Machine Reasoning using Bayesian Network

Ching-Yung Lin, Ph.D.
Columbia University and Graphen, Inc.
February 28, 2020
Outline

- Introduction
- Probability Review
- Bayesian Network
- Inference Methods
- Network Structure Learning
Evolution of Intelligence

Direction of Evolution

- perception
- reasoning
- strategy
- recognition
- sensors
- representation
- memory
Introduction

Suppose the doctor is trying to determine if a patient has inhalational anthrax. She observes the following symptoms:

• The patient has a cough
• The patient has difficulty in breathing
• The patient has a fever
Introduction

Dealing with uncertainty:

You would like to determine how likely the patient is infected with inhalational anthrax given that the patient has a cough, a fever, and difficulty breathing
Introduction

New evidence: X-ray image shows that the patient has a wide mediastinum.

Belief update: your belief that the patient is infected with inhalational anthrax is now much higher now.
Introduction

• In the previous slides, what you observed affected your belief that the patient is infected with anthrax
• This is called reasoning with uncertainty
• Wouldn’t it be nice if we had some tools for reasoning with uncertainty? In fact, we do...
Bayesian Network

- Need a representation and reasoning system that is based on conditional independence
  - Compact yet expressive representation
  - Efficient reasoning procedures
- Bayesian Network is such a representation
  - Named after Thomas Bayes (ca. 1702 – 1761)
  - Term coined in 1985 by Judea Pearl (1936 – , 2011 winner of the ACM Turing Award)
  - Many applications, e.g., spam filtering, speech recognition, robotics, diagnostic systems and even syndromic surveillance
Outline

• Introduction
• Probability Review
• Bayesian Network
• Inference methods
• Network Structure Learning
Probabilities

We will write $P(A = true)$ to mean the probability that $A = true$.

One definition of probability: the relative frequency with which an outcome would be obtained if the process were repeated a large number of times under similar conditions.

The sum of the red and blue areas is 1

$P(A = true)$

$P(A = false)$
Conditional Probability

• $P(A = \text{true} \mid B = \text{true})$: Out of all the outcomes in which $B$ is true, how many also have $A$ equal to true

• Read as: “Probability of $A$ given $B$”

$F =$ “Have a fever”  
$C =$ “Coming down with cold”

$P(F = \text{true}) = 1/10$
$P(C = \text{true}) = 1/15$
$P(F = \text{true} \mid C = \text{true}) = 1/2$

“Fever are rare and cold is rarer, but if you’re coming down with cold there’s a 50-50 chance you’ll have a headache.”
The Joint Probability Distribution

- \( P(A = true, B = true) : \) “the probability of \( A = true \) \textbf{and} \( B = true \)”

- Notice that:

\[
P(F = true \mid C = true)
\]

\[
P(F = true / C = true) = \frac{\text{Area of "C and F" region}}{\text{Area of "C" region}}
\]

\[
= \frac{P(C = true, F = true)}{P(C = true)}
\]
The Joint Probability Distribution

- Joint probabilities can be between any number of variables
e.g. $P(A = \text{true}, B = \text{true}, C = \text{true})$
- For each combination of variables, we need to say how probable that combination is

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>$P(A,B,C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>0.1</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>0.2</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>0.05</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>0.05</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>0.3</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>0.1</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>0.05</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>0.15</td>
</tr>
</tbody>
</table>

Sums to 1
The Joint Probability Distribution

- Once you have the joint probability distribution, you can calculate any probability involving $A$, $B$, and $C$
- Note: May need to use marginalization and Bayes rule,

Examples of things you can compute:
- $P(A=true) = \text{sum of } P(A,B,C) \text{ in rows with } A=true$
- $P(A=true, B = true \mid C=true) = \frac{P(A = true, B = true, C = true)}{P(C = true)}$
Independence

Variables $A$ and $B$ are independent if any of the following hold:

• $P(A,B) = P(A) P(B)$
• $P(A \mid B) = P(A)$
• $P(B \mid A) = P(B)$
Independence

How is independence useful?

