

Learning in Bayes Nets

Mausam

(Based on slides by Stuart Russell,
Marie desJardins, Subbarao
Kambhampati, Dan Weld)

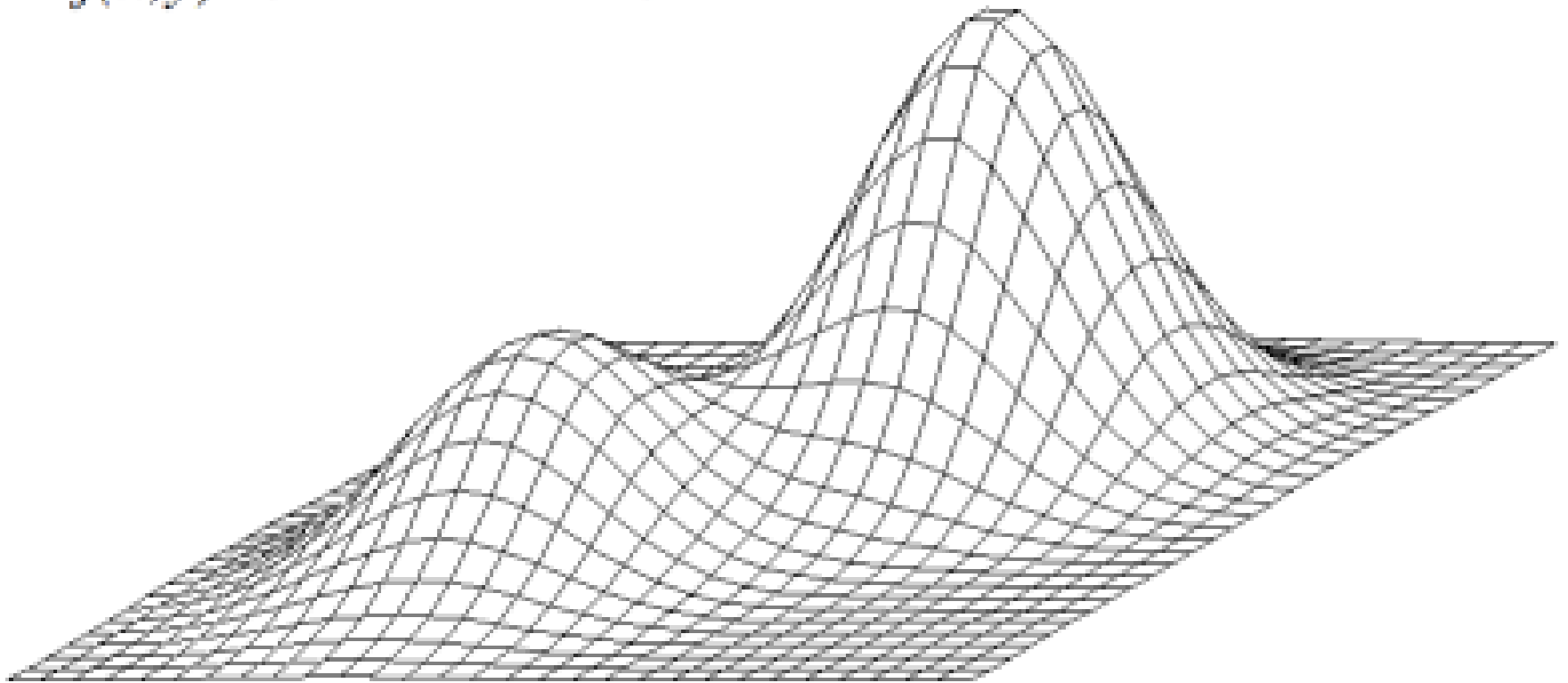
Parameter Estimation

- Learn all the CPTs in a Bayesian Net
- Data → Model → Queries
- Key idea: counting!

Optimization of Continuous Functions

- Discretization
 - use hill-climbing
- Gradient descent
 - make a move in the direction of the gradient
 - gradients: closed form or empirical

$$f(x,y) = e^{-(x^2+y^2)} + 2e^{-((x-1.7)^2+(y-1.7)^2)}$$



Gradient Descent

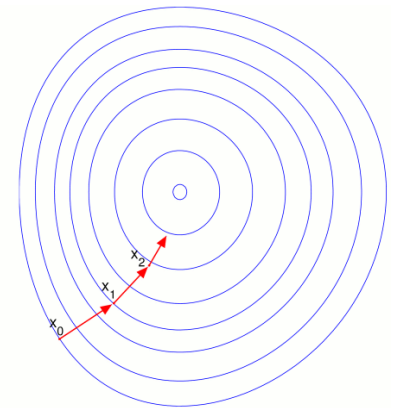
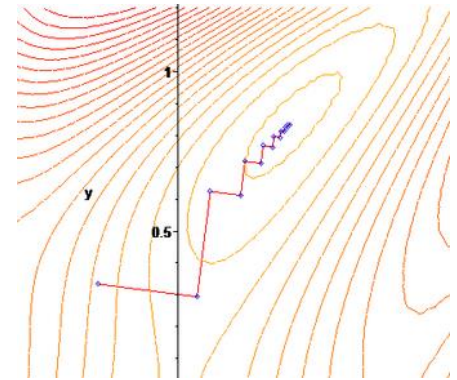
Assume we have a continuous function: $f(x_1, x_2, \dots, x_N)$
and we want minimize over continuous variables x_1, x_2, \dots, x_n

