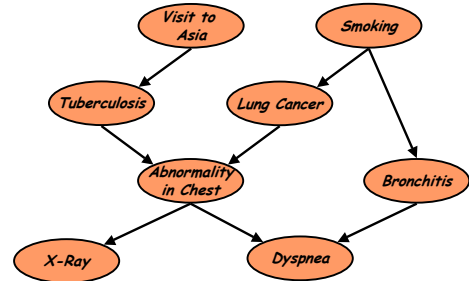


- Readings:
 - handouts
- Today's Lecture:
 - Inference
 - Learning BNs
 - slides courtesy of Nir Friedman
- Upcoming Dates:
 - Midterm Thursday

A More Complex Example

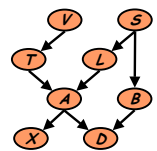
- "Asia" network:



Inference in Bayesian Networks

- We want to compute $P(d)$
- Need to eliminate: v, s, x, t, l, a, b

Initial factors



$$P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

Variable Elimination

General idea:

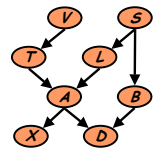
- Write query in the form

$$P(X_n, \mathbf{e}) = \sum_{x_k} \cdots \sum_{x_3} \sum_{x_2} \prod_i P(x_i | pa_i)$$

- Iteratively
 - Move all irrelevant terms outside of innermost sum
 - Perform innermost sum, getting a new term
 - Insert the new term into the product

- We want to compute $P(d)$
- Need to eliminate: v, s, x, t, l, a, b

Initial factors



$$P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

Eliminate: v

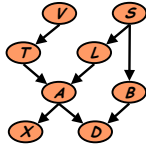
$$\text{Compute: } f_v(t) = \sum_v P(v)P(t|v)$$

$$\Rightarrow f_v(t)P(s)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

Note: $f_v(t) = P(t)$

In general, result of elimination is not necessarily a probability term

- We want to compute $P(d)$
- Need to eliminate: s, x, t, l, a, b
- Initial factors



$$P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)P(s)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

Eliminate: s

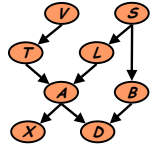
Compute: $f_s(b,l) = \sum_s P(s)P(b|s)P(l|s)$

$$\Rightarrow f_v(t)f_s(b,l)P(a|t,l)P(x|a)P(d|a,b)$$

Summing on s results in a factor with two arguments $f_s(b,l)$

In general, result of elimination may be a function of several variables

- We want to compute $P(d)$
- Need to eliminate: l, a, b
- Initial factors



$$P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)P(s)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)f_s(b,l)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)f_s(b,l)f_x(a)P(a|t,l)P(d|a,b)$$

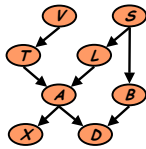
$$\Rightarrow f_s(b,l)f_x(a)f_l(a,l)P(d|a,b)$$

Eliminate: l

Compute: $f_l(a,b) = \sum_l f_s(b,l)f_l(a,l)$

$$\Rightarrow f_l(a,b)f_x(a)P(d|a,b)$$

- We want to compute $P(d)$
- Need to eliminate: x, t, l, a, b
- Initial factors



$$P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)P(s)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)f_s(b,l)P(a|t,l)P(x|a)P(d|a,b)$$

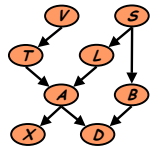
Eliminate: x

Compute: $f_x(a) = \sum_x P(x|a)$

$$\Rightarrow f_v(t)f_s(b,l)f_x(a)P(a|t,l)P(d|a,b)$$

Note: $f_x(a) = 1$ for all values of a !!