• Suppose you have n coin flips and you want to calculate the joint distribution $P(C_1, ..., C_n)$

• If the coin flips are not independent, you need $2^n$ values in the table

• If the coin flips are independent, then

$$P(C_1, ..., C_n) = \prod_{i=1}^{n} P(C_i)$$
Conditional Independence

• C and A are conditionally independent given B if the following holds:

\[ P(C \mid A, B) = P(C \mid B) \]

• Example: “Cancer is a common cause of the two symptoms: a positive X-ray and dyspnoea”

• Joint distribution: \[ P(A,B,C) = P(C/A,B)P(A,B) = P(C/B)P(A,B) = P(C/B)P(A/B)P(B) \]
Outline

• Introduction
• Probability Review
• Bayesian Network
• Inference methods
A Bayesian Network

A Bayesian network is made up of:

1. A Directed Acyclic Graph

2. A set of tables for each node in the graph: conditional probability table

<table>
<thead>
<tr>
<th>A</th>
<th>P(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>0.4</td>
</tr>
<tr>
<td>true</td>
<td>0.6</td>
</tr>
</tbody>
</table>

| A   | B   | P(B|A) |
|-----|-----|-------|
| false | false | 0.03  |
| false | true  | 0.97  |
| true  | false | 0.6   |
| true  | true  | 0.4   |

| B   | D   | P(D|B) |
|-----|-----|-------|
| false | false | 0.01  |
| false | true  | 0.99  |
| true  | false | 0.04  |
| true  | true  | 0.96  |

| B   | C   | P(C|B) |
|-----|-----|-------|
| false | false | 0.3   |
| false | true  | 0.7   |
| true  | false | 0.8   |
| true  | true  | 0.2   |
A Directed Acyclic Graph

Each node in the graph is a random variable

A node $X$ is a parent of another node $Y$ if there is an arrow from node $X$ to node $Y$ e.g. $A$ is a parent of $B$

an arrow from node $X$ to node $Y$ means $X$ has a direct influence on $Y$
A Set of Tables for Each Node

Each node $X_i$ has a conditional probability distribution $P(X_i | \text{Parents}(X_i))$ that quantifies the effect of the parents on the node except the root node.

<table>
<thead>
<tr>
<th>A</th>
<th>P(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>0.4</td>
</tr>
<tr>
<td>true</td>
<td>0.6</td>
</tr>
</tbody>
</table>

| B  | C   | P(C|B) |
|----|-----|-------|
| false | false | 0.3 |
| false | true  | 0.7 |
| true  | false | 0.8 |
| true  | true  | 0.2 |

| A  | B          | P(B|A) |
|----|------------|-------|
| false | false | 0.03 |
| false | true  | 0.97 |
| true  | false | 0.6  |
| true  | true  | 0.4  |

| B  | D   | P(D|B) |
|----|-----|-------|
| false | false | 0.01 |
| false | true  | 0.99 |
| true  | false | 0.04 |
| true  | true  | 0.96 |
Bayesian Networks

Two important properties:

1. Encodes the conditional independence relationships between the variables in the graph structure
2. Is a compact representation of the joint probability distribution over the variables
Conditional Independence

The probability distribution for each node depends only on its parents. $C_1$ and $C_2$ are conditionally independent given $X$. 
The Joint Probability Distribution

Due to the conditional independence property, the joint probability distribution over all the variables $X_1, \ldots, X_n$ in the Bayesian net can be computed using the formula:

$$P(X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} P(X_i = x_i \mid Parents(X_i))$$
Using a Bayesian Network Example

\[
P(A = \text{true}, B = \text{true}, C = \text{true}, D = \text{true})
= P(A = \text{true}) \times P(B = \text{true} \mid A = \text{true}) \times
  P(C = \text{true} \mid B = \text{true}) \times P(D = \text{true} \mid B = \text{true})
= (0.6) \times (0.4) \times (0.2) \times (0.96)
\]
Using a Bayesian Network Example

\[ P(A = \text{true}, B = \text{true}, C = \text{true}, D = \text{true}) \]
\[ = P(A = \text{true}) \times P(B = \text{true} \mid A = \text{true}) \times P(C = \text{true} \mid B = \text{true}) \times P(D = \text{true} \mid B = \text{true}) \]
\[ = (0.6) \times (0.4) \times (0.2) \times (0.96) \]

From the graph structure

From the conditional probability tables
Another example

- I'm at work, neighbor Jeff calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?

- Variables: Burglary, Earthquake, Alarm, JeffCalls, MaryCalls

- Network topology reflects "causal" knowledge:
  
  - A burglar can set the alarm off
  - An earthquake can set the alarm off
  - The alarm can cause Mary to call
  - The alarm can cause Jeff to call
Another example: Earthquake or Burglar

A: Alarm

B: Burglary
E: Earthquake

M: Mary Calls
J: Jeff Calls
Bayesian Network for Alarm Domain

```

P(B) = 0.001

P(E) = 0.002

P(A) =

<table>
<thead>
<tr>
<th>B</th>
<th>E</th>
<th>P(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>0.95</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>0.94</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>0.29</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>0.001</td>
</tr>
</tbody>
</table>

P(M) =

<table>
<thead>
<tr>
<th>A</th>
<th>P(M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>0.70</td>
</tr>
<tr>
<td>F</td>
<td>0.01</td>
</tr>
</tbody>
</table>

P(J) =

<table>
<thead>
<tr>
<th>A</th>
<th>P(J)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>0.90</td>
</tr>
<tr>
<td>F</td>
<td>0.05</td>
</tr>
</tbody>
</table>

P(J = true, M = true, A = true, B = false, E = false)

= P(J = true | A = true)P(M = true | A = true)P(A = true | B = false, E = false)P(B = false)P(E = false)

= 0.9 * 0.7 * 0.001 * 0.999 * 0.998 = 0.00062
```
Outline

• Introduction
• Probability Review
• Bayesian Network
• Inference methods
• Network Structure Learning
Inference

• How can one infer the (probabilities of) values of one or more network variables, given observed values of others?