1. Compute the *gradients* for all i : $\partial f(x_1, x_2, \dots, x_N) / \partial x_i$
2. Take a small step downhill in the direction of the gradient:

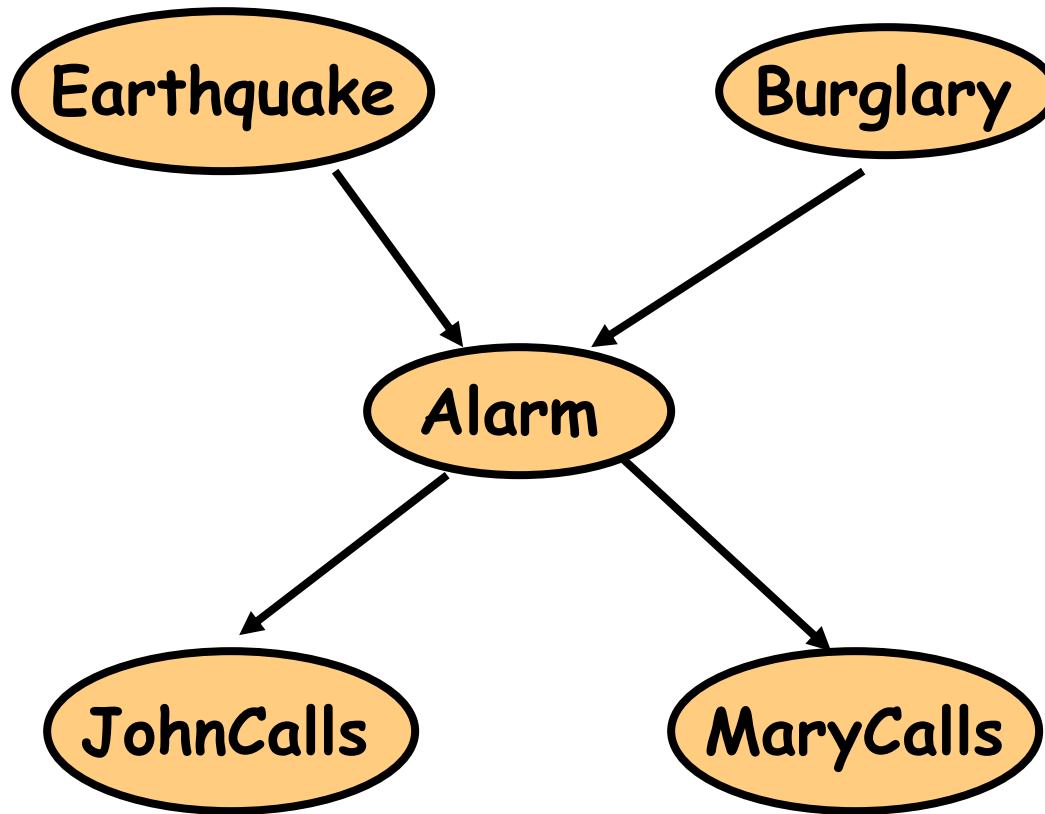
$$x_i \leftarrow x_i - \lambda \partial f(x_1, x_2, \dots, x_N) / \partial x_i$$

3. Repeat.

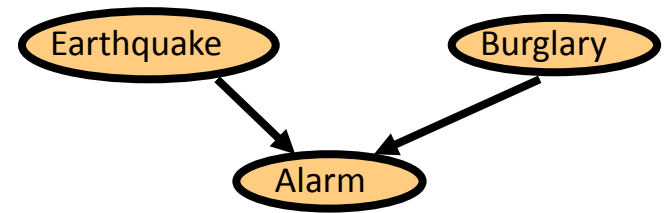
- How to select λ
 - Line search: successively double
 - until f starts to increase again



