CMSC 471
Fall 2012

Class #23

Thursday, November 15, 2012
Machine Learning II

Kevin Winner, winnerk1@umbc.edu

Instance-Based & Bayesian Learning

Chapter 18.8, 20 (parts)

Some material adapted
from lecture notes by
Lise Getoor and Ron Parr

Today's Class

- Project clarifications
- Extra credit assignment
- Guest lecture
- k-nearest neighbor
- k-means clustering
- Naïve Bayes
- Learning Bayes networks

Instance-Based Learning

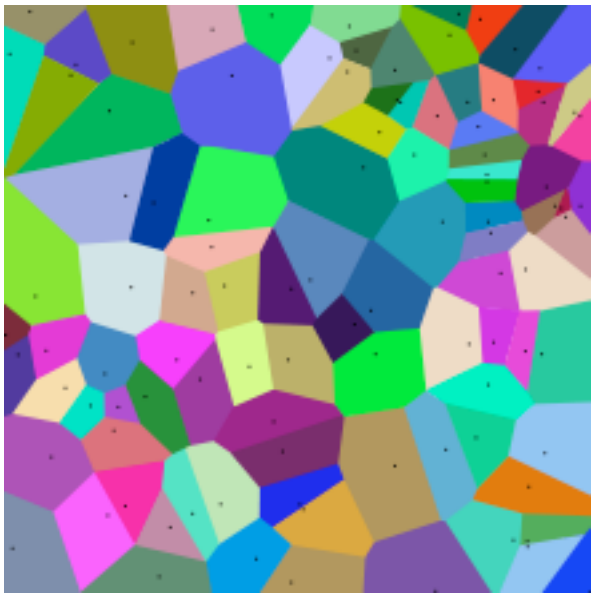
K-nearest neighbor

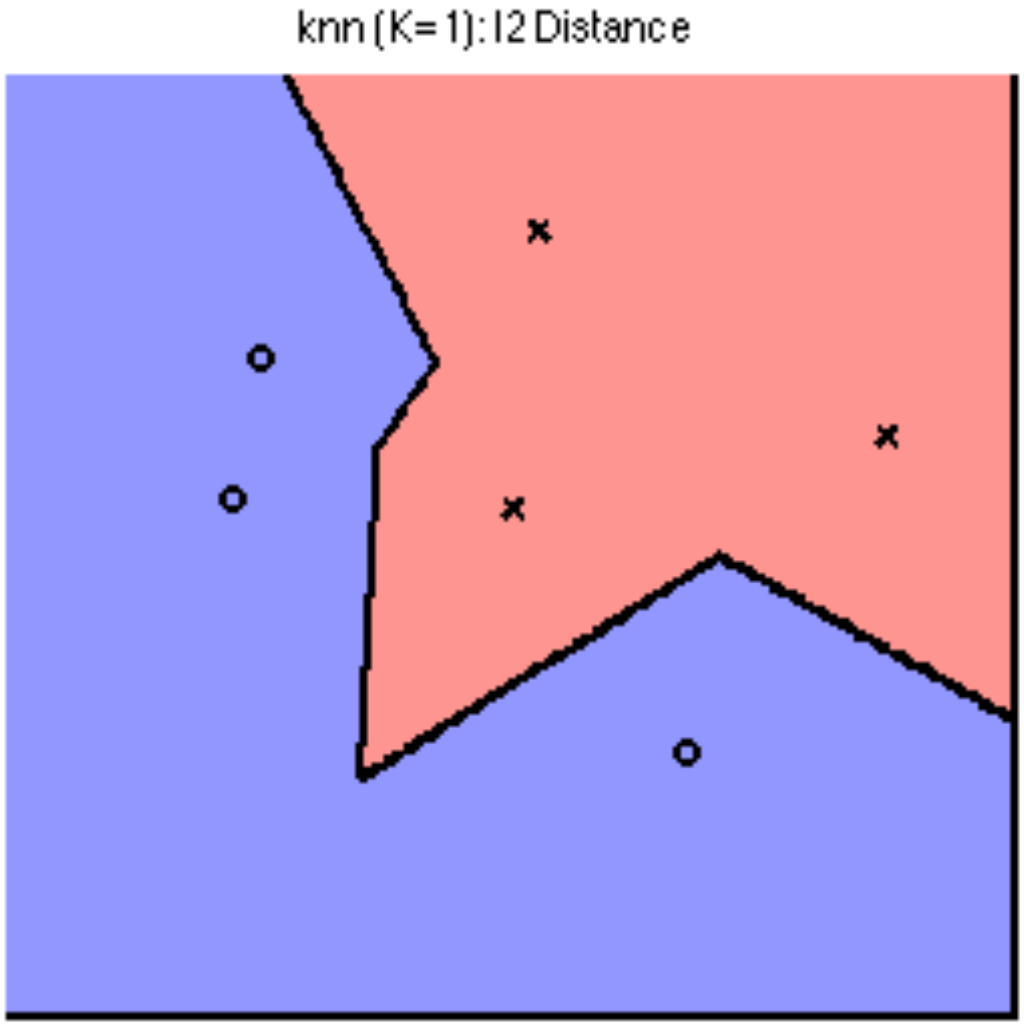
IBL

- Decision trees are a kind of *model*-based learning
  - We take the training instances and use them to build a *model* of the mapping from inputs to outputs
  - This model (e.g., a decision tree) can be used to make predictions on new (test) instances
- Another option is to do *instance*-based learning
  - Save all (or some subset) of the instances
  - Given a test instance, use some of the stored instances in some way to make a prediction
- Instance-based methods:
  - Nearest neighbor and its variants (today)
  - Support vector machines (if you take 671)

Nearest Neighbor

- Vanilla "Nearest Neighbor":
  - Save all training instances $X_i = (C_i, \boldsymbol{F_i})$ in T
  - Given a new test instance Y, find the instance X_j that is closest to Y
  - Predict class C_i
- What does "closest" mean?
  - Usually: Euclidean distance in feature space
  - Alternatively: Manhattan distance, or any other distance metric