• $P(X \mid E)$
  
  $E =$ The evidence variable(s)

  $X =$ The query variable(s)

• Bayes net contains all information needed for this inference
• If only one variable with unknown value, easy to infer it
• In general case, problem is NP hard
Inference: example

- An example of a query would be:
  \[ P(\text{Anthrax} = \text{true} \mid \text{Fever} = \text{true}, \text{Cough} = \text{true}) \]
- Note: Even though HasDifficultyBreathing and HasWideMediastinum are in the Bayesian network, they are not given values in the query.
- They are treated as unobserved variables.
Inference in Bayesian Network

• Exact inference:
  Variable Elimination
  Junction Tree

• Approximate inference:
  Markov Chain Monte Carlo
  Variational Methods
From Bayesian Network to Junction Tree – Exact Inference

- Conditional dependence among random variables allows information propagated from a node to another
  ⇒ foundation of probabilistic inference

Bayes’ theorem can not be applied directly to non-singly connected networks, as it would yield erroneous results.

Therefore, junction trees are used to implement exact inference.

Given evidence (observations) E, output the posterior probabilities of query $P(Q|E)$.
Conversion of Bayesian Network into Junction Tree

- **Parallel Moralization** connects all parents of each node
- **Parallel Triangulation** chordalizes cycles with more than 3 edges
- **Clique identification** finds **cliques** using node elimination
  - Node elimination is a step look-ahead algorithms that brings challenges in processing large scale graphs
- **Parallel Junction tree construction** builds a **hypergraph** of the Bayesian network based on running intersection property
Constructing Junction Trees

1. Moralization: construct an undirected graph from the DAG
2. Triangulation: Selectively add arcs to the moral graph
3. Build a junction graph by identifying the cliques and separators
4. Build the junction tree by find an appropriate spanning tree
Step 1: Moralization: marry the parents

1. For all $w \in V$:
   - For all $u,v \in \text{parents}(w)$ add an edge $e=u-v$.

2. Undirect all edges.
Step 2: Triangulation

Add edges to $G^M$ such that there is no cycle with length $\geq 4$ that does not contain a chord.
Step 3: Build the junction graph

- A junction graph for an undirected graph $G$ is an undirected, labeled graph.
- Clique: a subgraph that is complete and maximal.
- The nodes are the cliques in $G$.
- If two cliques intersect, they are joined in the junction graph by an edge labeled with their intersection. (separators)
Bayesian Network 
\( G = ( V, E ) \)

Moral graph \( G^M \)

Triangulated graph \( G^T \)

Junction graph \( G^I \) (not complete)

e.g. \( ceg \cap egh = eg \)

Cliques
Step 4: Junction Tree

• A junction tree is a sub-graph of the junction graph that
  • Is a tree
  • Contains all the cliques (spanning tree)
  • Satisfies the *running intersection* property:
    for each pair of nodes U, V, all nodes on the path between U and V contain \( U \cap V \)
Step 4: Junction Tree (cont.)

• Theorem: An undirected graph is triangulated if and only if its junction graph has a junction tree.

• Definition: The weight of a link in a junction graph is the number of variable in the label. The weight of a junction tree is the sum of weights of the labels.

• Theorem: A sub-tree of the junction graph of a triangulated graph is a junction tree if and only if it is a spanning of maximal weight.
There are several methods to find MST.

Kruskal’s algorithm: choose successively a link of maximal weight unless it creates a cycle.

\[ G^1 \text{ (not complete)} \]  \[ G^\text{JT} \]
Inference using junction tree

- Potential $\mathcal{Q}_X$: a function that maps each instantiation $x$ of a set of variables $X$ into a nonnegative real number.