- We want to compute $P(d)$
- Need to eliminate: b
- Initial factors



$$P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)P(s)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)f_s(b,l)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)f_s(b,l)f_x(a)P(a|t,l)P(d|a,b)$$

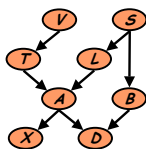
$$\Rightarrow f_s(b,l)f_x(a)f_l(a,l)P(d|a,b)$$

$$\Rightarrow f_l(a,b)f_x(a)P(d|a,b) \Rightarrow f_a(b,d) \Rightarrow f_b(d)$$

Eliminate: a, b

Compute: $f_a(b,d) = \sum_a f_l(a,b)f_x(a)P(d|a,b)$ $f_b(d) = \sum_b f_a(b,d)$

- We want to compute $P(d)$
- Need to eliminate: t, l, a, b
- Initial factors



$$P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)P(s)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)f_s(b,l)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)f_s(b,l)f_x(a)P(a|t,l)P(d|a,b)$$

Eliminate: t

Compute: $f_t(a,l) = \sum_t f_v(t)P(a|t,l)$

$$\Rightarrow f_s(b,l)f_x(a)f_t(a,l)P(d|a,b)$$

Variable Elimination

- We now understand variable elimination as a sequence of **rewriting** operations
- Actual computation is done in elimination step
- Exactly the same computation procedure applies to Markov networks
- Computation depends on order of elimination

Complexity of variable elimination

- Suppose in one elimination step we compute

$$f'_x(y_1, \dots, y_k) = \sum_x f_x(x, y_1, \dots, y_k)$$

$$f'_x(x, y_1, \dots, y_k) = \prod_{i=1}^m f_{i,j}(x, y_{1,1}, \dots, y_{1,i})$$

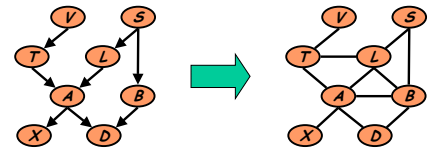
This requires

- $m \cdot |\text{Val}(X)| \cdot \prod_i |\text{Val}(Y_i)|$ multiplications
 - For each value for $x, y_1, \dots, y_m$ we do m multiplications
- $|\text{Val}(X)| \cdot \prod_i |\text{Val}(Y_i)|$ additions
 - For each value of $y_1, \dots, y_m$, we do $|\text{Val}(X)|$ additions

Complexity is exponential in number of variables in the intermediate factor!

Chordal Graphs

- elimination ordering $\Rightarrow$ undirected chordal graph



Graph:

- Maximal cliques are factors in elimination
- Factors in elimination are cliques in the graph
- Complexity is exponential in size of the largest clique in graph

Understanding Variable Elimination

- We want to select "good" elimination orderings that reduce complexity
- We start by attempting to understand variable elimination via the graph we are working with
- This will reduce the problem of finding good ordering to graph-theoretic operation that is well-understood

Induced Width

- The size of the largest clique in the induced graph is thus an indicator for the complexity of variable elimination
- This quantity is called the **induced width** of a graph according to the specified ordering
- Finding a good ordering for a graph is equivalent to finding the minimal induced width of the graph

Undirected graph representation

- At each stage of the procedure, we have an algebraic term that we need to evaluate
- In general this term is of the form:

$$P(x_1, \dots, x_k) = \sum_{y_1} \dots \sum_{y_m} \prod_i f_i(z_i)$$

where z_i are sets of variables

- We now plot a graph where there is undirected edge $X-Y$ if X, Y are arguments of some factor
 - that is, if X, Y are in some z_i
- Note: this is the Markov network that describes the probability on the variables we did not eliminate yet

General Networks

- From graph theory:

Thm:

- Finding an ordering that minimizes the induced width is NP-Hard

However,

- There are reasonable heuristic for finding "relatively" good ordering
- There are provable approximations to the best induced width
- If the graph has a small induced width, there are algorithms that find it in polynomial time

Elimination on Trees

- Formally, for any tree, there is an elimination ordering with induced width = 1

Thm

- Inference on trees is linear in number of variables

Stochastic simulation

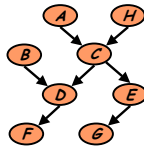
- Suppose you are given values for some subset of the variables, G , and want to infer values for unknown variables, U
- Randomly generate a very large number of instantiations from the BN
 - Generate instantiations for **all** variables – start at root variables and work your way “forward”
- Only keep those instantiations that are consistent with the values for G
- Use the frequency of values for U to get estimated probabilities
- Accuracy of the results depends on the size of the sample (asymptotically approaches exact results)

