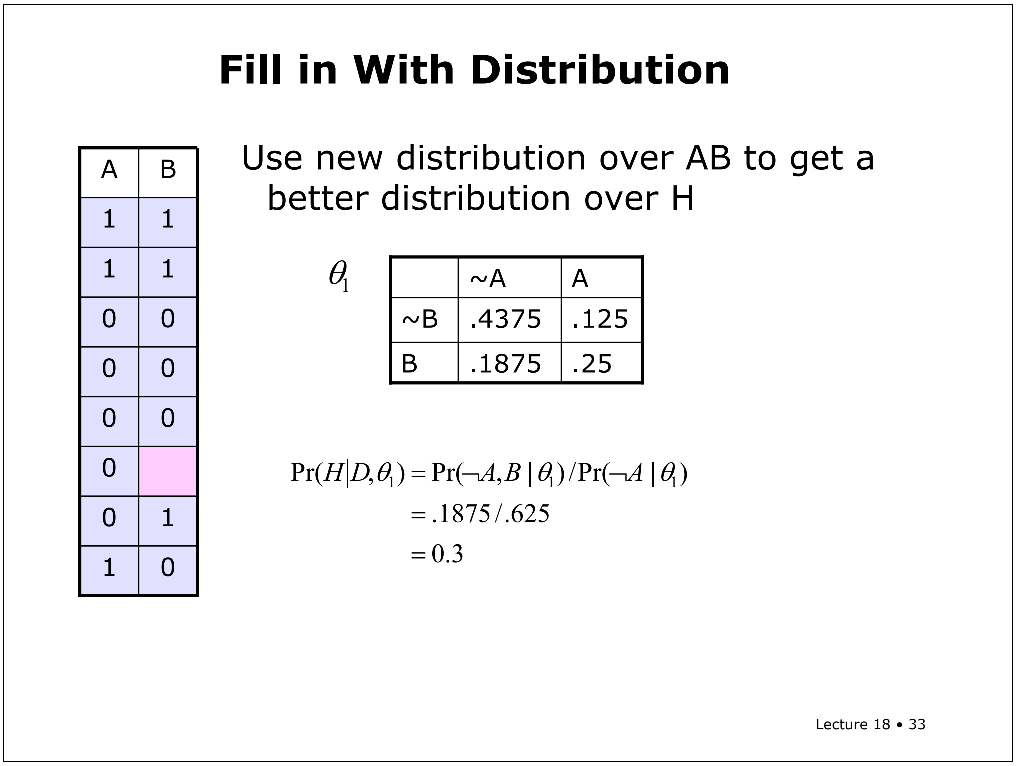
Maximum likelihood estimation using *expected counts*

θ_1

	~A	A
~B	3.5/8	1/8
B	1.5/8	2/8

	~A	A
~B	.4375	.125
B	.1875	.25

Given the expected counts, we can calculate a new model, theta 1.



Fill in With Distribution

A	B
1	1
1	1
0	0
0	0
0	0
0	
0	1
1	0

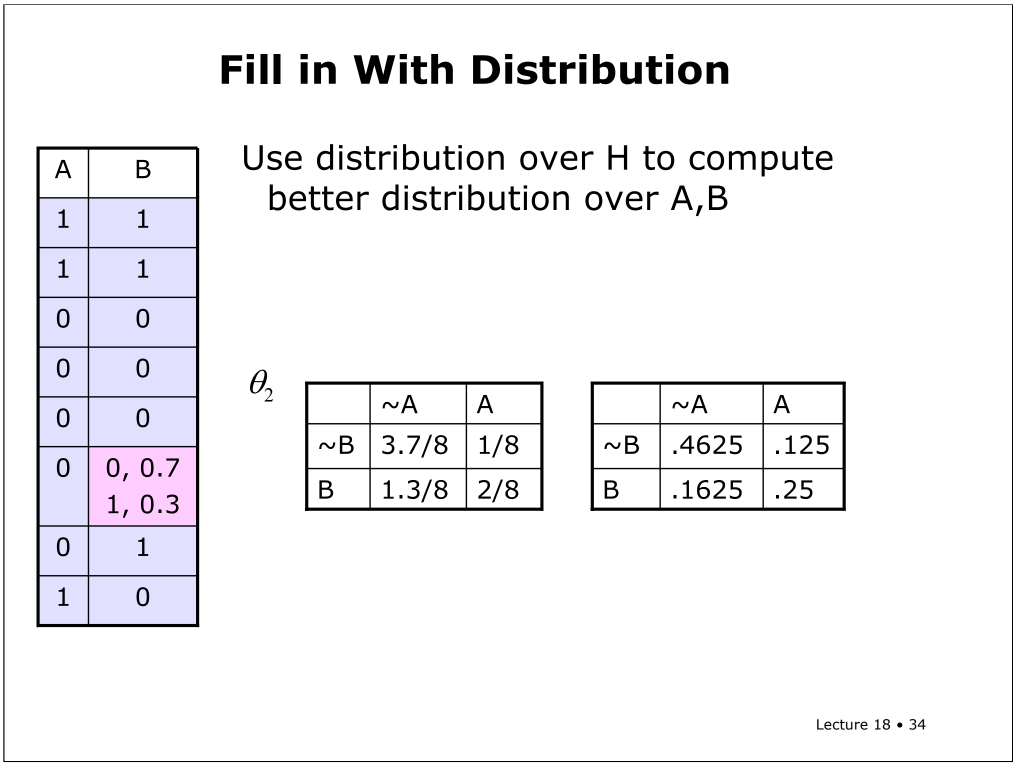
Use new distribution over AB to get a better distribution over H

θ_1

	~A	A
~B	.4375	.125
B	.1875	.25

$$\begin{aligned}\Pr(H|D,\theta_1) &= \Pr(\neg A, B \mid \theta_1)/\Pr(\neg A \mid \theta_1)\\ &= .1875/.625\\ &= 0.3\end{aligned}$$

Now, given our new distribution theta 1, we can do a better job of estimating a probability distribution over H. Our new estimate is that H is true with probability 0.3.



Fill in With Distribution

A	B
1	1
1	1
0	0
0	0
0	0
0	0, 0.7 1, 0.3
0	1
1	0

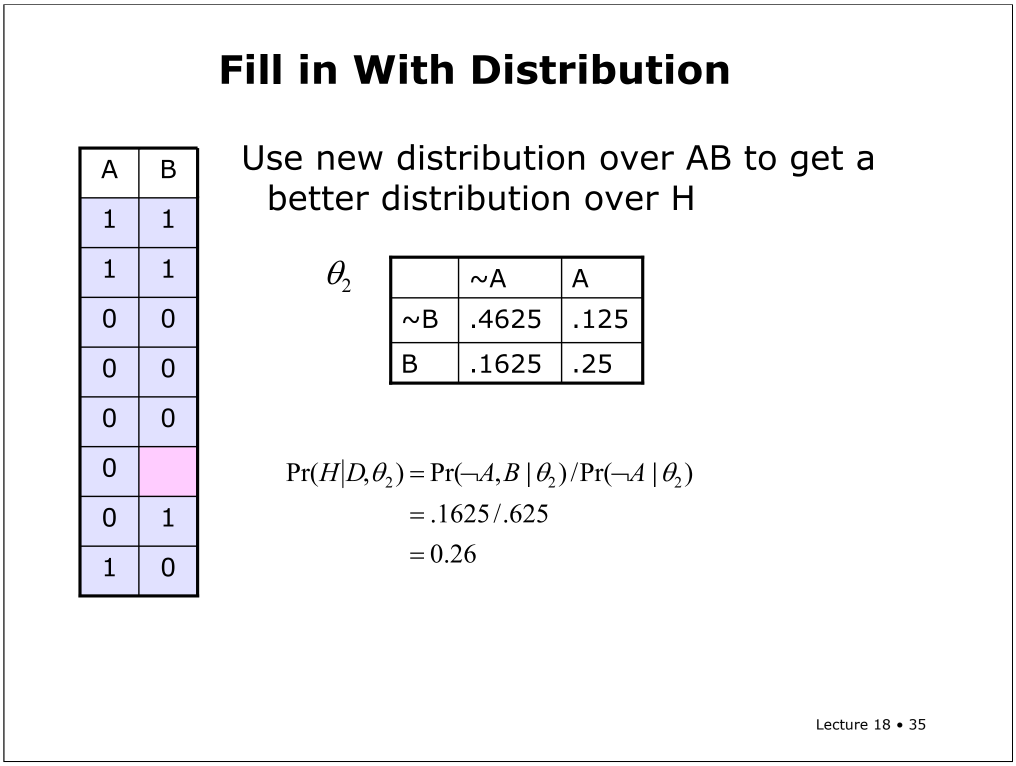
Use distribution over H to compute better distribution over A,B

θ_2

	~A	A
~B	3.7/8	1/8
B	1.3/8	2/8

	~A	A
~B	.4625	.125
B	.1625	.25

We plug the new estimate into the data set, compute new expected counts, and get a new model, theta 2.



Fill in With Distribution

A	B
1	1
1	1
0	0
0	0
0	0
0	
0	1
1	0

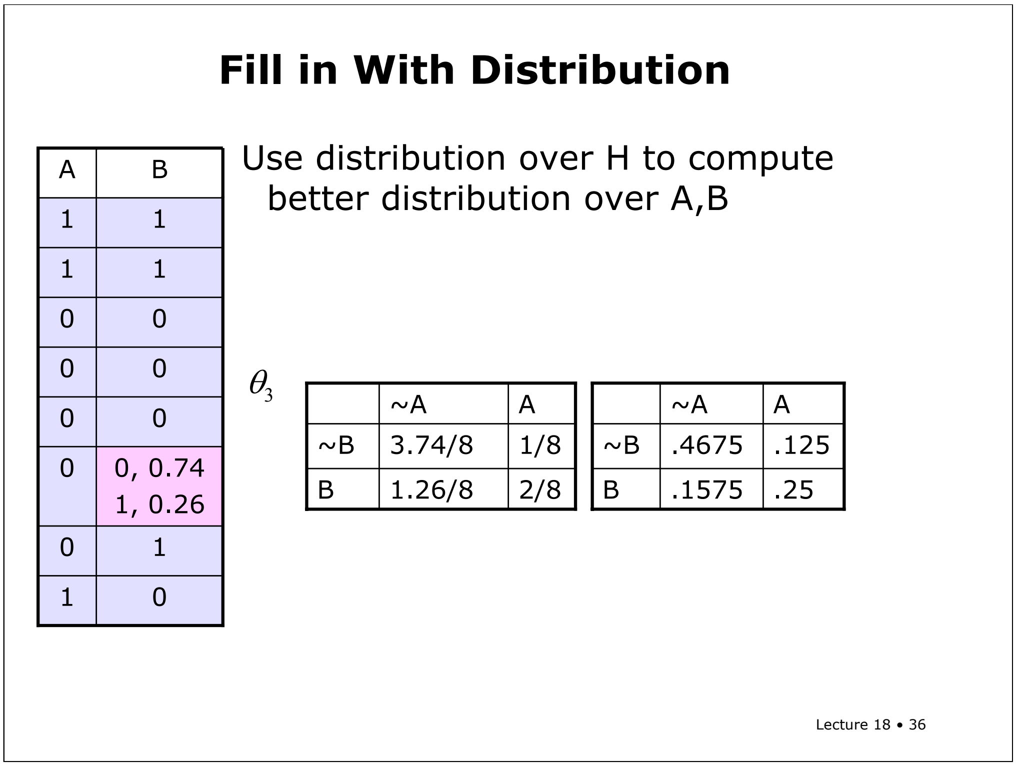
Use new distribution over AB to get a better distribution over H

θ_2

	~A	A
~B	.4625	.125
B	.1625	.25

$$\begin{aligned}\Pr(H|D,\theta_2) &= \Pr(\neg A, B \mid \theta_2)/\Pr(\neg A \mid \theta_2)\\ &= .1625/.625\\ &= 0.26\end{aligned}$$

Given theta 2, we now estimate the probability of H being true to be 0.26.



Fill in With Distribution

A	B
1	1
1	1
0	0
0	0
0	0
0	0, 0.74 1, 0.26
0	1
1	0

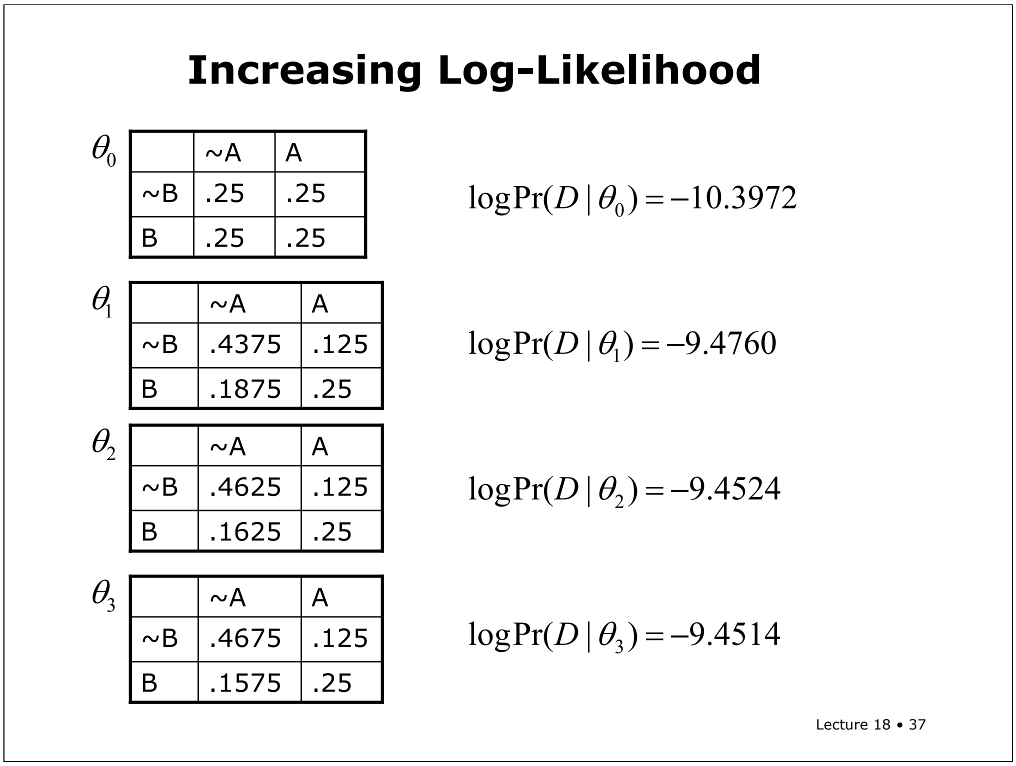
Use distribution over H to compute better distribution over A,B

θ_3

	~A	A
~B	3.74/8	1/8
B	1.26/8	2/8

	~A	A
~B	.4675	.125
B	.1575	.25

And that estimate leads us to a new theta 3.



Increasing Log-Likelihood

θ_0

	~A	A
~B	.25	.25
B	.25	.25

$$\log \Pr(D \mid \theta_0) = -10.3972$$

θ_1

	~A	A
~B	.4375	.125
B	.1875	.25

$$\log \Pr(D \mid \theta_1) = -9.4760$$

θ_2

	~A	A
~B	.4625	.125
B	.1625	.25

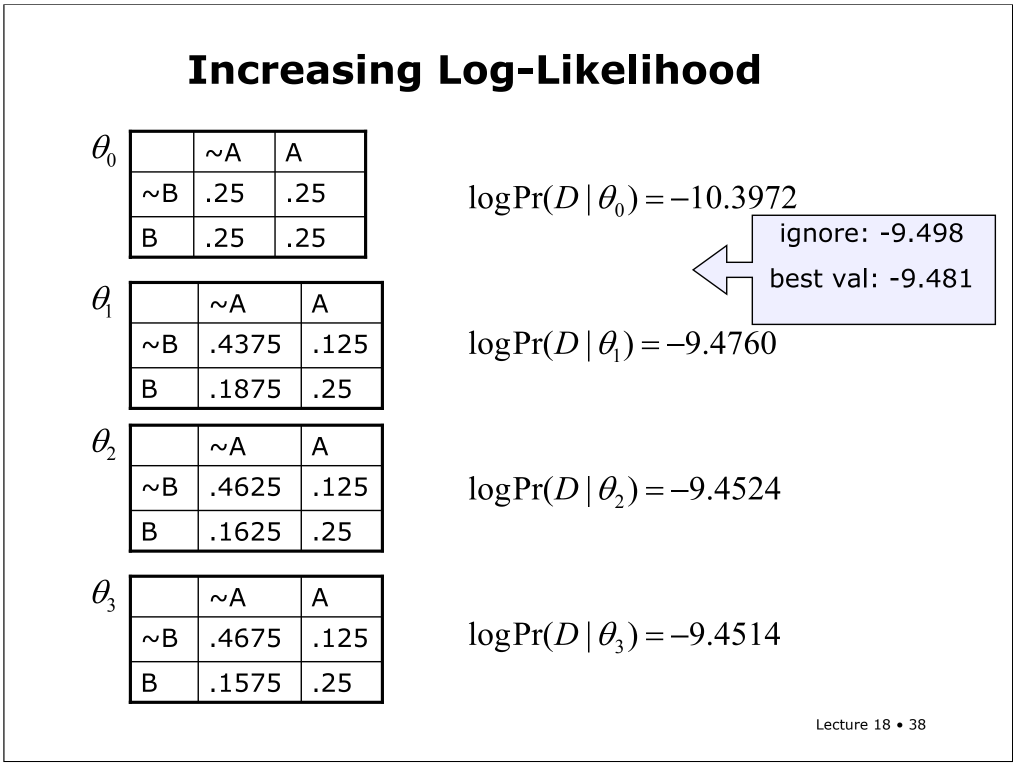
$$\log \Pr(D \mid \theta_2) = -9.4524$$

θ_3

	~A	A
~B	.4675	.125
B	.1575	.25

$$\log \Pr(D \mid \theta_3) = -9.4514$$

We can iterate this process until it converges or we get tired, or something. One important thing to notice is that the log-likelihood is increasing on each iteration.



