Inference in Bayesian Networks

Variable Elimination
General idea:
• Write query in the form
\[ P(X_e) = \sum_{x_1} \cdots \sum_{x_k} \prod_{x \in E} P(x | \mathcal{pa}_x) \]

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

A More Complex Example

Visit to Asia
Smoking
Tuberculosis
Lung Cancer
Abnormality in Chest
Bronchitis
Dyspnea

• "Asia" network:

\[ P(v) P(s) P(t | v) P(l | s) P(b | s) P(a | t, l) P(x | a) P(d | a, b) \]

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

Initial factors

\[ P(v) P(s) P(t | v) P(l | s) P(b | s) P(a | t, l) P(x | a) P(d | a, b) \]

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: \( v \)
Compute:
\[ f_t(t) = \sum_v P(v) P(t | v) \]
\[ \Rightarrow f_t(t) P(s) P(l | s) P(b | s) P(a | t, l) P(x | a) P(d | a, b) \]

Note: \( f_t(t) = P(t) \)
In general, result of elimination is not necessarily a probability term
• We want to compute $P(d)$
• Need to eliminate: $x, t, l, a, b$

Initial factors

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

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

Eliminate: $s$

Compute: $f(t) = \sum f(s)P(b|s)P(t|s)$

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

Summing on $s$ results in a factor with two arguments $f(b,t)$

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

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

Initial factors

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

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

Eliminate: $a$

Compute: $f(a) = \sum f(b)P(d|a,b)$

$\Rightarrow f(a)P(d|a,b)$

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

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

Initial factors

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

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

Eliminate: $a$

Compute: $f(a) = \sum f(b)P(d|a,b)$

$\Rightarrow f(a)P(d|a,b)$

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

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
  \[ f'_x(y_1, \ldots, y_n) = \sum f(x, y_1, \ldots, y_n) \]
  \[ f'_x(x, y_1, \ldots, y_n) = \prod f(x, y_1, \ldots, y_n) \]
  This requires
  - \[ m \cdot |\text{Val}(X)| \cdot |\text{Val}(Y)| \] multiplications
    - For each value for \( x, y_1, \ldots, y_n \), we do \( m \) multiplications
  - \[ |\text{Val}(X)| \cdot |\text{Val}(Y)| \] additions
    - For each value of \( y_1, \ldots, y_n \), we do \( |\text{Val}(X)| \) additions

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

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

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:
  \[ P(x_1, \ldots, x_n) = \sum \prod f(Z_i) \]
  where \( Z_i \) are sets of variables
- We now plot a graph where there is an undirected edge \( X \rightarrow Y \) 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

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

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

General Networks

- From graph theory:
  \textbf{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

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

Stochastic simulation

- Suppose you are given values for some subset of the variables, G, and want to infer values for unknown variables, U

  **Thm**

  - Inference on trees is linear in number of variables

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

  **Thm:**

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

- 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

  **Thm:**

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

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

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

- **Known Structure -- Incomplete Data**
  - Network structure is specified
  - Data contains missing values
  - We consider assignments to missing values

- **Known Structure / Complete Data**
  - Given a network structure G
  - And choice of parametric family for P(Xi|Pa_i)
  - Learn parameters for network

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

- **Learning Parameters for a Bayesian Network**
  - Training data has the form:
    
    $$
    D = \begin{bmatrix}
    \vdots & \vdots & \vdots & \vdots \\
    \end{bmatrix}
    $$
Learning Parameters for a Bayesian Network

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

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

Learning Parameters for a Bayesian Network (Cont.)