PolyTrees

- A polytree is a network where there is at most one path from one variable to another

Thm:

- Inference in a polytree is linear in the representation size of the network
 - This assumes tabular CPT representation



Markov chain Monte Carlo methods

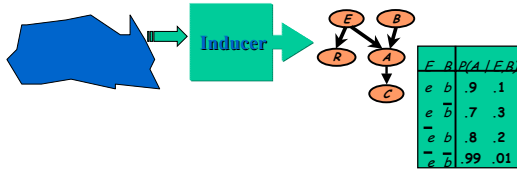
- So called because
 - Markov chain – each instance generated in the sample is dependent on the previous instance
 - Monte Carlo – statistical sampling method
- Perform a random walk through variable assignment space, collecting statistics as you go
 - Start with a random instantiation, consistent with evidence variables
 - At each step, for some nonevidence variable, randomly sample its value, consistent with the other current assignments
- Given enough samples, MCMC gives an accurate estimate of the true distribution of values

Approaches to inference

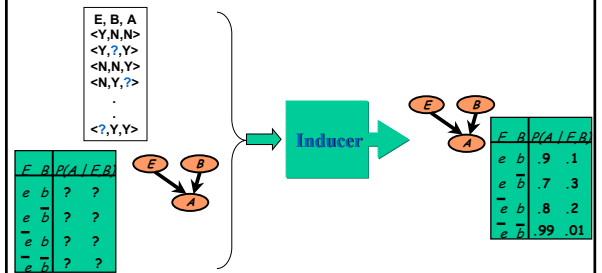
- Exact inference
 - Inference in Simple Chains
 - Variable elimination
 - Clustering / join tree algorithms
- Approximate inference**
 - Stochastic simulation / sampling methods
 - Markov chain Monte Carlo methods
 - Mean field theory

Learning Bayesian Networks

Learning Bayesian networks

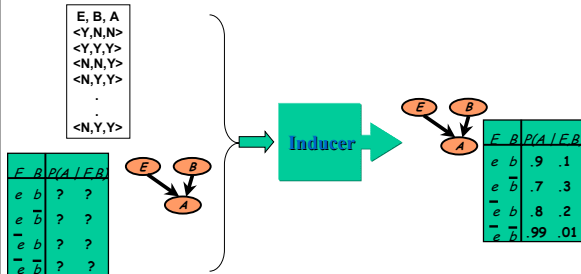


Known Structure -- Incomplete Data



- Network structure is specified
- Data contains missing values
 - We consider assignments to missing values

Known Structure -- Complete Data



- Network structure is specified
 - Inducer needs to estimate parameters
- Data does not contain missing values

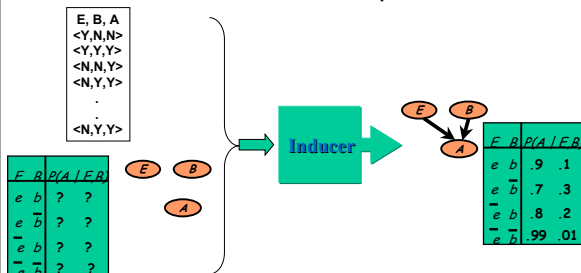
Known Structure / Complete Data

- Given a network structure G
 - And choice of parametric family for $P(X_i|Pa_i)$
- Learn parameters for network

Goal

- Construct a network that is "closest" to probability that generated the data

Unknown Structure -- Complete Data

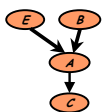


- Network structure is not specified
 - Inducer needs to select arcs & estimate parameters
- Data does not contain missing values

Learning Parameters for a Bayesian Network

- Training data has the form:

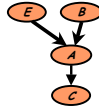
$$D = \begin{bmatrix} E[1] & B[1] & A[1] & C[1] \\ \vdots & \vdots & \vdots & \vdots \\ E[M] & B[M] & A[M] & C[M] \end{bmatrix}$$