Increasing Log-Likelihood

θ_0

	~A	A
~B	.25	.25
B	.25	.25

$$\log \Pr(D \mid \theta_0) = -10.3972$$

ignore: -9.498

best val: -9.481

θ_1

	~A	A
~B	.4375	.125
B	.1875	.25

$$\log \Pr(D \mid \theta_1) = -9.4760$$

θ_2

	~A	A
~B	.4625	.125
B	.1625	.25

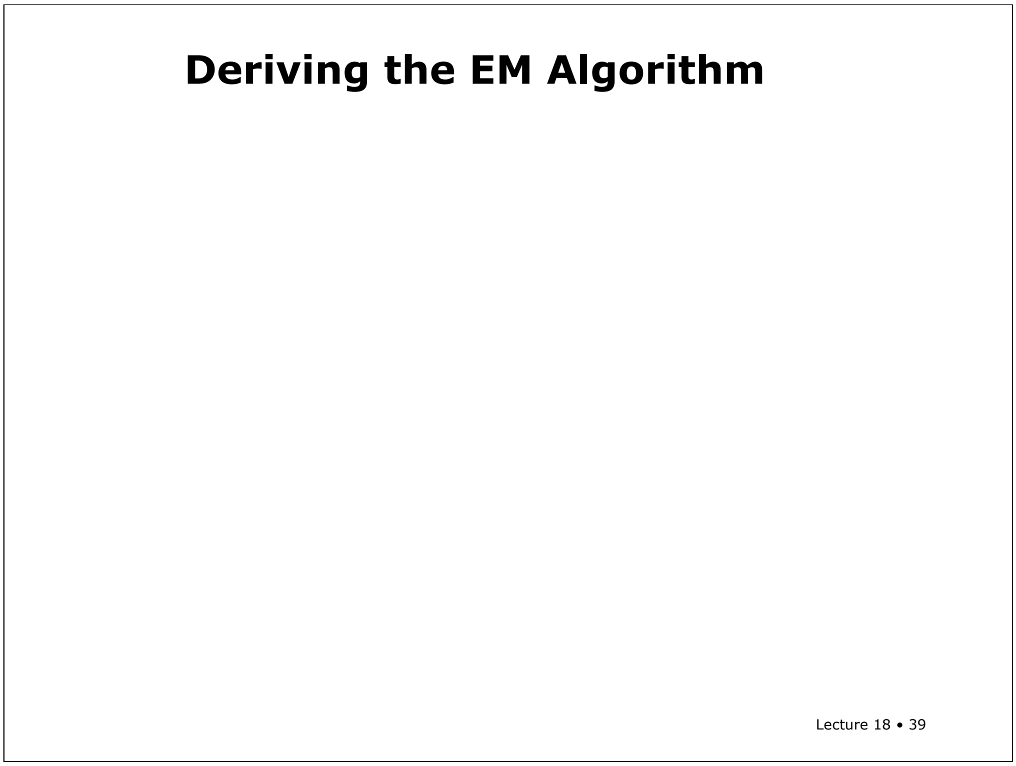
$$\log \Pr(D \mid \theta_2) = -9.4524$$

θ_3

	~A	A
~B	.4675	.125
B	.1575	.25

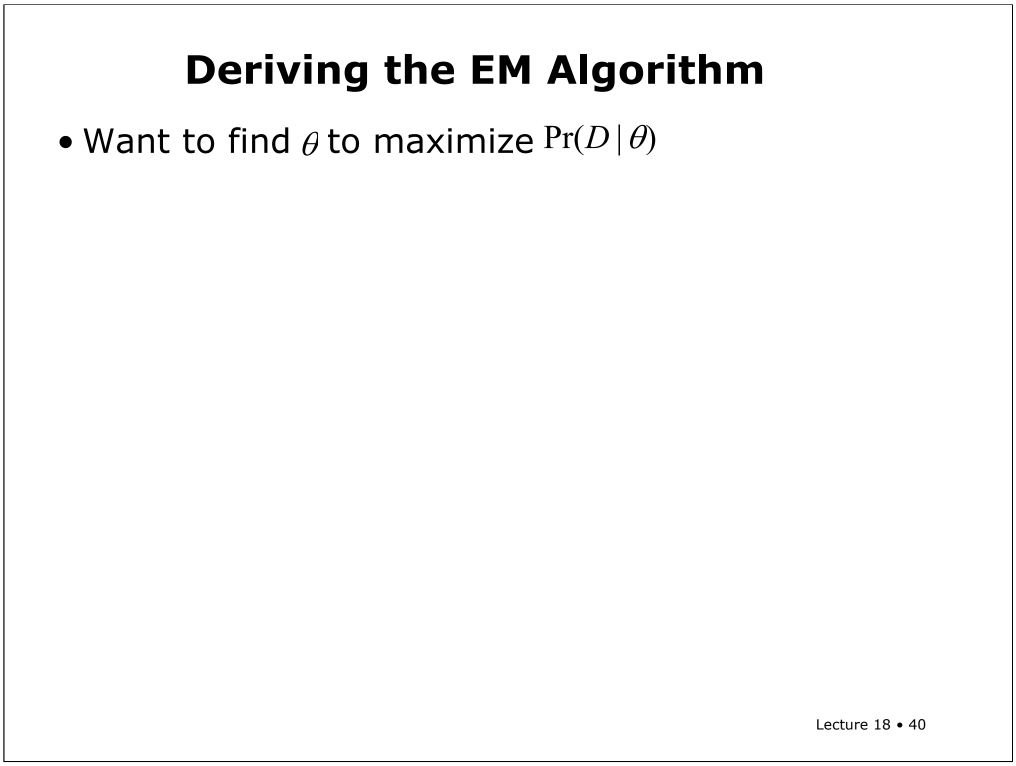
$$\log \Pr(D \mid \theta_3) = -9.4514$$

And even after one iteration, our model is better than the ones we derived by ignoring case 6 or by plugging in the best value for H.



Deriving the EM Algorithm

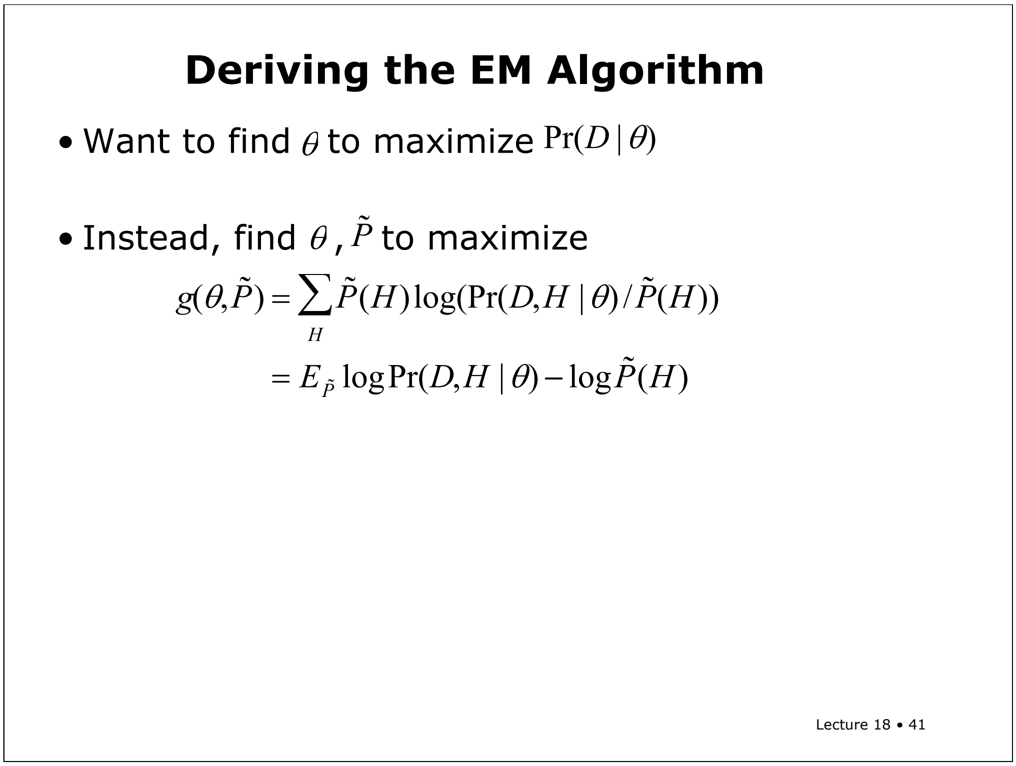
That iterative process that we just did is an instance of a general procedure, called the EM algorithm. It’s called EM for “expectation-maximization”, though the way we’ll look at it, it’s more like “maximization-maximization”.



Deriving the EM Algorithm

- Want to find θ to maximize $\Pr(D \mid \theta)$

So, our goal is to find the theta that maximizes the probability of data given theta.



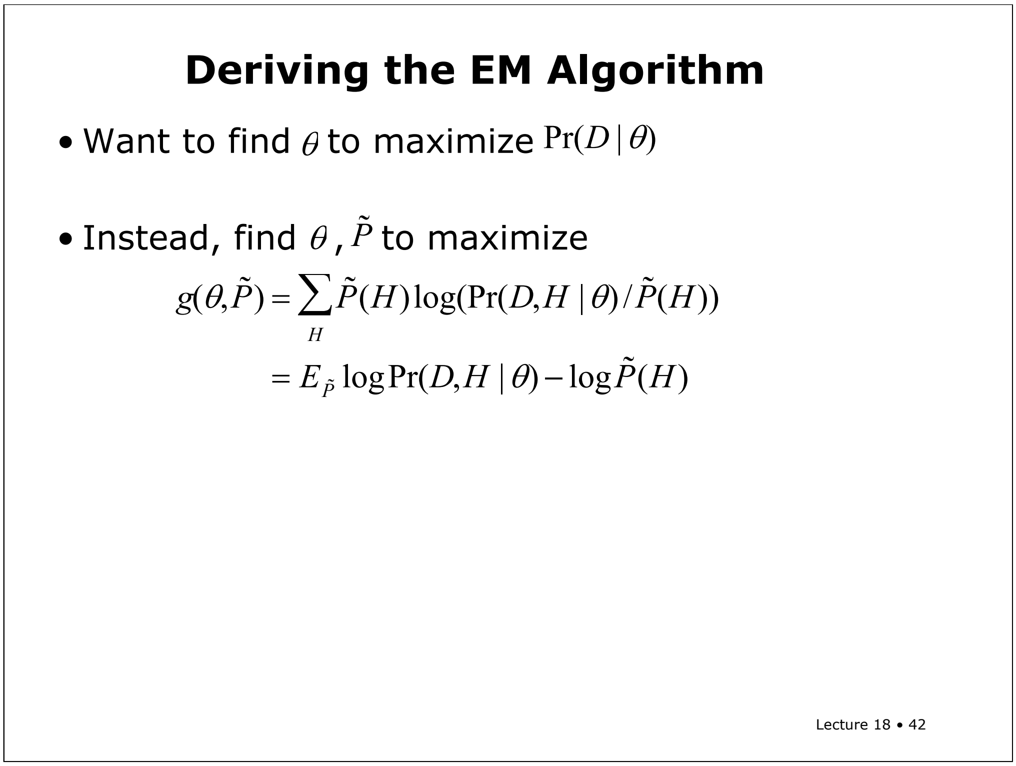
Deriving the EM Algorithm

- Want to find θ to maximize $\Pr(D \mid \theta)$

- Instead, find θ, $\tilde{P}$ to maximize

$$\begin{aligned} g(\theta,\tilde{P}) &= \sum_{H} \tilde{P}(H) \log(\Pr(D,H \mid \theta)/\tilde{P}(H)) \\ &= E_{\tilde{P}} \log \Pr(D,H \mid \theta) - \log \tilde{P}(H) \end{aligned}$$

The problem is that it’s hard to maximize that directly. Instead, some clever statistician found this expression, g of theta and P tilde. We’re going to try to maximize it instead.

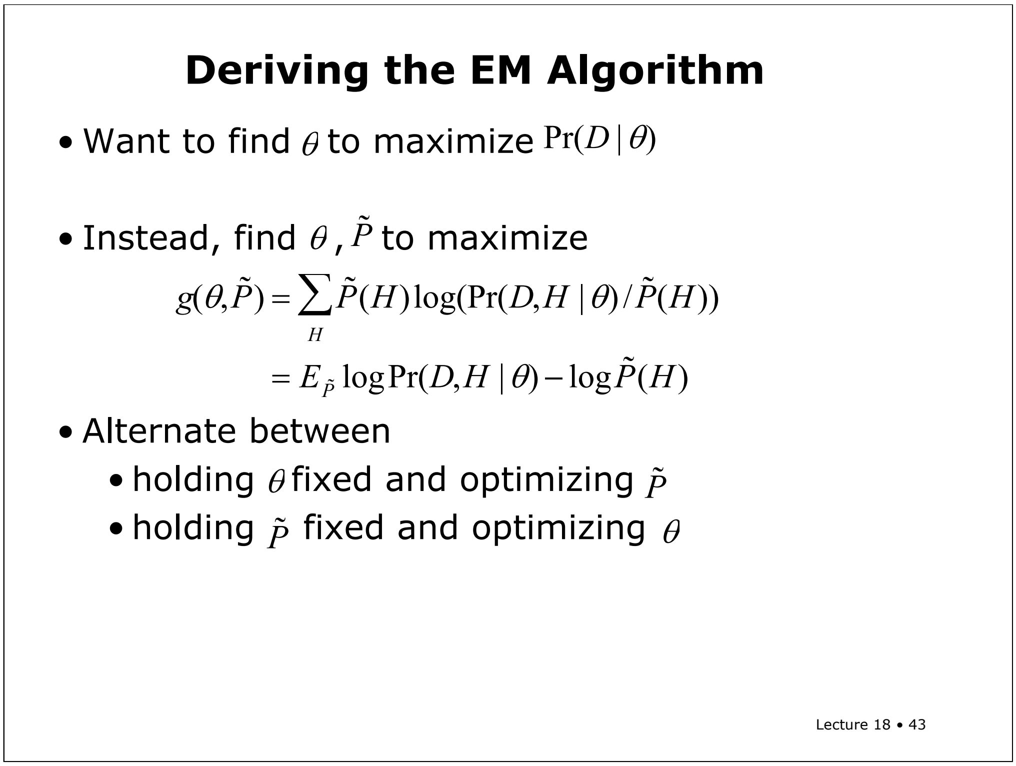


Deriving the EM Algorithm

- Want to find θ to maximize $\Pr(D \mid \theta)$
- Instead, find θ, $\tilde{P}$ to maximize

$$g(\theta,\tilde{P}) = \sum_{H} \tilde{P}(H) \log(\Pr(D,H \mid \theta) / \tilde{P}(H))$$
$$= E_{\tilde{P}} \log \Pr(D,H \mid \theta) - \log \tilde{P}(H)$$

P tilde is a probability distribution over the hidden variables.



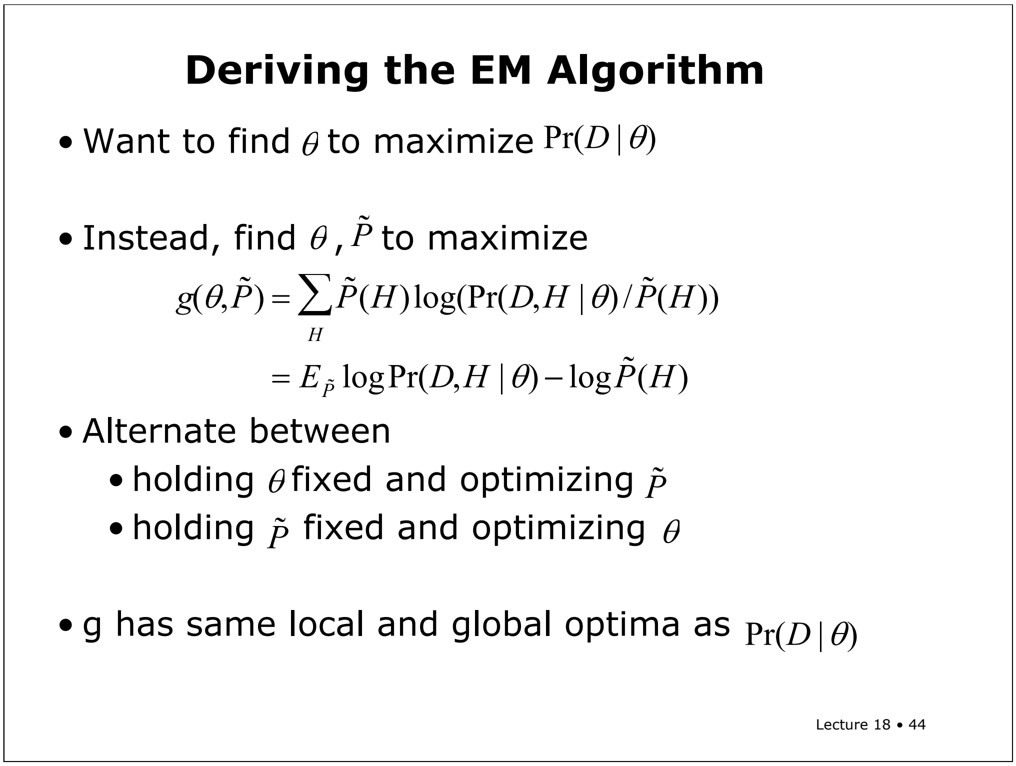
Deriving the EM Algorithm

- Want to find θ to maximize $\Pr(D \mid \theta)$
- Instead, find $\theta, \tilde{P}$ to maximize

$$
\begin{aligned}
g(\theta, \tilde{P}) &= \sum_{H} \tilde{P}(H) \log(\Pr(D, H \mid \theta) / \tilde{P}(H)) \\
&= E_{\tilde{P}} \log \Pr(D, H \mid \theta) - \log \tilde{P}(H)
\end{aligned}
$$

- Alternate between
  - holding θ fixed and optimizing $\tilde{P}$
  - holding $\tilde{P}$ fixed and optimizing θ

So, how are we going to find an optimum of g? We can do that by holding one argument fixed and finding an optimum with respect to the other, and repeating that procedure over and over.