- Marginalization: suppose $X \in Y$. $\mathcal{Q}_Y = \sum_{Y \setminus X} \mathcal{Q}_Y$

- Constraints on potentials
  1) Consistency property: for each clique $X$ and neighboring separator $S$, it holds that

$$\sum_{X \setminus S} \phi_X = \phi_S.$$

2) The potentials encode the joint distribution $P(U)$ of the network according to

$$P(U) = \frac{\prod_i \phi_{X_i}}{\prod_j \phi_{S_j}}.$$

- Property: for each clique/separator, it holds that $\mathcal{Q}_X = P(X)$

That means, for any variable $V \in X$, we can compute its marginal by

$$P(V) = \sum_{X \setminus \{V\}} \phi_X.$$
Inference without evidence

from Huang & Darwiche, 1996
Initialization

The following procedure assigns initial join tree potentials, using the conditional probabilities from the belief network:

1. For each cluster and sepset $X$, set each $\phi_X(x)$ to 1:
   \[
   \phi_X \leftarrow 1.
   \]

2. For each variable $V$, perform the following: Assign to $V$ a cluster $X$ that contains $F_V$; call $X$ the parent cluster of $F_V$. Multiply $\phi_X$ by $P(V \mid V)$:
   \[
   \phi_X \leftarrow \phi_X P(V \mid \Pi_V).
   \]

After initialization, the conditional distribution $P(V \mid \Pi_V)$ of each variable $V$ has been multiplied into some cluster potential. The initialization procedure satisfies Equation (2) as follows:

\[
\frac{\prod_{i=1}^{N} \phi_{X_i}}{\prod_{j=1}^{N-1} \phi_{S_j}} = \frac{\prod_{k=1}^{Q} P(V_k \mid C_{V_k})}{1} = P(U),
\]

where $N$ is the number of clusters, $Q$ is the number of variables, and $\phi_{X_i}$ and $\phi_{S_j}$ are the cluster and sepset potentials, respectively.

from Huang & Darwiche, 1996
Example for initialization

\[ \varphi_{\text{ACE}} \]

\begin{align*}
\text{a} & \quad \text{c} & \quad \text{e} & \quad \text{Initial Values} \\
\text{on} & \quad \text{on} & \quad \text{on} & \quad 1 \times .7 \times .3 = .21 \\
\text{on} & \quad \text{on} & \quad \text{off} & \quad 1 \times .7 \times .7 = .49 \\
\text{on} & \quad \text{off} & \quad \text{on} & \quad 1 \times .3 \times .6 = .18 \\
\text{on} & \quad \text{off} & \quad \text{off} & \quad 1 \times .3 \times .4 = .12 \\
\text{off} & \quad \text{on} & \quad \text{on} & \quad 1 \times .2 \times .3 = .06 \\
\text{off} & \quad \text{on} & \quad \text{off} & \quad 1 \times .2 \times .7 = .14 \\
\text{off} & \quad \text{off} & \quad \text{on} & \quad 1 \times .8 \times .6 = .48 \\
\text{off} & \quad \text{off} & \quad \text{off} & \quad 1 \times .8 \times .4 = .32 \\
\end{align*}

\[ \varphi_{\text{CE}} \]

\begin{align*}
\text{c} & \quad \text{e} & \quad \text{Initial Values} \\
\text{on} & \quad \text{on} & \quad 1 \\
\text{on} & \quad \text{off} & \quad 1 \\
\text{off} & \quad \text{on} & \quad 1 \\
\text{off} & \quad \text{off} & \quad 1 \\
\end{align*}

etc.

from Huang & Darwiche, 1996
Global propagation

• Single message pass

Consider two adjacent clusters X and Y with separator R. A message pass from X to Y occurs in two steps:

1. **Projection.** Assign a new table to \( R \), saving the old table:

\[
\phi_R^{old} \leftarrow \phi_R. \\
\phi_R \leftarrow \sum_{X \setminus R} \phi_X.
\]

2. **Absorption.** Assign a new table to \( Y \), using both the old and new tables of \( R \):

\[
\phi_Y \leftarrow \phi_Y \frac{\phi_R}{\phi_R^{old}}.
\]

from Huang&Darwiche, 1996
**Global propagation**

**COLLECT-EVIDENCE(X)**
1. Mark X.
2. Call **COLLECT-EVIDENCE** recursively on X's unmarked neighboring clusters, if any.
3. Pass a message from X to the cluster which invoked **COLLECT-EVIDENCE(X)**.

