Learning Bayesian Networks
Aspects in learning

- Learning the parameters of a Bayesian network
  - Marginalizing over all all parameters
  - Equivalent to choosing the expected parameters

- Learning the structure of a Bayesian network
  - Marginalizing over the structures not computationally feasible
  - Model selection
A Bayesian network

P(Cloudy)

Cloudy

Cloudy=no | Cloudy=yes
0.5 | 0.5

P(Sprinkler | Cloudy)

Cloudy | Sprinkler=on | Sprinkler=off
no | 0.5 | 0.5
yes | 0.9 | 0.1

P(Rain | Cloudy)

Cloudy | Rain=yes | Rain=no
no | 0.2 | 0.8
yes | 0.8 | 0.2

P(WetGrass | Sprinkler, Rain)

Sprinkler | Rain | WetGrass=yes | WetGrass=no
on | no | 0.90 | 0.10
on | yes | 0.99 | 0.01
off | no | 0.01 | 0.99
off | yes | 0.90 | 0.10
Learning the parameters

- Given the data D, how should I fill the conditional probability tables?
- Bayesian answer:
  - You should not. If you do not know them, you will have a priori and a posteriori distributions for them.
  - They are many, but again, the independence comes to rescue.
  - Once you have distribution of parameters, you can do the prediction by model averaging.
  - Very similar to Bernoulli case.
A Bayesian network revisited

Cloudy

<table>
<thead>
<tr>
<th>Cloudy=no</th>
<th>Cloudy=yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Theta_{C}$:</td>
<td>$\Theta_{C}$:</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Sprinkler

<table>
<thead>
<tr>
<th>Sprinkler=on</th>
<th>Sprinkler=off</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Theta_{S</td>
<td>C}^{C}$:</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>0.9</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Rain

<table>
<thead>
<tr>
<th>Rain=yes</th>
<th>Rain=no</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Theta_{R</td>
<td>C}$:</td>
</tr>
<tr>
<td>0.2</td>
<td>0.8</td>
</tr>
<tr>
<td>0.8</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Wet Grass

<table>
<thead>
<tr>
<th>WetGrass=yes</th>
<th>WetGrass=no</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Theta_{W</td>
<td>S,R}$:</td>
</tr>
<tr>
<td>0.90</td>
<td>0.10</td>
</tr>
<tr>
<td>0.99</td>
<td>0.01</td>
</tr>
<tr>
<td>0.01</td>
<td>0.99</td>
</tr>
<tr>
<td>0.90</td>
<td>0.10</td>
</tr>
</tbody>
</table>
A Bayesian network as a generative model

\[ \Theta_{C} \]
\[ \Theta_{S|C} \]
\[ \Theta_{R|C} \]
\[ \Theta_{W|S,R} \]
A Bayesian network as a generative model

- Cloudy
  - θ \( S|C=\text{yes} \)
  - θ \( S|C=\text{no} \)
- Sprinkler
  - θ \( W|S=\text{on},R=\text{no} \)
  - θ \( W|S=\text{on},R=\text{yes} \)
- Rain
  - θ \( R|C=\text{yes} \)
  - θ \( R|C=\text{no} \)
- Wet Grass
  - θ \( W|S=\text{off},R=\text{no} \)
  - θ \( W|S=\text{off},R=\text{yes} \)
A Bayesian network as a generative model

- Parameters are independent a priori:
  \[
P(\Theta) = \prod_{i=1}^{n} P(\Theta_i)
  = \prod_{i=1}^{n} \prod_{j=1}^{q_i} P(\Theta_{i|j}),
\]
  where
  \[
P(\Theta_{i|j}) = \text{Dir}(\alpha_1, \ldots, \alpha_{r_i}).
\]
Generating a data set

Plate notation:

\[
\begin{array}{ccc}
\Theta_c & \Theta_{R|C=yes} & \Theta_{R|C=no} \\
\alpha_c & \alpha_{R|C=yes} & \alpha_{R|C=no} \\
\end{array}
\]

\[
\begin{array}{ccc}
\text{Cloudy}_1 & \text{Rain}_1 & \text{Cloudy}_2 \\
\alpha & \alpha_{R|C=yes} & \alpha_{R|C=no} \\
\end{array}
\]

\[
\begin{array}{ccc}
\text{Cloudy}_N & \text{Rain}_N & \ldots \\
\alpha_c & \alpha_{R|C=yes} & \alpha_{R|C=no} \\
\end{array}
\]

i.i.d, isn't it

<table>
<thead>
<tr>
<th>D</th>
<th>Cloudy</th>
<th>Rain</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d_1)</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>(d_2)</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>(d_N)</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

