CMSC 671
Fall 2010

Thu 11/18/10

Learning probabilistic models
Chapter 20

Prof. Laura Zavala, laura.zavala@umbc.edu, ITE 373, 410-455-8775

Bayesian learning: Bayes' rule

- Given some model space (set of hypotheses h_i) and evidence (data D):
  - $P(h_i|D) = \alpha\, P(D|h_i)\, P(h_i)$
- We assume that observations are independent of each other, given a model (hypothesis), so:
  - $P(h_i|D) = \alpha \prod_j P(d_j|h_i)\, P(h_i)$
- To predict the value of some unknown quantity, X (e.g., the class label for a future observation):
  - $P(X|D) = \sum_i P(X|D, h_i)\, P(h_i|D) = \sum_i P(X|h_i)\, P(h_i|D)$

These are equal by our independence assumption

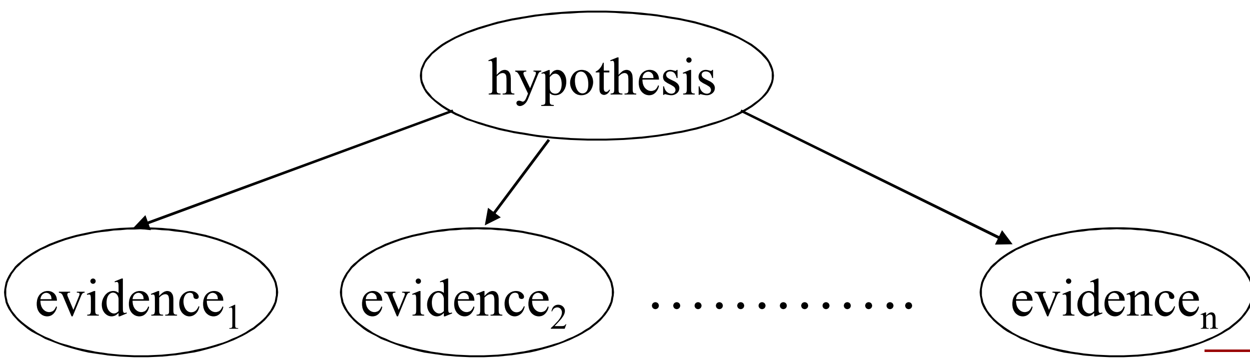
Bayesian learning

- We can apply Bayesian learning in three basic ways:
  - **BMA (Bayesian Model Averaging):** Don't just choose one hypothesis; instead, make predictions based on the weighted average of all hypotheses (or some set of best hypotheses)
  - **MAP (Maximum A Posteriori) hypothesis:** Choose the hypothesis with the highest *a posteriori* probability, given the data
  - **MLE (Maximum Likelihood Estimate):** Assume that all hypotheses are equally likely *a priori*; then the best hypothesis is just the one that maximizes the likelihood (i.e., the probability of the data given the hypothesis)
- **MDL (Minimum Description Length) principle:** Use some encoding to model the complexity of the hypothesis, and the fit of the data to the hypothesis, then minimize the overall description of $h_i + D$

Naïve Bayes

Naïve Bayes

- Use Bayesian modeling
- Make the simplest possible independence assumption:
  - Each attribute is independent of the values of the other attributes, given the class variable
  - In the restaurant domain: Cuisine is independent of Patrons, *given* a decision to stay (or not)



Bayesian Formulation

- $p(C \mid F_1, \ldots, F_n) = p(C)\, p(F_1, \ldots, F_n \mid C) \,/\, P(F_1, \ldots, F_n)$
  $= \alpha\, p(C)\, p(F_1, \ldots, F_n \mid C)$

- Naïve Bayes assumption
  - Assume that each feature F_i is **conditionally independent of the other features given the class C**.
- Then we have:
  $p(C \mid F_1, \ldots, F_n) = \alpha\, p(C)\, \Pi_i\, p(F_i \mid C)$
    - remember that α is a normalization factor
- We can estimate each of these conditional probabilities from the observed counts in the training data:
  $p(F_i \mid C) = \#(F_i \wedge C) \,/\, \#(C)$

Restaurant example (training set)

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

- Ten attributes: Alternative available? Bar in restaurant? Is it Friday? Are we hungry? How full is the restaurant? How expensive? Is it raining? Do we have a reservation? What type of restaurant is it? What's the purported waiting time?