**DISTRIBUTE-EVIDENCE(X)**
1. Mark X.
2. Pass a message from X to each of its unmarked neighboring clusters, if any.
3. Call **DISTRIBUTE-EVIDENCE** recursively on X's unmarked neighboring clusters, if any.

from Huang & Darwiche, 1996
Marginalization

- Once we have a consistent junction tree, we can compute $P(V)$ for each variable of interest $V$ by computing the marginals.

$$P(V) = \sum_{X \setminus \{V\}} \phi_X.$$
Inference with evidence

Belief Network

Graphical Transformation

Join Tree Structure

1. Initialization
2. Observation entry

Inconsistent Join Tree

Propagation

Consistent Join Tree

1. Marginalization
2. Normalization

\( P(V \mid e) \)

from Huang&Darwiche, 1996
Observations and Likelihoods

• An observation is a statement of the form $V = v$
• Observations are the simplest forms of evidence.
• Collections of observations denoted by $E$.
• Define likelihood to encode observations:

  • If $V \in E$—that is, if $V$ is observed—then assign each $\Lambda_V(v)$ as follows:
    $\Lambda_V(v) = \begin{cases} 1, & \text{when } v \text{ is the observed value of } V \\ 0, & \text{otherwise} \end{cases}$

  • If $V \notin E$—that is, if the value of $V$ is unknown—then assign $\Lambda_V(v) = 1$ for each value $v$.

from Huang & Darwiche, 1996
Example of likelihood encoding

• Suppose we have observations C = on, E = off

<table>
<thead>
<tr>
<th>Variable</th>
<th>$\Lambda_V(v)$</th>
<th>$v = on$</th>
<th>$v = off$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$B$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$C$</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$D$</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$E$</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$F$</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$G$</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$H$</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Initialization with observations

We keep track of observations by maintaining a likelihood for each variable. We initialize these likelihoods by adding step 2b to the initialization procedure below:

1. For each cluster and sepset $\mathbf{X}$, set each $\phi_{\mathbf{X}}(\mathbf{x})$ to 1:
   \[ \phi_{\mathbf{X}} \leftarrow 1. \]

2. For each variable $V$:
   
   \textbf{(a)} Assign to $V$ a cluster $\mathbf{X}$ that contains $\mathbf{F}_V$; multiply $\phi_{\mathbf{X}}$ by $P(V \mid \Pi_V)$:
   \[ \phi_{\mathbf{X}} \leftarrow \phi_{\mathbf{X}} P(V \mid \Pi_V). \]

   \textbf{(b)} Set each likelihood element $\Lambda_V(v)$ to 1:
   \[ \Lambda_V \leftarrow 1. \]
Observation entry

Note that upon completion of initialization, the likelihoods encode no observations. We incorporate each observation $V = v$ by encoding the observation as a likelihood, and then incorporating this likelihood into the join tree, as follows:

1. Encode the observation $V = v$ as a likelihood $\Lambda_V^{new}$.

2. Identify a cluster $X$ that contains $V$.$^{13}$

3. Update $\phi_X$ and $\Lambda_V$:

   $$
   \phi_X \leftarrow \phi_X \Lambda_V^{new}.
   $$

   $$
   \Lambda_V \leftarrow \Lambda_V^{new}.
   $$

By entering a set of observations $e$ as described above, we modify the join tree potentials, so that all subsequent probabilities derived from the join tree are probabilities of events that are conjoined with evidence $e$. In other words, instead of computing $P(X)$ and $P(V)$, we compute $P(X, e)$ and $P(V, e)$, respectively. Note also that the join tree encodes $P(U, e)$ instead of $P(U)$ (see Equation (2)).
Normalization

After the join tree is made consistent through global propagation, we have, for each cluster (or sepset) $\mathbf{X}$, $\phi_{\mathbf{X}} = P(\mathbf{X}, \mathbf{e})$, where $\mathbf{e}$ denotes the observations incorporated into the join tree according to Section 6.4 [2]. When we marginalize a cluster potential $\phi_{\mathbf{X}}$ into a variable $V$, we obtain the probability of $V$ and $\mathbf{e}$:

$$P(V, \mathbf{e}) = \sum_{\mathbf{X} \setminus \{V\}} \phi_{\mathbf{X}}.$$

Our goal is to compute $P(V | \mathbf{e})$, the probability of $V$ given $\mathbf{e}$. We obtain $P(V | \mathbf{e})$ from $P(V, \mathbf{e})$ by normalizing $P(V, \mathbf{e})$ as follows:

$$P(V | \mathbf{e}) = \frac{P(V, \mathbf{e})}{P(\mathbf{e})} = \frac{P(V, \mathbf{e})}{\sum_{V} P(V, \mathbf{e})}.$$

(2)

The probability of the observations $P(\mathbf{e})$ is often referred to as a normalizing constant.
Approximate inference

- Exact inference is feasible in small to medium-sized networks
- Takes a very long time for large networks
- Turn to approximate inference techniques which are much faster and give pretty good results
Sampling

- Input: Bayesian network with set of nodes $X$
- Sample = a tuple with assigned values
  
  $s= (X_1=x_1, X_2=x_2, ..., X_k=x_k)$

- Tuple may include all variables (except evidence) or a subset
- Sampling schemas dictate how to generate samples (tuples)
- Ideally, samples are distributed according to $P(X|E)$
Sampling algorithms