Burglars and Earthquakes



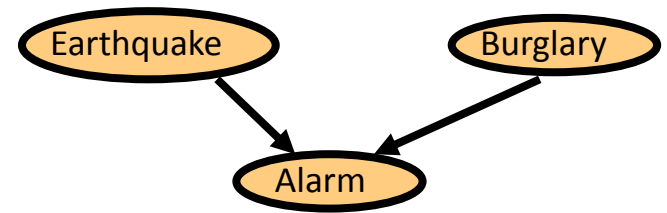
Counting



E	B	A	#
0	0	0	1000
0	0	1	10
0	1	0	20
0	1	1	100
1	0	0	200
1	0	1	50
1	1	0	0
1	1	1	5

	Pr(A E,B)
e,b	
e, \bar{b}	
\bar{e} ,b	
\bar{e} , \bar{b}	

Counting



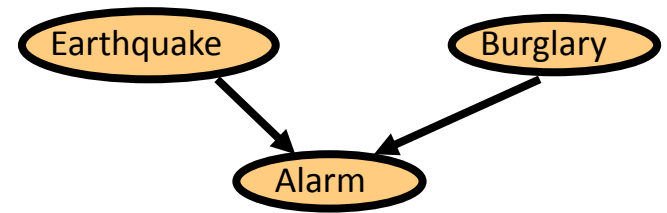
E	B	A	#
0	0	0	1000
0	0	1	10
0	1	0	20
0	1	1	100
1	0	0	200
1	0	1	50
1	1	0	0
1	1	1	5

	Pr(A E,B)
e,b	
e, \bar{b}	
\bar{e} ,b	
\bar{e} , \bar{b}	

$$P(a | \bar{e}, \bar{b}) = ?$$

$$= 10/1010$$

Counting



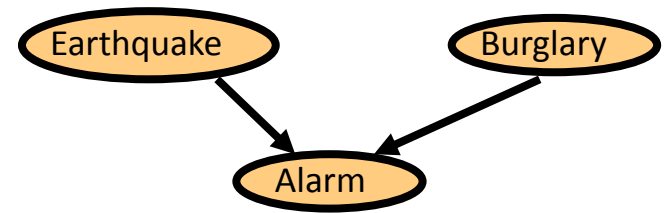
E	B	A	#
0	0	0	1000
0	0	1	10
0	1	0	20
0	1	1	100
1	0	0	200
1	0	1	50
1	1	0	0
1	1	1	5

	Pr(A E,B)
e,b	
e, \bar{b}	
\bar{e} ,b	
\bar{e} , \bar{b}	~0.01

$$P(a | \bar{e}, b) = ?$$

$$= 100/120$$

Counting

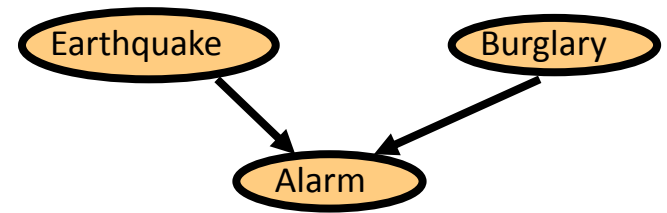


E	B	A	#
0	0	0	1000
0	0	1	10
0	1	0	20
0	1	1	100
1	0	0	200
1	0	1	50
1	1	0	0
1	1	1	5

	Pr(A E,B)
e,b	
e, \bar{b}	
\bar{e} ,b	0.83
\bar{e} , \bar{b}	~ 0.01

$$\begin{aligned}
 P(a|e, \bar{b}) &= ? \\
 &= 50/250
 \end{aligned}$$

Counting



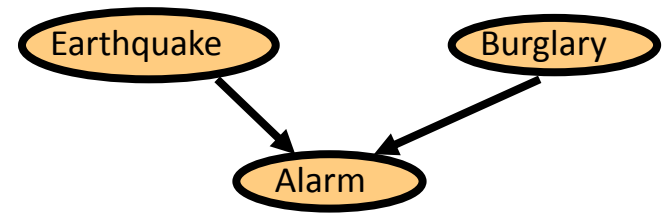
E	B	A	#
0	0	0	1000
0	0	1	10
0	1	0	20
0	1	1	100
1	0	0	200
1	0	1	50
1	1	0	0
1	1	1	5

	Pr(A E,B)
e,b	
e, \bar{b}	0.2
\bar{e} ,b	0.83
\bar{e} , \bar{b}	~0.01

$$P(a|e, b) = ?$$

$$= 5/5$$

Counting



E	B	A	#
0	0	0	1000
0	0	1	10
0	1	0	20
0	1	1	100
1	0	0	200
1	0	1	50
1	1	0	0
1	1	1	5

	Pr(A E,B)
e,b	1
e, \bar{b}	0.2
\bar{e} ,b	0.83
\bar{e} , \bar{b}	~0.01

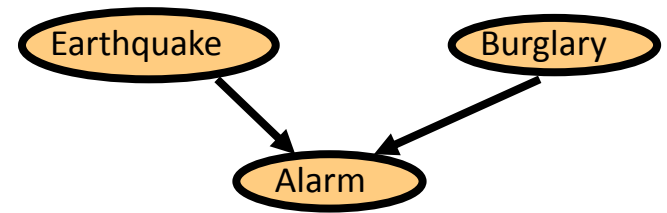
Bad idea to have prob as 0 or 1

- stumps Gibbs sampling
- low prob states become impossible

Solution: Smoothing

- Why?
 - To deal with events observed zero times.
 - “event”: a particular ngram
- How?
 - To shave a little bit of probability mass from the higher counts, and pile it instead on the zero counts
- Laplace Smoothing/Add-one smoothing
 - assume each event was observed at least once.
 - add 1 to all frequency counts
- Add m instead of 1 (m could be $>$ or $<$ 1)