Learning Parameters for a Bayesian Network

- Since we assume i.i.d. samples, likelihood function is

$$\mathcal{L}(\Theta : D) = \prod_m P(E[m], B[m], A[m], C[m] : \Theta)$$



General Bayesian Networks

Generalizing for any Bayesian network:

$$\begin{aligned} \mathcal{L}(\Theta : D) &= \prod_m P(x_1[m], \dots, x_n[m] : \Theta) \\ &= \prod_m \prod_i P(x_i[m] | \text{pa}_i[m] : \Theta_i) \\ &= \prod_i \prod_m P(x_i[m] | \text{pa}_i[m] : \Theta_i) \\ &= \prod_i \mathcal{L}_i(\Theta_i : D) \end{aligned}$$

i.i.d. samples

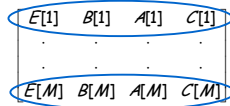
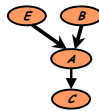
Network factorization

- The likelihood **decomposes** according to the structure of the network.

Learning Parameters for a Bayesian Network

- By definition of network, we get

$$\begin{aligned} \mathcal{L}(\Theta : D) &= \prod_m P(E[m], B[m], A[m], C[m] : \Theta) \\ &= \prod_m P(E[m] : \Theta) P(B[m] : \Theta) \\ &\quad P(A[m] | B[m], E[m] : \Theta) P(C[m] | A[m] : \Theta) \end{aligned}$$



General Bayesian Networks (Cont.)

Decomposition

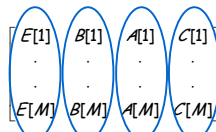
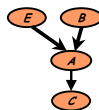
⇒ Independent Estimation Problems

If the parameters for each family are not related, then they can be estimated independently of each other.

Learning Parameters for a Bayesian Network

- Rewriting terms, we get

$$\begin{aligned} \mathcal{L}(\Theta : D) &= \prod_m P(E[m], B[m], A[m], C[m] : \Theta) \\ &= \prod_m P(E[m] : \Theta) P(B[m] : \Theta) \\ &\quad P(A[m] | B[m], E[m] : \Theta) P(C[m] | A[m] : \Theta) \end{aligned}$$



From Binomial to Multinomial

- For example, suppose X can have the values $1, 2, \dots, K$
- We want to learn the parameters $\theta_1, \theta_2, \dots, \theta_K$

Sufficient statistics:

- $N_1, N_2, \dots, N_K$ - the number of times each outcome is observed

Likelihood function:

MLE:

$$\mathcal{L}(\theta : D) = \prod_{k=1}^K \theta_k^{N_k}$$

$$\hat{\theta}_k = \frac{N_k}{\sum_{\ell} N_{\ell}}$$

Likelihood for Multinomial Networks

- When we assume that $P(X_i / Pa_i)$ is multinomial, we get further decomposition:

$$\begin{aligned} \mathcal{L}(\Theta_i : D) &= \prod_m P(x_i[m] | Pa_i[m] : \Theta_i) \\ &= \prod_{pa_i, m, Pa_i[m]=pa_i} P(x_i[m] | pa_i : \Theta_i) \\ &= \prod_{pa_i} \prod_{x_i} P(x_i | pa_i : \Theta_i)^{N(x_i, pa_i)} \\ &= \prod_{pa_i} \prod_{x_i} \theta_{x_i | pa_i}^{N(x_i, pa_i)} \end{aligned}$$

KL-Divergence

- Let P and Q be two distributions over $\mathcal{X}$
- A measure of distance between P and Q is the Kullback-Leibler Divergence

$$KL(P || Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$
- $KL(P || Q) = 1$ (when logs are in base 2) =
 - The probability P assigns to an instance is, on average, half the probability Q assigns to it
- $KL(P || Q) \geq 0$
- $KL(P || Q) = 0$ iff P and Q are equal

Likelihood for Multinomial Networks

- When we assume that $P(X_i / Pa_i)$ is multinomial, we get further decomposition:

$$\mathcal{L}(\Theta_i : D) = \prod_{pa_i} \prod_{x_i} \theta_{x_i | pa_i}^{N(x_i, pa_i)}$$