- Gibbs Sampling (MCMC)
- Importance Sampling
- Sequential Monte-Carlo (Particle Filtering) in Dynamic Bayesian Networks
- etc.
## A list of Python libraries

<table>
<thead>
<tr>
<th>Library</th>
<th>Algorithm</th>
<th>Algorithm Type</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>BayesPy</td>
<td>variational message passing</td>
<td>approximate</td>
<td>MIT</td>
</tr>
<tr>
<td>pomegranate</td>
<td>loopy belief</td>
<td>approximate</td>
<td>MIT</td>
</tr>
<tr>
<td>pgmpy</td>
<td>multiple</td>
<td>approximate/exact</td>
<td>MIT</td>
</tr>
<tr>
<td>libpgm</td>
<td>likelihood sampling</td>
<td>approximate</td>
<td>Proprietary</td>
</tr>
<tr>
<td>bayesnetinference</td>
<td>variable elimination</td>
<td>exact</td>
<td>None</td>
</tr>
</tbody>
</table>
Outline

• Introduction
• Probability Review
• Bayesian Network
• Inference methods
• Network Structure Learning
Original Graph – Asia Example

In [18]: bnh.get_model_from_bif('/datadrive/masa/bayesian/asia.bif')

In [19]: bnh.plot_model()
This example is the well known Asia Bayesian network.

The Bayesian network below will update when you click the check boxes to set evidence. The on very small subset of the features of the full User Interface and APIs.
Prediction

- Setting one (or more) column to None and let Pomegranate predict the value based on other observations
- Need to already have Bayesian network with conditional probabilities (through .bif file or generated network)
- Right example: original Asia network, 64% accuracy
- Get random samples from Asia network, then re-feed the samples to generate graph estimate
- Run prediction again on “smoke”
- 49% accuracy
- Takeaway: generated networks and conditional probabilities from samples may generate completely different networks
Generating Bayesian Network = NP-Hard

PGMPY

**Exhaustive Search**

```python
class pgmpy.esters.ExhaustiveSearch.ExhaustiveSearch(data, scoring_method=\textit{None}, **kwargs)
    all_dags(nodes=\textit{None})
```

Computes all possible directed acyclic graphs with a given set of nodes, sparse ones first. $2^{n(n-1)}$ graphs need to be searched, given $n$ nodes, so this is likely not feasible for $n>6$. This is a generator.

Pomegranate

However, one can also initialize a Bayesian network based completely on data. As mentioned before, the exact version of this algorithm takes exponential time with the number of variables and typically can't be done on more than $\sim25$ variables. This is because there are a super-exponential number of directed acyclic graphs that one could define over a set of variables, but fortunately one can use dynamic programming in order to reduce this complexity down to "simply exponential." The implementation of the exact algorithm actually goes further than the original dynamic
Generate Models with Pomegranate (greedy)

- Results can highly depend on samples

```
In [91]: bnh.get_model_from_sample(X_train)
In [92]: bnh.plot_model()
```
Generate Acyclic Permutations

- For each node, randomly assign it to level 1, 2, ..., K
- Randomly pick two nodes
  - One node from level k
  - Other node from level k+1
  - Add a directed edge from first to second node
  - Edges cannot skip levels or connect nodes in the same level
- This leveling system prevents generating networks with cycles
- Use PGMPY’s K2Score to quantify network fit
Bayesian scoring functions

- Compute the posterior probability distribution, starting from a prior probability distribution on the possible networks, conditioned to data $T$, that is, $P(B|T)$.
- The best network is the one that maximizes the posterior probability.
- Since the term $P(T)$ is the same for all possible networks, in practice, for comparative purposes, computing $P(B, T)$ is sufficient.
- As it is easier to work in the logarithmic space, the scoring functions use the value $\log(P(B, T))$ instead of $P(B, T)$.

K2 scoring function

Cooper and Herskovits (1992) proposed a particular case of the BD score, called the K2 score,

\[ K2(B, T) = \log(P(B)) + \sum_{i=1}^{n} \sum_{j=1}^{q_i} \left( \log \left( \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \right) + \sum_{k=1}^{r_i} \log(N_{ijk}!) \right), \]

