Readings: K&F 18.3, 18.4, 18.5, 18.6

Structure Learning

Lecture 11 – May 2, 2011

CSE 515, Statistical Methods, Spring 2011

Instructor: Su-In Lee

University of Washington, Seattle

Last Time

- Score-based structure learning
  - Candidate structures; Score function; Search for the high-scoring structure
- Scoring functions
  - Maximum likelihood score
    - $\text{Score}_L(G:D)=\log P(D \mid G, \theta'_G)$ where θ'_G is MLE for G
    - Prone to overfitting
  - Bayesian score

Bayesian Score

$p(\theta)$ $p(\theta|D)$

- Main principle of the Bayesian approach $p(X[M+1] | D)$
  - Whenever we have uncertainty over anything, place a distribution over it.
  - What uncertainty? (G, Θ_G) $p(G)$ $p(\theta_G|G)$

Marginal likelihood

Prior over structures

θ_G

$$P(G | D) = \frac{P(D | G) P(G)}{P(D)}$$

Marginal probability of Data

P(D) does not depend on the network

$$\boxed{\text{Bayesian Score: } Score_B(G : D) = \log P(D | G) + \log P(G)}$$

3

Marginal Likelihood of Data Given G

$X_i \rightarrow X_j$

$$\boxed{\text{Bayesian Score: } Score_B(G : D) = \log P(D | G) + \log P(G)}$$

Likelihood

Prior over parameters

Marginal likelihood

$$\boxed{P(D | G)} = \int_{\theta_G} P(D | G, \theta_G) P(\theta_G | G) d\theta_G$$

$p(D | G, \hat{\theta}_G) = \max_{\theta_G} p(D | G, \theta_G) \rightarrow$ ML score

Note similarity to maximum likelihood score, but with the key difference that ML finds maximum of likelihood and here we compute average of the terms over parameter space

4

Marginal Likelihood: Binomial Case

- Assume a sequence of m coin tosses D
- By the chain rule for probabilities

X

$$P(x[1],\ldots,x[m]) = P(x[1])\cdot\ldots\cdot P(x[m] \mid x[1],\ldots,x[m-1])$$

Likelihood **Prior over parameters**

$$P(D \mid G) = \int_{\theta_G} P(D \mid G,\theta_G)P(\theta_G \mid G)d\theta_G$$

Marginal Likelihood: Binomial Case

- Assume a sequence of m coin tosses
- By the chain rule for probabilities P(x[1], x[2] ... x[?-1])

X

$$P(x[1],\ldots,x[m]) = P(x[1])\cdot\ldots\cdot P(x[m] \mid x[1],\ldots,x[m-1])$$

- Recall that for Dirichlet priors

$$P(x[m+1] = H \mid x[1],\ldots,x[m]) = \frac{M_H^m + \alpha_H}{m + \alpha_H + \alpha_T}$$

hyper-para Dirichlet.

  - Where M^m_H is number of heads in first m examples

$$P(x[1],\ldots,x[m]) = \frac{[\alpha_H\cdot\ldots\cdot(\alpha_H + M_H - 1)][\alpha_T\cdot\ldots\cdot(\alpha_T + M_T - 1)]}{\alpha\cdot\ldots\cdot(\alpha + M - 1)}$$

Marginal Likelihood: Binomial Case

$$P(x[1],\ldots,x[m]) = \frac{[\alpha_H \cdot\ldots\cdot(\alpha_H + M_H - 1)][\alpha_T \cdot\ldots\cdot(\alpha_T + M_T - 1)]}{\alpha\cdot\ldots\cdot(\alpha + M - 1)}$$