- For each value pa_i of the parents of X_i we get an independent multinomial problem
- The MLE is

$$\hat{\theta}_{x_i | pa_i} = \frac{N(x_i, pa_i)}{N(pa_i)}$$

Consistency

- Let $P(X / \theta)$ be a parametric family
 - We need to make various regularity condition we won't go into now
- Let $P^*(X)$ be the distribution that generates the data
- Let $\hat{\theta}_D$ be the MLE estimate given a dataset D

Thm

- As $N \rightarrow \infty$, $\hat{\theta}_D \rightarrow \theta^*$ where

with probability 1

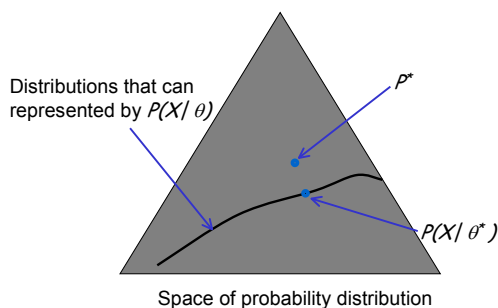
$$\theta^* = \arg \min_{\theta} KL(P^*(X) || P(X : \theta))$$

Maximum Likelihood Estimation

Consistency

- Estimate converges to best possible value as the number of examples grow
- To make this formal, we need to introduce some definitions

Consistency -- Geometric Interpretation



Bayesian Inference

Frequentist Approach:

- Assumes there is an unknown but fixed parameter θ
- Estimates θ with some confidence
- Prediction by using the estimated parameter value

Bayesian Approach:

- Represents uncertainty about the unknown parameter
- Uses probability to quantify this uncertainty:
 - Unknown parameters as **random variables**
- Prediction follows from the rules of probability:
 - Expectation over the unknown parameters

Dirichlet Priors

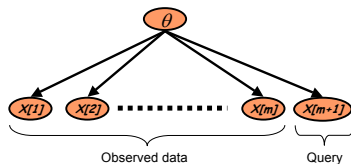
- Recall that the likelihood function is $L(\Theta : D) = \prod_{k=1}^K \theta_k^{N_k}$
- A **Dirichlet** prior with hyperparameters $\alpha_1, \dots, \alpha_K$ is defined as $P(\Theta) \propto \prod_{k=1}^K \theta_k^{\alpha_k - 1}$ for legal $\theta_1, \dots, \theta_K$

Then the posterior has the same form, with hyperparameters $\alpha_1 + N_1, \dots, \alpha_K + N_K$

$$P(\Theta | D) \propto P(\Theta) L(D | \Theta) \propto \prod_{k=1}^K \theta_k^{\alpha_k - 1} \prod_{k=1}^K \theta_k^{N_k} = \prod_{k=1}^K \theta_k^{\alpha_k + N_k - 1}$$

Bayesian Inference (cont.)

- We can represent our uncertainty about the sampling process using a Bayesian network



- The values of X are independent given θ
- The conditional probabilities, $P(x[m] | \theta)$, are the parameters in the model
- Prediction is now inference in this network

Dirichlet Priors (cont.)

- We can compute the prediction on a new event in closed form:
- If $P(\Theta)$ is Dirichlet with hyperparameters $\alpha_1, \dots, \alpha_K$ then

$$P(X[1] = k) = \int \theta_k P(\Theta) d\Theta = \frac{\alpha_k}{\sum_i \alpha_i}$$

- Since the posterior is also Dirichlet, we get

$$P(X[M+1] = k | D) = \int \theta_k P(\Theta | D) d\Theta = \frac{\alpha_k + N_k}{\sum_i (\alpha_i + N_i)}$$

Bayesian Inference (cont.)

Prediction as **inference** in this network

$$\begin{aligned} P(x[M+1] | x[1], \dots, x[M]) &= \int P(x[M+1] | \theta, x[1], \dots, x[M]) P(\theta | x[1], \dots, x[M]) d\theta \\ &= \int P(x[M+1] | \theta) P(\theta | x[1], \dots, x[M]) d\theta \end{aligned}$$

where

$$P(\theta | x[1], \dots, x[M]) = \frac{P(x[1], \dots, x[M] | \theta) P(\theta)}{P(x[1], \dots, x[M])}$$

where $P(x[1], \dots, x[M] | \theta) P(\theta)$ is labeled 'Probability of data' and $P(\theta | x[1], \dots, x[M])$ is labeled 'Posterior'.