with the uninformative assignment \( N_{ijk}^\prime = 1 \) (corresponding to zero pseudo-counts).
Example where max_level = 3
Network with best score
How to handle larger datasets with many columns/nodes and cardinality?

```
In [2]:
csv_file = '/datadrive/masa/bayesian/hidden_rel_new_1.csv'
df = pd.read_csv(csv_file)
df
```

```
Out[2]:

<table>
<thead>
<tr>
<th>date</th>
<th>source</th>
<th>target</th>
<th>source_in_degree</th>
<th>source_out_degree</th>
<th>source_all_degree</th>
<th>target_in_degree</th>
<th>target_out_degree</th>
<th>target_all_degree</th>
<th>source_closeness</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2018/8/30</td>
<td>39</td>
<td>28</td>
<td>6.0</td>
<td>2.0</td>
<td>8.0</td>
<td>6.0</td>
<td>5.0</td>
<td>11.0</td>
</tr>
<tr>
<td>1</td>
<td>2018/8/30</td>
<td>28</td>
<td>39</td>
<td>6.0</td>
<td>5.0</td>
<td>11.0</td>
<td>6.0</td>
<td>2.0</td>
<td>8.0</td>
</tr>
<tr>
<td>2</td>
<td>2018/8/30</td>
<td>45</td>
<td>28</td>
<td>0.0</td>
<td>4.0</td>
<td>4.0</td>
<td>6.0</td>
<td>5.0</td>
<td>11.0</td>
</tr>
<tr>
<td>3</td>
<td>2018/8/30</td>
<td>28</td>
<td>45</td>
<td>6.0</td>
<td>5.0</td>
<td>11.0</td>
<td>0.0</td>
<td>4.0</td>
<td>4.0</td>
</tr>
<tr>
<td>4</td>
<td>2018/8/30</td>
<td>45</td>
<td>39</td>
<td>0.0</td>
<td>4.0</td>
<td>4.0</td>
<td>6.0</td>
<td>2.0</td>
<td>8.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>276</td>
<td>2018/8/30</td>
<td>25</td>
<td>33</td>
<td>4.0</td>
<td>2.0</td>
<td>6.0</td>
<td>4.0</td>
<td>2.0</td>
<td>6.0</td>
</tr>
<tr>
<td>277</td>
<td>2018/8/30</td>
<td>33</td>
<td>25</td>
<td>4.0</td>
<td>2.0</td>
<td>6.0</td>
<td>4.0</td>
<td>2.0</td>
<td>6.0</td>
</tr>
<tr>
<td>278</td>
<td>2018/8/30</td>
<td>23</td>
<td>49</td>
<td>0.0</td>
<td>2.0</td>
<td>2.0</td>
<td>4.0</td>
<td>0.0</td>
<td>4.0</td>
</tr>
<tr>
<td>279</td>
<td>2018/8/30</td>
<td>38</td>
<td>49</td>
<td>0.0</td>
<td>2.0</td>
<td>2.0</td>
<td>4.0</td>
<td>0.0</td>
<td>4.0</td>
</tr>
<tr>
<td>280</td>
<td>2018/8/30</td>
<td>23</td>
<td>40</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

281 rows x 40 columns
Use histogram binning to reduce the cardinality

• Instead of having too many cardinality, reduce it using histogram (fixed # bins or percentage).

• Also reduces chance of having value only appear once or twice

In [5]: simp = Simplify(df_valid)

In [6]: simp.df_hist

Out[6]:

<table>
<thead>
<tr>
<th>preferential_attachment</th>
<th>direct_hitting_time</th>
<th>CC_num_vertices</th>
<th>leicht</th>
<th>source</th>
<th>allpaths</th>
<th>edge_indicator</th>
<th>source_all_degree</th>
<th>adar</th>
<th>salton_e</th>
<th>...</th>
<th>target_between</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7.2</td>
<td>0.231071</td>
<td>6.4</td>
<td>0.0</td>
<td>30.4</td>
<td>50.0</td>
<td>1.0</td>
<td>5.8</td>
<td>-0.5</td>
<td>0.0</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>28.8</td>
<td>0.231071</td>
<td>6.4</td>
<td>0.0</td>
<td>20.6</td>
<td>50.0</td>
<td>1.0</td>
<td>10.6</td>
<td>-0.5</td>
<td>0.0</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>21.6</td>
<td>0.000000</td>
<td>6.4</td>
<td>0.0</td>
<td>40.2</td>
<td>50.0</td>
<td>0.0</td>
<td>3.4</td>
<td>-0.5</td>
<td>1.2</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>0.0</td>
<td>0.231071</td>
<td>6.4</td>
<td>0.0</td>
<td>20.6</td>
<td>50.0</td>
<td>0.0</td>
<td>10.6</td>
<td>-0.5</td>
<td>0.0</td>
<td>...</td>
</tr>
<tr>
<td>4</td>
<td>21.6</td>
<td>0.000000</td>
<td>6.4</td>
<td>0.0</td>
<td>40.2</td>
<td>50.0</td>
<td>1.0</td>
<td>3.4</td>
<td>-0.5</td>
<td>0.0</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>276</td>