Simplify using $\Gamma(x+1)=x\Gamma(x)$ $\quad x = \frac{\Gamma(x+1)}{\Gamma(x)}$

$$(\alpha)(\alpha+1)\cdots(\alpha+M-1) = \frac{\Gamma(\alpha+M)}{\Gamma(\alpha)}$$

$$P(x[1],\ldots,x[m]) = \frac{\Gamma(\alpha)}{\Gamma(\alpha+M)}\cdot\left(\frac{\Gamma(\alpha_H+M_H)}{\Gamma(\alpha_H)}\right)\left(\frac{\Gamma(\alpha_T+M_T)}{\Gamma(\alpha_T)}\right)$$

x $\{H,T\}$

For multinomials with Dirichlet prior $\quad x$

$$P(x[1],\ldots,x[m]) = \frac{\Gamma(\alpha)}{\Gamma(\alpha+M)}\cdot\prod_{i=1}^{k}\frac{\Gamma(\alpha_i+M[x^i])}{\Gamma(\alpha_i)}$$

Marginal Likelihood: BayesNets

- Network structure determines form of marginal likelihood P(D|G)

$P(D|G_0)$

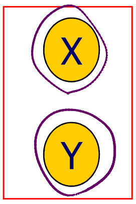
D	1	2	3	4	5	6	7
X	*H*	*T*	*T*	*H*	*T*	*H*	*H*
Y	*H*	*T*	*H*	*H*	*T*	*T*	*H*

Network 1: Two Dirichlet marginal likelihoods

P(X[1],…,X[7])

P(Y[1],…,Y[7])

Network G0



Marginal Likelihood: BayesNets

- Network structure determines form of marginal likelihood P(D|G)

D	1	2	3	4	5	6	7
X	H	T	T	H	T	H	H
Y	H	T	H	H	T	T	H

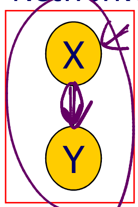
Network 2: Three Dirichlet marginal likelihoods

P(X[1],…,X[7])

P(Y[1]Y[4]Y[6]Y[7]) x=H

P(Y[2]Y[3]Y[5]) x=T

Network G1



Y	
H	T
θ_H	θ_T

$p(D|G_1) > p(D|G_0)$

X	Y: H	T
H	$\Theta_{y=H\|x=H}$	$\Theta_{y=T\|x=H}$
T	$\Theta_{y=H\|x=T}$	$\Theta_{y=T\|x=T}$

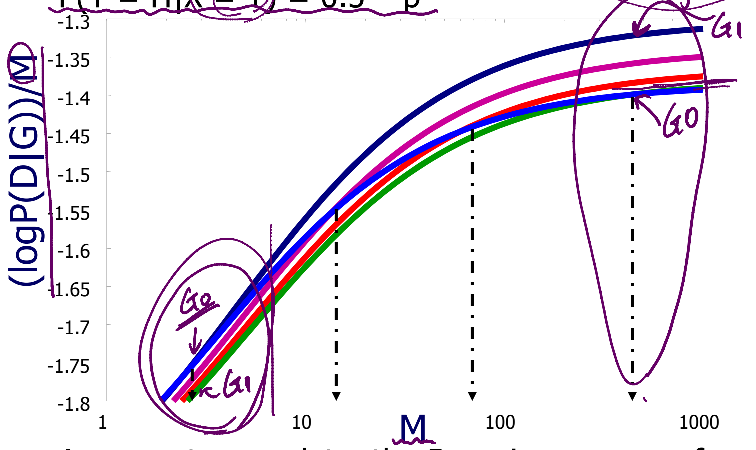
col 1

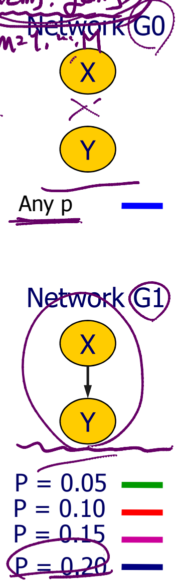
col 2

Idealized Experiment

D: <X[m], Y[m]> m=1,…,M

- P(X = H) = 0.5   X,Y P(Y=H) = 0.5   p=0: X⊥Y
- P(Y = H|X = H) = 0.5 + p |p|↑ : dependency ↑
  P(Y = H|X = T) = 0.5 – p





- As we get more data, the Bayesian score prefers G1 where X and Y are dependent.

