BAYESIAN NETWORKS STRUCTURE LEARNING

Xiannian Fan
Uncertainty Reasoning Lab (URL)
Department of Computer Science
Queens College/City University of New York
http://url.cs.qc.cuny.edu
Overview

- Introduction: Bayesian Networks
  - Bayesian Networks Structure Learning
  - Local Search Strategies
  - Optimal Search Strategies
  - Summary
Many research problems involve understanding the uncertain relations between a set of random variables

- Ignorance: Limited knowledge
- Laziness: Details ignored
A Bayesian network [Pearl 1988] is a **directed acyclic graph (DAG)** consisting of two parts:

- **The qualitative part**, encoding a domain's variables (nodes) and the probabilistic (usually causal) influences among them (arcs).

- **The quantitative part**, encoding a joint probability distribution over these variables.
Bayesian networks: Numerical parameters

Prior probability distributions for nodes without predecessors (Obesity, alcoholism, ...)

<table>
<thead>
<tr>
<th></th>
<th>Present</th>
<th>Absent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Present</td>
<td>0.15</td>
<td>0.85</td>
</tr>
</tbody>
</table>

Conditional probability distributions for nodes with predecessors (fatigue, jaundice, ...)

<table>
<thead>
<tr>
<th>Chronic hepatitis</th>
<th>Present</th>
<th>Absent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Present</td>
<td>0.6</td>
<td>0.2</td>
</tr>
<tr>
<td>Absent</td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>
Inference in Bayesian networks

Inference: Compute the probability of a hypothesis (e.g., a diagnosis) given new evidence (e.g., medical findings, test results).

- **Belief updating:**
  - $P(\text{Chronic hepatitis} \mid \text{Jaundice, Ascites})$?

- **Which disease is most likely?**

- **Maximum a Posteriori assignment (MAP)**
  - What is the most likely joint state of the diseases?
Why Learn Bayesian networks?

• Provide **compact** and **intuitive** graphical representations of the uncertain relations between the random variables

• Offer well-understood principles for solving many challenging tasks, e.g.
  – Incorporating prior knowledge with observational data
  – Handling missing data
  – Knowledge discovery from data
Applications of Bayesian networks

- Computer vision
- Natural language processing
- Computer security
- Bioinformatics
- Data mining
- User modeling
- Robotics

- Medical and machine diagnosis/prognosis
- Information retrieval
- Planning
- Fraud detection
- Planning
- ...

Introduction
BN Structure Learning
Local Search Strategies
Optimal Search Strategies
Summary
Learning Bayesian networks

- Very often we have data sets.
- We can extract knowledge from these data.

<table>
<thead>
<tr>
<th>VisitToAsia</th>
<th>Smoking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuberculosis</td>
<td>LungCancer</td>
</tr>
<tr>
<td>TuberDrCa.</td>
<td>Bronchitis</td>
</tr>
<tr>
<td>XRay</td>
<td>Dyspnea</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Success</th>
<th>Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Success</th>
<th>Success</th>
<th>Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good</td>
<td>0.4</td>
<td>0.1</td>
</tr>
<tr>
<td>Moderate</td>
<td>0.4</td>
<td>0.3</td>
</tr>
<tr>
<td>Poor</td>
<td>0.2</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Data:

| 100 100 100 90 | 390 | 97.5% |
| 100 95 100 80  | 375 | 93.8% |
| 100 100 100 90 | 390 | 97.5% |
| 80 95 100 90   | 365 | 91.3% |
| 100 100 100 100| 400 | 100.0%|
| 100 100 100 100| 400 | 100.0%|
| 90 95 100 90   | 375 | 93.8% |
| 90 95 100 90   | 375 | 93.8% |
| 100 100 100 90 | 390 | 97.5% |
| 100 100 100 100| 400 | 100.0%|
| 100 90 100 90  | 360 | 96.0% |
| 95 90 100 80   | 365 | 91.3% |
| 100 95 100 90  | 375 | 93.8% |
| 100 95 100 90  | 375 | 93.8% |
| 100 100 100 100| 400 | 100.0%|
Why struggle for accurate structure