Naive Bayes: Example

- $p(\text{Wait} \mid \text{Cuisine, Patrons, Rain}) =$

  $\alpha\ p(\text{Wait})\ p(\text{Cuisine} \mid \text{Wait})\ p(\text{Patrons} \mid \text{Wait})\ p(\text{Rain} \mid \text{Wait})$

  - remember that α is a normalization factor

- $P(\text{Rain}|\text{Wait}) = \#(\text{Wait} \wedge \text{Rain}) / \#(\text{Rain})$
- $P(\text{Patrons}|\text{Wait}) = \#(\text{Patrons} \wedge \text{Rain}) / \#(\text{Rain})$

  - Substitute *Wait* for *Wait=T* and *Wait=F*
  - This gives us the actual probabilities (of *Wait=T* and *Wait=F*)

Naïve Bayes Classifier

- Assume target function $f: X \rightarrow V$, where each instance (example) x is described by attributes <a1, a2, …, an>
- The most probable value of $f(x)$ is:

$$v_{MAP} = \operatorname*{argmax}_{v_j \in V} P(v_j | a_1, a_2 \ldots a_n)$$

$$\begin{aligned} v_{MAP} &= \operatorname*{argmax}_{v_j \in V} \frac{P(a_1, a_2 \ldots a_n | v_j) P(v_j)}{P(a_1, a_2 \ldots a_n)} \\ &= \operatorname*{argmax}_{v_j \in V} P(a_1, a_2 \ldots a_n | v_j) P(v_j) \end{aligned}$$

Naive Bayes assumption:

$$P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j)$$

which gives

Naive Bayes classifier: $v_{NB} = \operatorname*{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$

Naïve Bayes Classifier

$$\textbf{Naive Bayes classifier: } v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i|v_j)$$

- v_{NB}: Naïve Bayes value
  - The value returned by the Naïve Bayes classifier (T or F for the restaurant example)

- Remember that we can estimate each of these conditional probabilities from the observed counts in the training data:

$$p(a_i \mid v_j) = \#(a_i \wedge v_j) / \#(v_j)$$

Naïve Bayes Learning

- To learn from the examples, we estimate the probabilities from the observed counts in the training data:

Naive_Bayes_Learn($examples$)

For each target value v_j

$\hat{P}(v_j) \leftarrow$ estimate $P(v_j)$

For each attribute value a_i of each attribute a

$\hat{P}(a_i|v_j) \leftarrow$ estimate $P(a_i|v_j)$

Classify_New_Instance(x)

$$v_{NB} = \operatorname*{argmax}_{v_j \in V} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i|v_j)$$

Learning to classify text

Why?

- Learn which news articles are of interest
- Learn to classify web pages by topic

Naive Bayes is among most effective algorithms

What attributes shall we use to represent text documents??

Learning to classify text

Target concept $Interesting? : Document \rightarrow \{+, -\}$

1. Represent each document by vector of words
   - one attribute per word position in document
2. Learning: Use training examples to estimate
   - $P(+)$
   - $P(-)$
   - $P(doc|+)$
   - $P(doc|-)$

Naive Bayes conditional independence assumption

$$P(doc|v_j) = \prod_{i=1}^{length(doc)} P(a_i = w_k|v_j)$$

where $P(a_i = w_k|v_j)$ is probability that word in position i is w_k, given v_j

Learning to classify text

$\text{CLASSIFY\_NAIVE\_BAYES\_TEXT}(Doc)$

- $positions \leftarrow$ all word positions in Doc that contain tokens found in $Vocabulary$
- Return v_{NB}, where

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_{i \in positions} P(a_i | v_j)$$

Learning to classify text

LEARN_NAIVE_BAYES_TEXT($Examples, V$)

1. collect all words and other tokens that occur in Examples

- $Vocabulary \leftarrow$ all distinct words and other tokens in $Examples$

2. calculate the required $P(v_j)$ and $P(w_k|v_j)$ probability terms

- For each target value v_j in V do
  - $docs_j \leftarrow$ subset of $Examples$ for which the target value is v_j
  - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
  - $Text_j \leftarrow$ a single document created by concatenating all members of $docs_j$
  - $n \leftarrow$ total number of words in $Text_j$ (counting duplicate words multiple times)
  - for each word w_k in $Vocabulary$
    - $n_k \leftarrow$ number of times word w_k occurs in $Text_j$
    - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

Naive Bayes: Analysis

- Naive Bayes is amazingly easy to implement (once you understand the bit of math behind it)
- Remarkably, naive Bayes can outperform many much more complex algorithms—it's a baseline that should pretty much always be used for comparison
- Naive Bayes can't capture interdependencies between variables (obviously)—for that, we need Bayes nets!

Naïve Bayes in practice

- When to use
  - Moderate or large training set available
  - Attributes that describe instances are conditionally independent given classification
- Successful applications
  - Diagnosis
  - Classifying text documents
  - Detecting spam email

Exercise: Play Tennis example again!