Marginal Likelihood: BayesNets

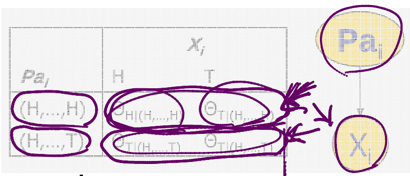
The marginal likelihood has the form:

"Decomposability of Bayesian Score"

$\int P(D \mid G, \theta_G) P(\theta_G \mid G) \, d\theta_G = \int_{\theta_G} \prod_i p(X_i \mid Pa_{X_i}, \theta_{X_i}) \prod_i P(\theta_{X_i} \mid G) \, d\theta_G = \prod_i \int p(X_i \mid Pa_{X_i}, \theta_{X_i}) p(\theta_{X_i} \mid G) \, d\theta_{X_1} \cdots d\theta_{X_n}$

log

$$P(D \mid G) = \prod_i \prod_{pa_i^G} \underbrace{\frac{\Gamma(\alpha_{pa_i^G})}{\Gamma(\alpha_{pa_i^G} + M[pa_i^G])} \prod_{x_i} \frac{\Gamma(\alpha_{x_i, pa_i^G} + M[x_i, pa_i^G])}{\Gamma(\alpha_{x_i, pa_i^G})}}$$



Dirichlet Marginal Likelihood
For the sequence of values of X_i when X_i's parents have a particular value

X_i

where

- *M(..)* are the counts from the data
- *α(..)* are hyperparameters for each family

Bayesian Score: Asymptotic Behavior

- For M→∞, a network G with Dirichlet priors satisfies

$$\log P(D \mid G) = l(\hat{\theta}_G : D) - \frac{\log M}{2} Dim(G) + O(1)$$

Dim(G): number of independent parameters in G

- Approximation is called BIC score

$$Score_{BIC}(G : D) = l(\hat{\theta}_G : D) - \frac{\log M}{2} Dim(G)$$

M

  - Score exhibits tradeoff between fit to data and complexity
  - Mutual information grows linearly with M while complexity grows logarithmically with M
    - As M grows, more emphasis is given to the fit to the data

Bayesian Score: Asymptotic Behavior

- For M→∞, a network G with Dirichlet priors satisfies

$$\log P(D \mid G) = l(\hat{\theta}_G : D) - \frac{\log M}{2} Dim(G) + O(1)$$

$$= M \sum_{i=1}^{n} \mathbf{I}_{\hat{P}}(X_i, Pa_{X_i}) - M \sum_{i=1}^{n} \mathbf{H}_{\hat{P}}(X_i) - \frac{\log M}{2} Dim(G) + O(1)$$

- Bayesian score is consistent
  - As M→∞, the true structure G* maximizes the score
    - Spurious edges will not contribute to likelihood and will be penalized
    - Required edges will be added due to linear growth of likelihood term relative to M compared to logarithmic growth of model complexity

Priors

Bayesian Score: $Score_B(G : D) = \log P(D \mid G) + \log P(G)$

M

$|G| \log c$

- Structure prior P(G)
  - Uniform prior: P(G) ∝ constant
  - Prior penalizing number of edges: P(G) ∝ $c^{|G|}$ (0<c<1)
  - Normalizing constant across networks is similar and can thus be ignored

#edges

Priors

Bayesian Score: $Score_B(G : D) = \log P(D \mid G) + \log P(G)$

max

$P(\theta_G \mid G)$ $\quad \int_{\theta_G} P(D \mid G, \theta_G) P(\theta_G \mid G)$

$\alpha_H + \alpha_T$

- Parameter prior P(θ|G)
  - BDe prior
    - M_0: equivalent sample size
    - B_0: prior network representing the prior probability of events G
    - Set $\alpha(x_i, pa_i^G) = M_0 P(x_i, pa_i^G \mid B_0)$ B_0
      - Note: pa_i^G may not the same as parents of X_i in B_0
      - Compute $P(x_i, pa_i^G \mid B_0)$ using standard inference in B_0
    - BDe requires assessing prior network B_0
      - Can naturally incorporate prior knowledge
    - BDe is consistent and asymptotically equivalent (up to a constant) to BIC

Summary: Network Scores

Pa_i

X_i

- **Decomposability**
  - Likelihood, BIC, (log) BDe have the form

$$Score(G : D) = \sum_i Score(X_i \mid Pa_i^G : D)$$

- All are **score-equivalent**
  - *G* I-equivalent to *G′* ⇒ *Score(G) = Score(G′)*

So far, we discussed scores for evaluating the quality of different candidate BN structures… Let's now examine how to find a structure with a high score.