• Missing an arc
  – Cannot be compensated by fitting parameters
  – Wrong assumptions about domain structure

• Adding an arc
  – Increases the number of parameters to be estimated
  – Wrong assumptions about domain structure
Major learning approaches

- **Score-based structure learning**
  - Find the highest structure network structure

- **Constraint-based structure learning**
  - Find a network that best explains the dependencies and independencies in the data

- **Hybrid approaches**
  - Integrated constraint and score based structure learning
Score-based learning

- Define scoring function that evaluates how well a structure matches the data

- Search for a structure that optimizes a given scoring function
Scoring function

- Minimum Description Length (MDL)
- Bayesian Dirichlet Family (BD)
- Factorized Normalized Maximum Likelihood (fNML)

MDL consistently outperforms other scoring functions in recovering the underlying Bayesian network structures.

[Liu, Malone, Yuan, BMC-2012]
Minimum Description Length (MDL)

MDL views learning as data compression.

MDL consists of two components:

- Model encoding

- Data encoding, using the model

\[
MDL(G) = \sum_i MDL(X_i | PA_i).
\]

\[
MDL(X_i | PA_i) = H(X_i | PA_i) + \frac{\log N}{2} K(X_i | PA_i).
\]

\[
K(X_i | PA_i) = (r_i - 1) \prod_{x_i \in PA_i} r_i
\]

Search for a structure that optimizes the score
Decomposability

- For all the scoring function we mentioned:
  - MDL: \( MDL(G) = \sum_i MDL(X_i | PA_i) \)
  - BDeu: \[ \sum_i \sum_j \log \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} + \sum_k \log \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})} \]
  - fNML: \[ \sum_i \sum_j \sum_k -N_{ijk} \log \frac{N_{ijk}}{N_{ij}} - C(r_i, N_{ij}) \]

All of these are expressed as a sum over the individual variables.

This property is called decomposability and will be quite important for structure learning.

[Heckerman1995, etc.]
Score-based learning

- Define scoring function that evaluates how well a structure matches the data

- Search for a structure that optimizes a given scoring function
Search space of DAGs

- Number of possible DAGs containing \( n \) nodes

\[
f(n) = \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} f(n-i)
\]

Curse of dimensionality

<table>
<thead>
<tr>
<th># of variables</th>
<th># of the possible DAGs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>8</td>
<td>78,370,2329,343</td>
</tr>
<tr>
<td>9</td>
<td>1,213,442,454,842,881</td>
</tr>
<tr>
<td>10</td>
<td>4,175,098,976,430,598,100</td>
</tr>
</tbody>
</table>
Search strategies

- **Local search strategies**
  - Hill Climbing
  - Max-Min Hill Climbing
  - Others, e.g., stochastic local search, genetic algorithm.

- **Optimal search strategies**
  - Branch-and-Bound
  - Dynamic Programming
  - Integer Linear Programming
  - Heuristic Search
Local search: Hill Climbing

- Start from an initial state
- Iteratively Update: Add, delete and reverse.
Local search: Max-Min Hill Climbing

First stage: Max-Min Parent Children (MMPC):
- use statistical test of conditional independence to find the probable sets of parents and children for each variable x in the domain.

Second stage: greedy local search
- search in the space of DAGs, but restricted to the skeleton identified by first stage MMPC.
Search strategies

- **Local search strategies**
  - Hill Climbing
  - Max-Min Hill Climbing
  - Others, e.g. stochastic local search, genetic algorithm.

- **Optimal search strategies**
  - Branch-and-Bound
  - Dynamic Programming
  - Integer Linear Programming
  - Heuristic Search
Optimal Search Strategies

• General form, for scores $s(x, PA)$

$$\sum_{x, PA} s(x, PA)$$

subject to: $PA$ is possible parents set of $x$ in an optimal BN

• Begin by calculating optimal parents sets for all variables

• Number of optimal parents sets $s(x, PA)$: $2^{n-1}$ scores for each variable

• Most scores can be pruned: not possible in any optimal BNs
  – Bounds on Parents Sets for MDL [Tian, UAI-00])
  – Properties of Decomposable Score Functions [de Campos and Ji, JMLR-11]
  – Sparse Parent Graph [Yuan and Malone, UAI-12]

[Yuan, Malone, UAI-12]
Input $s(\cdot,\cdot)$: Potential Optimal Parents Set

- Example: Potential Optimal Parent Set for $X_1$

Prune useless local scores.