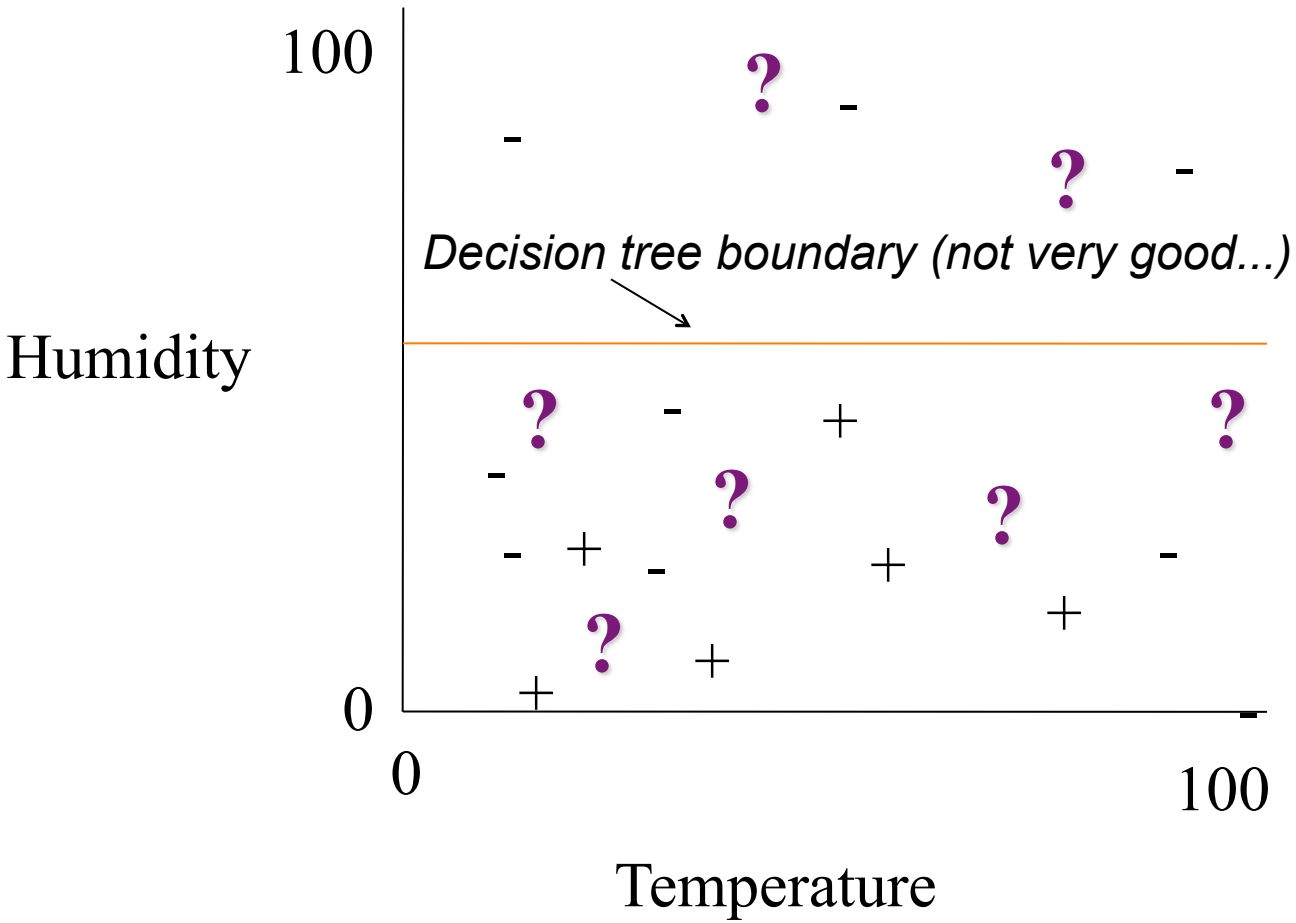


Borrowed from Ben Taskar of UPenn

K-Nearest Neighbor

- What if the data is noisy?
  - Generalize to k-nearest neighbor
  - Find the *k* closest training instances to *Y*
  - Use majority voting to predict the class label of *Y*
  - Better yet: use weighted (by distance) voting to predict the class label
    - Kernel Regression

Nearest Neighbor Example: Run Outside (+) or Inside (-)



- Noisy data
- Not linearly separable

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

- **Hierarchical clustering**
  - **Agglomerative clustering**
    - Single-link clustering
    - Complete-link clustering
    - Average-link clustering
  - Divisive clustering
- **Partitional clustering**
  - k-means clustering
- Spectral clustering
  - Dimension reduction

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

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 our restaurant domain: Cuisine is independent of Patrons, *given* a decision to stay (or not)

Bayesian Formulation

- $p(C \mid F_1, ..., F_n) = p(C)\, p(F_1, ..., F_n \mid C) / P(F_1, ..., F_n)$
  $= \alpha\, p(C)\, p(F_1, ..., F_n \mid C)$
- Assume that each feature F_i is conditionally independent of the other features given the class C. Then:
  $p(C \mid F_1, ..., F_n) = \alpha\, p(C)\, \Pi_i\, p(F_i \mid C)$
- We can estimate each of these conditional probabilities from the observed counts in the training data:
  $p(F_i \mid C) = N(F_i \wedge C) / N(C)$
  - One subtlety of using the algorithm in practice: When your estimated probabilities are zero, ugly things happen
  - The fix: Add one to every count (aka "Laplacian smoothing")

Naive Bayes: Example

- p(Wait | Cuisine, Patrons, Rainy?) =
  α p(Wait) p(Cuisine | Wait) p(Patrons | Wait)
  p(Rainy? | Wait)

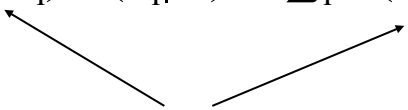
Naive Bayes: Analysis

- 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!