Counting w/ Smoothing



E	B	A	#
0	0	0	1000+1
0	0	1	10+1
0	1	0	20+1
0	1	1	100+1
1	0	0	200+1
1	0	1	50+1
1	1	0	0+1
1	1	1	5+1

	Pr(A E,B)
e,b	0.86
e, \bar{b}	~ 0.2
\bar{e} ,b	~ 0.83
\bar{e} , \bar{b}	~ 0.01

ML vs. MAP Learning

- **ML: maximum likelihood (what we just did)**
 - find parameters that maximize the prob of seeing the data D
 - $\operatorname{argmax}_{\theta} P(D | \theta)$
 - easy to compute (for example, just counting)
 - assumes **uniform prior**
- **Prior: your belief before seeing any data**
 - **Uniform prior:** all parameters equally likely
- **MAP: maximum a posteriori estimate**
 - maximize prob of parameters after seeing data D
 - $\operatorname{argmax}_{\theta} P(\theta | D) = \operatorname{argmax}_{\theta} P(D | \theta)P(\theta)$
 - allows user to input additional domain knowledge
 - better parameters when data is sparse...
 - reduces to ML when infinite data

Example

Suppose there are five kinds of bags of candies:

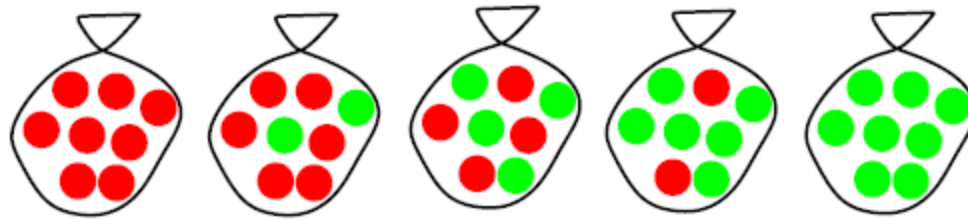
10% are h_1 : 100% cherry candies

20% are h_2 : 75% cherry candies + 25% lime candies

40% are h_3 : 50% cherry candies + 50% lime candies

20% are h_4 : 25% cherry candies + 75% lime candies

10% are h_5 : 100% lime candies



Then we observe candies drawn from some bag: ● ● ● ● ● ● ● ● ● ●

What kind of bag is it? What flavour will the next candy be?

Learning

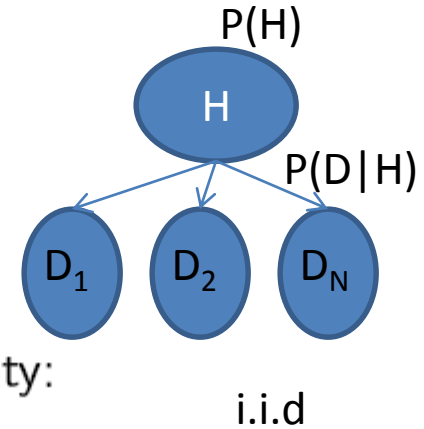
Inference

Full Bayesian learning

View learning as Bayesian updating of a probability distribution over the hypothesis space

H is the hypothesis variable, values h_1, h_2, \dots , prior $\mathbf{P}(H)$

j th observation d_j gives the outcome of random variable D_j
training data $\mathbf{d} = d_1, \dots, d_N$



Given the data so far, each hypothesis has a posterior probability:

$$P(h_i|\mathbf{d}) = \alpha P(\mathbf{d}|h_i)P(h_i)$$

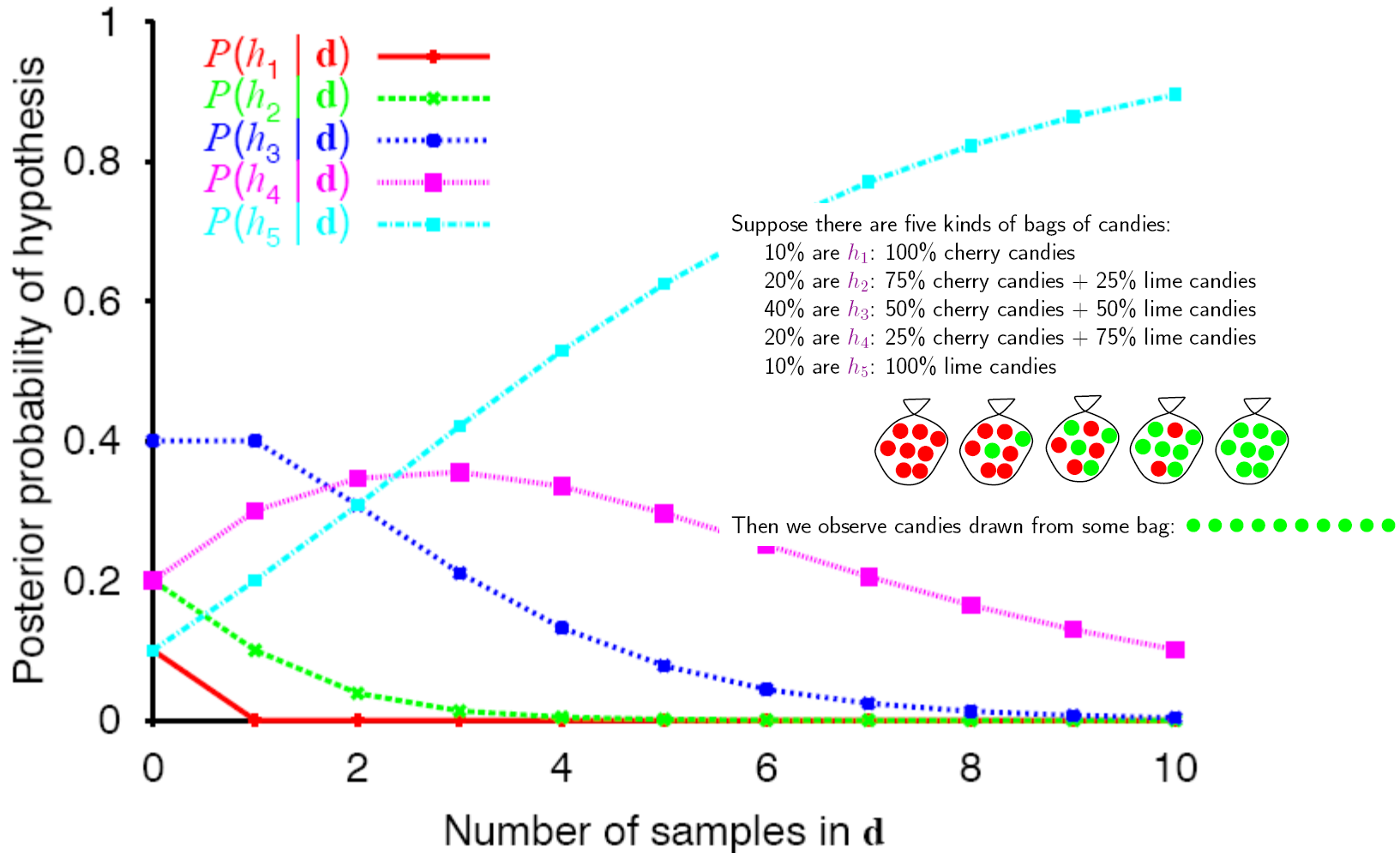
where $P(\mathbf{d}|h_i)$ is called the likelihood

Predictions use a likelihood-weighted average over the hypotheses:

$$\mathbf{P}(X|\mathbf{d}) = \sum_i \mathbf{P}(X|\mathbf{d}, h_i)P(h_i|\mathbf{d}) = \sum_i \mathbf{P}(X|h_i)P(h_i|\mathbf{d})$$