STRUCTURE SEARCH

Optimization Problem

Input:

$\langle x_1[1], \dots, x_n[1] \rangle$

- Training data D = {**X**[1],…,**X**[M]}
- Scoring function (including priors, if needed)
- Set of possible structures (search space)
  - Including prior knowledge about structure

Output:

- A network (or networks) that maximize the score

Key Property:

- Decomposability: the score of a network is a sum of terms.

$$Score(G : D) = \sum_i Score(X_i \mid Pa_i^G : D)$$

Learning Trees

- Trees
  - At most one parent per variable
- Why trees?
  - Elegant math

    ⇒we can solve the optimization problem efficiently (with a greedy algorithm)
  - Sparse parameterization

    ⇒avoid overfitting while adapting to the data

Learning Trees

- *Let p(i)* denote parent of X_i, or 0 if X_i has no parent
- We can write the score as

$$Score(G:D) = \sum_i Score(X_i : Pa_i)$$

$$= \sum_{i:p(i)>0} Score(X_i : X_{p(i)}) + \sum_{i:p(i)=0} Score(X_i)$$

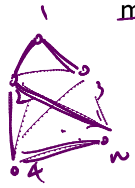
$$= \sum_{i:p(i)>0} \left(Score(X_i : X_{p(i)}) - Score(X_i)\right) + \sum_i Score(X_i)$$

$w(p(i)\to i)$

- Score = sum of edge scores + constant

Learning Trees

- Algorithm
  - Construct graph with vertices: 1,…,n
  - For all (i,j), set edge score $w(i\to j) = Score(X_j \mid X_i) - Score(X_j)$
  - If the score satisfies score equivalence, $w(i\to j) = w(j\to i)$
  - Structure learning problem: Find the tree structure with maximum sum of weights $\sum w(i\to j)$
    - Solve an undirected spanning tree (forest) problem and determine directions of edges afterwards.
    - This can be done using standard algorithms in low-order polynomial time by building a tree in a greedy fashion (e.g. Kruskal's maximum spanning tree algorithm)



- Theorem: Procedure finds the tree with maximal score (sum of $w(i\to j)$ for all edges $i\to j$)
- When score is likelihood, then $w(i\to j)$ is proportional to $I(X_i; X_j)$. This is known as the Chow & Liu method.

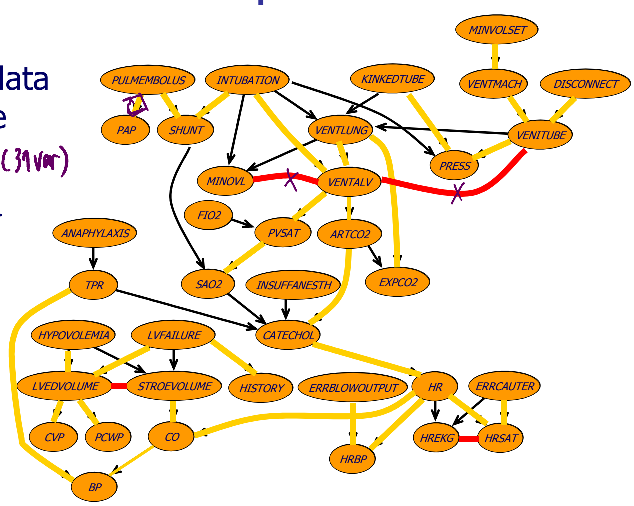
$p(X_1 \cdots X_n) \underset{KL}{\approx} \prod_i p(X_i \mid X_j)$

Learning Trees: Example

Tree learned from data D sampled from the ICU-Alarm network (39 var)

