Bayesian Learning

Chapter 20.1-20.4

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

Simple case: 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 each feature F_i is *conditionally independent* of others given the class C. Then:
  $p(C \mid F_1, ..., F_n) = \alpha\, p(C)\, \Pi_i\, p(F_i \mid C)$
- 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
  - Fix: Add one to every count (aka Laplace smoothing—they have a different name for *everything*!)

Naive Bayes: Example

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

$$= \alpha \bullet p(\text{Wait}) \bullet p(\text{Cuisine}|\text{Wait}) \bullet p(\text{Patrons}|\text{Wait}) \bullet p(\text{Rainy?}|\text{Wait})$$

$$= \frac{p(\text{Wait}) \bullet p(\text{Cuisine}|\text{Wait}) \bullet p(\text{Patrons}|\text{Wait}) \bullet p(\text{Rainy?}|\text{Wait})}{p(\text{Cuisine}) \bullet p(\text{Patrons}) \bullet p(\text{Rainy?})}$$

We can estimate all of the parameters (p(F) and p(C) just by counting from the training examples

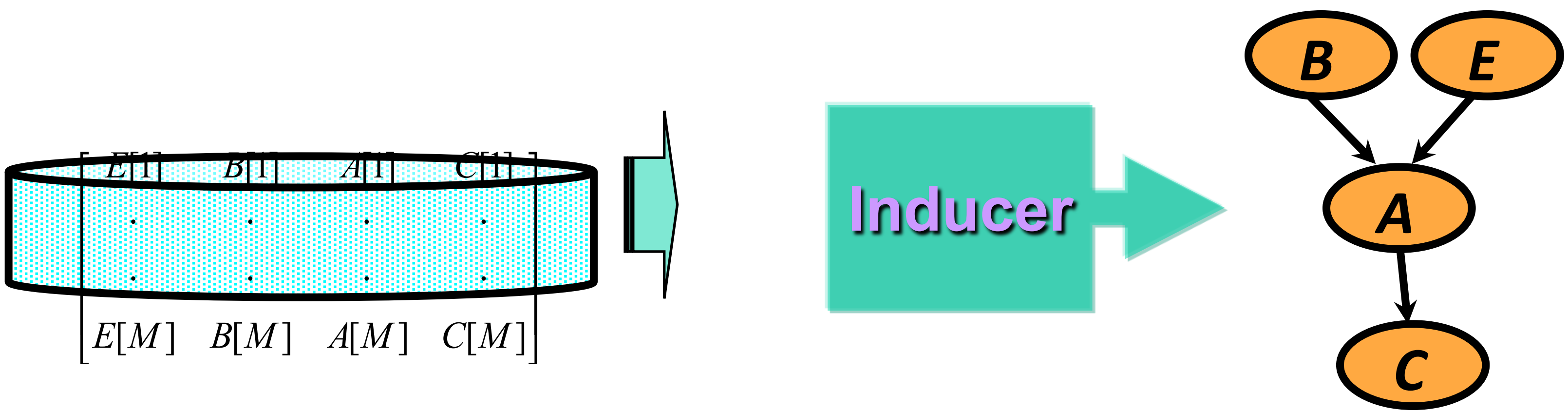
Naive Bayes: Analysis

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

Learning Bayesian networks

- Given training set
- Find B that best matches ***D***
  - model selection
  - parameter estimation

$$D = \{x[1], \ldots, x[M]\}$$



Data D

Learning Bayesian Networks

- Describe a BN by specifying its (1) structure and (2) conditional probability tables (CPTs)
- Both can be learned from data, but
  - learning structure much harder than learning parameters
  - learning when some nodes are hidden, or with missing data harder still
- Four cases:

Structure	*Observability*	*Method*
Known	Full	Maximum Likelihood Estimation
Known	Partial	EM (or gradient ascent)
Unknown	Full	Search through model space
Unknown	Partial	EM + search through model space

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 ***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