- We to use NaiveBayes to decide whether the weather is amenable to playing tennis. Over the course of 2 weeks, data is collected to help ID3 build a decision tree.
- The target (binary) classification is
  - "should we play **PlayTennis**?" which can be Yes or No
- The weather attributes are outlook, temperature, humidity, and wind. They can have the following values:
  - Outlook = { sunny, overcast, rain }
  - Temperature = {hot, mild, cool }
  - Humidity = { high, normal }
  - Wind = {weak, strong }

Training examples for the target concept PlayTennis

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Exercise

- Consider a new instance (observation):

<Outlook=sunny, Temperature=cool, Humidity= high, Wind=strong>

- We want to compute (v_{NB}: Naïve Bayes value):

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i|v_j)$$

  - where V = {y, n} and the a_i's are the values for each attribute given in the observation vector (<Outlook=sunny, Temperature=cool, Humidity= high, Wind=strong>)

- That is, the *max* of:

$$P(y)\ P(sun|y)\ P(cool|y)\ P(high|y)\ P(strong|y)\ :$$

$$P(n)\ P(sun|n)\ P(cool|n)\ P(high|n)\ P(strong|n)$$

Exercise

- P(strong | y) = 3/9 = .33
  - Remember this is abbrev. for P(Wind=strong|PlayTennis=yes)
  - Also remember that $p(a_i \mid v_j) = \#(a_i \wedge v_j) / \#(v_j)$
  - Then, we have: p(strong | y) = #(strong $\wedge$ y) / #(y) = 3/9
- P(strong | n) = 3/5 = .60
- Therefore: $P(y)\ P(sun|y)\ P(cool|y)\ P(high|y)\ P(strong|y) = .005$

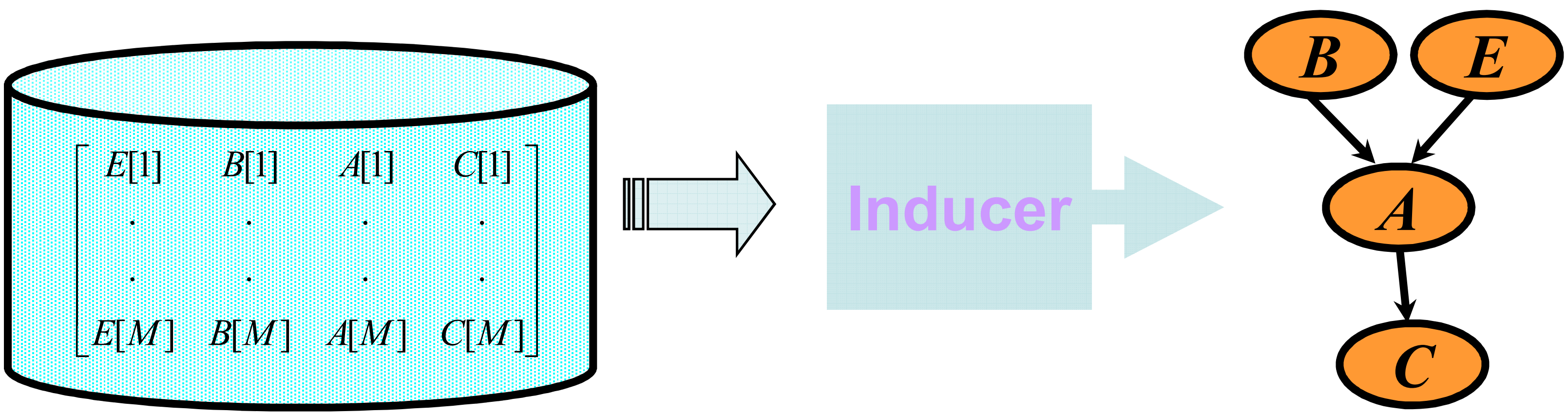
  $P(n)\ P(sun|n)\ P(cool|n)\ P(high|n)\ P(strong|n) = .021$
- So, the Naïve Bayes value (v_{NB}) is:
  - *Playtennis=**no***
  - What is its probability?
    - $\dfrac{.021}{.021+.005} = .795$

$\alpha\ P(v_j) \prod_i P(a_i|v_j)$

Learning Bayesian Networks

Learning Bayesian networks

- Given training set
- Find B that best matches ***D***
  - Learn the structure - *model selection*
  - Structure is given, learn the conditional probabilities - *parameter estimation*