• Plate notation:
Likelihood $P(D|\Theta,G)$

- For one data vector it was:

$$P(x_1, x_2, \ldots, x_n|G) = \prod_{i=1}^{n} P(x_i|pa_G(x_i)), \text{ or}$$

$$P(d_1|G, \theta) = \prod_{i=1}^{n} \theta_{d_{1i}|pa_{1i}}, \text{ where } d_{1i} \text{ and } pa_{1i} \text{ are the value and the parent configuration of the variable } i \text{ in data vector } d_1.$$ 

$$P(d_1, d_2, \ldots, d_N|G, \theta) = \prod_{j=1}^{N} \prod_{i=1}^{n} \theta_{d_{ji}|pa_{ji}} = \prod_{i=1}^{n} \prod_{k=1}^{r_i} \prod_{j=1}^{q_i} \prod_{k=1}^{R_{ijk}} \theta_{ik|j},$$

where $N_{ijk}$ is the number of data vectors with parent configuration $j$ when variable $i$ has the value $k$, $r_i$ and $q_i$ are the numbers of values and parent configurations of the variable $i$. 
Bayesian network learning

\[ N_C(q_C=1, r_C=2) \]

\[ N_{S|C}(q_S=2, r_S=2) \]

\[ P(D|G, \theta) = \prod_{i=1}^{n} \prod_{k=1}^{r_i} \prod_{j=1}^{q_i} \theta^{N_{ijk}}_{ikj} \]

- i picks the variable (table)
- j picks the row
- k picks the column

\[ N_R(q_R=2, r_R=2) \]

\[ N_{W|S,R}(q_W=4, r_W=2) \]
Bayesian network learning after 
\((C,S,R,W) = [(\text{no, on, yes, yes}), (\text{no, on, no, no})]\)

\[
P(D|G, \theta) = \prod_{i=1}^{n} \prod_{k=1}^{r_i} \prod_{j=1}^{q_i} \theta_{ikj}^{N_{ijk}}
\]

\begin{align*}
N_{S|C}(q_s=2, r_s=2) & \quad N_{C}(q_c=1, r_c=2) & \quad N_{R|C}(q_r=2, r_r=2) & \quad N_{W|S,R}(q_w=4, r_w=2) \\
\begin{array}{c|c}
\text{Sprinkler=on} & \text{Sprinkler=off} \\
\hline
N_{S|C=\text{no}} & 1+1=2 & 0 \\
N_{S|C=\text{yes}} & 0 & 0 \\
\end{array}
\end{align*}

\begin{align*}
\begin{array}{c|c}
\text{Cloudy=\text{no}} & \text{Cloudy=\text{yes}} \\
\hline
N_{C} & 1+1=2 & 0 \\
\end{array}
\end{align*}

\begin{align*}
\begin{array}{c|c}
\text{Rain=\text{yes}} & \text{Rain=\text{no}} \\
\hline
N_{R|C} & 1 & 1 \\
0 & 0 \\
\end{array}
\end{align*}

\begin{align*}
\begin{array}{c|c}
\text{WetGrass=\text{yes}} & \text{WetGrass=\text{no}} \\
\hline
N_{W|S,R} & 0 & 1 \\
N_{W|S=\text{on},R=\text{no}} & 0 & 0 \\
N_{W|S=\text{on},R=\text{yes}} & 1 & 0 \\
N_{W|S=\text{off},R=\text{no}} & 0 & 0 \\
N_{W|S=\text{off},R=\text{yes}} & 0 & 0 \\
\end{array}
\end{align*}

- i picks the variable (table)
- j picks the row
- k picks the column
- \(r_i\), number of columns in table i
- \(q_i\), number of rows in table i
Bayesian network learning after a while (20 data vectors)

\[ P(D|G, \theta) = \prod_{i=1}^{n} \prod_{k=1}^{r_i} \prod_{j=1}^{q_i} \theta_{ijk}^{N_{ijk}} \]

- i picks the variable (table)
- j picks the row
- k picks the column
- \( r_i \), number of columns in table i
- \( q_i \), number of rows in table i

\[ N_{C|\text{no}} = 16 \quad N_{C|\text{yes}} = 4 \quad = 20 \]
\[ N_{S|C|\text{no}} = 10 \quad N_{S|C|\text{yes}} = 1 \quad = 11 \]
\[ N_{R|C|\text{no}} = 3 \quad N_{R|C|\text{yes}} = 4 \quad = 7 \]
\[ N_{W|S,R} = \]

\[ \begin{array}{c|c|c}
\text{WetGrass} & \text{yes} & \text{no} \\
\hline
\text{W|S=on,R=no} & 2 & 3 \quad = 5 \\
\text{W|S=on,R=yes} & 1 & 5 \quad = 6 \\
\text{W|S=off,R=no} & 6 & 2 \quad = 8 \\
\text{W|S=off,R=yes} & 0 & 1 \quad = 1 \\
\end{array} \]
Maximum likelihood