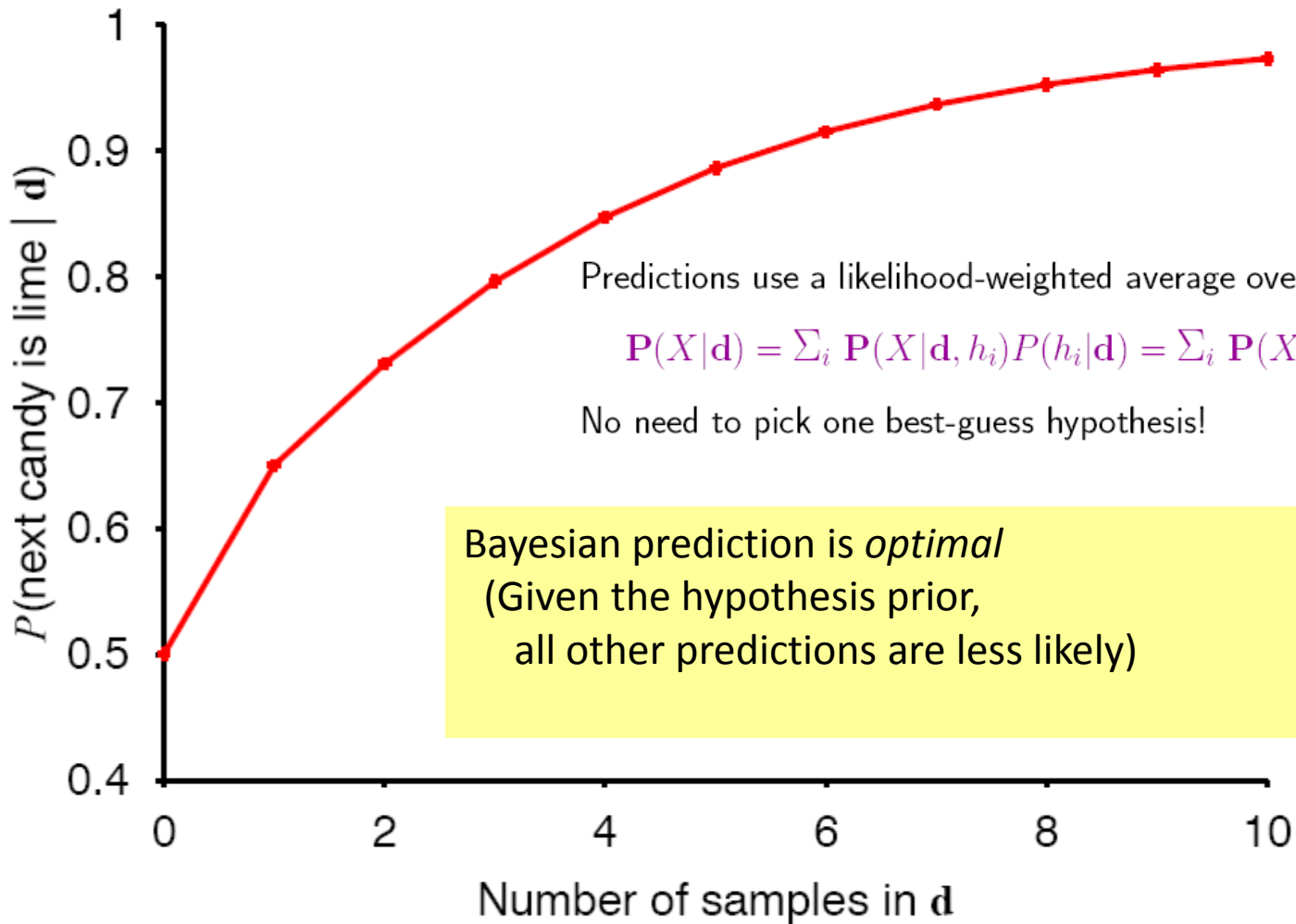
No need to pick one best-guess hypothesis!

Posterior probability of hypotheses



True hypothesis eventually dominates...
 probability of indefinitely producing uncharacteristic data $\rightarrow 0$

Prediction probability



ML vs. MAP Learning

- **ML: maximum likelihood (what we just did)**
 - find parameters that maximize the prob of seeing the data D
 - $\operatorname{argmax}_{\theta} P(D | \theta)$
 - easy to compute (for example, just counting)
 - assumes **uniform prior**
- **Prior: your belief before seeing any data**
 - **Uniform prior:** all parameters equally likely
- **MAP: maximum a posteriori estimate**
 - maximize prob of parameters after seeing data D
 - $\operatorname{argmax}_{\theta} P(\theta | D) = \operatorname{argmax}_{\theta} P(D | \theta)P(\theta)$
 - allows user to input additional domain knowledge
 - better parameters when data is sparse...
 - reduces to ML when infinite data