Data D

Learning Bayesian networks

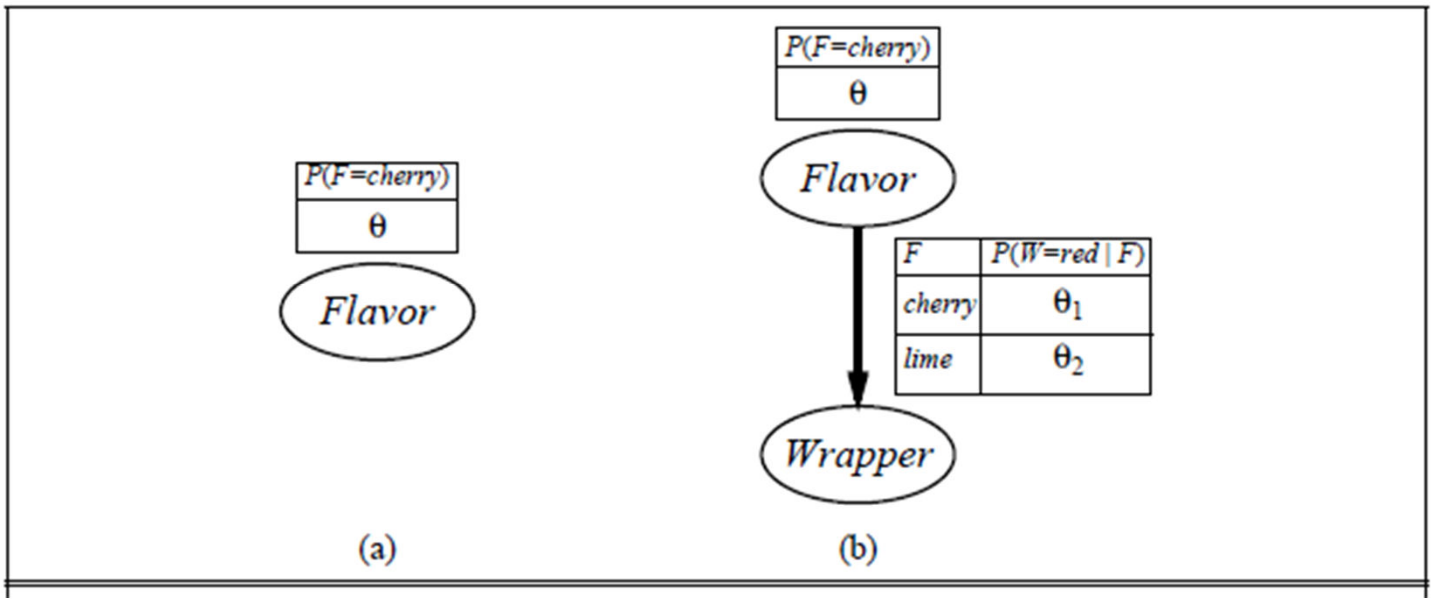
- **Known structure, fully observable**: only need to do parameter estimation
- **Unknown structure, fully observable:** do heuristic search through structure space, then parameter estimation
- **Known structure, missing values:** use expectation maximization (EM) to estimate parameters
- **Known structure,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

Learning Bayesian networks

- **Known structure, fully observable**: only need to do parameter estimation
- **Unknown structure, fully observable:** do heuristic 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!

Parameter estimation

- **Known structure, fully observable**: only need to do parameter estimation



Parameter estimation

- Assume known structure
- Goal: estimate BN parameters θ
  - entries in local probability models, $P(X \mid \text{Parents}(X))$
- A parameterization θ is good if it is likely to generate the observed data:

$$L(\theta : D) = P(D \mid \theta) = \prod_m P(d_m \mid \theta)$$

- Maximum Likelihood Estimation (MLE) Principle: Choose θ^* so as to maximize L

Parameter estimation II

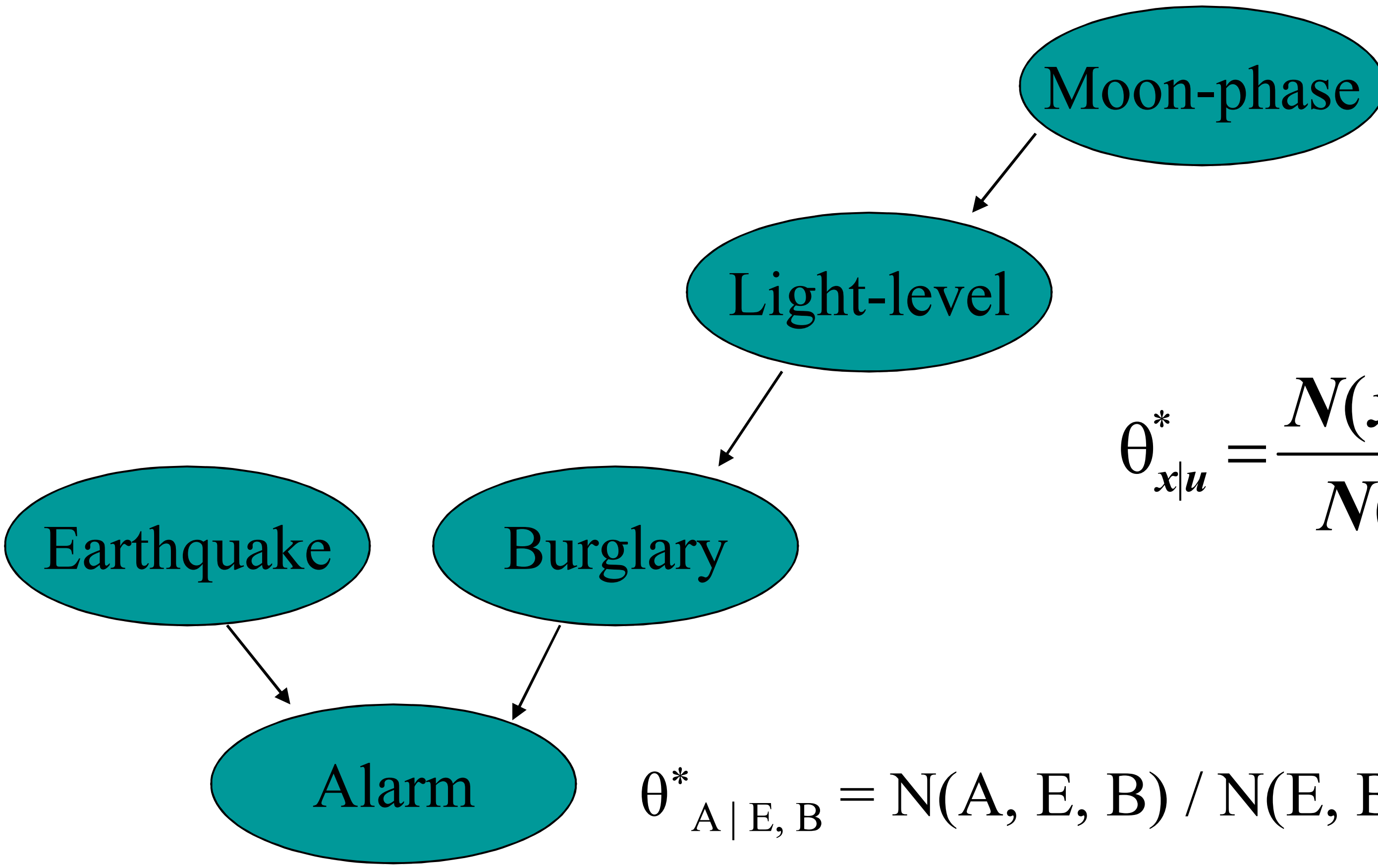
- The likelihood **decomposes** according to the structure of the network
  → we get a separate estimation task for each parameter
- The MLE (maximum likelihood estimate) solution:
  - for each value x of a node X
  - and each instantiation $\boldsymbol{u}$ of $Parents(X)$

$$\theta^*_{\boldsymbol{x}|\boldsymbol{u}} = \frac{N(\boldsymbol{x},\boldsymbol{u})}{N(\boldsymbol{u})}$$

sufficient statistics

  - Just need to collect the counts for every combination of parents and children observed in the data
  - MLE is equivalent to an assumption of a uniform prior over parameter values

Sufficient statistics: Example



$$\theta^*_{\boldsymbol{x}|\boldsymbol{u}} = \frac{\boldsymbol{N}(\boldsymbol{x},\boldsymbol{u})}{\boldsymbol{N}(\boldsymbol{u})}$$

$\theta^*_{A\,|\,E,\,B} = N(A, E, B) \,/\, N(E, B)$

Learning Bayesian networks

- **Known structure, fully observable**: only need to do parameter estimation
- **Unknown structure, fully observable:** do heuristic 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!

Model selection

- **Unknown structure, fully observable:** do heuristic search through structure space, then parameter estimation

Goal: Select the best network structure, given the data

Input:

- Training data
- Scoring function

Output:

- A network that maximizes the score

Structure selection: Scoring

- Bayesian: prior over parameters and structure
  - get balance between model complexity and fit to data as a byproduct