- By definition of network, we get

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

  \[ P(E[m] : \Theta) \]
  \[ = \prod_{m} P(E[m] | B[m], A[m], C[m] : \Theta) \]

From Binomial to Multinomial

- For example, suppose \( X \) can have the values 1, 2, ..., \( K \)
- We want to learn the parameters \( \theta_1, \theta_2, \ldots, \theta_K \)

**Sufficient statistics:**

- \( N_1, N_2, \ldots, N_K \) - the number of times each outcome is observed

**Likelihood function:**

\[ \mathcal{L}(\Theta : D) = \prod_{k=1}^{K} \theta_k^{N_k} \]

**MLE:**

\[ \hat{\theta}_k = \frac{N_k}{\sum N_k} \]
Likelihood for Multinomial Networks

- When we assume that $P(X_i | P_{ai})$ is multinomial, we get further decomposition:

$$L(\Theta_i : D) = \prod_{m} P(x_i[m] | p_{ai}[m] : \Theta_i)$$

$$= \prod_{p_{ai}, \Theta_i} \prod_{m} P(x_i[m] | p_{ai}, \Theta_i)$$

$$= \prod_{p_{ai}, \Theta_i} \prod_{m, x_i} P(x_i | p_{ai}, \Theta_i)^{N(x_i, p_{ai})}$$

$$= \prod_{p_{ai}, x_i} \Theta = \prod_{p_{ai}, x_i} \Theta = \Theta = \Theta$$

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

KL-Divergence

- Let $P$ and $Q$ be two distributions over $X$

- 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

Consistency -- Geometric Interpretation

- Distributions that can be represented by $P(X | \theta)$

- Space of probability distribution

- $\theta^*$ is the MLE estimate given a dataset $D$
Bayesian Inference

**Frequentist Approach:**
- Assumes there is an unknown but fixed parameter \( \theta \)
- Estimates \( \theta \) 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

**Bayesian Inference (cont.)**
- We can represent our uncertainty about the sampling process using a Bayesian network

![Bayesian Network Diagram]

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

**Dirichlet Priors**
- Recall that the likelihood function is
  \[ L(\theta \mid D) = \prod_{k=1}^{K} \theta_k^{N_k} \]
- A Dirichlet prior with hyperparameters \( \alpha_1, \ldots, \alpha_K \) is defined as
  \[ \mathcal{P}(\theta) \propto \prod_{k=1}^{K} \theta_k^{\alpha_k-1} \]
  for legal \( \theta \), \( \theta \)

Then the posterior has the same form, with hyperparameters \( \alpha_1 + N_1, \ldots, \alpha_K + N_K \)

\[ \mathcal{P}(\theta \mid D) \propto \mathcal{P}(\theta) \mathcal{P}(D \mid \theta) \propto \prod_{k=1}^{K} \theta_k^{\alpha_k+N_k-1} \]

**Dirichlet Priors (cont.)**
- We can compute the prediction on a new event in closed form:
  - If \( \mathcal{P}(\theta) \) is Dirichlet with hyperparameters \( \alpha_1, \ldots, \alpha_K \)
    then
    \[ P(X[k+1]) = \int \mathcal{P}(\theta) \mathcal{P}(D \mid \theta) \theta \]
    \[ = \int \mathcal{P}(\theta) \mathcal{P}(D \mid \theta) \theta = \frac{\alpha_k}{\sum N_k} \]

- Since the posterior is also Dirichlet, we get
  \[ P(X[k+1] \mid D) = \frac{\alpha_k + N_k}{\sum_{k=1}^{K} (\alpha_k + N_k)} \]

**Prior Knowledge**
- The hyperparameters \( \alpha_1, \ldots, \alpha_K \) can be thought of as "imaginary" counts from our prior experience
- Equivalent sample size = \( \alpha_1 + \cdots + \alpha_K \)
- 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 $\theta_{x_i | p_{ai}}$ independently
  - The posterior is Dirichlet with parameters $\alpha_{(x_i = 1 | p_{ai})} + N(x_i = 1 | p_{ai})$, ..., $\alpha_{(x_i = k | p_{ai})} + N(x_i = k | p_{ai})$

- The predictive distribution is then represented by
  $$\tilde{\theta}_{x_i | p_{ai}} = \frac{\alpha(x_i, p_{ai}) + N(x_i, p_{ai})}{\alpha(p_{ai}) + N(p_{ai})}$$

Learning Parameters: Summary

- Estimation relies on **sufficient statistics**
  - For multinomial these are of the form $N(x_i, p_{ai})$
  - Parameter estimation
    $$\hat{\theta}_{x_i | p_{ai}} = \frac{N(x_i, p_{ai})}{N(p_{ai})}$$
    $$\tilde{\theta}_{x_i | p_{ai}} = \frac{\alpha(x_i, p_{ai}) + N(x_i, p_{ai})}{\alpha(p_{ai}) + N(p_{ai})}$$

- 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

Why Struggle for Accurate Structure?