Learning Bayesian Networks

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):
  - $P(X|D) = \sum_i P(X|D, h_i)\, P(h_i|D) = \sum_i P(X|h_i)\, P(h_i|D)$



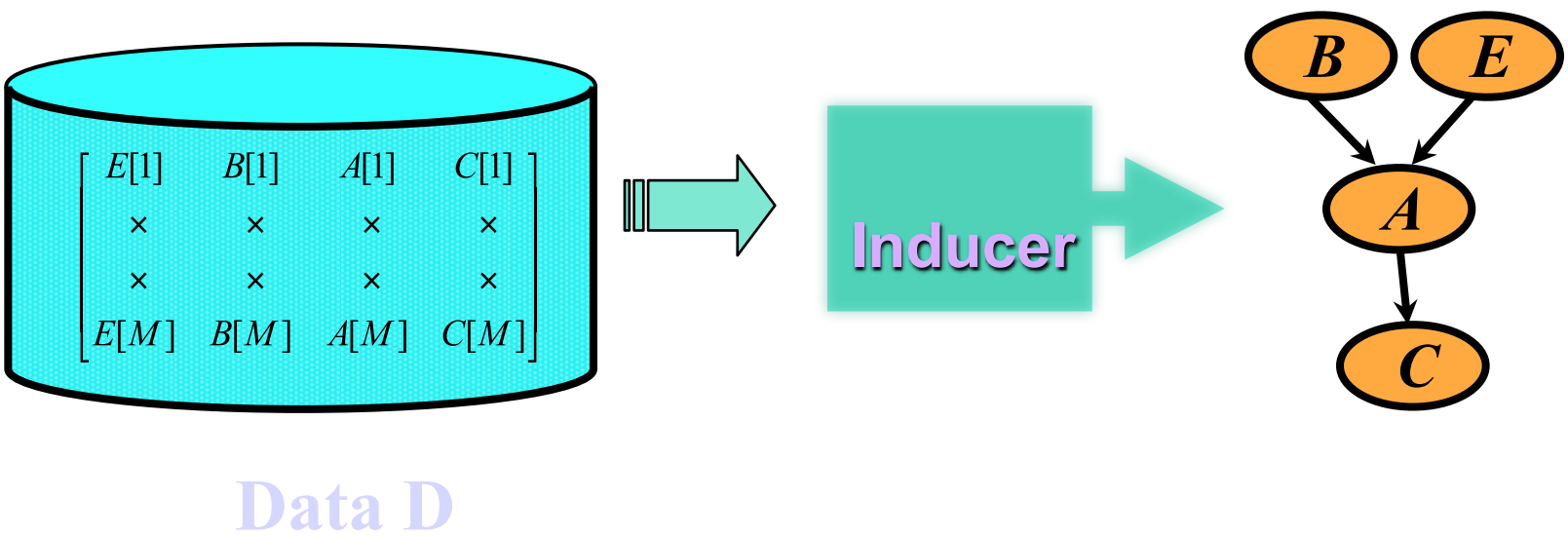
These are equal by our independence assumption

Bayesian Learning

- We can apply Bayesian learning in three basic ways:
  - **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)
  - **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)
- **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 $h_i + D$

Learning Bayesian Networks

- Given training set $\boldsymbol{D} = \{\boldsymbol{x}[1],\ldots,\boldsymbol{x}[M]\}$
- Find B that best matches $\boldsymbol{D}$
  - model selection
  - parameter estimation