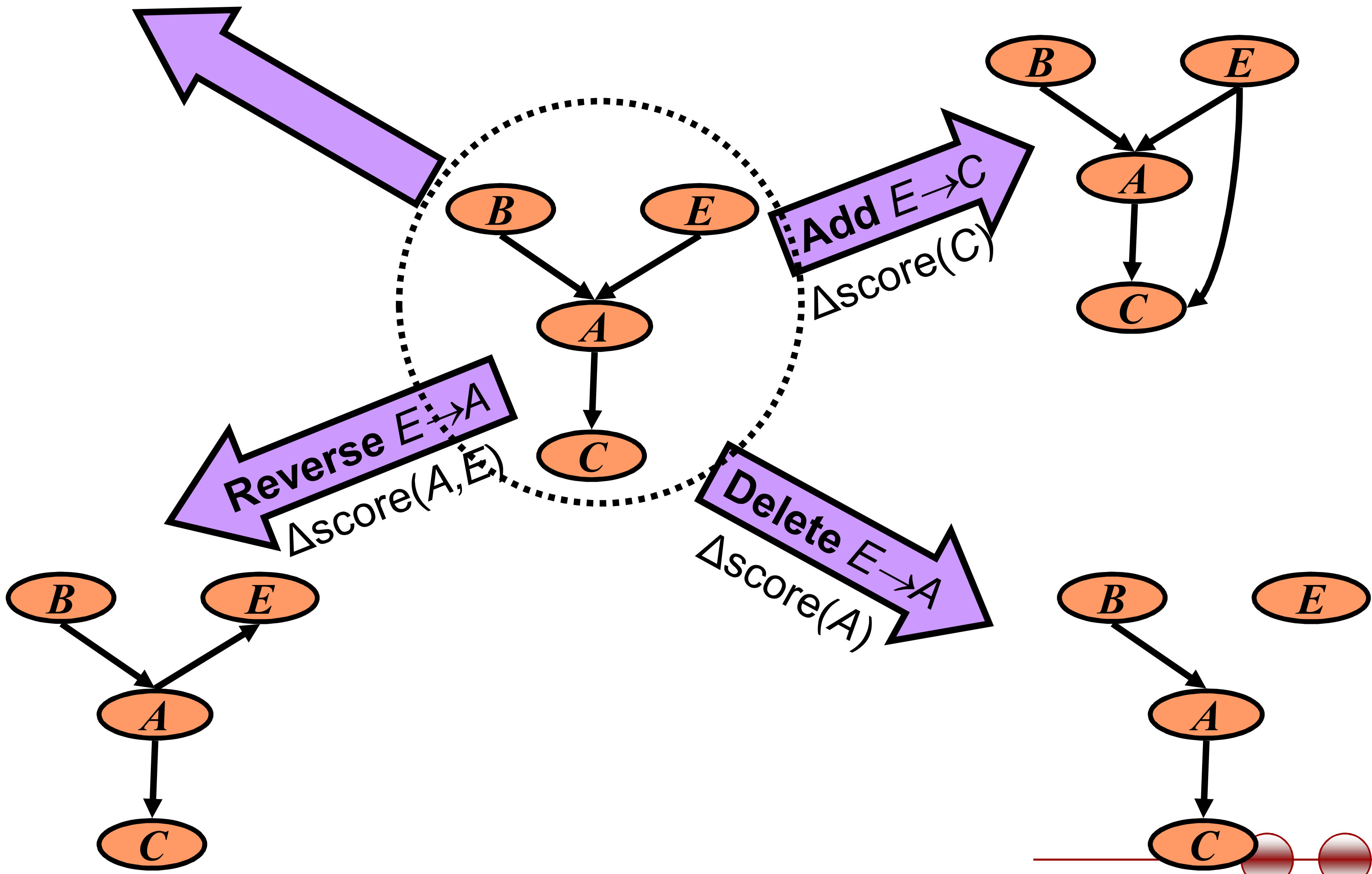
Marginal likelihood ***Prior***

- Score (G:D) = log P(G|D) α log [P(D|G) P(G)]
- Marginal likelihood just comes from our parameter estimates
- Prior on structure can be any measure we want; typically a function of the network complexity