- Since the parameters are occur separately in likelihood we can maximize the terms independently:

\[ P(D|G, \theta) = \prod_{i=1}^{n} \prod_{k=1}^{r_i} \prod_{j=1}^{q_i} \theta_{ijk}^{N_{ijk}} \Rightarrow \hat{\theta}_{ijk} = \frac{N_{ijk}}{\sum_{k'=1}^{r_i} N_{ijk'}} \]

- So you simply normalize the rows in the sufficient statistics tables to get ML-parameters

- But these parameters may have zero probabilities:
  - not good for prediction; hear the Bayes call ....
Learning the parameters - again

• Given the data D, how should I fill the conditional probability tables?

• Bayesian answer:
  - You should not. If you do not know them, you will have a priori and a posteriori distributions for them.
  - They are many, but again, the independence comes to rescue.
  - Once you have distribution of parameters, you can do the prediction by model averaging.
  - Very similar to the Bernoulli case.
Prior x Likelihood

- A priori parameters independently Dirichlet:

\[
P(\Theta|\alpha) = \prod_{i=1}^{n} P(\Theta_i) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} P(\Theta_{ij}|\alpha) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma\left(\sum_{k=1}^{r_i} \alpha_{ijk}\right)}{\prod_{k=1}^{r_i} \Gamma(\alpha_{ijk})} \prod_{k=1}^{r_i} \theta_{ijk}^{\alpha_{ijk}-1}
\]

- Likelihood compatible with conjugate prior:

\[
P(D|G,\theta) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}}
\]

- Yields a simple posterior

\[
P(\Theta|D,\alpha) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} P(\Theta_{ij}|N_{ij},\alpha_{ij}),
\]

where \( P(\Theta_{ij}|N_{ij},\alpha_{ij}) = \text{Dir}(N_{ij} + \alpha_{ij}) \)
Predictive distribution $P(d|D,\alpha,G)$

- **Posterior:**
  
  
  $$P(\theta|D,\alpha) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma\left(\sum_{k=1}^{r_i} N_{ijk} + \alpha_{ijk}\right)}{\prod_{k=1}^{r_i} \Gamma\left(N_{ijk} + \alpha_{ijk}\right)} \prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk} + \alpha_{ijk} - 1}$$

- **Predictive distribution:**

  $$P(d|D,\alpha,G) = \int_{\theta} P(d,\theta|D,\alpha) d\theta = \int_{\theta} P(d|\theta) P(\theta|D,\alpha) d\theta$$

  $$= \int_{\theta} \prod_{i=1}^{n} P(d_i|\theta_i) P(\theta_i|D,\alpha) d\theta$$

  $$= \prod_{i=1}^{n} \int_{\theta_{ipa,d_i}} \theta_{ipa,d_i} P(\theta_{ipa,d_i}|N_{ipa,d_i},\alpha_{ipa,d_i}) d\theta_{ipa,d_i}$$

  $$= \prod_{i=1}^{n} \bar{\theta}_{ipa,d_i} = \prod_{i=1}^{n} \frac{N_{ipa,d_i} + \alpha_{ipa,d_i}}{\sum_{k=1}^{r_i} N_{ipa,k} + \alpha_{ipa,k}}$$
Predictive distribution

• This means that predictive distribution

\[ P(d|D, \alpha, G) = \prod_{i=1}^{n} \frac{N_{ipa_i d_i} + \alpha_{ipa_i d_i}}{\sum_{k=1}^{r_i} N_{ipa_i k} + \alpha_{ipa_i k}} \]

can be achieved by just setting

\[ \theta_{ijk} = \frac{N_{ijk} + \alpha_{ijk}}{N_{ij} + \alpha_{ij}} \]

• So just gather counts \( N_{ijk} \), add \( \alpha_{ijk} \) to them and normalize.
Being uncertain about the Bayesian network structure

- Bayesian says again:
  - If you do not know it, you should have an a priori and the a posteriori distribution for it.