Parameter Estimation

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

$$L(\theta : D) = P(D \mid \theta) = \prod_m P(x[m] \mid \theta)$$

i.i.d. samples

- 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)*

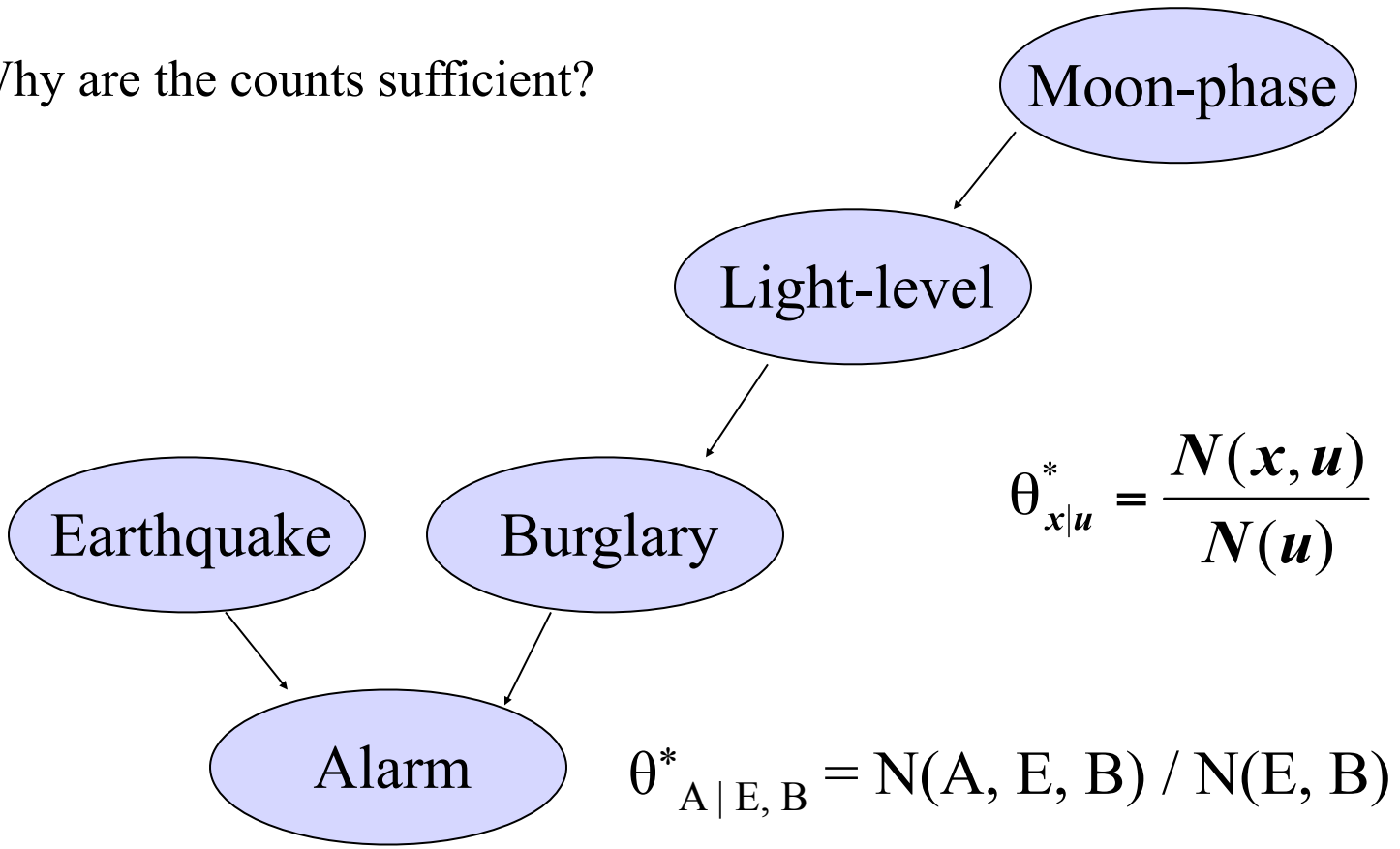
$$\theta^{*}_{x|\boldsymbol{u}} = \frac{N(x,\boldsymbol{u})}{N(\boldsymbol{u})}$$

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

- Why are the counts sufficient?