D = {**X**[1],…,**X**[M]}

↳ $\langle X_1 \cdots X_{39} \rangle$



Correct edges

Spurious edges

Not every edge in tree is in the original network
Tree direction is arbitrary --- we can't learn about arc direction

Beyond Trees

- Problem is not easy for more complex networks
  - Example: Allowing two parents, greedy algorithm is no longer guaranteed to find the optimal network
- Theorem:
  - Finding maximal scoring network structure with at most *k* parents for each variable is NP-hard for *k>1*
- In fact, no efficient algorithm exists

23

Fixed Ordering

- For any decomposable scoring function Score(G:D)

$$Score(G:D) = \sum_i Score(X_i \mid Pa_i^G : D)$$

and ordering α the maximal scoring network has:

If $X_i \in Pa_{X_j} \rightarrow X_i \prec X_j$ in α

$$Pa_i^G = \arg\max_{\mathbf{U} \subseteq \{X_j : X_j < X_i\}} Score(X_i \mid \mathbf{U}_i : D)$$

(since choice at X_i does not constrain other choices)

→ For fixed ordering, the structure learning problem becomes a set of independent problems of finding parents of X_i.

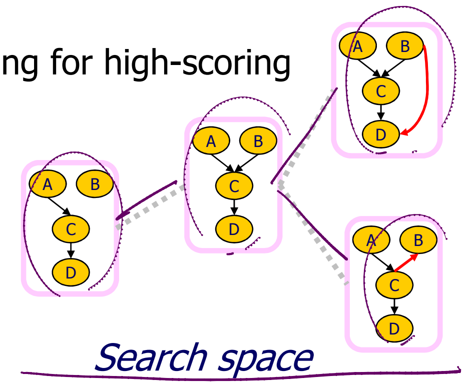
- If we bound the in-degree per variable by d, then complexity is exponential in d

24

Heuristic Search

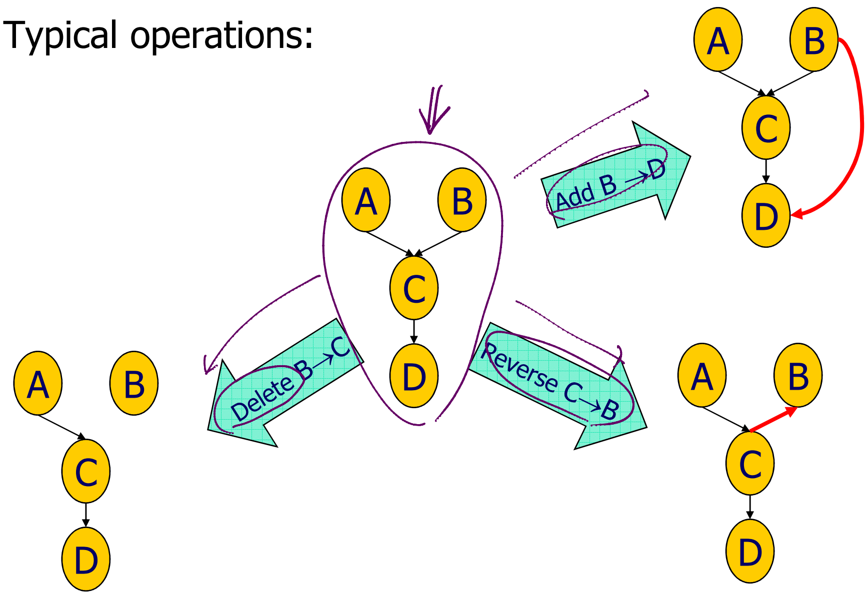
We address the problem by using heuristic search

- Define a search space:
  - nodes are possible structures
  - edges denote adjacency of structures
- Traverse this space looking for high-scoring structures
- Search techniques:
  - Greedy hill-climbing
  - Best first search
  - Simulated Annealing
  - ...

