Bayesian Learning

Chapter 20.1-20.4

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

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 each feature F_i is *conditionally independent* of the other 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
  - The fix: Add one to every count (aka "Laplacian 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 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):
  - $P(X|D) = \sum_i P(X|D, h_i)\, P(h_i|D) = \sum_i P(X|h_i)\, P(h_i|D)$

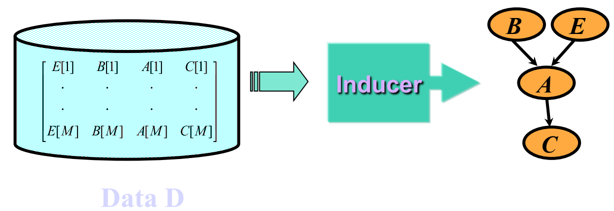
These are equal by our independence assumption

Bayesian learning

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

Learning Bayesian networks

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



Learning Bayesian Networks

- Describe a BN by specifying its (1) structure and (2) CPD tables
- Both can be learned from data, but
  - learning structure is much harder than learning parameters
  - learning when some nodes are hidden, or with missing data harder still
- We have 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 $\boldsymbol{u}$ of *Parents(X)*

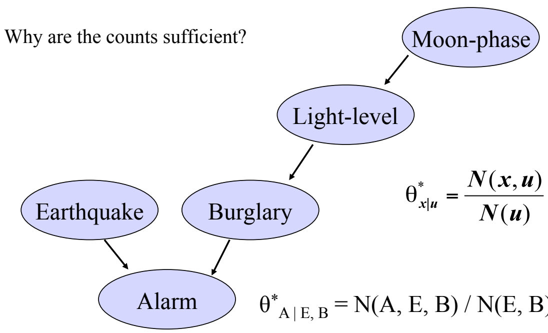
$$\theta^*_{x|\boldsymbol{u}} = \frac{N(\boldsymbol{x},\boldsymbol{u})}{N(\boldsymbol{u})}$$

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

- Why are the counts sufficient?



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

$\theta^*_{A\,|\,E,\,B}$ = 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

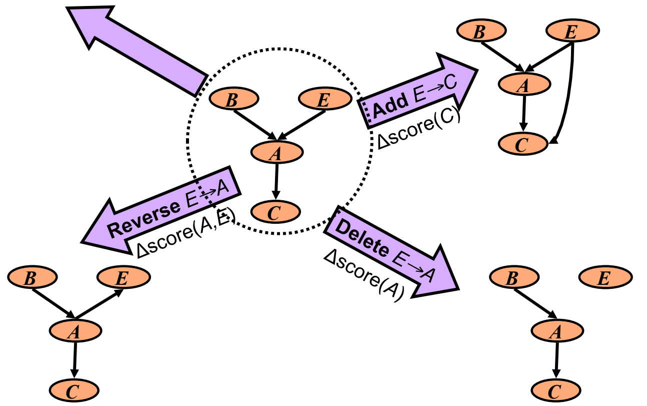
Structure selection: Scoring

- Bayesian: prior over parameters and structure
  - get balance between model complexity and fit to data as a byproduct
- Score (G:D) = log P(G|D) α log $[\underbrace{P(D|G)}_{\textit{Marginal likelihood}}\ \underbrace{P(G)}_{\textit{Prior}}]$
- Marginal likelihood just comes from our parameter estimates
- Prior on structure can be any measure we want; typically a function of the network complexity

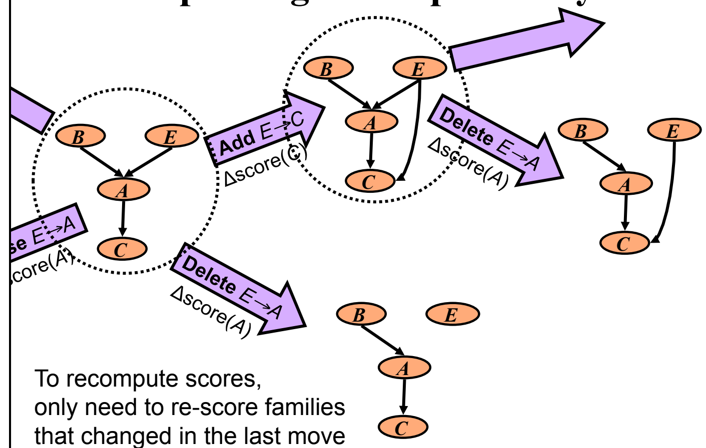
Same key property: Decomposability

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

Heuristic search



Exploiting decomposability



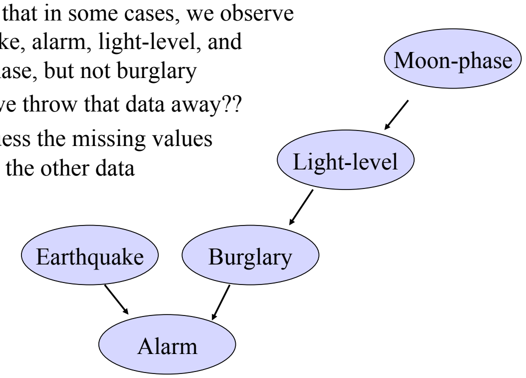
To recompute scores,
only need to re-score families
that changed in the last move

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!