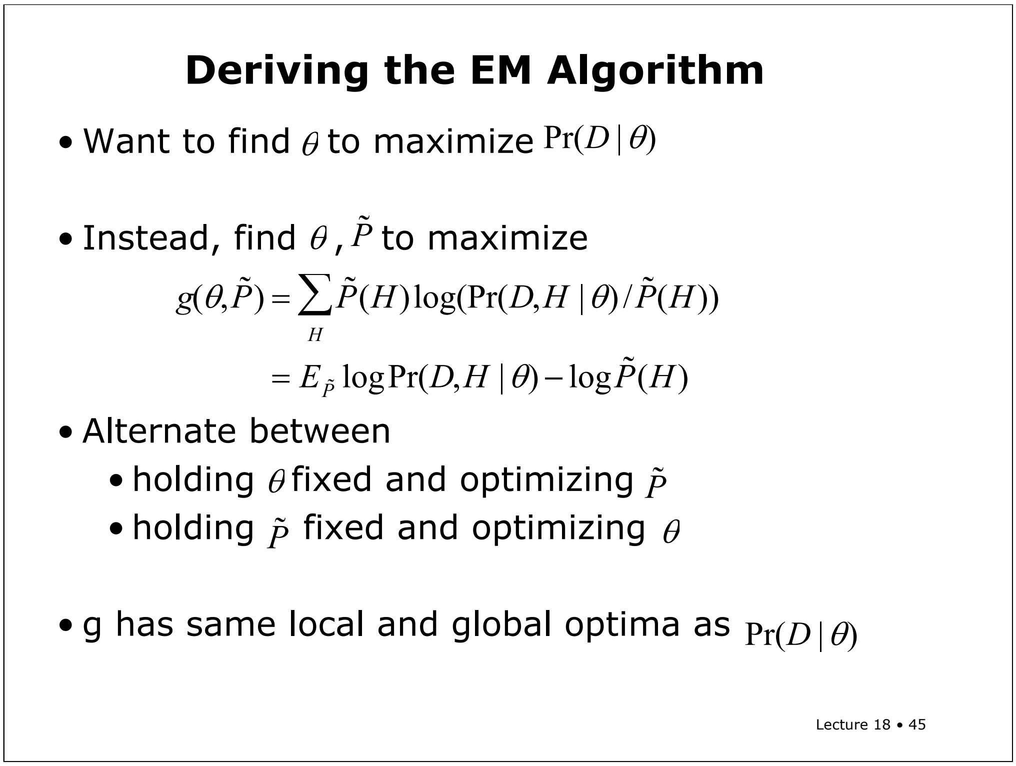
Deriving the EM Algorithm

- Want to find θ to maximize $\Pr(D \mid \theta)$
- Instead, find $\theta, \tilde{P}$ to maximize

$$g(\theta,\tilde{P}) = \sum_{H} \tilde{P}(H) \log(\Pr(D,H \mid \theta)/\tilde{P}(H))$$
$$= E_{\tilde{P}} \log \Pr(D,H \mid \theta) - \log \tilde{P}(H)$$

- Alternate between
  - holding θ fixed and optimizing $\tilde{P}$
  - holding $\tilde{P}$ fixed and optimizing θ
- g has same local and global optima as $\Pr(D \mid \theta)$

So, in our algorithm, we’ll hold theta (the model) fixed and find the best distribution over the hidden variables. Then we’ll hold the distribution over the hidden variables fixed and find the best model.



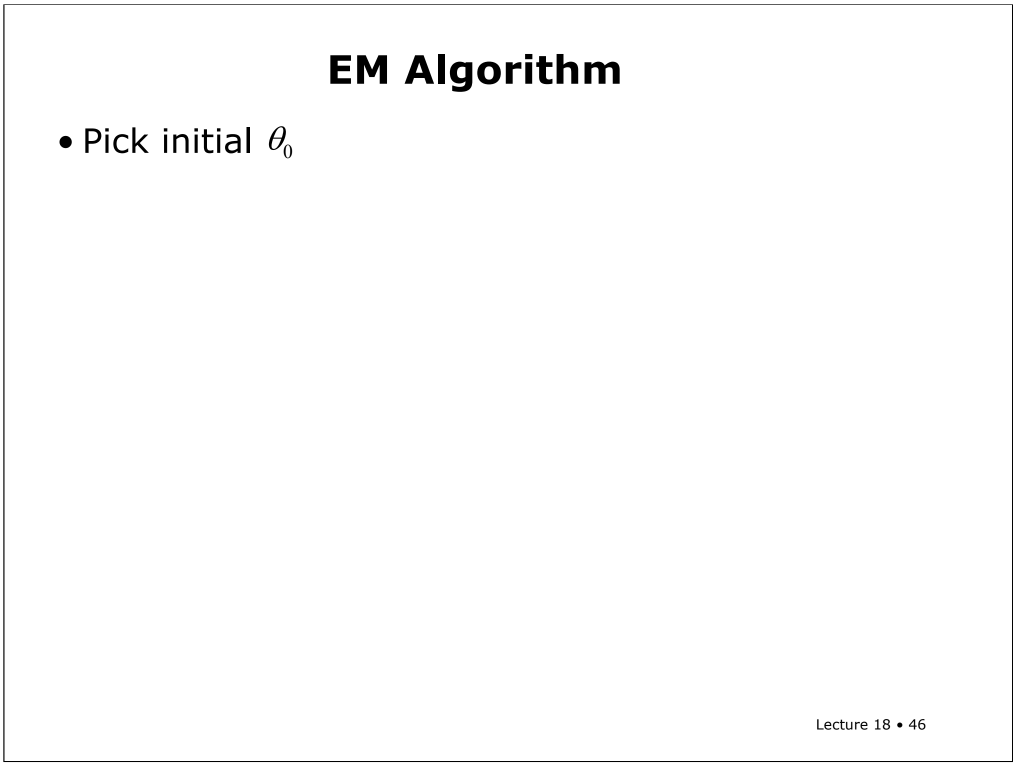
Deriving the EM Algorithm

- Want to find θ to maximize $\Pr(D \mid \theta)$
- Instead, find θ, $\tilde{P}$ to maximize

$$g(\theta, \tilde{P}) = \sum_{H} \tilde{P}(H) \log(\Pr(D, H \mid \theta) / \tilde{P}(H))$$
$$= E_{\tilde{P}} \log \Pr(D, H \mid \theta) - \log \tilde{P}(H)$$

- Alternate between
  - holding θ fixed and optimizing $\tilde{P}$
  - holding $\tilde{P}$ fixed and optimizing θ
- g has same local and global optima as $\Pr(D \mid \theta)$

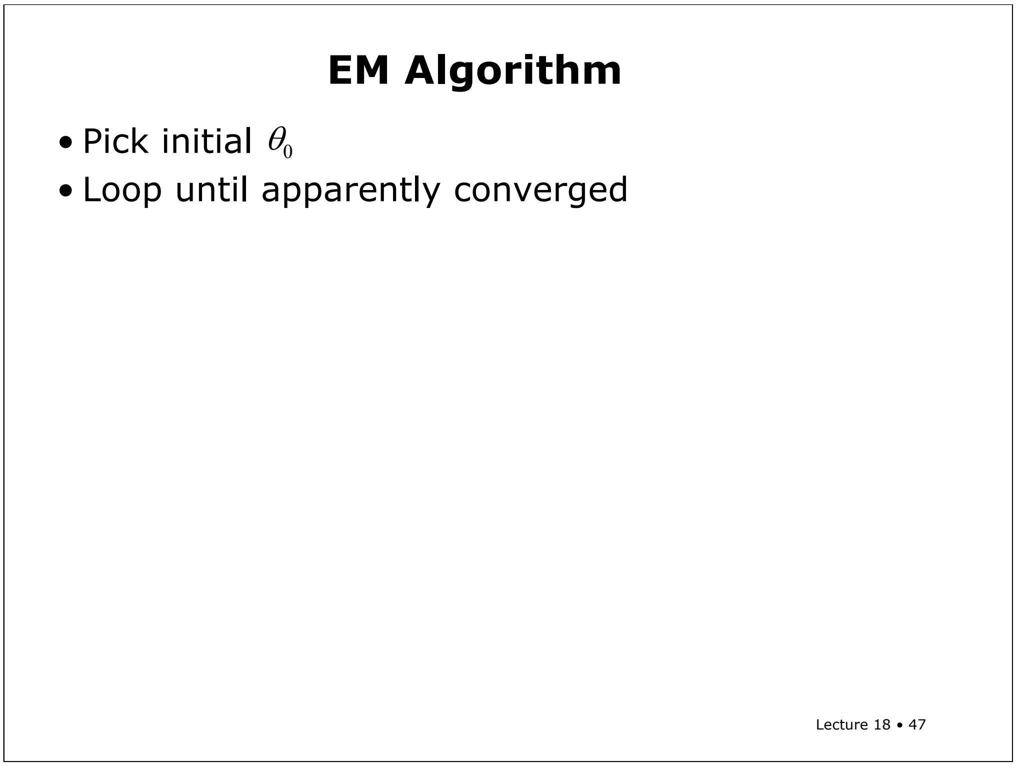
The clever statisticians that invented the g function proved that it has the same local and global optima with respect to theta as the likelihood function that we really want to optimize. So, working with g should get us the answer we need, and it’s easier to work with than the straight likelihood.



EM Algorithm

- Pick initial θ_0

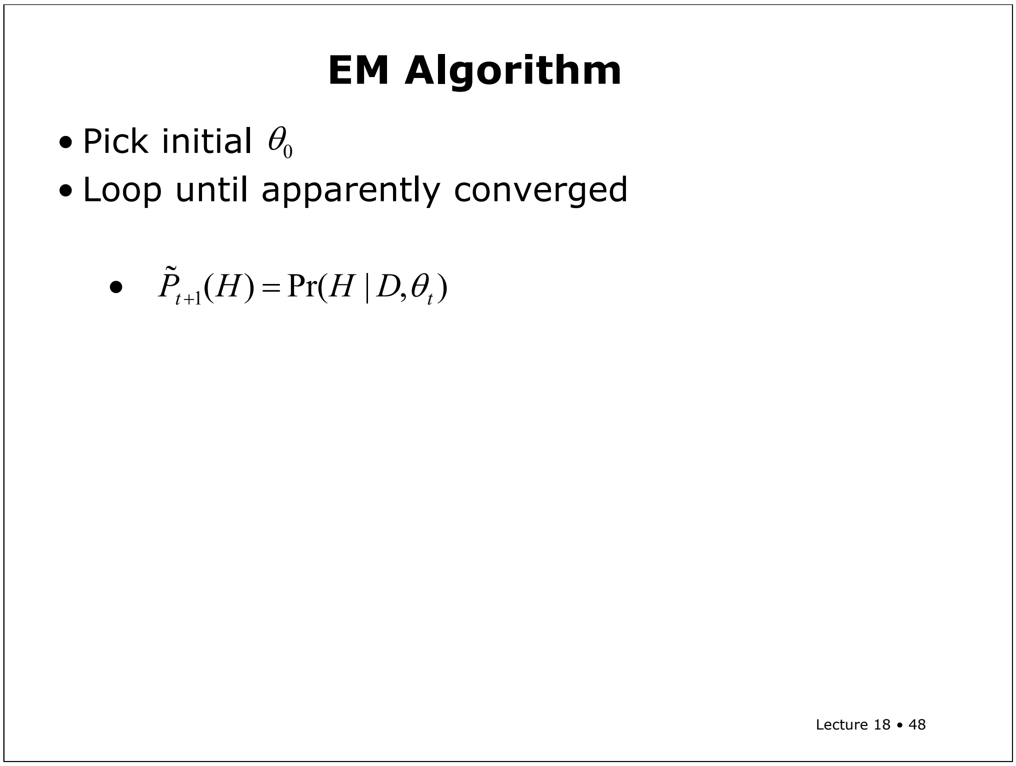
So, here’s the algorithm in a bit more detail. We start by picking some initial model, theta 0.



EM Algorithm

- Pick initial θ_0
- Loop until apparently converged

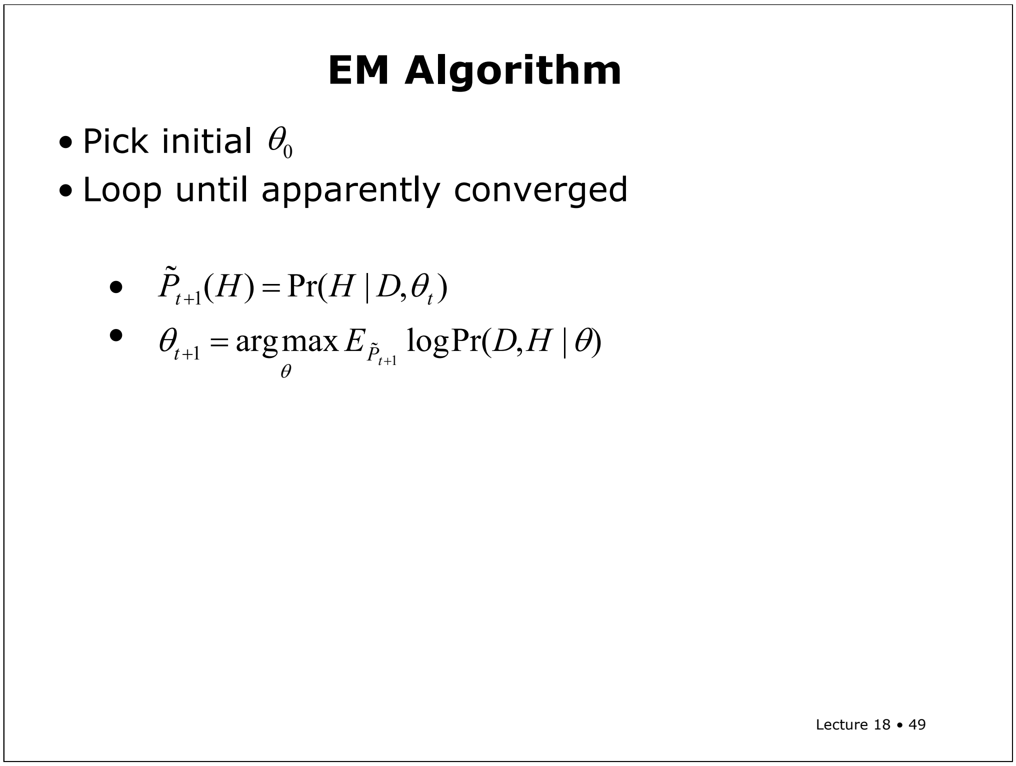
Then, we loop until we think the process has converged, alternating between two steps.