Prior Knowledge

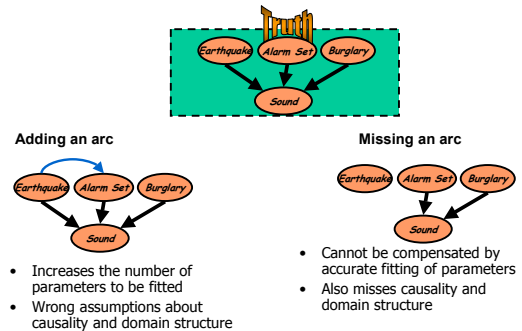
- The hyperparameters $\alpha_1, \dots, \alpha_K$ can be thought of as "imaginary" counts from our prior experience
- Equivalent sample size = $\alpha_1 + \dots + \alpha_K$
- The larger the **equivalent sample size** the more confident we are in our prior

Bayesian Prediction(cont.)

- Given these observations, we can compute the posterior for each multinomial $\theta_{x_i | pa_i}$ independently
 - The posterior is Dirichlet with parameters $\alpha(X_i=1|pa_i)+N(X_i=1|pa_i), \dots, \alpha(X_i=k|pa_i)+N(X_i=k|pa_i)$
- The predictive distribution is then represented by the parameters

$$\tilde{\theta}_{x_i | pa_i} = \frac{\alpha(x_i, pa_i) + N(x_i, pa_i)}{\alpha(pa_i) + N(pa_i)}$$

Why Struggle for Accurate Structure?



Learning Parameters: Summary

- Estimation relies on **sufficient statistics**
 - For multinomial these are of the form $N(x_i, pa_i)$
 - Parameter estimation
- $$\hat{\theta}_{x_i | pa_i} = \frac{N(x_i, pa_i)}{N(pa_i)} \quad \tilde{\theta}_{x_i | pa_i} = \frac{\alpha(x_i, pa_i) + N(x_i, pa_i)}{\alpha(pa_i) + N(pa_i)}$$
- MLE
- Bayesian (Dirichlet)
- Bayesian methods also require choice of priors
 - Both MLE and Bayesian are asymptotically equivalent and consistent
 - Both can be implemented in an **on-line** manner by accumulating sufficient statistics

Approaches to Learning Structure

- Constraint based**
 - Perform tests of conditional independence
 - Search for a network that is consistent with the observed dependencies and independencies
- Pros & Cons**
 - + Intuitive, follows closely the construction of BNs
 - + Separates structure learning from the form of the independence tests
 - Sensitive to errors in individual tests

Learning Structure from Complete Data

Approaches to Learning Structure

- Score based**
 - Define a score that evaluates how well the (in)dependencies in a structure match the observations
 - Search for a structure that maximizes the score
- Pros & Cons**
 - + Statistically motivated
 - + Can make compromises
 - + Takes the structure of conditional probabilities into account
 - Computationally hard

Likelihood Score for Structures

First cut approach:

- Use likelihood function

- Recall, the likelihood score for a network structure and parameters is

$$L(G, \Theta_G : D) = \prod_m P(x_1[m], \dots, x_n[m] : G, \Theta_G) \\ = \prod_m \prod_i P(x_i[m] | \text{Pa}_i^G[m] : G, \Theta_{G,i})$$

- Since we know how to maximize parameters from now we assume

$$L(G : D) = \max_{\Theta_G} L(G, \Theta_G : D)$$

Posterior Score

Using Bayes rule:

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

Diagram labels: Marginal likelihood (points to $P(D | G)$), Prior over structures (points to $P(G)$), Probability of Data (points to $P(D)$)

$P(D)$ is the same for all structures G
Can be ignored when comparing structures

Avoiding Overfitting

“Classic” issue in learning.