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

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

Model Selection

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

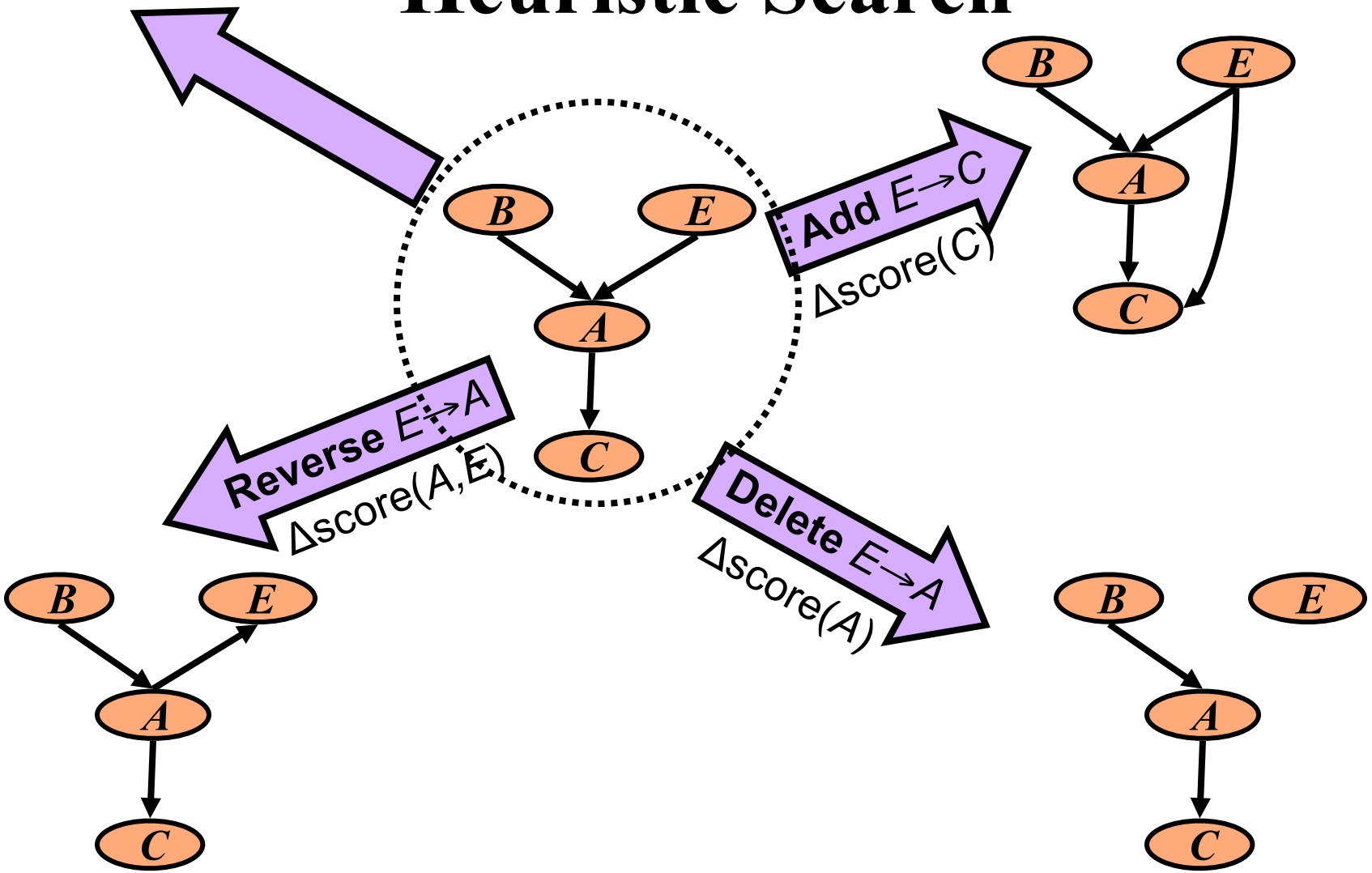