(b) $s(X_1, PA)$

(c) $s(X_1, PA)$
Input: Potential Optimal Parents Set (POPS)

POPS for another 8-variable example

<table>
<thead>
<tr>
<th>variable</th>
<th>POPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>${X_2}$</td>
</tr>
<tr>
<td>$X_2$</td>
<td>${X_1}$</td>
</tr>
<tr>
<td>$X_3$</td>
<td>${X_1, X_2}$</td>
</tr>
<tr>
<td>$X_4$</td>
<td>${X_1, X_3}$</td>
</tr>
<tr>
<td>$X_5$</td>
<td>${X_4}$</td>
</tr>
<tr>
<td>$X_6$</td>
<td>${X_2, X_5}$</td>
</tr>
</tbody>
</table>

[Yuan, Malone, Wu, IJCAI-11]
Optimal search: Branch-and Bound

- Begin by calculating optimal parents sets for all variables.

- These sets are represented as a directed graph $G'$ that may have cycles.

<table>
<thead>
<tr>
<th>variable</th>
<th>POPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>${X_2}$ ${}$</td>
</tr>
<tr>
<td>$X_2$</td>
<td>${X_1}$ ${}$</td>
</tr>
<tr>
<td>$X_3$</td>
<td>${X_1, X_2}$ ${X_2, X_6}$ ${X_1, X_6}$ ${X_2}$ ${X_6}$ ${}$</td>
</tr>
<tr>
<td>$X_4$</td>
<td>${X_1, X_3}$ ${X_1}$ ${X_3}$ ${}$</td>
</tr>
<tr>
<td>$X_5$</td>
<td>${X_4}$ ${X_2}$ ${}$</td>
</tr>
<tr>
<td>$X_6$</td>
<td>${X_2, X_5}$ ${X_2}$ ${}$</td>
</tr>
</tbody>
</table>
Optimal search: Branch-and-Bound

- Search over all possible directed graph which is a sub-graph of $G$, by removing one edge at a time.
  - Use bounds to prune search space
  - Claim to be optimal [de Campos and Ji, JMLR-11]

Actually, not optimal
Optimal Search: Dynamic Programming

- Dynamic programming: optimal Bayesian network = a best leaf with optimal parents + an optimal subnetwork.

\[
\text{MDL}(\mathbf{V}) = \min_{X \in \mathbf{V}} \{ \text{MDL}(\mathbf{V} \setminus \{X\}) + \text{BestMDL}(X, \mathbf{V} \setminus \{X\}) \}
\]

\[
\text{BestMDL}(X, \mathbf{V} \setminus \{X\}) = \min_{PA_X \subseteq \mathbf{V} \setminus \{X\}} \text{MDL}(X | PA_X)
\]

[Singh & Moore 2005, Silander & Myllymaki 2006]
Formulate the learning task as an Integer Linear Programming (ILP) problem

- Encode any graph by creating a binary ILP variables $I(PA \rightarrow x)$ for each variable $x$ and each candidate parent set $PA$. $I(PA \rightarrow x) = 1$ if and only if $PA$ are the parents of $x$ in an optimal BN.
- Each $I(PA \rightarrow x)$ has a corresponding local score $s(x, PA)$.