EM Algorithm

- Pick initial θ_0
- Loop until apparently converged
  - $\tilde{P}_{t+1}(H) = \Pr(H \mid D, \theta_t)$

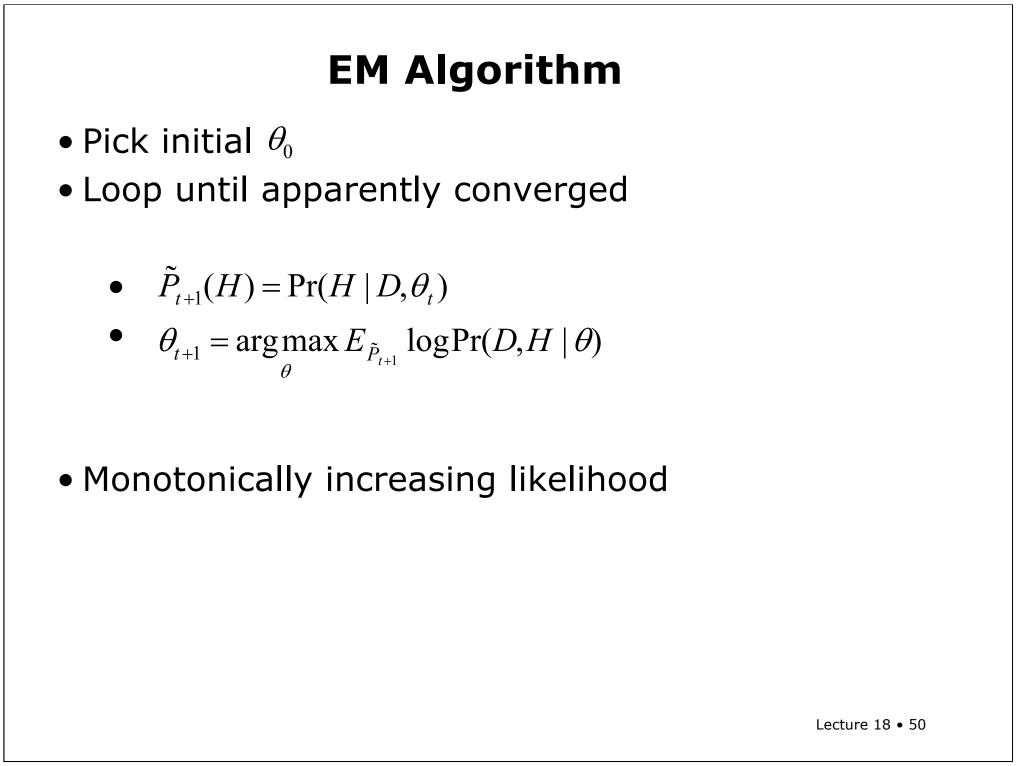
In the first step, we set our distribution over the hidden variables to be the probability of the hidden variables given the observed data and the current model.



EM Algorithm

- Pick initial θ_0
- Loop until apparently converged
  - $\tilde{P}_{t+1}(H) = \Pr(H \mid D, \theta_t)$
  - $\theta_{t+1} = \arg\max_{\theta} E_{\tilde{P}_{t+1}} \log \Pr(D, H \mid \theta)$

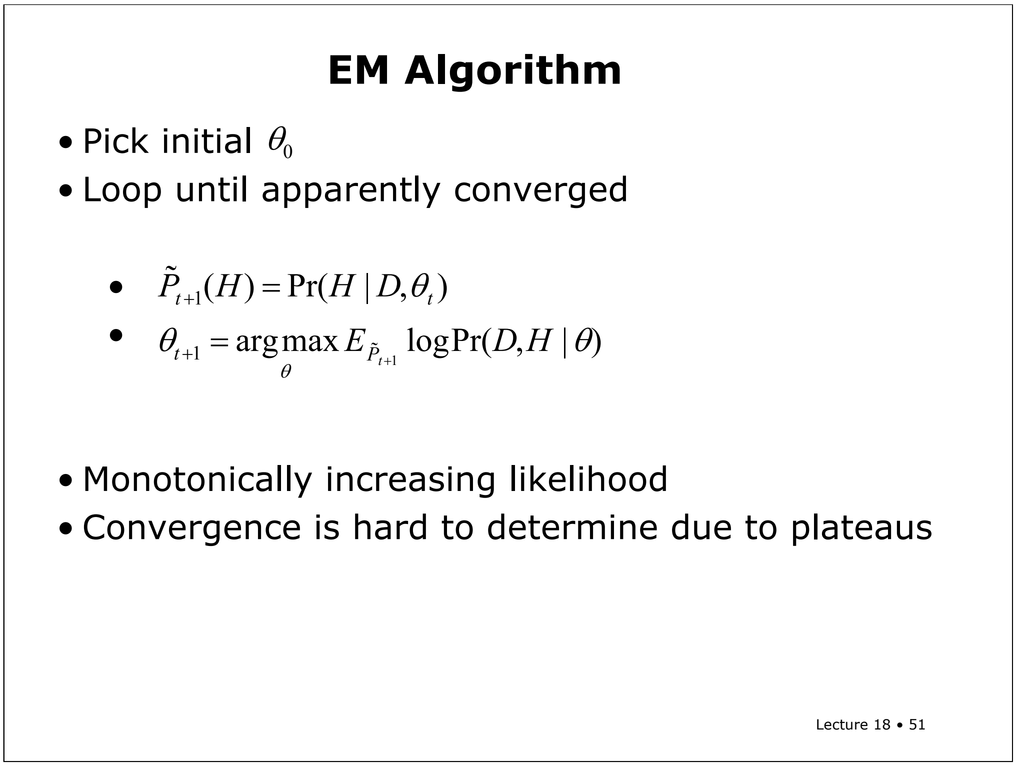
In the second step, we find the maximum likelihood model for the “expected data”, using the distribution over H to generate expected counts for the different cases.



EM Algorithm

- Pick initial θ_0
- Loop until apparently converged
  - $\tilde{P}_{t+1}(H) = \Pr(H \mid D, \theta_t)$
  - $\theta_{t+1} = \arg\max_{\theta} E_{\tilde{P}_{t+1}} \log \Pr(D, H \mid \theta)$
- Monotonically increasing likelihood

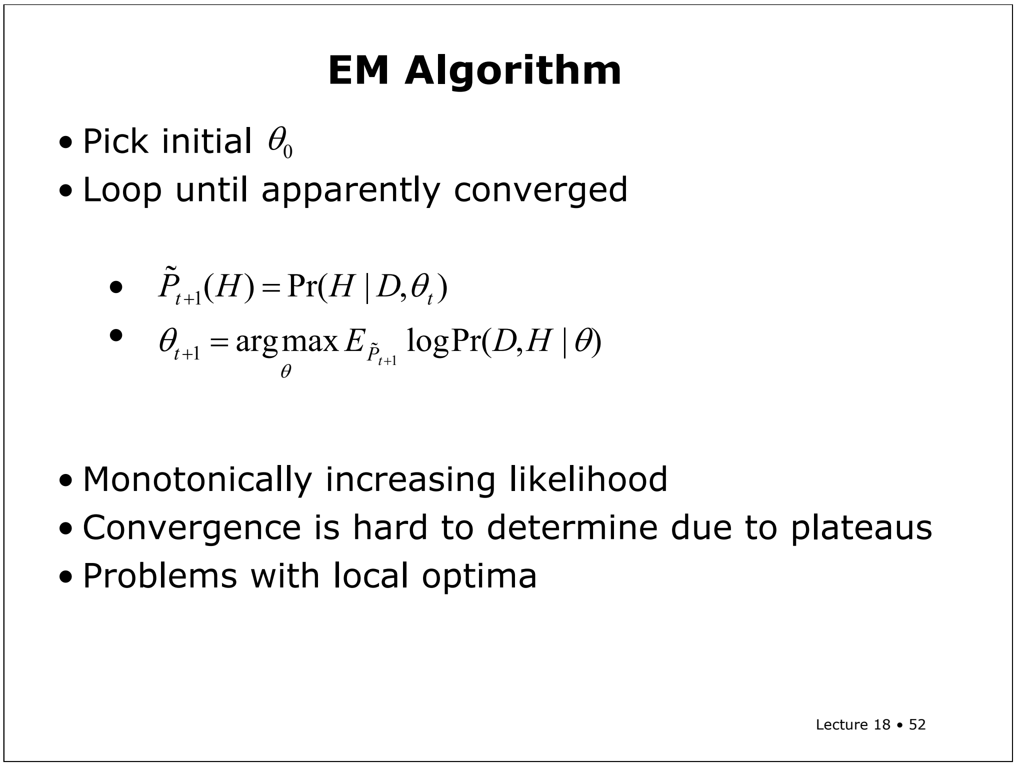
It’s possible to prove that this algorithm generates models with monotonically increasing likelihood. So, things always get better.



EM Algorithm

- Pick initial θ_0
- Loop until apparently converged
    - $\tilde{P}_{t+1}(H) = \Pr(H \mid D, \theta_t)$
    - $\theta_{t+1} = \arg\max_{\theta} E_{\tilde{P}_{t+1}} \log \Pr(D, H \mid \theta)$
- Monotonically increasing likelihood
- Convergence is hard to determine due to plateaus

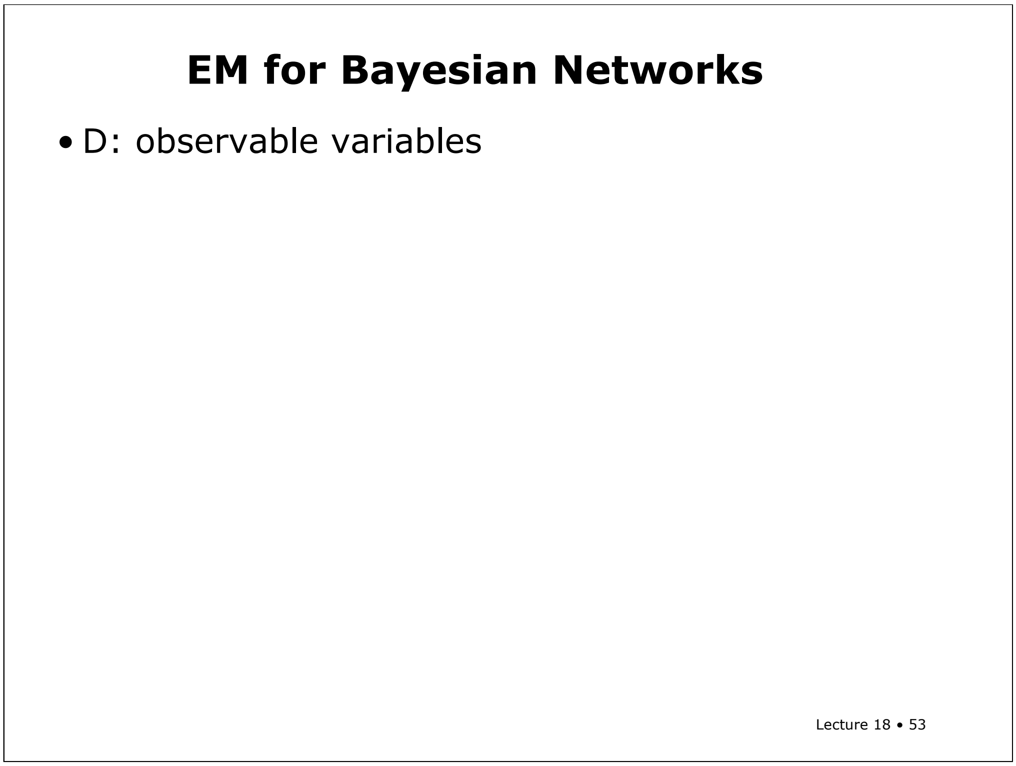
It can be hard to tell when EM has converged, though. Sometimes, the models just get a tiny bit better for a long time, and you think the process is done, and there's a sudden increase in likelihood. There's no real way to know whether that's going to happen or not.



EM Algorithm

- Pick initial θ_0
- Loop until apparently converged
  - $\tilde{P}_{t+1}(H) = \Pr(H \mid D, \theta_t)$
  - $\theta_{t+1} = \arg\max_{\theta} E_{\tilde{P}_{t+1}} \log \Pr(D, H \mid \theta)$
- Monotonically increasing likelihood
- Convergence is hard to determine due to plateaus
- Problems with local optima

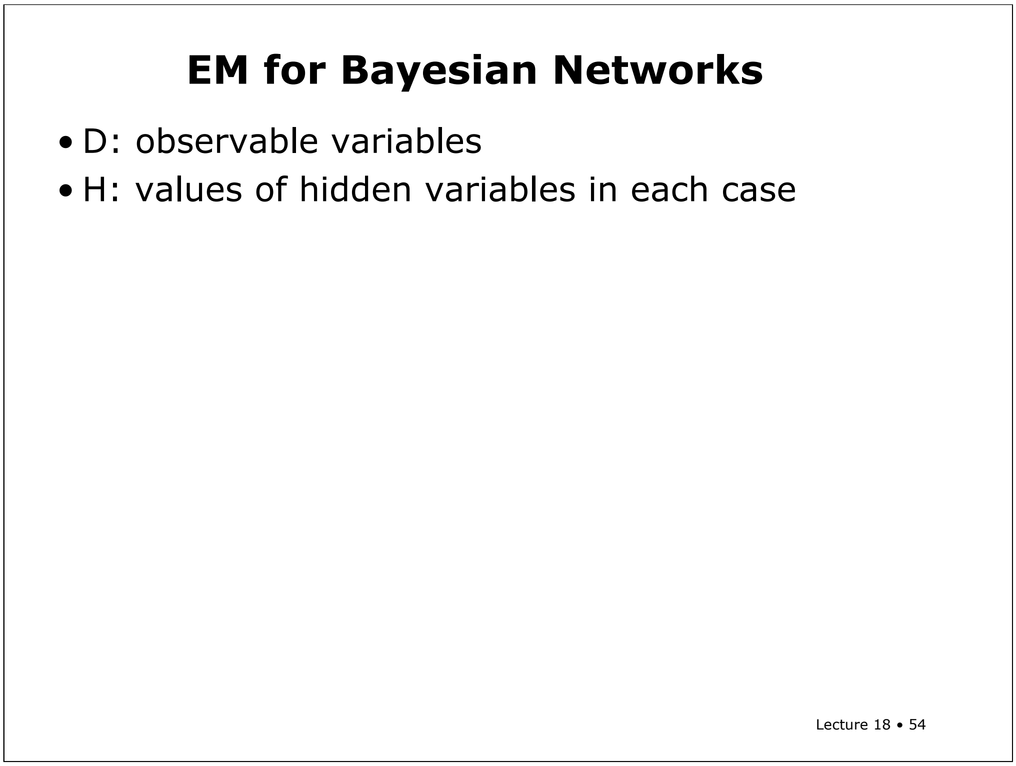
Another problem with EM is that it is subject to local minima. Sometimes it converges quite effectively to the maximum model that's near the one it started with, but there's a much better model somewhere else in the space. For this reason, it can be important either to start from multiple different initial models, or to initialize your model based on some insight into the domain.



EM for Bayesian Networks

- D: observable variables

Okay, so now let's look at how to apply EM to Bayesian networks. Our data will be a set of cases of observations of some observable variables, D.



EM for Bayesian Networks

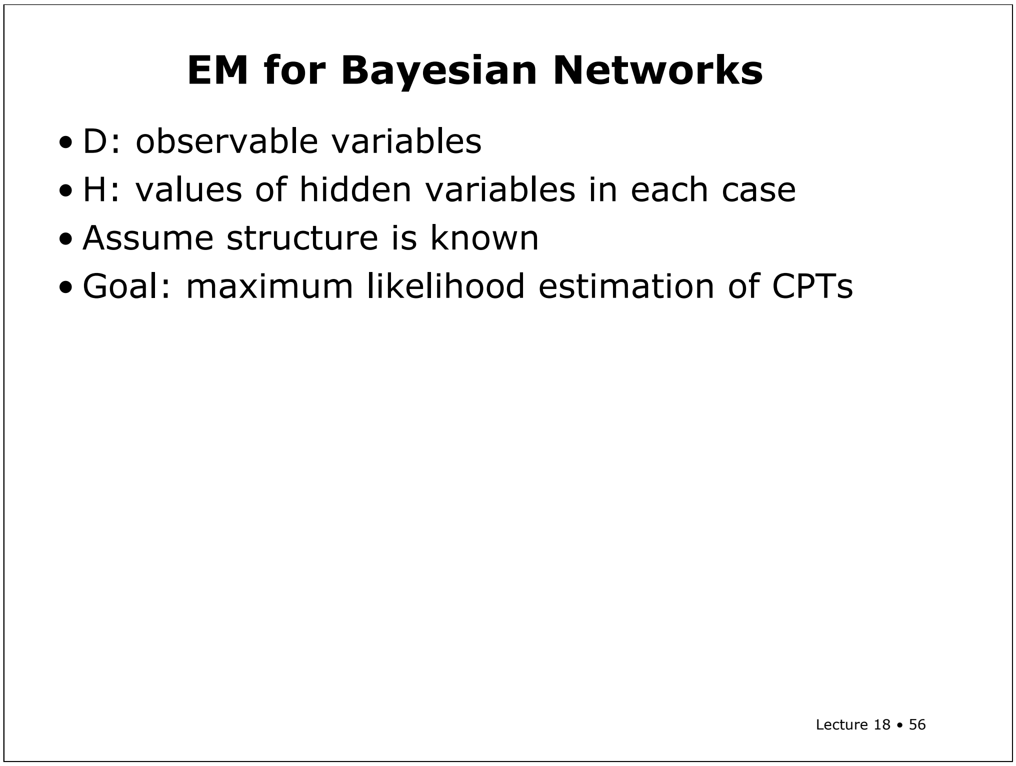
- D: observable variables
- H: values of hidden variables in each case

Our hidden variables will actually be the values of the hidden nodes in each case. (So, if we have 10 data cases and a network with one hidden node, we’ll really have 10 hidden variables, or missing pieces of data).