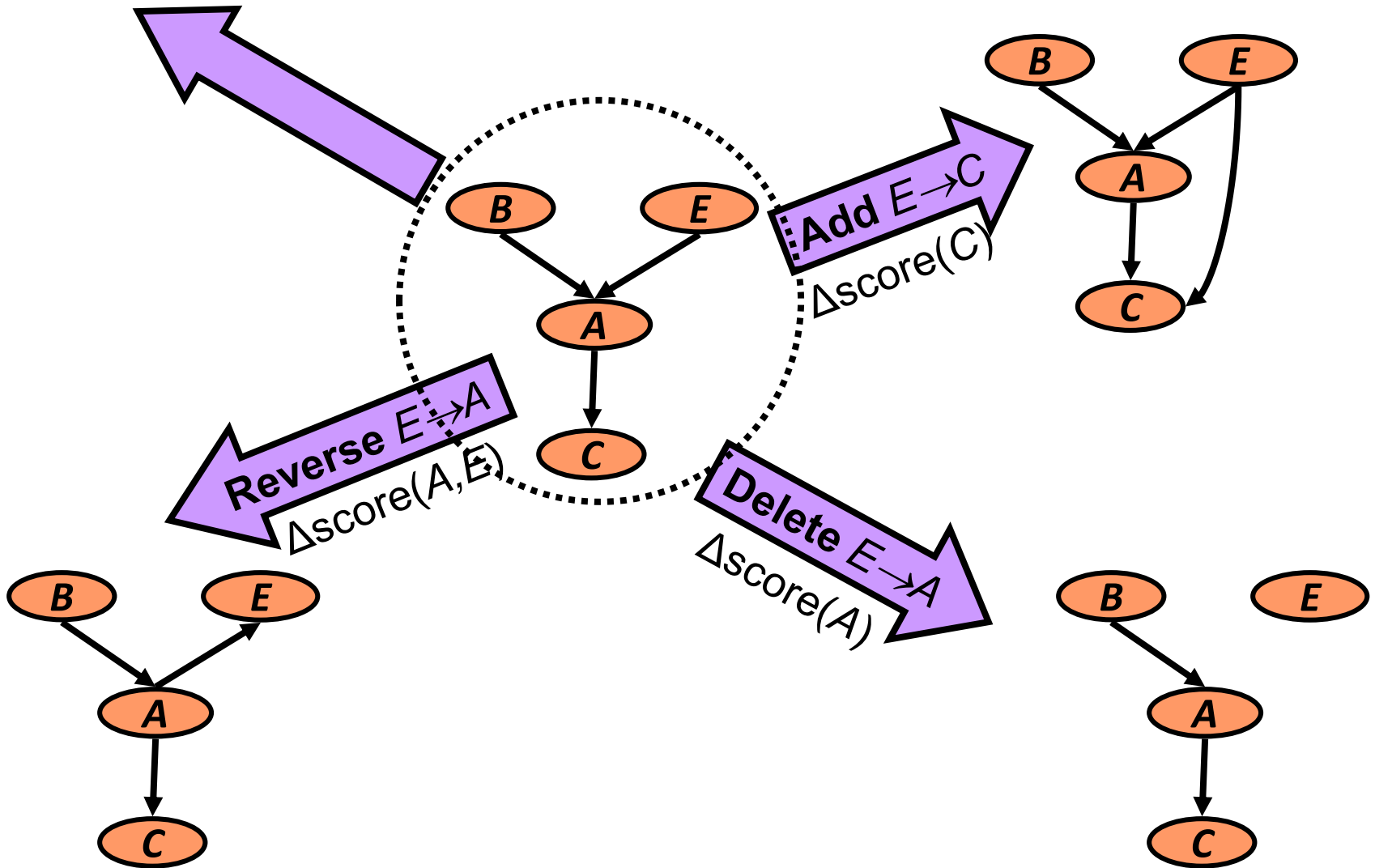
Learning the Structure

- Problem: learn the structure of Bayes nets
- Search thru the space...
 - of possible network structures!
 - Heuristic search/local search
- For each structure, learn parameters
- Pick the one that fits observed data best
 - Caveat – won't we end up fully connected????

When scoring, add a penalty

\propto model complexity

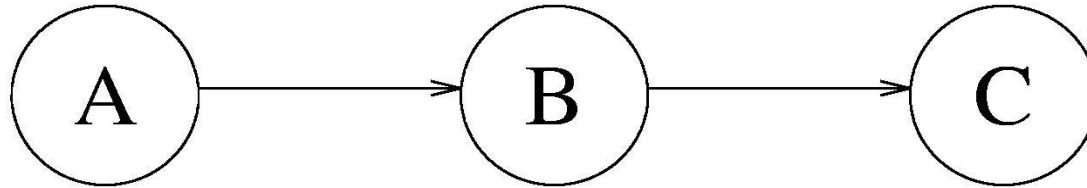
Local Search



How to learn when some data missing?

- Expectation Maximization (EM)

Example



Examples:	0	1	1
	1	0	0
	1	1	1
	1	?	0

Initialization: $P(B|A) =$ $P(C|B) =$
 $P(A) =$ $P(B|\neg A) =$ $P(C|\neg B) =$

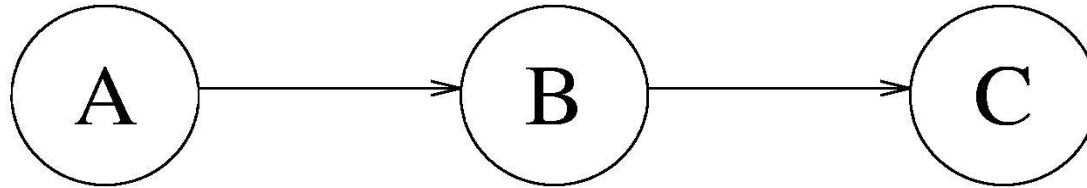
Chicken & Egg Problem

- If we knew the missing value
 - It would be easy to learn CPT

- If we knew the CPT
 - Then it'd be easy to infer the (probability of) missing value

- But we do not know either!