- Increases the number of parameters to be fitted
- Wrong assumptions about causality and domain structure

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

- **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

Learning Structure from Complete Data
**Likelihood Score for Structures**

First cut approach:
- Use likelihood function

- Recall, the likelihood score for a network structure and parameters is
  \[ L(G, \theta : D) = \prod_x P(x[m], \ldots, x[m] : G, \theta) \]

- Since we know how to maximize parameters from now we assume
  \[ L(G : D) = \max_\theta L(G, \theta : D) \]

**Posterior Score**

Using Bayes rule:

\[ P(G | D) = \frac{P(D | G) P(G)}{P(D)} \]

- Prior over structures
- Marginal likelihood
- Probability of Data

\[ P(D) \text{ is the same for all structures } G \]

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

- 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 \( \alpha_1, \ldots, \alpha_K \)
- \( D \) is a dataset with sufficient statistics \( N_1, \ldots, N_K \)

Then

\[ P(D) = \frac{\Gamma \left( \sum \alpha_i \right)}{\prod \Gamma (\alpha_i + N)} \]
Marginal Likelihood for General Network

The marginal likelihood has the form:

\[ P(D | \mathcal{G}) = \prod_{\mathcal{P}} \left( \frac{I(\alpha(\mathcal{P}))}{I(\alpha(\mathcal{P})) + N(\mathcal{P})} \right) \prod_{\mathcal{X}} \left( \frac{I(\alpha(\mathcal{X}, \mathcal{P}))}{I(\alpha(\mathcal{X}, \mathcal{P})) + N(\mathcal{X}, \mathcal{P})} \right) \]

where

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

\[ \gamma_{i \mathcal{G}} \prod \gamma_{i \mathcal{G}} = \prod \gamma_{i \mathcal{G}} \]

Dirichlet Marginal Likelihood

For the sequence of values of \( X_i \) when \( X_i \)’s parents have a particular value

Priors

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

BDe Score

\[ \text{Intuition: } M_0 \text{ prior examples distributed by } B_0 \]

- Set \( \alpha(x, \mathcal{P}) = M_0 P(x, \mathcal{P} \mid \mathcal{B}_0) \)
  - Note that \( \mathcal{P} \) are not the same as the parents of \( X_i \) in \( \mathcal{B}_0 \)
  - Compute \( P(x, \mathcal{P} \mid \mathcal{B}_0) \) using standard inference procedures
- Such priors have desirable theoretical properties
  - Equivalent networks are assigned the same score

Bayesian Score: Asymptotic Behavior

\[ \log P(D \mid \mathcal{G}) = I(\mathcal{G} : D) - \frac{\log M}{2} \text{dim}(\mathcal{G}) + O(1) \]

Asymptotic Behavior: Consequences

\[ \log P(D \mid \mathcal{G}) = I(\mathcal{G} : D) - \frac{\log M}{2} \text{dim}(\mathcal{G}) + O(1) \]

- Bayesian score is consistent
  - As \( M \to \infty \), 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

\[ \text{Score}(G : D) = I(G : D) - \frac{\log M}{2} \text{dim}(G) \]

- 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

Scores -- Summary

- Likelihood, MDL, (log) BDe have the form
  \[ \text{Score}(G : D) = \sum_i \text{Score}(X_i \mid Pa_i^G : N(X_i, Pa_i)) \]
- 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
  - \( G \) equivalent to \( G' \) \( \Rightarrow \) Score(\( G \)) = Score(\( G' \))

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

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
- ...

Difficulty

Theorem: Finding maximal scoring network structure with at most \( k \) parents for each variables is NP-hard for \( k \geq 1 \)
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

---

**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 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: 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**

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

---

**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

\[
\frac{P(G | D)}{P(G | D)} \cdot \frac{P(G) P(D | G)}{P(G) P(D | G)} = \frac{P(D)}{P(D)}
\]

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

- Nir Friedman’s excellent lecture notes, http://www.cs.huji.ac.il/~pmail/