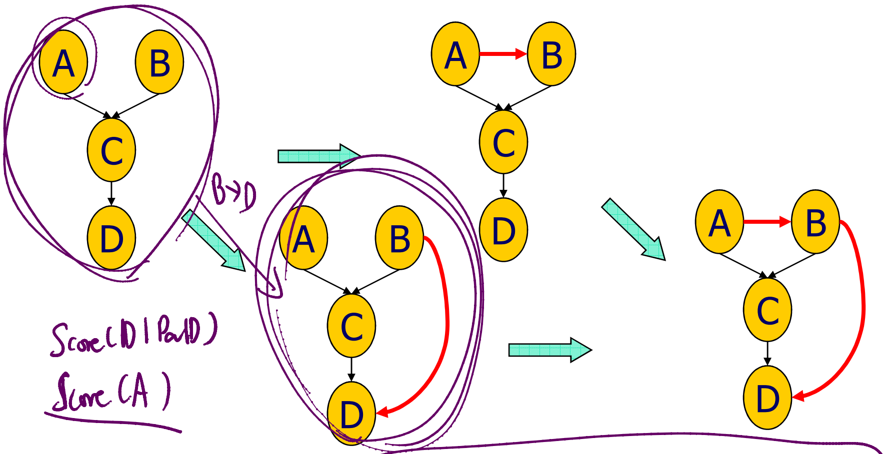


Heuristic Search

- Typical operations:



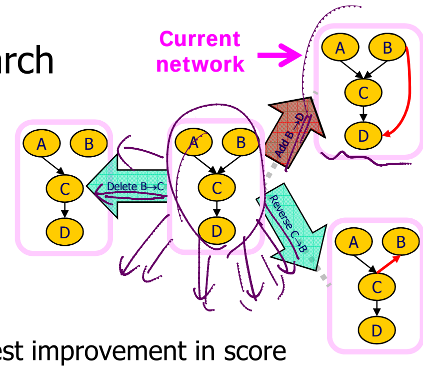
Exploiting Decomposability



- Decomposability: $Score(G:D) = \sum_i Score(X_i \mid Pa_i^G : D)$
- Caching: To update the score after a local change, we only need to re-score the families that were changed

Greedy Hill Climbing

- Simplest heuristic local search
  - Start with a given network
    - empty network
    - best tree (tree learning)
    - a random network
  - At each iteration
    - Evaluate all possible changes
    - Apply change that leads to best improvement in score
    - Reiterate
  - Stop when no modification improves score
- Each step requires evaluating $O(n^2)$ new changes



Greedy Hill Climbing Pitfalls

- Greedy Hill-Climbing can get stuck in:
  - Local Maxima
    - All one-edge changes reduce the score
  - Plateaus
    - Some one-edge changes leave the score unchanged
    - Happens because I-equivalent networks received the same score and are neighbors in the search space
- Both occur during structure search
- Standard heuristics can escape from both
  - Randomization and restart
  - TABU search: Keep a list of recent operators we applied, and in each step, we do not consider operators that reverse the effect of recently applied operators.

Model Selection

- So far, we focused on single model
  - Given D={**X**[1],…,**X**[M]}, find best scoring model

$$\tilde{G} = \arg\max_G P(G \mid D)$$

  - Use it to predict next example $P(\mathbf{X}[M+1] \mid D) \approx P(\mathbf{X}[M+1] \mid D, \tilde{G})$
- Implicit assumption
  - Making predictions based on the Bayesian estimation rule:

$$P(\mathbf{X}[M+1] \mid D) = \sum_G P(\mathbf{X}[M+1] \mid D, G) P(G \mid D)$$

  - Best scoring model dominates the weighted sum
    - Valid with many data instances (very large M)
- Pros:
  - We get a single structure
  - Allows for efficient use in our tasks
- Cons:
  - We are committing to the independencies of a particular structure
  - Other structures might be as probable given the data

Announcements

- Solution for PS #1 uploaded.
- Typo in Q5 of PS #2
  - Let C_i be some clique such that **Scope[ϕ']**…
  - 1 free late day for PS #2 (due 5/3 at noon; CSE536)
- PS #3 is ready (please pick it up).

Acknowledgement

- These lecture notes were generated based on the slides from Prof Eran Segal.