Example



Examples:	0	1	1
	1	0	0
	1	1	1
	1	?	0

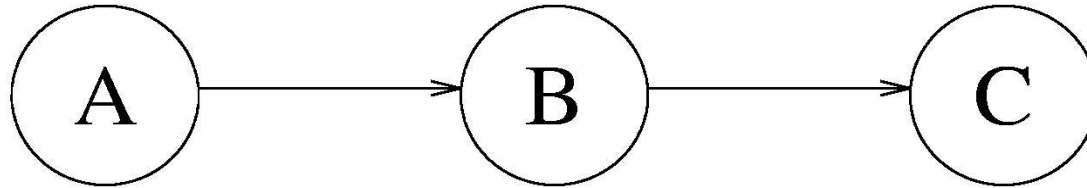
Initialization: $P(B|A) = 0$ $P(C|B) = 0$
 $P(A) = 0.75$ $P(B|\neg A) = 0$ $P(C|\neg B) = 0$

E-step: $P(? = 1) = P(B|A, \neg C) = \frac{P(A, B, \neg C)}{P(A, \neg C)} = \dots = 0$

M-step: $P(B|A) =$ $P(C|B) =$
 $P(A) =$ $P(B|\neg A) =$ $P(C|\neg B) =$

E-step: $P(? = 1) =$

Example



Examples:	0	1	1
	1	0	0
	1	1	1
	1	0	0

Initialization: $P(B|A) = 0$ $P(C|B) = 0$
 $P(A) = 0.75$ $P(B|\neg A) = 0$ $P(C|\neg B) = 0$

E-step: $P(? = 1) = P(B|A, \neg C) = \frac{P(A, B, \neg C)}{P(A, \neg C)} = \dots = 0$

M-step: $P(B|A) = 0.33$ $P(C|B) = 1$
 $P(A) = 0.75$ $P(B|\neg A) = 1$ $P(C|\neg B) = 0$

E-step: $P(? = 1) =$

Expectation Maximization

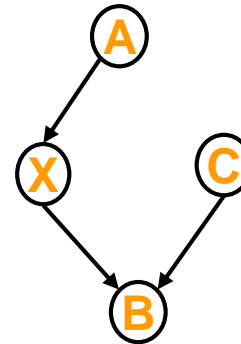
- **Guess** probabilities for nodes with **missing values** (e.g., based on other observations)
- **Compute the probability distribution** over the missing values, given our guess
- **Update the probabilities** based on the guessed values
- **Repeat** until convergence
- Guaranteed to converge to local optimum

Learning Summary

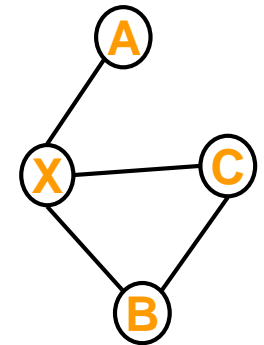
- **Known structure, fully observable:** only need to do parameter estimation
- **Unknown structure, fully observable:** do heuristic/local search through structure space, then parameter estimation
- **Known structure, missing values:** use expectation maximization (EM) to estimate parameters
- **Known structure, hidden variables:** apply adaptive probabilistic network (APN) techniques
- **Unknown structure, hidden variables:** too hard to solve!

Other Graphical Models

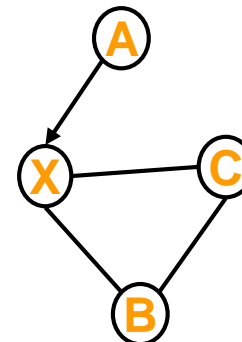
- Directed
 - Bayesian Networks



- Undirected
 - Markov Network (Markov Random Field)
 - BN \rightarrow MN (**moralization**: marry all co-parents)

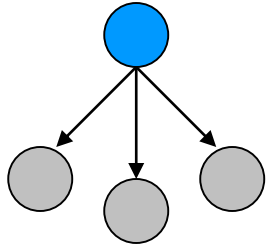


- Mixed
 - Chain Graph



Other Graphical Models

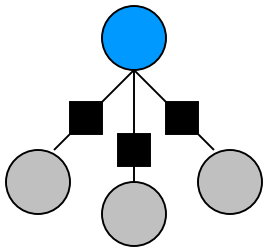
Naïve Bayes



Conditional



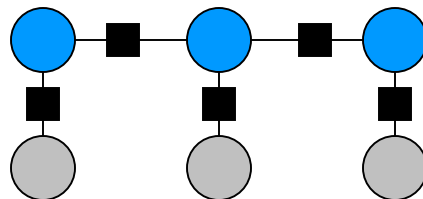
Logistic Regression



Sequence



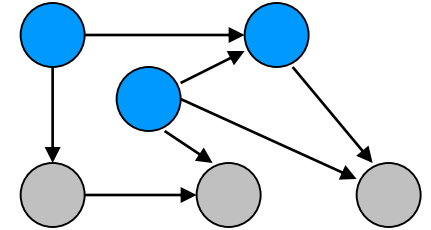
Linear-chain CRFs



General Graphs



Generative directed models



Conditional



General CRFs