Approaches:

- **Restricting the hypotheses space**
 - Limits the overfitting capability of the learner
 - Example: restrict # of parents or # of parameters
- **Minimum description length**
 - Description length measures complexity
 - Prefer models that compactly describes the training data
- **Bayesian methods**
 - Average over all possible parameter values
 - Use prior knowledge

Marginal Likelihood

- By introduction of variables, we have that

$$P(D | G) = \int P(D | G, \theta) P(\theta | G) d\theta$$

Diagram labels: Likelihood (points to $P(D | G, \theta)$), Prior over parameters (points to $P(\theta | G)$)

- This integral measures sensitivity to choice of parameters

Bayesian Inference

- Bayesian Reasoning---compute expectation over unknown G

$$P(x[M+1] | D) = \sum_G P(x[M+1] | D, G) P(G | D)$$

- **Assumption:** G s are mutually exclusive and exhaustive
- We know how to compute $P(x[M+1] | G, D)$
 - Same as prediction with fixed structure
- How do we compute $P(G | D)$?

Marginal Likelihood: Multinomials

The same argument generalizes to multinomials with Dirichlet prior

- $P(\theta)$ is Dirichlet with hyperparameters $\alpha_1, \dots, \alpha_K$
 - D is a dataset with sufficient statistics $N_1, \dots, N_K$
- Then

$$P(D) = \frac{\Gamma\left(\sum_i \alpha_i\right)}{\Gamma\left(\sum_i (\alpha_i + N_i)\right)} \prod_i \frac{\Gamma(\alpha_i + N_i)}{\Gamma(\alpha_i)}$$

Marginal Likelihood for General Network

The marginal likelihood has the form:

$$P(D | \mathcal{G}) = \prod_i \prod_{pa_i^{\mathcal{G}}} \underbrace{\frac{\Gamma(\alpha(pa_i^{\mathcal{G}}))}{\Gamma(\alpha(pa_i^{\mathcal{G}}) + N(pa_i^{\mathcal{G}}))}}_{\text{Dirichlet Marginal Likelihood}} \prod_{x_i} \frac{\Gamma(\alpha(x_i, pa_i^{\mathcal{G}}) + N(x_i, pa_i^{\mathcal{G}}))}{\Gamma(\alpha(x_i, pa_i^{\mathcal{G}}))}$$

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

where

- $N(.,.)$ are the counts from the data
- $\alpha(.,.)$ are the hyperparameters for each family **given**
 $\mathcal{G}$

BDe Score

Intuition: M_0 prior examples distributed by B_0

- Set $\alpha(x_i, pa_i^{\mathcal{G}}) = M_0 P(x_i, pa_i^{\mathcal{G}} | B_0)$
 - Note that $pa_i^{\mathcal{G}}$ are not the same as the parents of X_i in B_0
 - Compute $P(x_i, pa_i^{\mathcal{G}} | B_0)$ using standard inference procedures
- Such priors have desirable theoretical properties
 - Equivalent networks are assigned the same score

Priors

- We need: prior counts $\alpha(.,.)$ for each network structure $\mathcal{G}$
- This can be a formidable task
 - There are exponentially many structures...

Bayesian Score: Asymptotic Behavior

Theorem: If the prior $P(\theta | \mathcal{G})$ is “well-behaved”, then

$$\log P(D | \mathcal{G}) = I(\mathcal{G} : D) - \frac{\log M}{2} \dim(\mathcal{G}) + O(1)$$

BDe Score

Possible solution: The **BDe prior**

- Represent prior using two elements M_0, B_0
 - M_0 - **equivalent sample size**
 - B_0 - network representing the prior probability of events

Asymptotic Behavior: Consequences

$$\log P(D | \mathcal{G}) = I(\mathcal{G} : D) - \frac{\log M}{2} \dim(\mathcal{G}) + O(1)$$

- Bayesian score is **consistent**
 - As $M \rightarrow \infty$ the “true” structure $\mathcal{G}^*$ maximizes the score (almost surely)
 - For sufficiently large M , the maximal scoring structures are **equivalent** to $\mathcal{G}^*$
- Observed data eventually overrides prior information
 - Assuming that the prior assigns positive probability to all cases