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

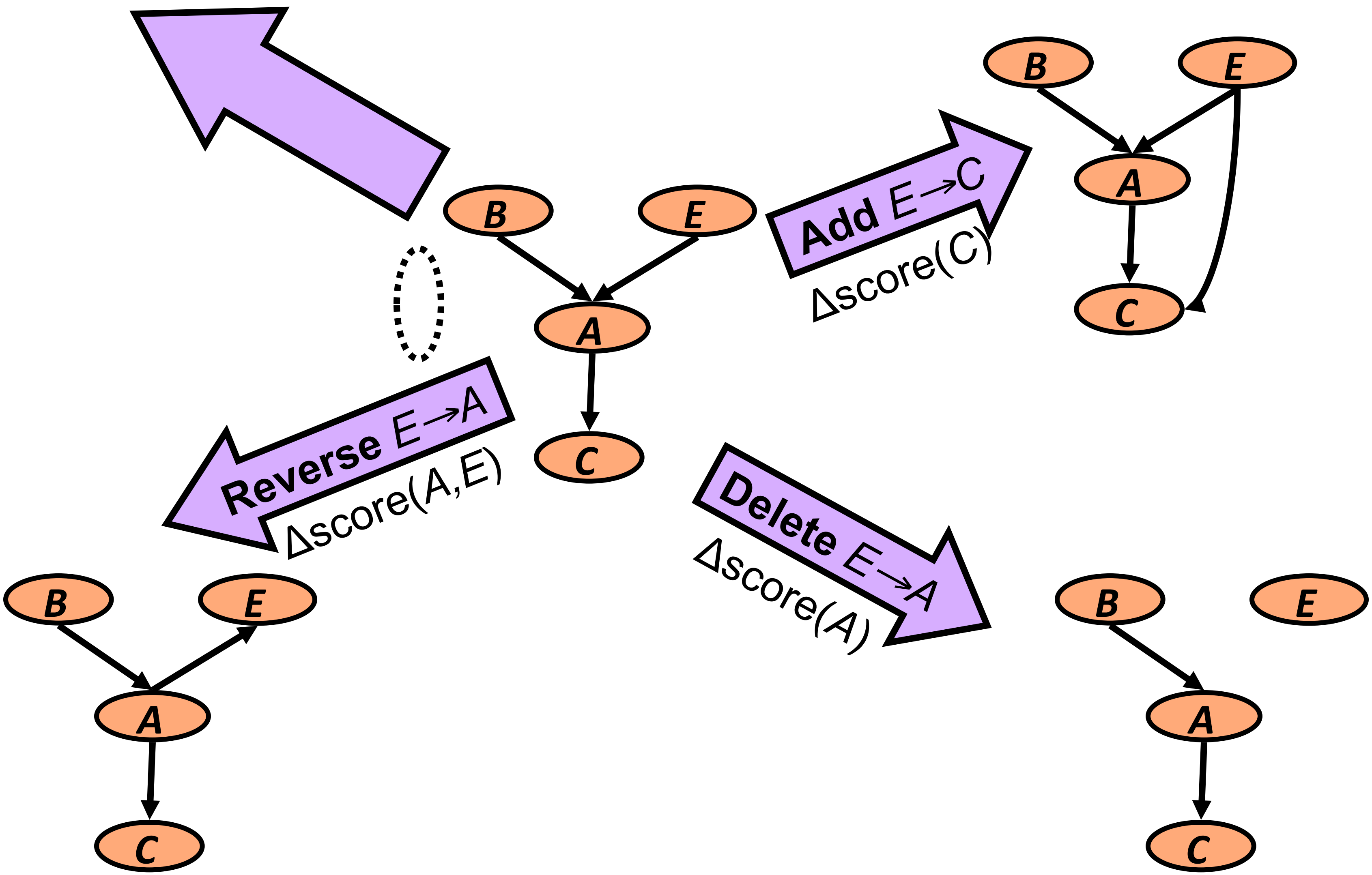
Marginal likelihood ***Prior***

- 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