EM for Bayesian Networks

- D: observable variables
- H: values of hidden variables in each case
- Assume structure is known

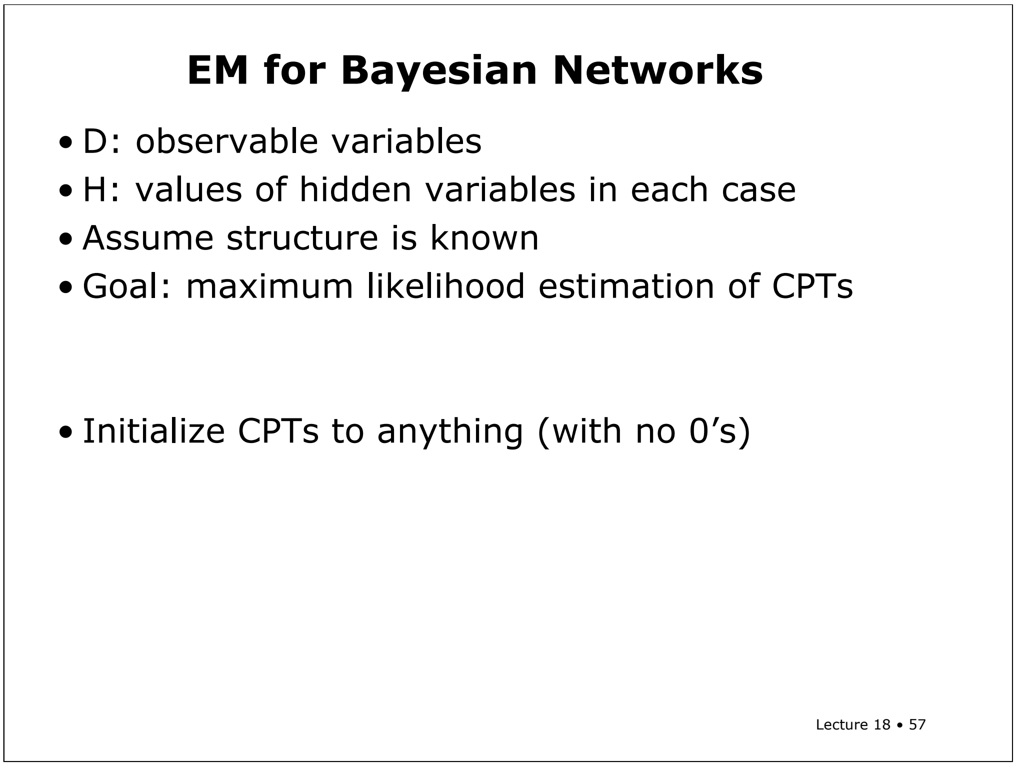
We’ll assume that the structure is known.



EM for Bayesian Networks

- D: observable variables
- H: values of hidden variables in each case
- Assume structure is known
- Goal: maximum likelihood estimation of CPTs

And we want to find the CPTs that maximize the probability of the observed data D.

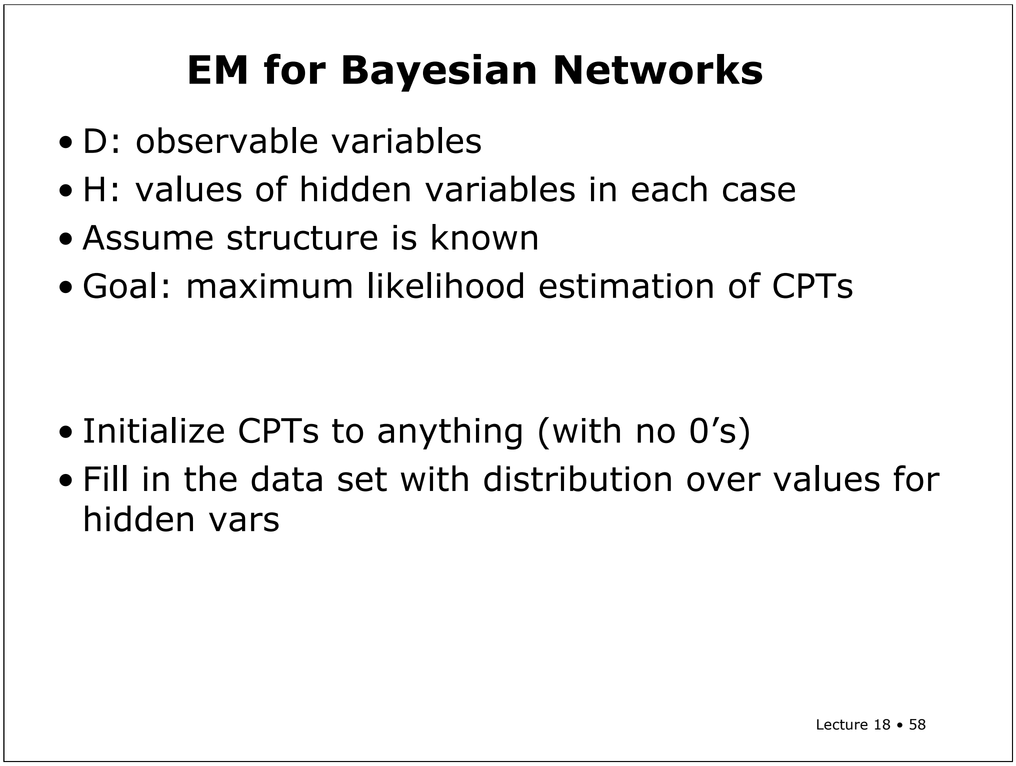


EM for Bayesian Networks

- D: observable variables
- H: values of hidden variables in each case
- Assume structure is known
- Goal: maximum likelihood estimation of CPTs

- Initialize CPTs to anything (with no 0’s)

So, we’ll initialize the CPTs to have any values we want (without any zeros, unless we’re absolutely certain that they are true in our domain).

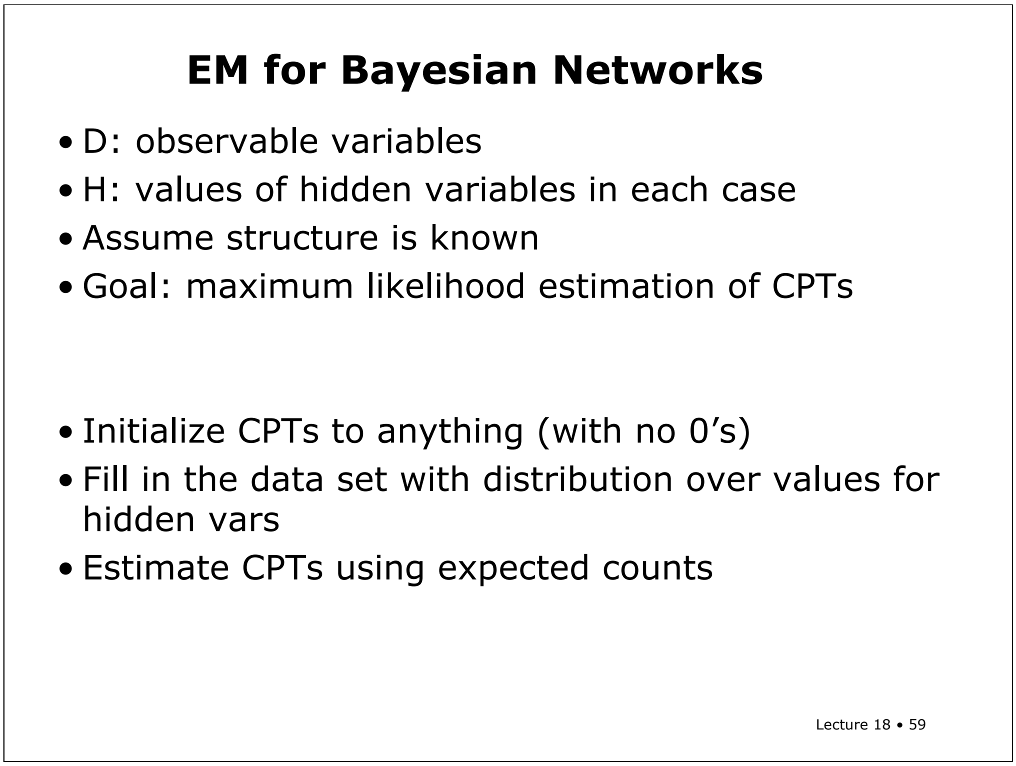


EM for Bayesian Networks

- D: observable variables
- H: values of hidden variables in each case
- Assume structure is known
- Goal: maximum likelihood estimation of CPTs

- Initialize CPTs to anything (with no 0’s)
- Fill in the data set with distribution over values for hidden vars

We can fill in the data set with distributions over values for the hidden variables.

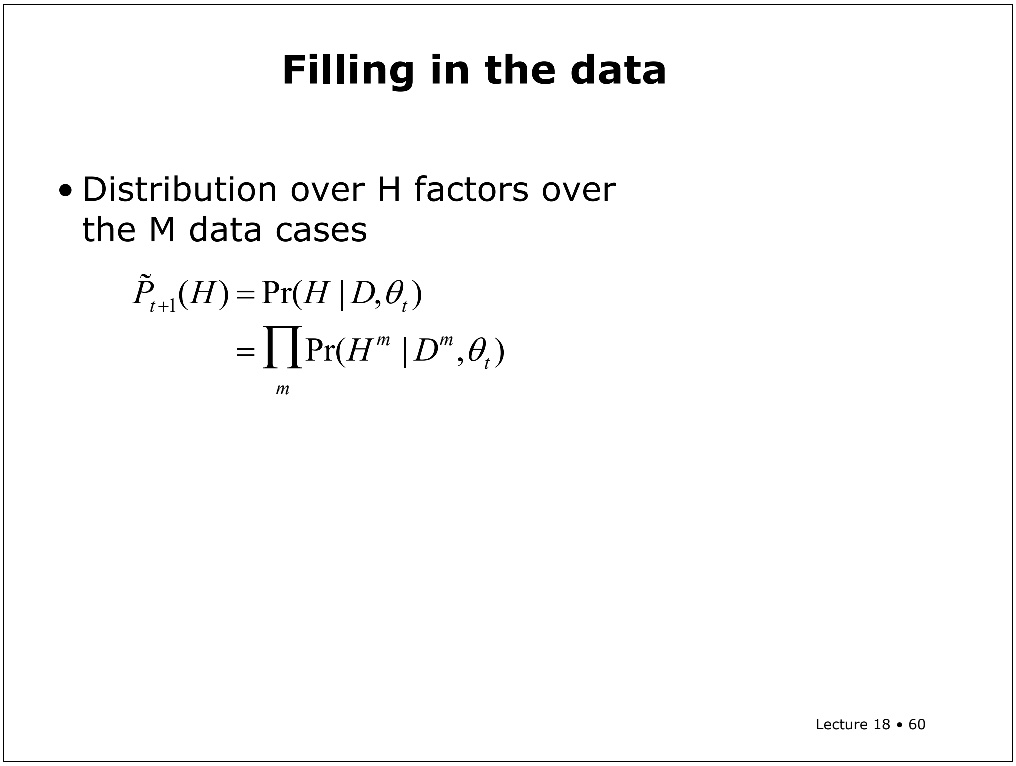


EM for Bayesian Networks

- D: observable variables
- H: values of hidden variables in each case
- Assume structure is known
- Goal: maximum likelihood estimation of CPTs

- Initialize CPTs to anything (with no 0’s)
- Fill in the data set with distribution over values for hidden vars
- Estimate CPTs using expected counts

And then estimate the CPTs using expected counts.

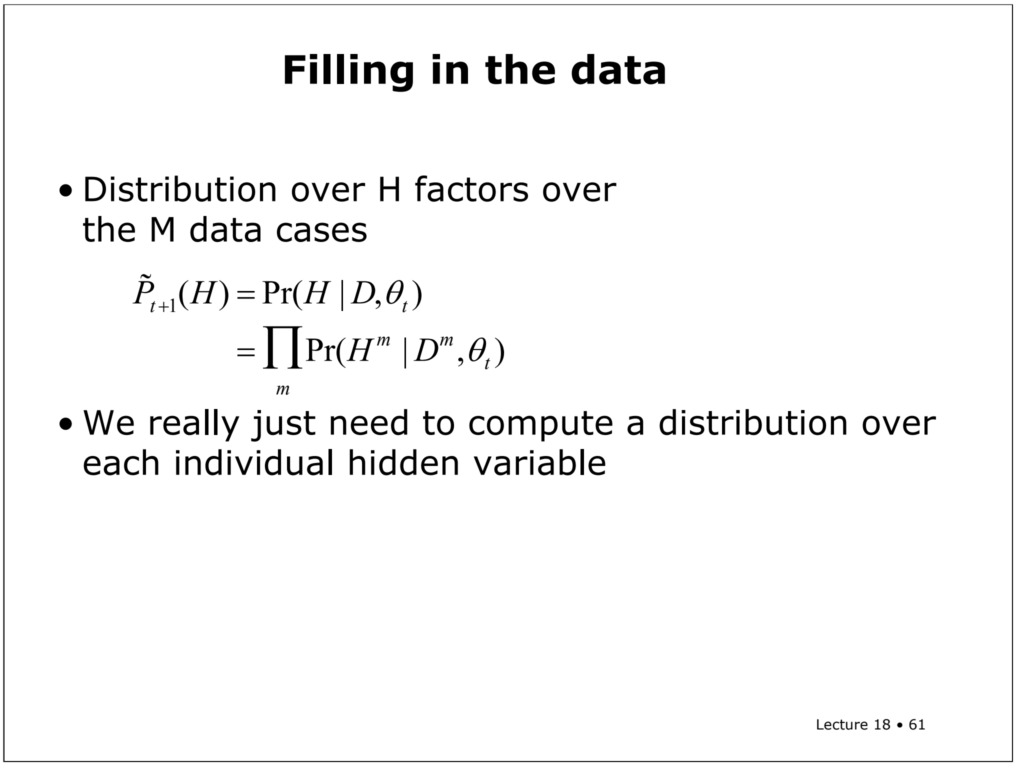


Filling in the data

- Distribution over H factors over the M data cases

$$\begin{aligned}\tilde{P}_{t+1}(H) &= \Pr(H \mid D, \theta_t) \\ &= \prod_m \Pr(H^m \mid D^m, \theta_t)\end{aligned}$$

When it’s time to compute the probability distribution over H given D and theta, it seems hard, because we’ll have m different hidden variables: one for the value of node H in each of the m data cases.



Filling in the data

- Distribution over H factors over the M data cases

$$\tilde{P}_{t+1}(H) = \Pr(H \mid D, \theta_t)$$
$$= \prod_m \Pr(H^m \mid D^m, \theta_t)$$

- We really just need to compute a distribution over each individual hidden variable

Luckily, this distribution factors out. Each hidden variable depends only on the observed variables in its case, given the model. So, we really only have to worry about coming up with the individual distributions over each hidden variable in each case.

Filling in the data

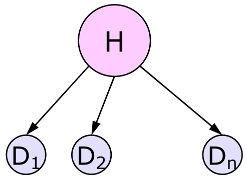
- Distribution over H factors over the M data cases

$$\tilde{P}_{t+1}(H) = \Pr(H \mid D, \theta_t)$$
$$= \prod_m \Pr(H^m \mid D^m, \theta_t)$$

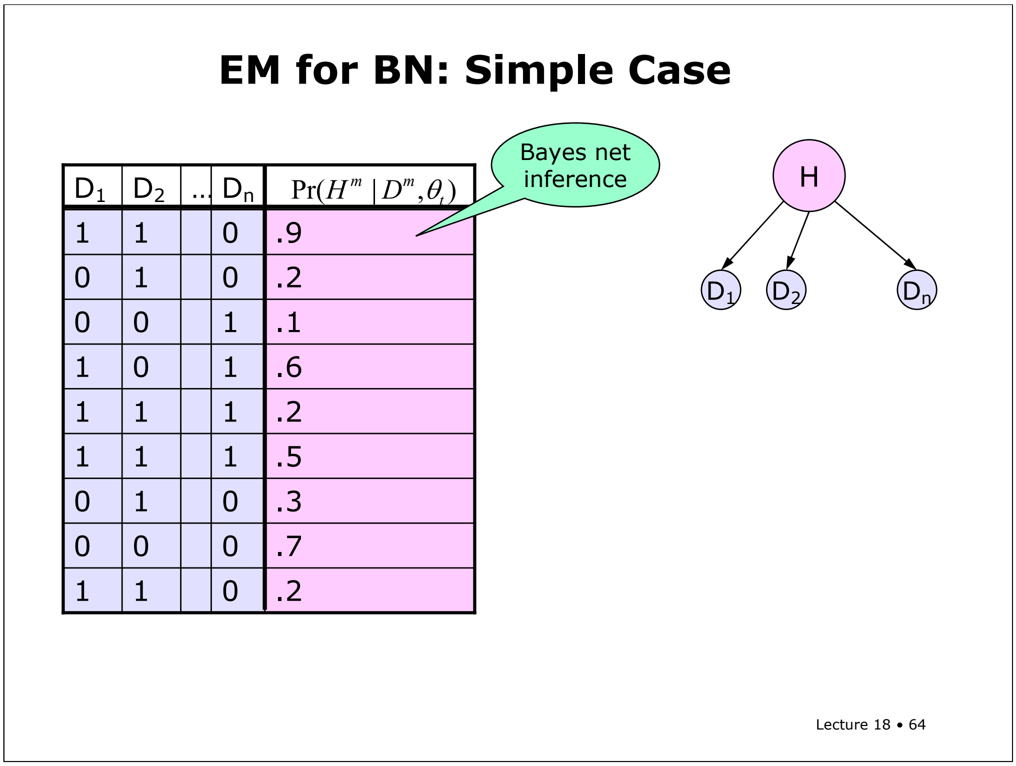
- We really just need to compute a distribution over each individual hidden variable
- Each factor is a call to Bayes net inference

Now, how can we compute Pr(Hm | dm, theta)? That's just a call to a bayes net inference procedure. We're given all the parameters of the network, and an assignment to some of the variables, D. We need to find a probability distribution over the other variables, H. We can use variable elimination, or any other technique available to us.