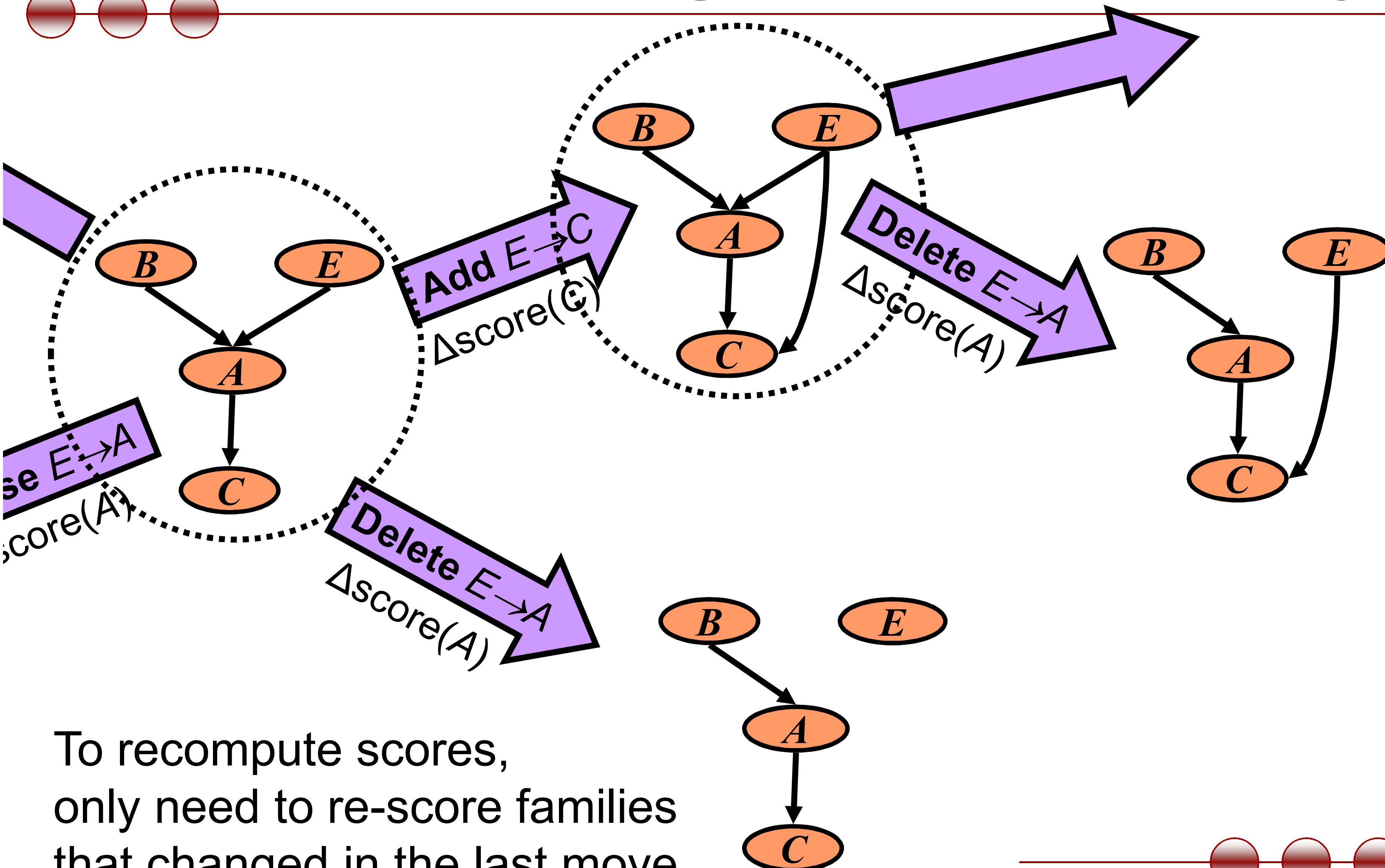
Same key property: Decomposability

Score(structure) = Σ_i Score(family of X_i)

Heuristic search



Exploiting decomposability



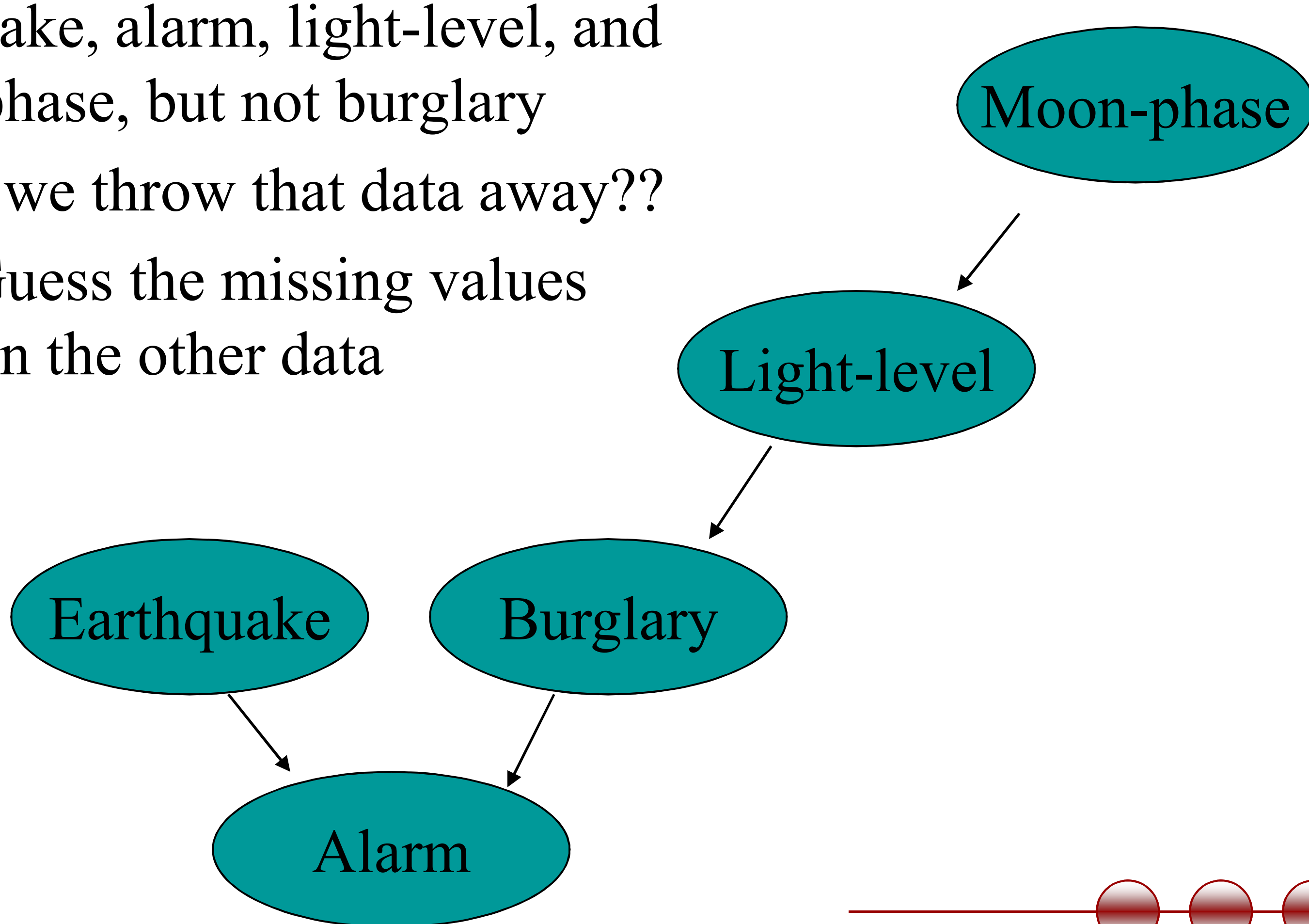
To recompute scores,
only need to re-score families
that changed in the last move