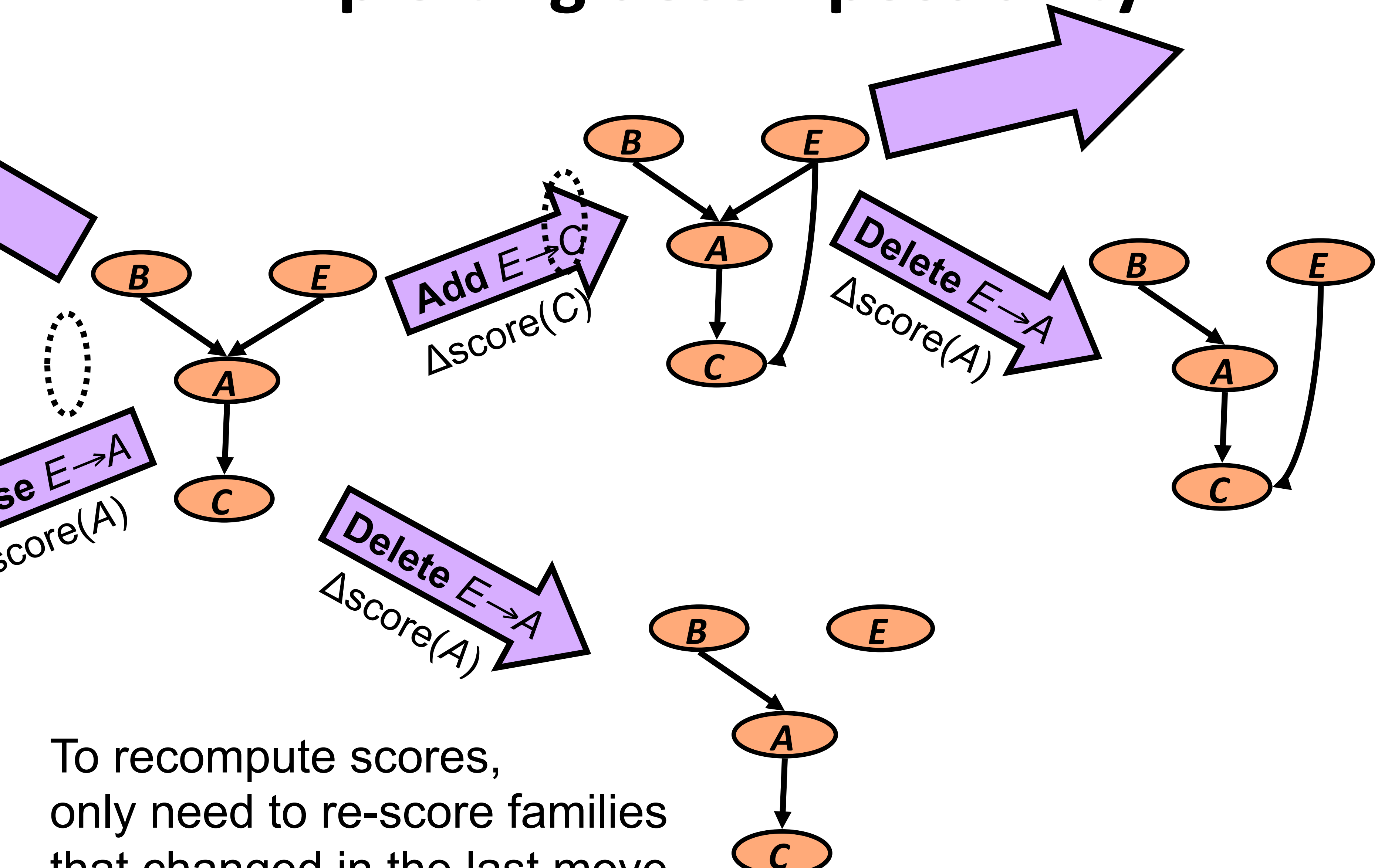
Same key property: Decomposability

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

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!