EM for BN: Simple Case



Let's just consider a simple case with a single hidden node (things get a bit more complicated when we have more than one; but not qualitatively different). We'll use the same network structure we talked about at the beginning of this lecture: one hidden cause directly controlling a whole set of possible effects. And for further simplicity, we'll assume all the nodes are binary.



EM for BN: Simple Case

D$_1$	D$_2$	…	D$_n$	$\Pr(H^m \mid D^m, \theta_t)$
1	1		0	.9
0	1		0	.2
0	0		1	.1
1	0		1	.6
1	1		1	.2
1	1		1	.5
0	1		0	.3
0	0		0	.7
1	1		0	.2

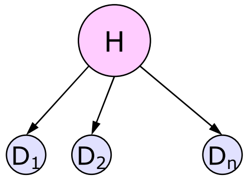
Bayes net inference

So, given a model, theta, we can use bayes net inference to compute, for each case in our data set, the probability that H would be true, given the values of the observed variables.

EM for BN: Simple Case

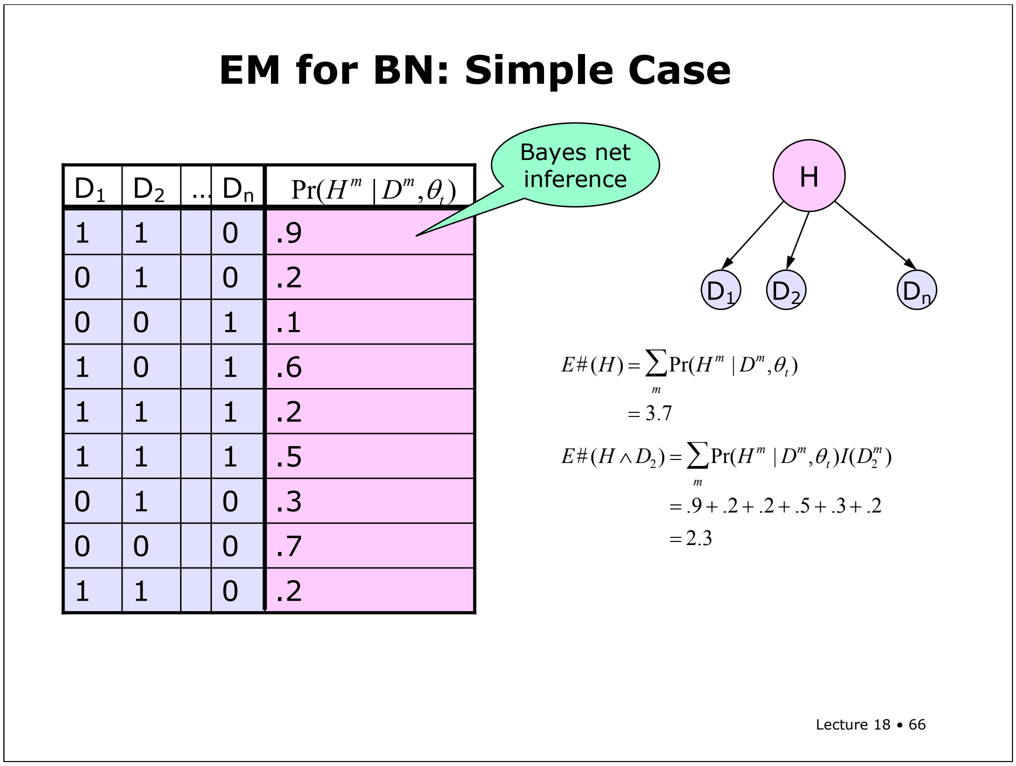
Bayes net inference

D_1	D_2	...	D_n	$\Pr(H^m \mid D^m, \theta_t)$
1	1		0	.9
0	1		0	.2
0	0		1	.1
1	0		1	.6
1	1		1	.2
1	1		1	.5
0	1		0	.3
0	0		0	.7
1	1		0	.2



$$
\begin{aligned}
E\#(H) &= \sum_m \Pr(H^m \mid D^m, \theta_t) \\
&= 3.7
\end{aligned}
$$

Then, we can use these distributions to compute expected counts. So, for instance, to get the expected number of times H is true, we'd just add up with probabilities of H being true in each case.



EM for BN: Simple Case

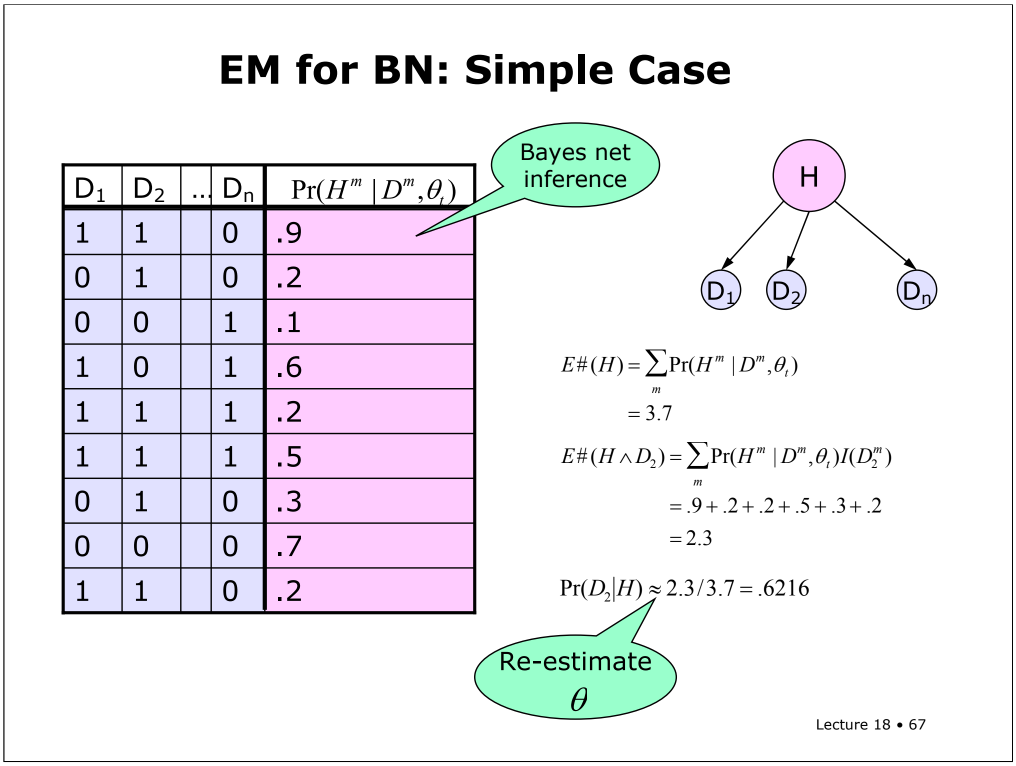
D_1	D_2	…	D_n	$\Pr(H^m \mid D^m, \theta_t)$
1	1		0	.9
0	1		0	.2
0	0		1	.1
1	0		1	.6
1	1		1	.2
1	1		1	.5
0	1		0	.3
0	0		0	.7
1	1		0	.2

Bayes net inference

$$
\begin{aligned}
E\#(H) &= \sum_m \Pr(H^m \mid D^m, \theta_t) \\
&= 3.7
\end{aligned}
$$

$$
\begin{aligned}
E\#(H \wedge D_2) &= \sum_m \Pr(H^m \mid D^m, \theta_t) I(D_2^m) \\
&= .9 + .2 + .2 + .5 + .3 + .2 \\
&= 2.3
\end{aligned}
$$

To get the expected number of times that H and D2 are true, we find all the cases in which D2 is true, and add up their probabilities of H being true.



EM for BN: Simple Case

D_1	D_2	…	D_n	$\Pr(H^m \mid D^m, \theta_t)$
1	1		0	.9
0	1		0	.2
0	0		1	.1
1	0		1	.6
1	1		1	.2
1	1		1	.5
0	1		0	.3
0	0		0	.7
1	1		0	.2

Bayes net inference

$$E\#(H) = \sum_m \Pr(H^m \mid D^m, \theta_t)$$
$$= 3.7$$

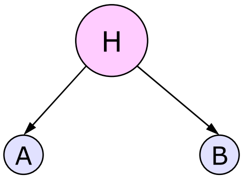
$$E\#(H \wedge D_2) = \sum_m \Pr(H^m \mid D^m, \theta_t) I(D_2^m)$$
$$= .9 + .2 + .2 + .5 + .3 + .2$$
$$= 2.3$$

$$\Pr(D_2 | H) \approx 2.3 / 3.7 = .6216$$

Re-estimate θ

Those two expected counts will let us re-estimate theta. The component of theta that represents the probability of D2 given H will be estimated by dividing the two counts we just computed.

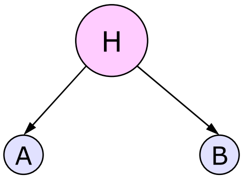
EM for BN: Worked Example



Now, to make everything concrete, we'll go all the way through a very simple example. Let's assume there's a hidden cause, H, and two observable variables, A and B.

EM for BN: Worked Example

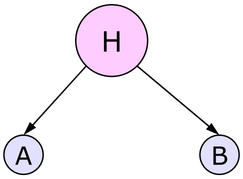
A	B	#	$\Pr(H^m \mid D^m, \theta_t)$
0	0	6	
0	1	1	
1	0	1	
1	1	4	



I’ve summarized our data set in this table, indicating that we saw the combination 0,0 6 times, the combination 0,1 once, etc. If we have a domain with more data cases than possible assignments to the observable variables, it’s usually more efficient to store the data this way. But quite typically we never see the same data case more than once, and most of them we never see at all!

EM for BN: Worked Example

A	B	#	$\Pr(H^m \mid D^m, \theta_t)$
0	0	6	
0	1	1	
1	0	1	
1	1	4	

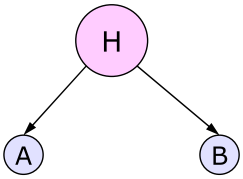


$$\theta_1 = \Pr(H)$$
$$\theta_2 = \Pr(A \mid H)$$
$$\theta_3 = \Pr(A \mid \neg H)$$
$$\theta_4 = \Pr(B \mid H)$$
$$\theta_5 = \Pr(B \mid \neg H)$$

We’ll let the thetas be these probabilities, which make up all the CPTs for our simple network.

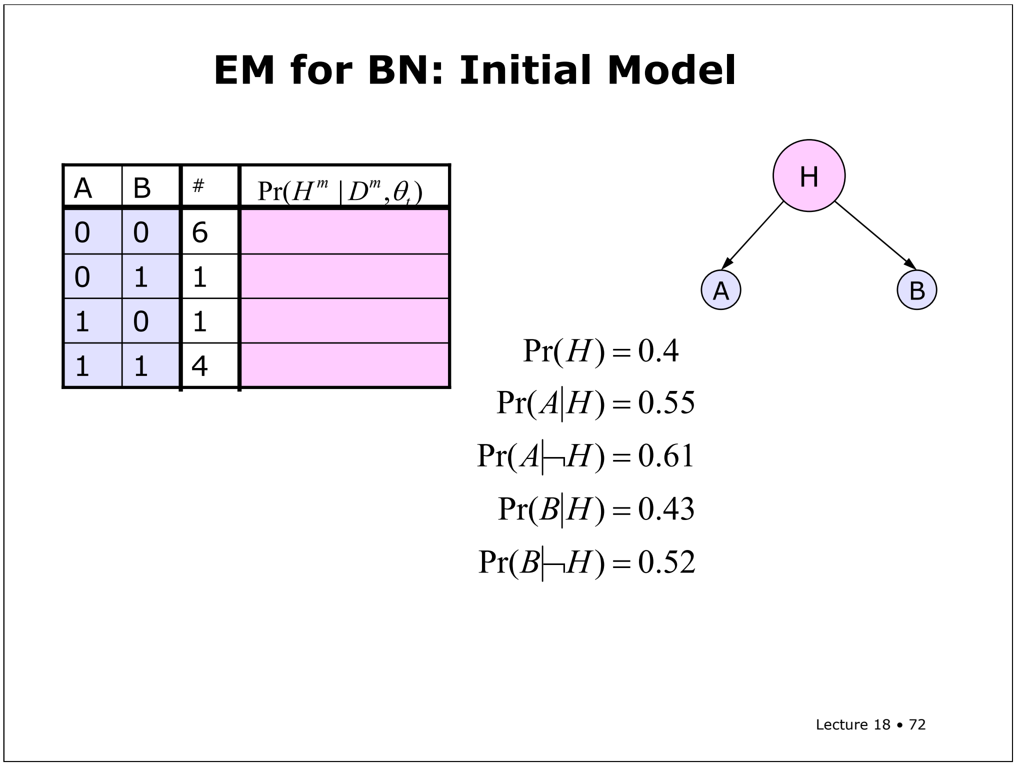
EM for BN: Worked Example

A	B	#	$\Pr(H^m \mid D^m, \theta_t)$
0	0	6	
0	1	1	
1	0	1	
1	1	4	



$$\theta_1 = \Pr(H)$$
$$\theta_2 = \Pr(A \mid H)$$
$$\theta_3 = \Pr(A \mid \neg H)$$
$$\theta_4 = \Pr(B \mid H)$$
$$\theta_5 = \Pr(B \mid \neg H)$$

Note that we have a lot of cases of 00 and of 11, but not many with 01 or 10. We can guess that the hidden node is going to play the role of choosing whether we output a 00 or a 11. And that there are roughly two reasonable solutions: A and B are both on when H is off, or A and B are both on when H is on. Let's see what learning does for us.



EM for BN: Initial Model

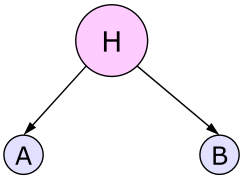
A	B	#	$\Pr(H^m \mid D^m, \theta_t)$
0	0	6	
0	1	1	
1	0	1	
1	1	4	

$$\Pr(H) = 0.4$$
$$\Pr(A|H) = 0.55$$
$$\Pr(A|\neg H) = 0.61$$
$$\Pr(B|H) = 0.43$$
$$\Pr(B|\neg H) = 0.52$$

I picked an initial model to be this set of probabilities, which are sort of near, but not equal to 0.5. We’ll see why I did this, later on.

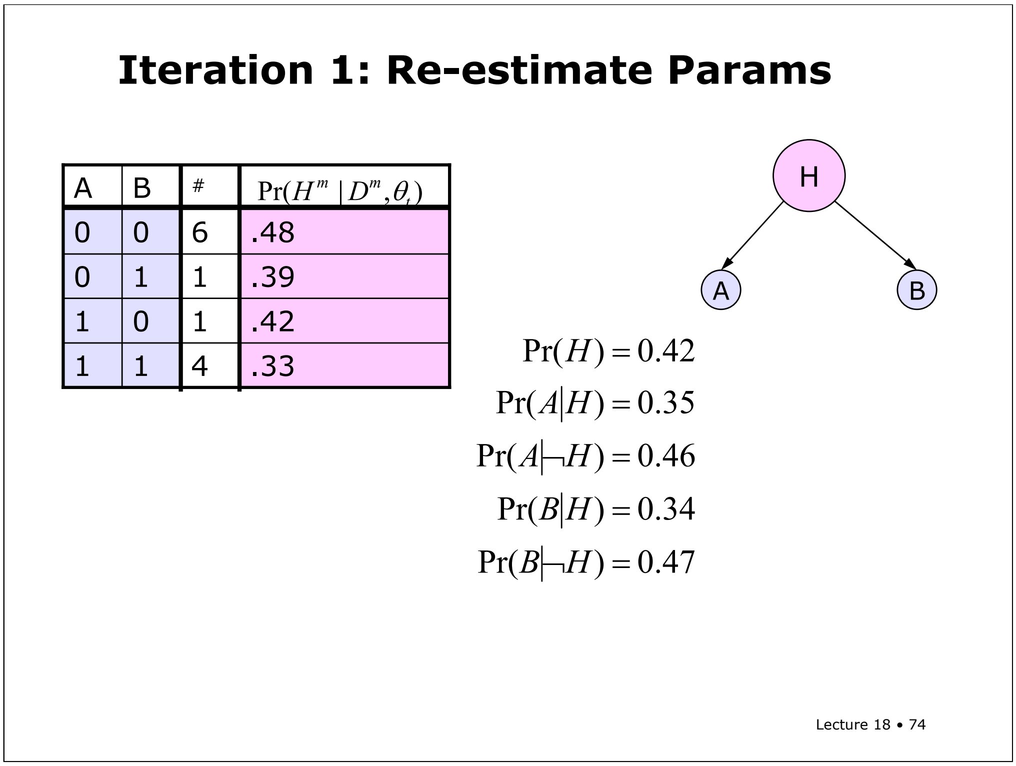
Iteration 1: Fill in data

A	B	#	$\Pr(H^m \mid D^m, \theta_t)$
0	0	6	.48
0	1	1	.39
1	0	1	.42
1	1	4	.33



$$\Pr(H) = 0.4$$
$$\Pr(A|H) = 0.55$$
$$\Pr(A|\neg H) = 0.61$$
$$\Pr(B|H) = 0.43$$
$$\Pr(B|\neg H) = 0.52$$

Given that initial model, we can compute the probability of H given A and B, for every combination of A and B, and put those probabilities into our table.