Learning Bayesian networks

- **Known structure, fully observable**: only need to do parameter estimation
- **Unknown structure, fully observable:** do heuristic 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!

Handling missing data

- Suppose that in some cases, we observe earthquake, alarm, light-level, and moon-phase, but not burglary
- Should we throw that data away??
- **Idea**: Guess the missing values based on the other data

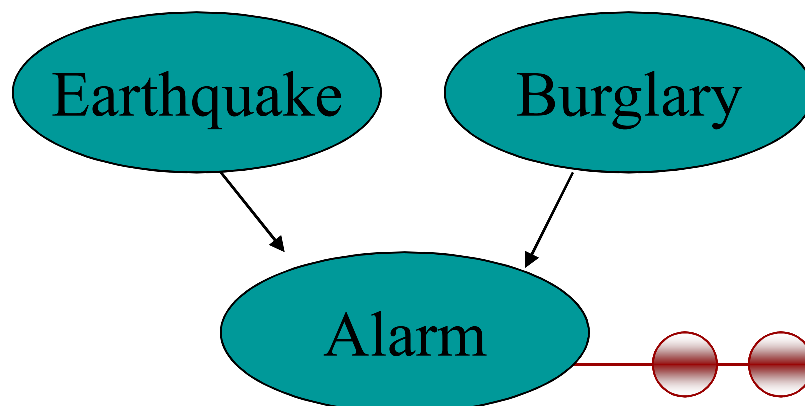


EM (expectation maximization)

- **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

EM example

- Suppose we have observed Earthquake and Alarm but not Burglary for an observation on November 27
- We estimate the CPTs based on the *rest* of the data
- We then estimate P(Burglary) for November 27 from those CPTs
- Now we recompute the CPTs as if that estimated value had been observed
- Repeat until convergence!



Unsupervised Learning: Clustering

Unsupervised learning

- Learn without a “supervisor” who labels instances
  - Clustering
  - Scientific discovery
  - Pattern discovery
  - Associative learning
- Clustering:
  - Given a set of instances *without labels*, partition them such that each instance is:
    - *similar* to other instances in its partition (intra-cluster similarity)
    - *dissimilar* from instances in other partitions (inter-cluster dissimilarity)

Clustering techniques

- **Agglomerative clustering**
  - Single-link clustering
  - Complete-link clustering
  - Average-link clustering
- **Partitional clustering**
  - k-means clustering
- Spectral clustering

Agglomerative clustering

- Agglomerative:
  - Start with each instance in a cluster by itself
  - Repeatedly combine pairs of clusters until some stopping criterion is reached (or until one “super-cluster” with substructure is found)
  - Often used for non-fully-connected data sets (e.g., clustering in a social network)
- Single-link:
  - At each step, combine the two clusters with the smallest minimum distance between any pair of instances in the two clusters (i.e., find the shortest “edge” between each pair of clusters
- Average-link:
  - Combine the two clusters with the smallest average distance between all pairs of instances
- Complete-link:
  - Combine the two clusters with the smallest *maximum* distance between any pair of instances

k-Means

- Partitional:
  - Start with all instances in a set, and find the “best” partition
- k-Means:
  - Basic idea: use expectation maximization to find the best clusters
  - Objective function: Minimize the within-cluster sum of squared distances
  - Initialize *k* clusters by choosing *k* random instances as cluster “centroids” (where *k* is an input parameter)
  - E-step: Assign each instance to its nearest cluster (using Euclidean distance to the centroid)
  - M-step: Recompute the centroid as the center of mass of the instances in the cluster
  - Repeat until convergence is achieved