Marginal likelihood ***Prior***

- 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

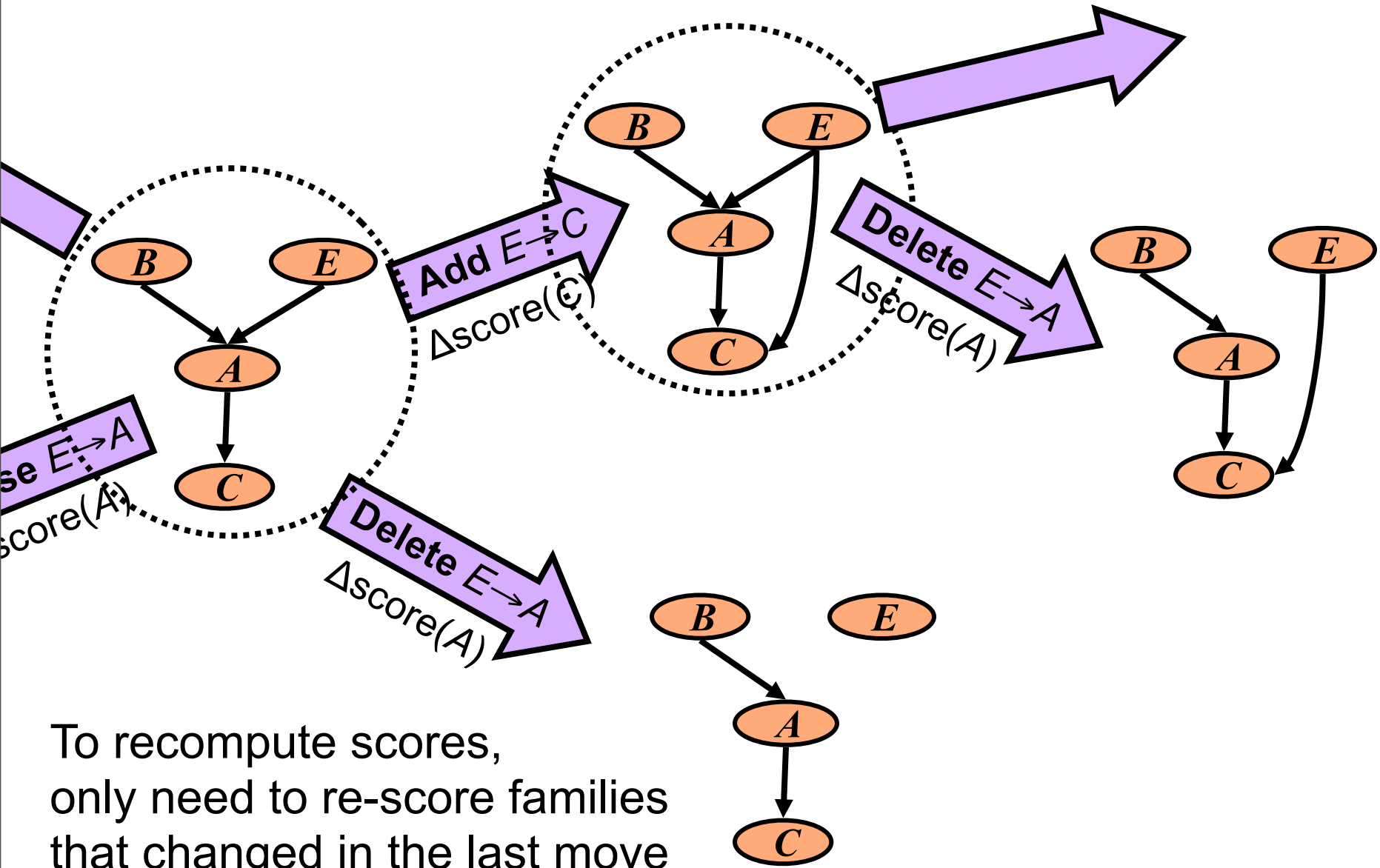
Same key property: Decomposability

$$\text{Score(structure)} = \sum_i \text{Score(family of } X_i)$$

Heuristic Search