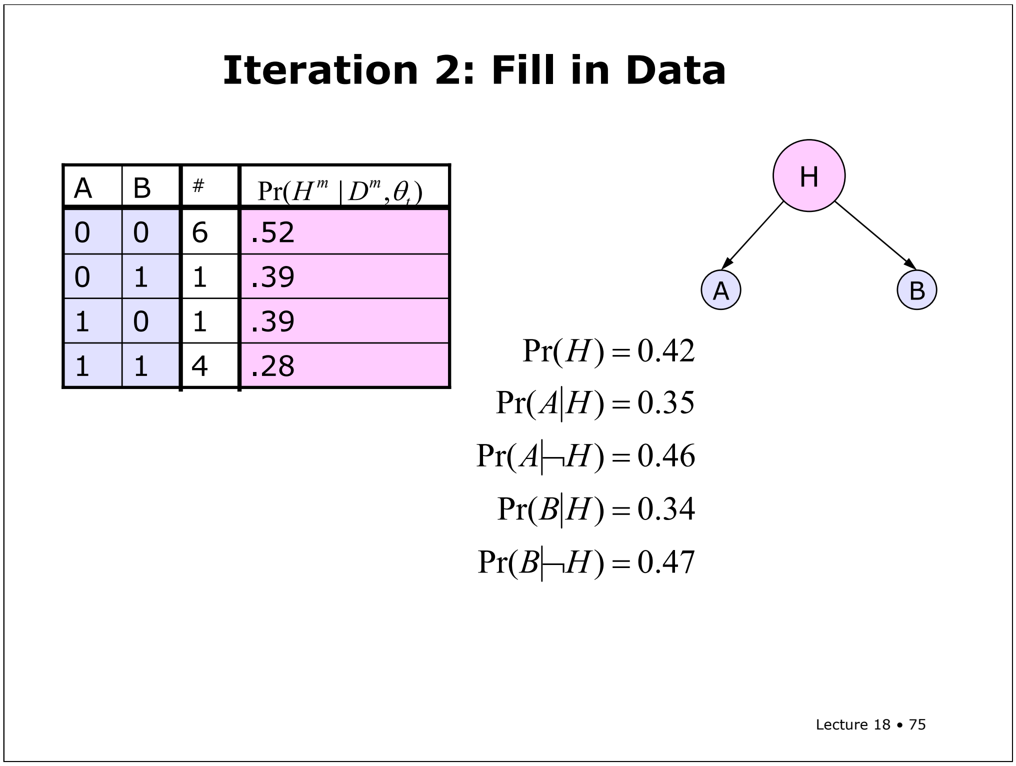


Iteration 1: Re-estimate Params

A	B	#	$\Pr(H^m \mid D^m, \theta_t)$
0	0	6	.48
0	1	1	.39
1	0	1	.42
1	1	4	.33

$$\Pr(H) = 0.42$$
$$\Pr(A|H) = 0.35$$
$$\Pr(A|\neg H) = 0.46$$
$$\Pr(B|H) = 0.34$$
$$\Pr(B|\neg H) = 0.47$$

Now we can re-estimate the parameters of the model using the expected values of H. Here’s what we get (I used a computer program to do this, so it’s probably right; but I wrote the program, so maybe not…)



Iteration 2: Fill in Data

A	B	#	$\Pr(H^m \mid D^m, \theta_t)$
0	0	6	.52
0	1	1	.39
1	0	1	.39
1	1	4	.28

$$\Pr(H) = 0.42$$

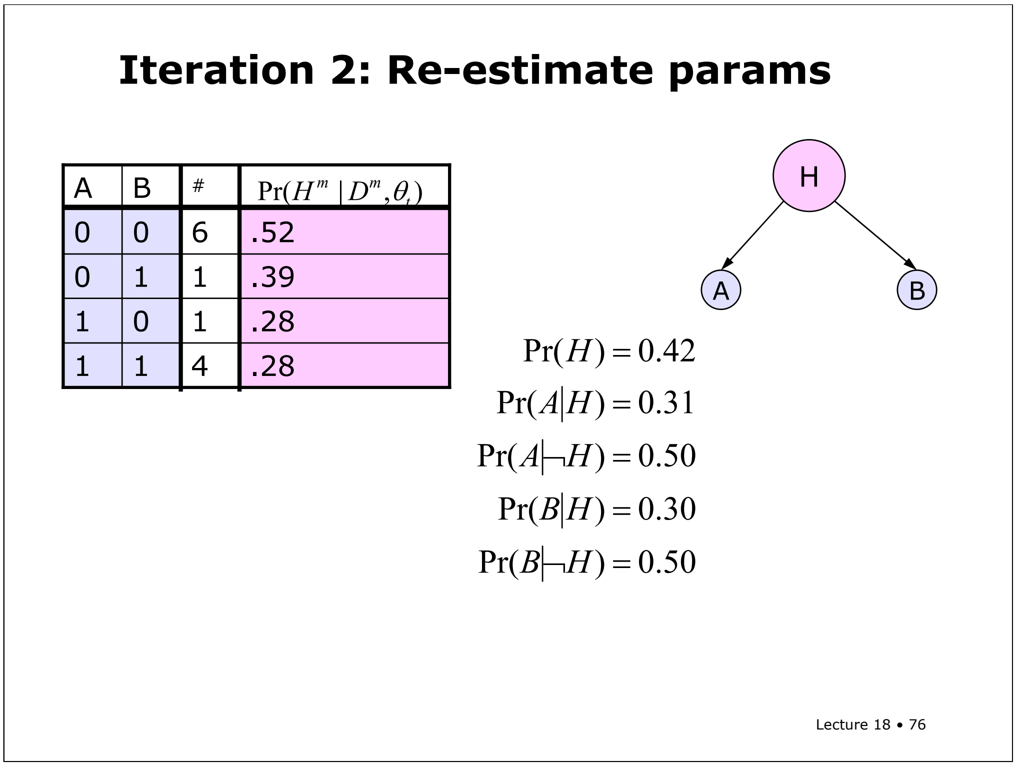
$$\Pr(A|H) = 0.35$$

$$\Pr(A|\neg H) = 0.46$$

$$\Pr(B|H) = 0.34$$

$$\Pr(B|\neg H) = 0.47$$

Now we can fill in new values of the data. We can start to see a tendency for H to want to be on when A and B are off, and vice versa.

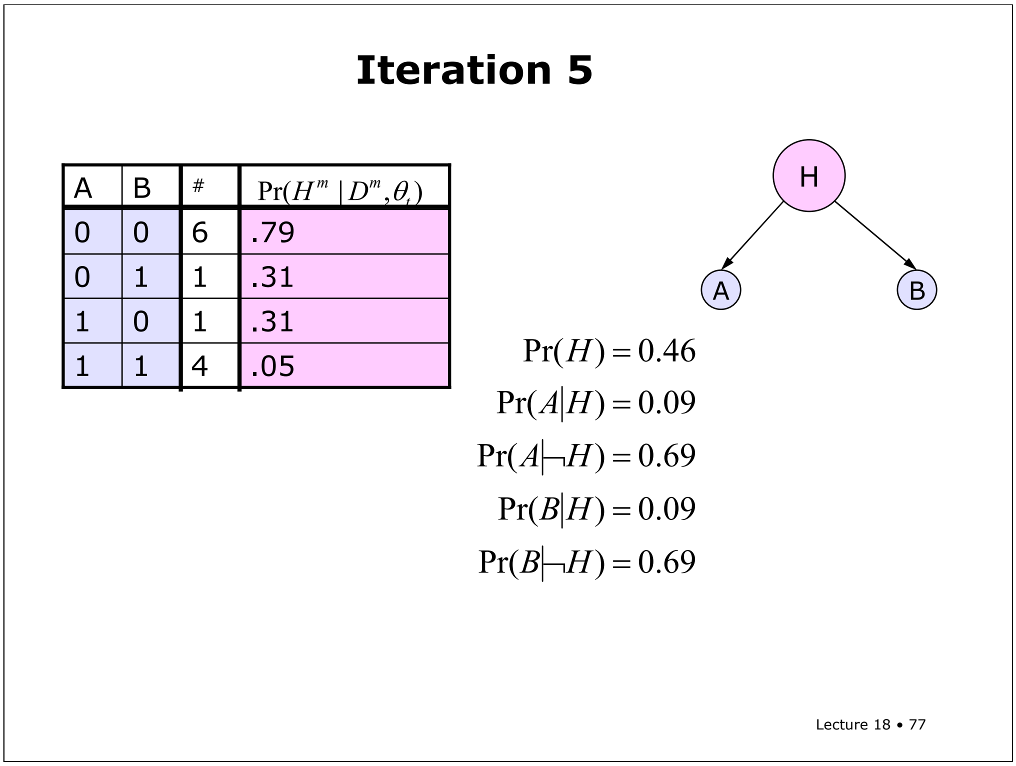


Iteration 2: Re-estimate params

A	B	#	$\Pr(H^m \mid D^m, \theta_t)$
0	0	6	.52
0	1	1	.39
1	0	1	.28
1	1	4	.28

$$\Pr(H) = 0.42$$
$$\Pr(A|H) = 0.31$$
$$\Pr(A|\neg H) = 0.50$$
$$\Pr(B|H) = 0.30$$
$$\Pr(B|\neg H) = 0.50$$

Now we recomputed the probabilities in the model. They're moving away from their initial values.

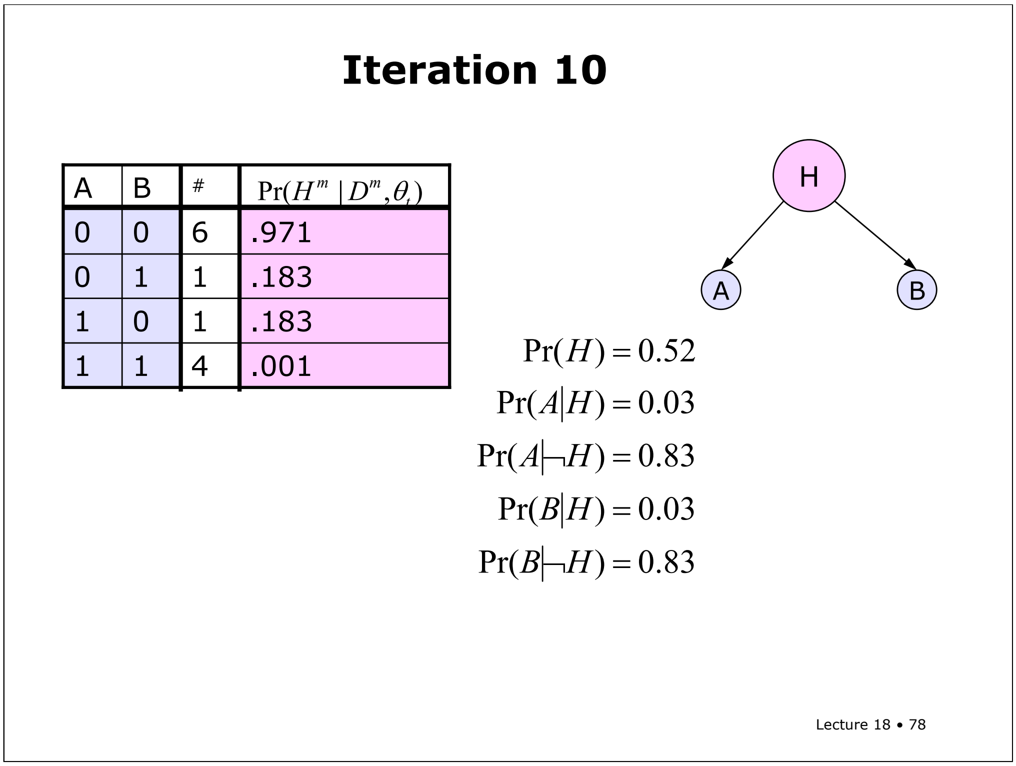


Iteration 5

A	B	#	$\Pr(H^m \mid D^m, \theta_t)$
0	0	6	.79
0	1	1	.31
1	0	1	.31
1	1	4	.05

$$\Pr(H) = 0.46$$
$$\Pr(A|H) = 0.09$$
$$\Pr(A|\neg H) = 0.69$$
$$\Pr(B|H) = 0.09$$
$$\Pr(B|\neg H) = 0.69$$

Now we skip ahead to iteration 5. Here are the missing-data distributions and the model. The tendency for H to be on when A and B are off, and for it to be off when they are on is considerably strengthened, as we can see in both distributions.

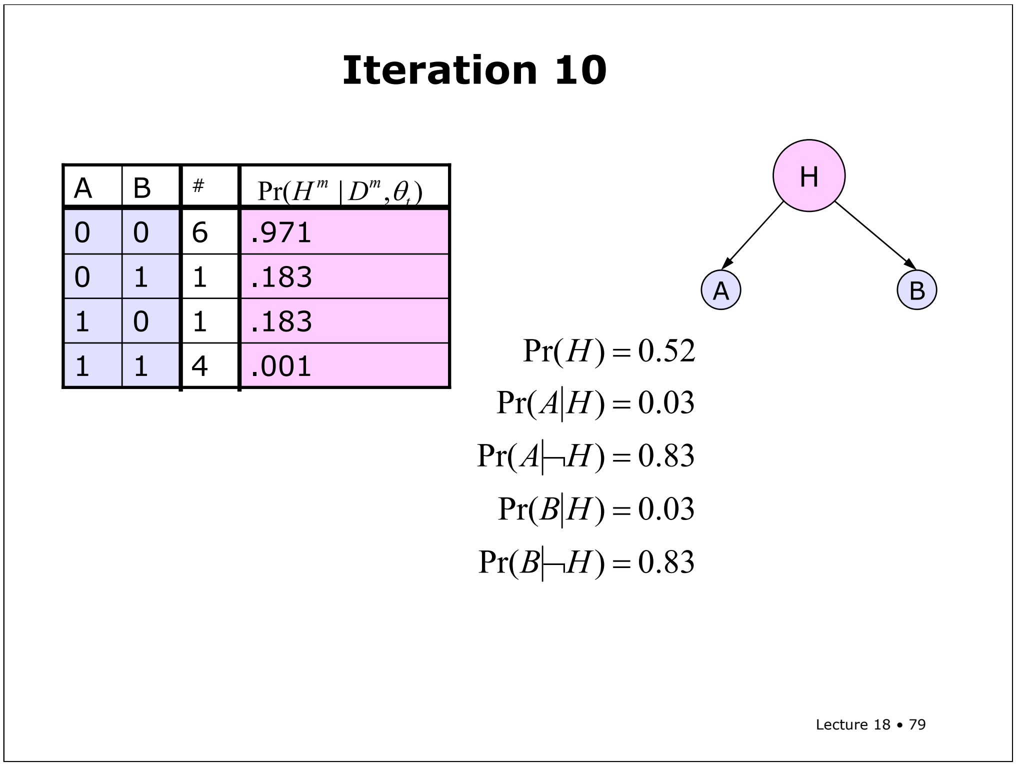


Iteration 10

A	B	#	$\Pr(H^m \mid D^m, \theta_t)$
0	0	6	.971
0	1	1	.183
1	0	1	.183
1	1	4	.001

$$\Pr(H) = 0.52$$
$$\Pr(A|H) = 0.03$$
$$\Pr(A|\neg H) = 0.83$$
$$\Pr(B|H) = 0.03$$
$$\Pr(B|\neg H) = 0.83$$

After 10 iterations, the process is pretty well converged. The prior probability of H is just over 50 percent (which makes sense, since about half of the data cases are 00, when it is almost certainly on, and it has some chance of being on in a couple of the other cases).