<table>
<thead>
<tr>
<th>$I(PA_{1,1} \rightarrow X_1)$</th>
<th>$I(PA_{1,2} \rightarrow X_1)$</th>
<th>...</th>
<th>$I(PA_{1,k_1} \rightarrow X_1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I(PA_{2,1} \rightarrow X_2)$</td>
<td>$I(PA_{2,2} \rightarrow X_2)$</td>
<td>...</td>
<td>$I(PA_{2,k_2} \rightarrow X_2)$</td>
</tr>
<tr>
<td>$I(PA_{3,1} \rightarrow X_3)$</td>
<td>$I(PA_{3,2} \rightarrow X_1)$</td>
<td>...</td>
<td>$I(PA_{3,k_3} \rightarrow X_3)$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$I(PA_{n,1} \rightarrow X_n)$</td>
<td>$I(PA_{n,2} \rightarrow X_n)$</td>
<td>...</td>
<td>$I(PA_{n,k_n} \rightarrow X_n)$</td>
</tr>
</tbody>
</table>

Table 1. Coding for family variables

[Yuan, Malone, Wu, IJCAI-11]
Optimal Search: Integer Linear Programming

- **Learning formulation:**
  
  Instantiate the $I(PA \rightarrow X)$ to maximize
  
  \[
  \sum_{s,PA} s(x, PA) \cdot I(PA \rightarrow X)
  \]
  
  subject to all $I(PA \rightarrow X)$ represent a DAG.

- **Clusters Constraint:**
  
  $\sum_{PA} I(PA \rightarrow X) = 1$ for each $x \in V$
  
  $\sum_{x \in V} \sum_{PA : PA \cap C = \phi} s(x, PA) \cdot I(PA \rightarrow X) \geq 1$, where $C \subseteq V$
  
  any subset $C$ of nodes in a DAG must contains at least one node who has parent in that subset.

- **Learning method:** **GOBNILP** using Cutting Plane
  
  [Cussens, etc, UAI-2011]
Optimal Search: Heuristic Search

• Formulate the learning task as a **shortest path finding** problem
  – The shortest path solution to a graph search problem corresponds to an optimal Bayesian network

[Yuan, Malone, Wu, IJCAI-11]
Motivation

- Observation: *Every DAG has at least one leaf.*
Search graph (Order graph)

Formulation:
Search space: Variable subsets
Start node: Empty set
Goal node: Complete set
Edges: Select parents
Edge cost: $\text{BestMDL}(X, U)$ for edge $U \rightarrow U \cup \{X\}$

Task: find the shortest path between start and goal nodes

[Yuan, Malone, Wu, IJCAI-11]
**A* search:** Expands the nodes in the order of promisingness: \( f = g + h \)

\[
g(U) = MDL(U)
\]

\[
h(U) = \sum_{X \in V \setminus U} \text{BestMDL}(X, V \setminus \{X\})
\]

\[h(\{2,3\}):\]

[Yuan, Malone, Wu, IJCAI-11]
Properties of simple heuristic

• Theorem: The simple heuristic function $h$ is admissible
  – Optimistic estimation

• Theorem: $h$ is also consistent
  – Monotonic $f$ values
  – Consistency $\Rightarrow$ admissibility
BFBnB Algorithm

Breadth-first branch and bound search (BFBnB):

- **Motivation:** Exponential-size order&parent graphs
- **Observation:** Natural layered structure
- **Solution:** Search one layer at a time

[Malone, Yuan, Hansen, UAI-11]
BFBnB Algorithm

[Malone, Yuan, Hansen, UAI-11]
BFBnB Algorithm

[Malone, Yuan, Hansen, UAI-11]
BFBnB Algorithm

[Malone, Yuan, Hansen, UAI-11]
BFBnB Algorithm

[Malone, Yuan, Hansen, UAI-11]
Pruning in BFBnB

- For pruning, estimate an **upper bound** solution before search
  - Can be done using greedy local search
- Prune a node when $f$-cost > upper bound

[Malone, Yuan, Hansen, UAI-11]
Critiquing the simple heuristic

- Drawback of the simple heuristic
  - Let each variable to choose optimal parents from all the other variables
  - Completely relaxes the acyclic constraint

Bayesian network

Heuristic estimation

Relaxation
Potential solution

- Breaking the cycles to obtain tighter heuristic

BestMDL(1, \{2,3,4\})=\{2,3,4\}
+ BestMDL(2, \{1,3,4\})=\{1,4\}

\[
\min \Rightarrow c(\{1,2\})
\]