Asymptotic Behavior

$$\text{Score}(\mathcal{G} : D) = l(\mathcal{G} : D) - \frac{\log M}{2} \dim(\mathcal{G})$$

- This score can also be justified by the **Minimal Description Length (MDL)** principle
- This equation explicitly shows the tradeoff between
 - Fitness to data --- likelihood term
 - Penalty for complexity --- regularization term

Difficulty

Theorem: Finding maximal scoring network structure with at most k parents for each variables is NP-hard for $k > 1$

Scores -- Summary

- Likelihood, MDL, (log) BDe have the form

$$\text{Score}(\mathcal{G} : D) = \sum_i \text{Score}(X_i | Pa_i^{\mathcal{G}} : N(X_i, Pa_i))$$
- BDe requires assessing prior network. It can naturally incorporate prior knowledge and previous experience
- BDe is consistent and asymptotically equivalent (up to a constant) to MDL
- All are **score-equivalent**
 - $\mathcal{G}$ equivalent to $\mathcal{G}' \Rightarrow \text{Score}(\mathcal{G}) = \text{Score}(\mathcal{G}')$

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
- ...

Optimization Problem

Input:

- Training data
- Scoring function (including priors, if needed)
- Set of possible structures
 - Including prior knowledge about structure

Output:

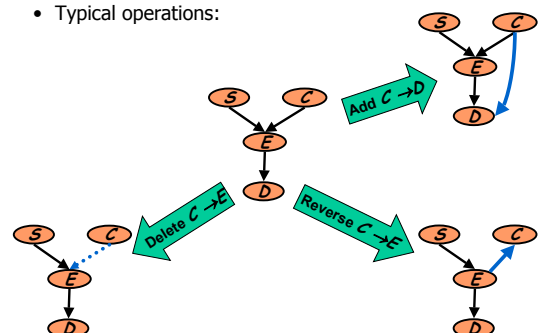
- A network (or networks) that maximize the score

Key Property:

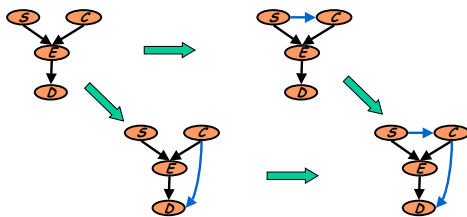
- **Decomposability:** the score of a network is a sum of terms.

Heuristic Search (cont.)

- Typical operations:



Exploiting Decomposability in Local Search



- **Caching:** To update the score of after a local change, we only need to re-score the families that were changed in the last move

Model Selection

- So far, we focused on single model
 - Find best scoring model
 - Use it to predict next example
- Implicit assumption:
 - Best scoring model dominates the weighted sum
- **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

Greedy Hill-Climbing

- Simplest heuristic local search
 - Start with a given network
 - empty network
 - best tree
 - 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 approximately n new changes

Model Averaging

- Recall, Bayesian analysis started with

$$P(x[M+1] | D) = \sum_{\mathcal{G}} P(x[M+1] | D, \mathcal{G}) P(\mathcal{G} | D)$$
 - This requires us to average over all possible models

Greedy Hill-Climbing: Possible 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 equivalent networks received the same score and are neighbors in the search space
- Both occur during structure search
- Standard heuristics can escape both
 - Random restarts
 - TABU search

Model Averaging (cont.)

- **Full Averaging**
 - Sum over all structures
 - Usually intractable—there are exponentially many structures
- **Approximate Averaging**
 - Find K largest scoring structures
 - Approximate the sum by averaging over their prediction
 - Weight of each structure determined by the **Bayes Factor**

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

The actual score we compute

Search: Summary

- Discrete optimization problem
- In general, NP-Hard
 - Need to resort to heuristic search
 - In practice, search is relatively fast (~100 vars in ~10 min):
 - Decomposability
 - Sufficient statistics
- In some cases, we can reduce the search problem to an easy optimization problem
 - Example: learning trees

References

- *Principles of Data Mining*, Hand, Mannila, Smyth. MIT Press, 2001.
- Nir Friedman's excellent lecture notes, <http://www.cs.huji.ac.il/~pmaj/>