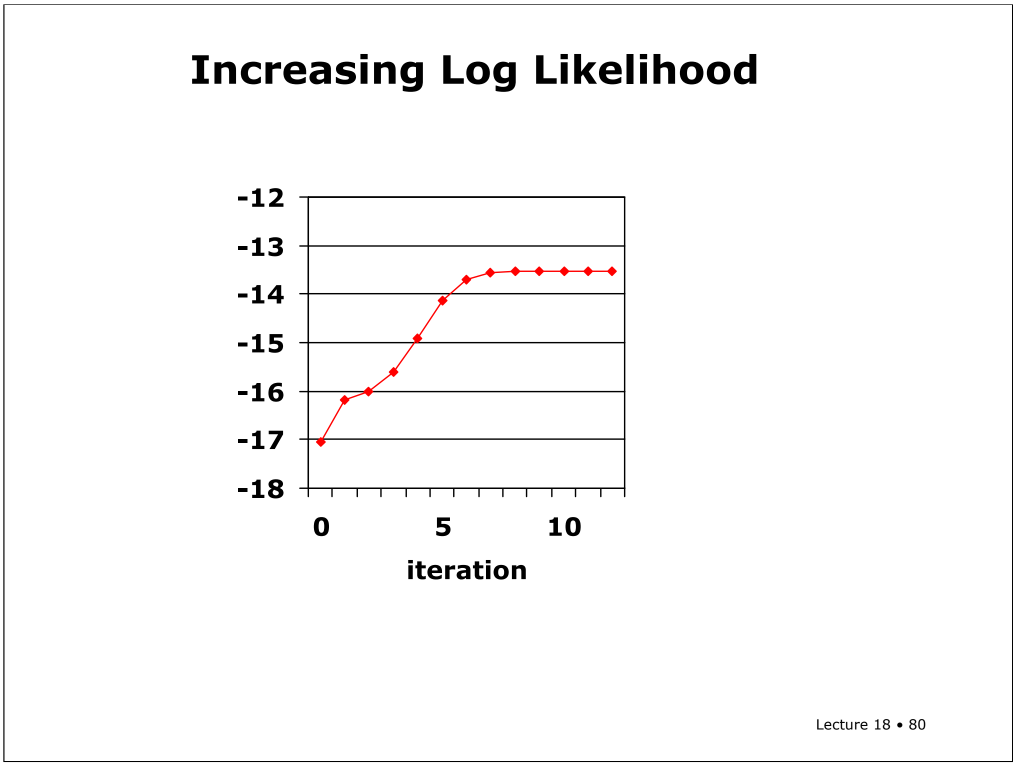


Iteration 10

A	B	#	$\Pr(H^m \mid D^m, \theta_t)$
0	0	6	.971
0	1	1	.183
1	0	1	.183
1	1	4	.001

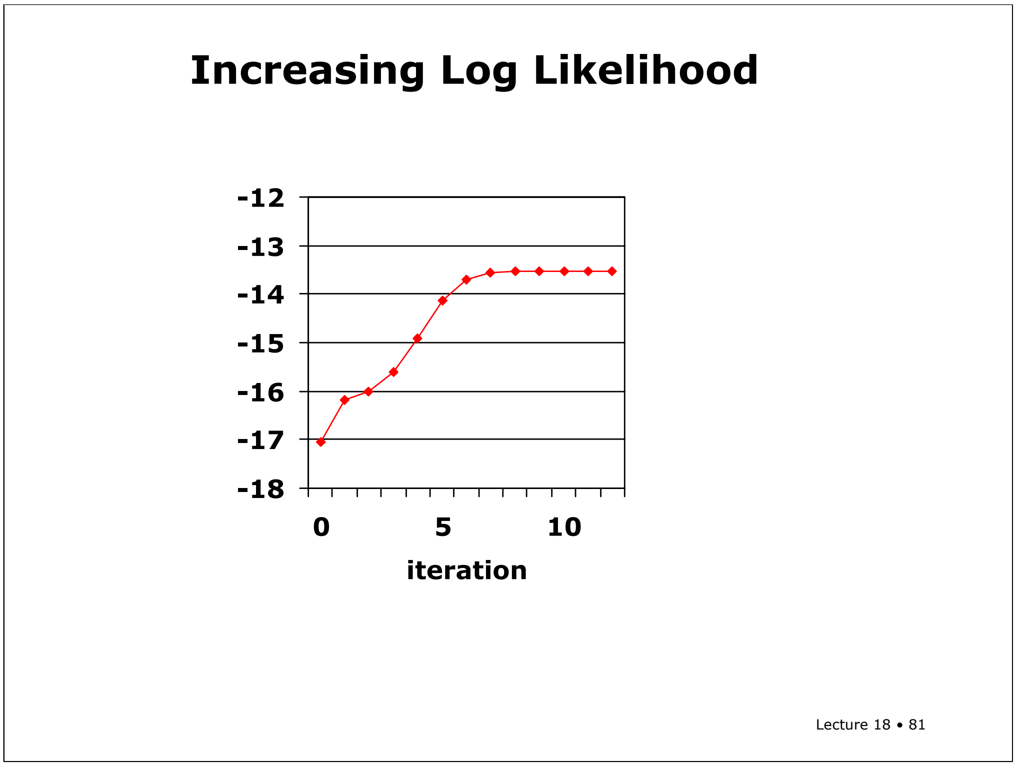
$$\Pr(H) = 0.52$$
$$\Pr(A|H) = 0.03$$
$$\Pr(A|\neg H) = 0.83$$
$$\Pr(B|H) = 0.03$$
$$\Pr(B|\neg H) = 0.83$$

The CPTs for A and B are the same, which also makes sense, since the data is completely symmetric for A and B. When H is on, A and B are almost certainly off. When H is off, A and B have a moderately high probability of being on.



Increasing Log Likelihood

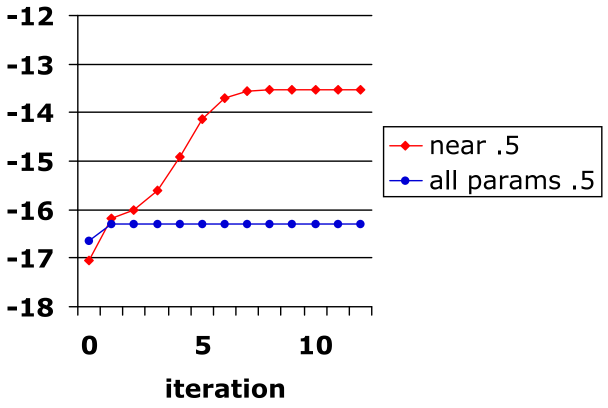
If we plot the log likelihood of the observed data given the model as a function of the iteration, we can see that it increases monotonically. It flattens out somewhere around iteration 8, and I don’t think it’s going to improve much after that.



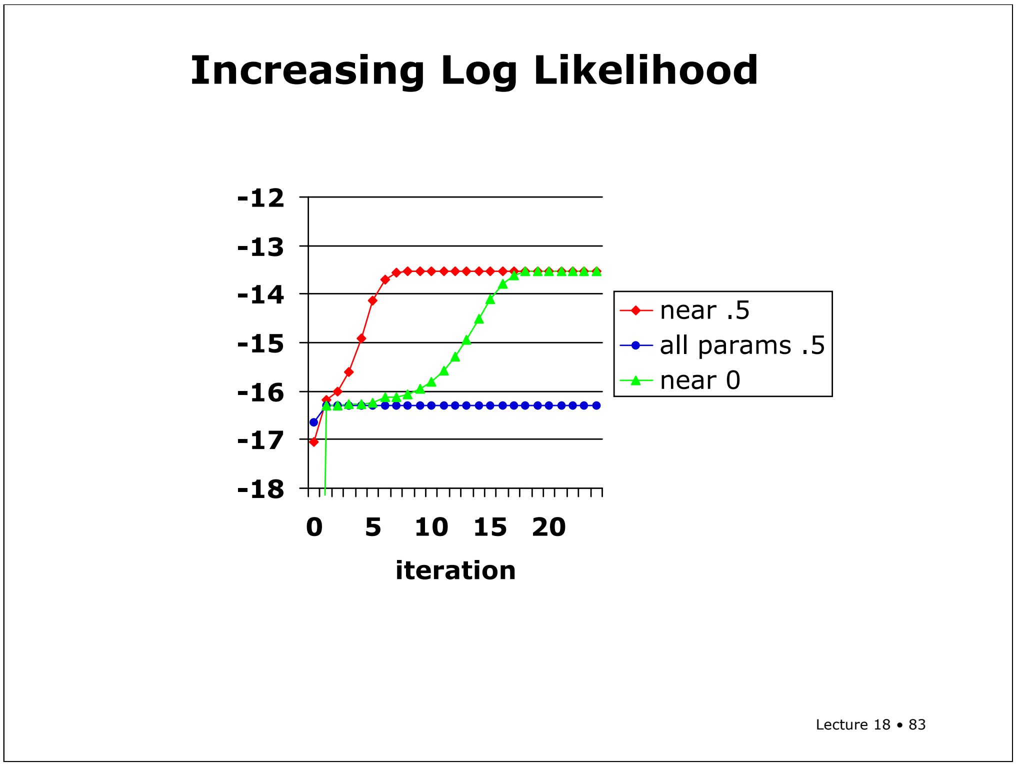
Increasing Log Likelihood

You can see that, although it’s always improving, the amount of improvement per iteration is variable.

Increasing Log Likelihood

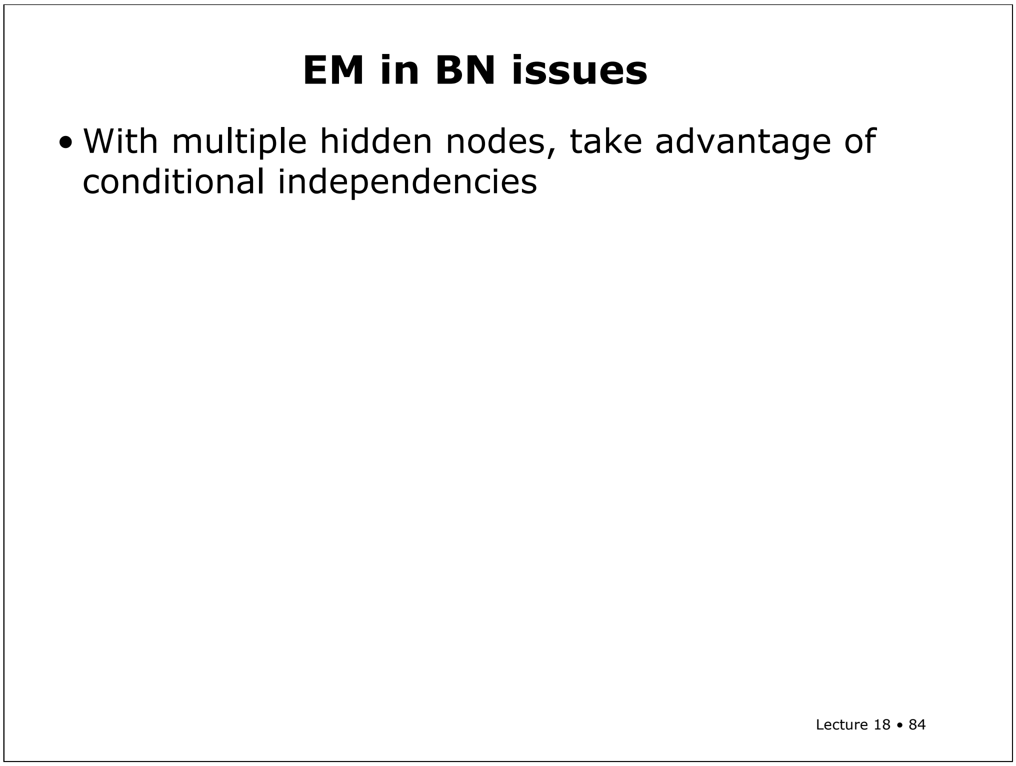


To illustrate the problems with local optima, even with such a simple model as this, I tried to solve the same problem, with the same data set, but initializing all of the parameters in the model to 0.5. Because of the symmetry in the parameters and the symmetry in the data set, parameters theta 2 through theta 5 remain at 0.5 forever. It takes just a little bit of asymmetry to tip the iterative process toward one or the other reasonable solution. This is an unstable equilibrium, which is unlikely to arise in practice. But just to be safe, it’s often wise to initialize your parameters to be nearly, but not quite uniform.



Increasing Log Likelihood

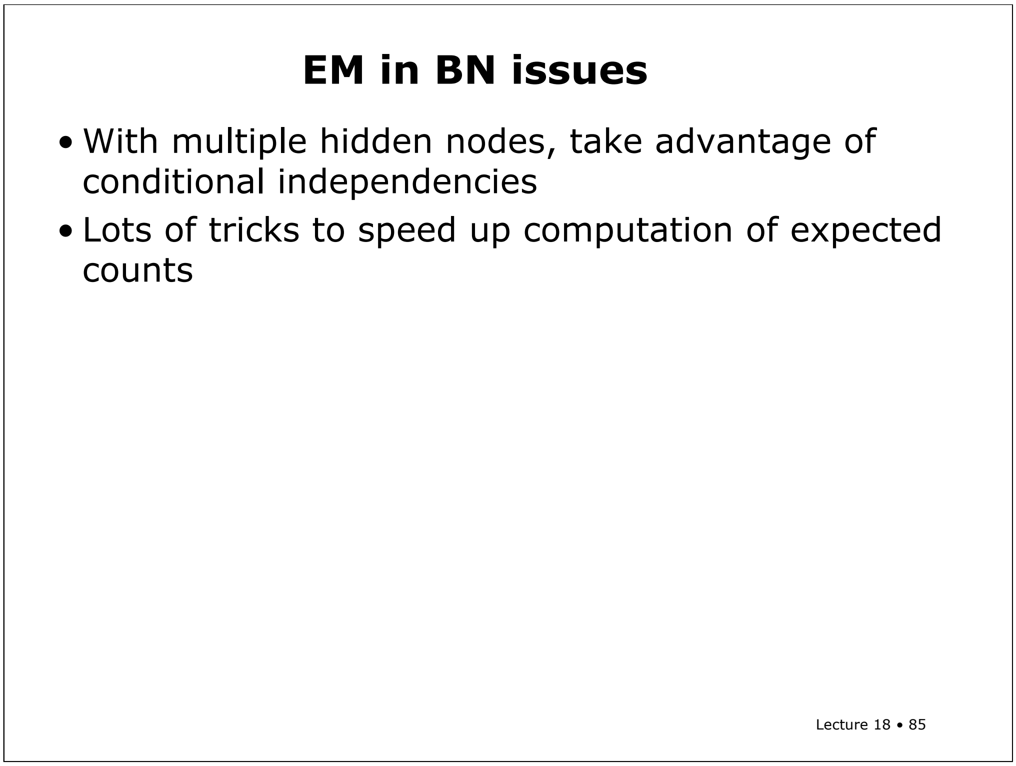
Finally, just for fun, I tried initializing all the parameters near (but not equal to 0). The log likelihood of that model is terrible (something like –35), but then it jumps up to around –16, which is where the completely symmetric model was. Eventually, it manages to break the symmetry, and come up to the same asymptote as the first run.



EM in BN issues

- With multiple hidden nodes, take advantage of conditional independencies

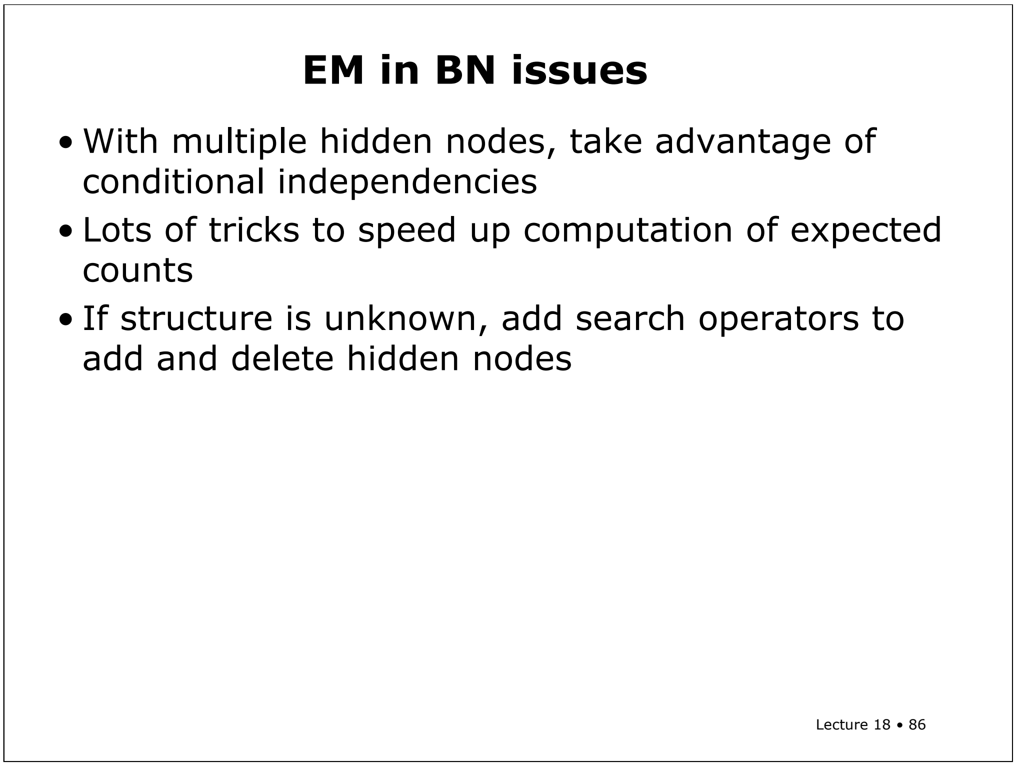
When you have multiple hidden nodes, it's important to take advantage of conditional independencies among the hidden nodes given the observables, to avoid having to compute joint distributions over many hidden variables.



EM in BN issues

- With multiple hidden nodes, take advantage of conditional independencies
- Lots of tricks to speed up computation of expected counts

The way we described this algorithm, including filling in all of the partial counts, is very inefficient. There are lots of methods, and a fair amount of current research, devoted to making that process much more efficient.



EM in BN issues

- With multiple hidden nodes, take advantage of conditional independencies
- Lots of tricks to speed up computation of expected counts
- If structure is unknown, add search operators to add and delete hidden nodes

What if the structure of the network is unknown? Then we can do structure search, but add to our repertoire of search steps the option of adding or deleting hidden nodes. Then, given a structure, we can use EM to estimate the parameters, and use them to compute a score on the final model.

EM in BN issues

- With multiple hidden nodes, take advantage of conditional independencies
- Lots of tricks to speed up computation of expected counts
- If structure is unknown, add search operators to add and delete hidden nodes
- There are clever ways of search with unknown structure and hidden nodes

Another topic of current research is how to make search with both unknown structure and hidden nodes more efficient by considering them both simultaneously.