BestMDL(1, \{3,4\})=\{3,4\}
+ BestMDL(2, \{1,3,4\})

[Yuan, Malone, UAI-12]
**$k$-cycle conflict heuristic**

- **Compute costs for all 2-variable groups**
  - Each group, called a pattern, has a tighter score

  \[
  \begin{align*}
  &1,2 & 1,3 & 2,3 & 1,4 & 2,4 & 3,4 \\
  \end{align*}
  \]

- **Based on dynamically partitioned pattern database** [Felner et al. 2004]
  - Avoid cycles for all 3-variable groups
  - Avoid cycles for variable groups with size up to $k$

[Yuan, Malone, UAI-12]
Compute $k$-cycle conflict heuristic

- Compute the pattern database with a **backward breadth-first search** in the order graph for $k$ layers

\[
P1: \text{BestMDL}(1, \{2,3,4\}) + \text{BestMDL}(2, \{3,4\})
\]

\[
P2: \text{BestMDL}(2, \{1,3,4\}) + \text{BestMDL}(1, \{3,4\})
\]

Reverse $g$ cost:

\[
g^r(\{3,4\}) = \min(P1, P2)
\]

\[
g^r(\{3,4\}) = c(\{1,2\})!
\]

[Yuan, Malone, UAI-12]
Computing heuristic value using dynamic PD

- To compute the heuristic for a node, find non-overlapping patterns and sum their costs

\[
h(\{1\}) = c(\{2,3\}) + c(\{4\})
\]

\[
h(\{1\}) = c(\{2,4\}) + c(\{3\})
\]

\[
h(\{1\}) = c(\{3,4\}) + c(\{2\})
\]

- Can be formulated as a maximum-weight matching problem
  - \(O(N^3)\) for \(k=2\) [Papadimitriou&Steiglitz 1982]
  - NP-hard for \(k \geq 3\) [Garey&Johnson 1979]

- Greedy method
  - Order the patterns based on improvements
  - Greedily find the best available pattern at each step

[Yuan, Malone, UAI-12]
Statically partitioned pattern database

- Calculate **full** pattern databases for non-overlapping static groups: \(\{1,2,3,4,5,6\} \Rightarrow \{1,2,3\}, \{4,5,6\}\)
Computing heuristic value using static PD

- Sum costs of pattern databases according to static grouping

\[ h(\{1,5,6\}) = c(\{2,3\}) + c(\{4\}) = g^r(\{1\}) + g^r(\{5,6\}) \]

[Yuan, Malone, UAI-12]
Properties of $k$-cycle conflict heuristic

• Theorem: The $k$-cycle conflict heuristic is admissible
  – Dynamic version
  – Static version

• Theorem: The static $k$-cycle conflict heuristic is consistent

• Theorem: The dynamic $k$-cycle conflict heuristic is consistent
  – Given that we use an optimal maximum matching algorithm
  – Will lose the consistency property if using an approximation algorithm
Comparison of optimal search algorithms

- **Branch-and-Bound**
  - search all sub-graphs of the directed graph \( G' \) that may have cycles, getting from optimal parents sets. *(Not optimal solution).*
  - Prune by bound.

- **DP**
  - Search all DAGs with DP techniques

- **Integer Linear Programming (ILP)**
  - Number of family variables

- **Heuristic Search**
  - Search all subsets of variables
  - Tightness of heuristic function

Heuristic Search and ILP are much more efficient. [Yuan, Malone, JAIR-12]
Comparison of optimal search algorithms
Summary

• A survey on the problem of Bayesian networks structure learning.

• Topics covered
  – Basics of Bayesian networks
  – Scoring function
  – Local search strategies
    ▪ hill climbing
    ▪ mmhc
  – Optimal Search Strategies
    ▪ Branch-and-bound
    ▪ Dynamic programming
    ▪ Integer linear programming
    ▪ Heuristic search

• Efficient Packages/software
  – GOBNILP, using Integer linear programming
  – URLearning, using Heuristic search
Thank you