<td>7.2</td>
<td>0.231071</td>
<td>8.6</td>
<td>0.0</td>
<td>20.6</td>
<td>50.0</td>
<td>1.0</td>
<td>5.8</td>
<td>-0.5</td>
<td>0.6</td>
<td>...</td>
</tr>
<tr>
<td>277</td>
<td>7.2</td>
<td>0.462143</td>
<td>8.6</td>
<td>0.0</td>
<td>30.4</td>
<td>50.0</td>
<td>1.0</td>
<td>5.8</td>
<td>-0.5</td>
<td>0.6</td>
<td>...</td>
</tr>
<tr>
<td>278</td>
<td>7.2</td>
<td>0.462143</td>
<td>2.0</td>
<td>0.0</td>
<td>20.6</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>-0.5</td>
<td>0.0</td>
<td>...</td>
</tr>
<tr>
<td>279</td>
<td>7.2</td>
<td>0.924286</td>
<td>2.0</td>
<td>0.0</td>
<td>30.4</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>-0.5</td>
<td>0.6</td>
<td>...</td>
</tr>
<tr>
<td>280</td>
<td>0.0</td>
<td>0.000000</td>
<td>2.0</td>
<td>0.0</td>
<td>20.6</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>-0.5</td>
<td>0.0</td>
<td>...</td>
</tr>
</tbody>
</table>

281 rows × 38 columns
Example with max_level = 5
Permutation + Pruning Algorithm

• Have a loop where we choose 1, 2, ... tuple of nodes and run through each permutation of node-edge connections with the tuples

• After every loop, look through the permutations that are generated and pick the ones with highest scores
  • Scores are calculated by doing prediction on the leaf nodes (nodes without any children)
Networks with high scores

- 'property': {'category': 'Classification', 'trainer': None},
- 'model_name': 'my_bayes'},
- 'model_metrics': {0: (0.996969696969697, ('tub', 'asia', 'either', 'lung')),
  1: (1.0, ('lung', 'tub', 'either')),
  2: (0.9924242424242424, ('lung', 'tub', 'either')),
  3: (0.9757575757575757, ('tub', 'lung', 'asia')),
  4: (0.9681818181818181, ('tub', 'asia', 'either')),
  5: (0.9681818181818181, ('tub', 'asia', 'either')),
  6: (0.9636363636363636, ('lung', 'tub', 'asia')),
  7: (0.9424242424242424, ('lung', 'tub', 'tub')),
  8: (0.9424242424242424, ('lung', 'tub', 'either')),
  9: (0.9424242424242424, ('tub', 'tub', 'either')),
  10: (0.9393939393939394, ('tub', 'tub', 'asia')),
  11: (0.9924242424242424, ('tub', 'tub')),
  12: (0.9909090909090909, ('tub', 'either')),
  13: (0.9681818181818181, ('tub', 'asia')),
  14: (0.9636363636363636, ('tub', 'lung')),
  15: (0.9609090909090909, ('tub', 'tub')),
  16: (0.9424242424242424, ('tub', 'either')),
  17: (0.9424242424242424, ('tub', 'tub')),
  18: (0.9348484848484848, ('tub', 'tub')),
  19: (0.9333333333333333, ('lung', 'tub'))},

'log_path': '/datadrive/mace/ard/machine_learning/client/mlcompute/user_output/'
Network with high score

In [150]: bayes_analysis.bayes_network_handler.get_model_from_sample(X_train[list(score_dict[6][0][1])])
bayes_analysis.bayes_network_handler.plot_model()

[DEBUG] PngImagePlugin.call().146 STREAM b'IHDR' 16 13
[DEBUG] PngImagePlugin.call().146 STREAM b'bKGD' 41 6
[DEBUG] PngImagePlugin._open().592 b'bKGD' 41 6 (unknown)
[DEBUG] PngImagePlugin.call().146 STREAM b'IDAT' 59 8192

[(0.6103286384976526,  
'sorensen_e',  
'ground_truth',  
'edge_indicator',  
'Katz',  
'target_betweenness',  
'salton_e'))]
Ardi Machine Learning includes Bayesian Networks

ml-ui

ml-manager
- Get file from UI
- Process parameters
- Store entry in database
- Wait for available worker to take the work
- Read results

requests

ml-db
- Stores data and parameters

ml-worker
- Read files and parameters
- Run Bayesian code
- Return results

BayesNetworkHandler.py
- PGMPY
- Pomegranate
Acknowledgements

• Some of the materials are based on work by the following:
  • Dr.Cheng, Dr. Wong, Dr. Hamo, Dr. Silberstein, Dr. Huang, Mr. Chang-Ogimoto, etc...