Exploiting Decomposability



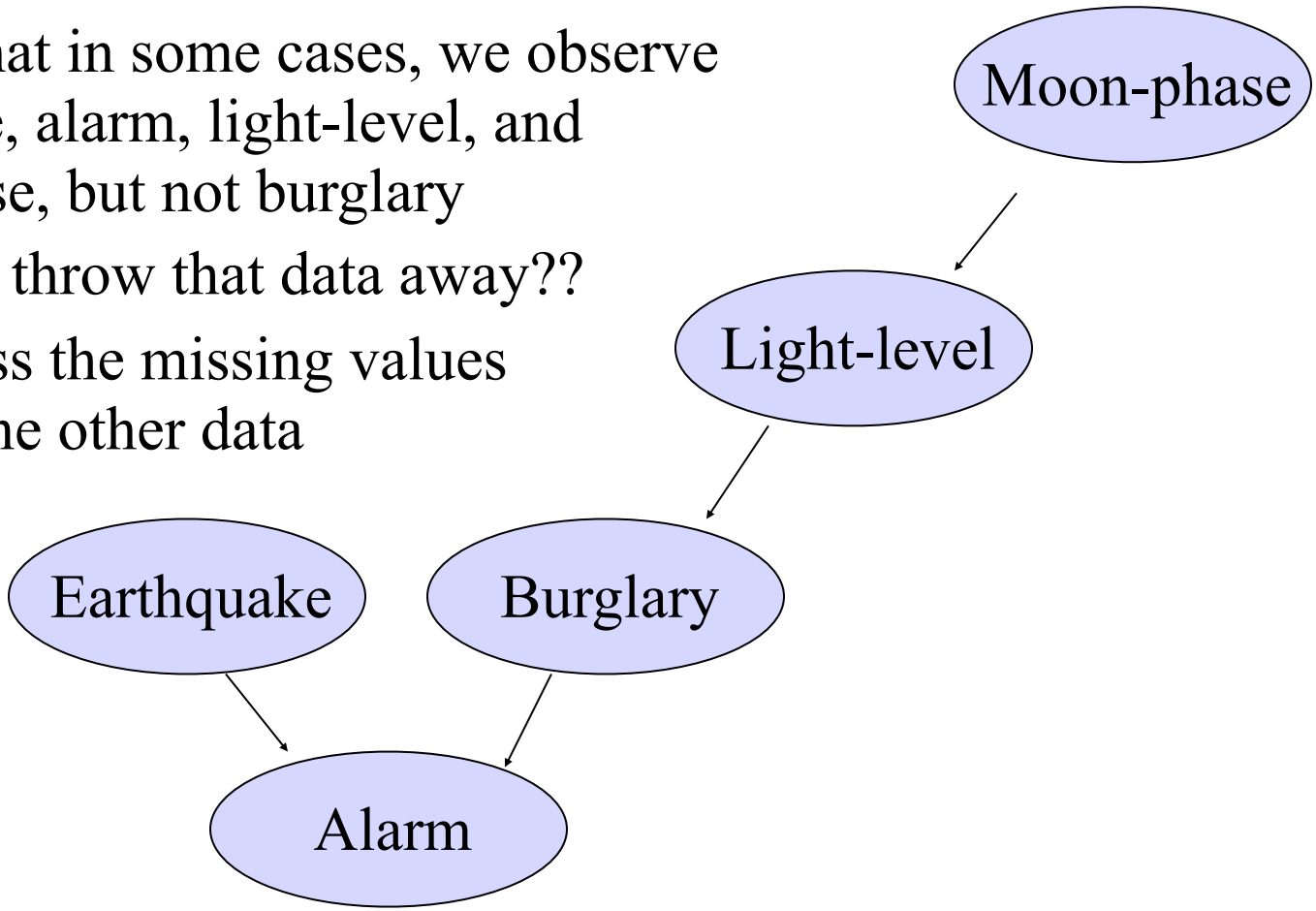
To recompute scores,
only need to re-score families
that changed in the last move

Variations on a Theme

- **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!