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

- Likelihood \(P(D|G)\) is called the *marginal likelihood* and with certain assumptions, it can be computed in closed form

- Normalizer we can just ignore.
Prediction over model structures

\[ P(X|D) = \sum_M P(X|M, D) P(M|D) \]

\[ = \sum_M \int_M P(X|\Theta, M, D) P(\Theta|M, D) d\Theta P(M|D) \]

\[ \propto \sum_M P(X|\overline{\Theta}(D), M) P(D|M) P(M) \]

\[ = \sum_M P(X|\overline{\Theta}(D), M) \int_\Theta P(D|\Theta, M) P(\Theta|M) d\Theta P(M) \]

- This summation is not feasible as it goes over a super-exponential number of model structures
- Does NOT reduce to using a single expected model structure, like what happens with the parameters
- Typically use only one (or a few) models with high posterior probability \( P(M|D) \)
Averaging over an equivalence class

• Boils down to using a single model (assuming uniform prior over the models within the equivalence class):

\[ P(X|E) = \sum_{M \in E} P(X|M, E) P(M|E) \]

\[ = |E| P(X|M) \frac{1}{|E|} \]

\[ = P(X|M) \]
Model Selection

- Problem: The number of possible structures for a given domain is more than exponential in the number of variables
- Solution: Use only one or a handful of "good" models
- Necessary components:
  - Scoring method (what is "good"?)
  - Search method (how to find good models?)
Learning the structure: scoring
Good models?

- In marginalization/summation/model averaging over all the model structures, the predictions are weighted by $P(M \mid D)$, the posteriors of the models given the data.

- If have to select one (a few) model(s), it sounds reasonable to use model(s) with the largest weight(s).

- $P(M \mid D) = \frac{P(D \mid M)P(M)}{P(D)}$

- Relevant components:
  - The structure prior $P(M)$
  - The marginal likelihood (the "evidence") $P(D \mid M)$
How to set the structure prior $P(M)$?

- The "standard" solution: use the uniform prior (i.e., ignore the structure prior)

- Sometimes suggested: $P(M)$ proportional to the number of arcs so that simple models more probable
  - Justification???

- Uniform over the equivalence classes? Proportional to the size of the equivalence class? What about the nestedness (full networks "contain" all the other networks)...

- ...still very much an open issue
Marginal likelihood $P(D|G, \alpha)$

$$P(D|G, \alpha) = P(d_1|G, \alpha)P(d_2|d_1, G, \alpha)\ldots P(d_N|d_1,\ldots,d_{N-1}, G, \alpha)$$

$$= \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(N_{ij} + \alpha_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + \alpha_{ijk})}{\Gamma(\alpha_{ijk})}$$

<table>
<thead>
<tr>
<th>Cloudy=no</th>
<th>Cloudy=yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_C$</td>
<td>1+1=2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sprinkler=on</th>
<th>Sprinkler=off</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{S</td>
<td>C=\text{no}}$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$N_{S</td>
<td>C=\text{yes}}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rain=yes</th>
<th>Rain=no</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WetGrass=yes</th>
<th>WetGrass=no</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{W</td>
<td>S=\text{on},R=\text{no}}$</td>
</tr>
<tr>
<td>$N_{W</td>
<td>S=\text{on},R=\text{yes}}$</td>
</tr>
<tr>
<td>$N_{W</td>
<td>S=\text{off},R=\text{no}}$</td>
</tr>
<tr>
<td>$N_{W</td>
<td>S=\text{off},R=\text{yes}}$</td>
</tr>
</tbody>
</table>
Computing the marginal likelihood

• Two choices:

  1. Calculate the sufficient statistics $N_{ijk}$ and compute $P(D \mid M)$ directly using the (gamma) formula on the previous slide.

  2. Use the chain rule, and compute $P(d_1, \ldots, d_n \mid M) = P(d_1 \mid M)P(d_2 \mid d_1, M)\ldots P(d_n \mid d_1, \ldots, d_{n-1} \mid M)$ by using iteratively the predictive distribution (slide 18).

• OBS! The latter can be done in any order, and the result will be the same (remember Exercise 2?)!
How to set the hyperparameters $\alpha$?

- Assuming...
  - a multinomial sample,
  - independent parameters,
  - modular parameters,
  - complete data,
  - likelihood equivalence,

...implies a certain parameter prior: BDe ("Bayesian Dirichlet with likelihood equivalence")
BDeu

- Likelihood equivalence: two Markov equivalent model structures produce to the same predictive distribution.
- Means also that $P(D \mid M) = P(D \mid M')$ if $M$ and $M'$ equivalent.
- Let $\alpha_i = \sum_j \alpha_{ij}$, where $\alpha_{ij} = \sum_k \alpha_{ijk}$.
- BDe means that $\alpha_i = \alpha$ for all $i$, and $\alpha$ is the equivalent sample size.
- An important special case: BDeu ("u" for "uniform"): $\alpha_{ijk} = \frac{\alpha}{q_i r_i}$, $\alpha_{ij} = \frac{\alpha}{q_i}$. 
Model selection in the Bernoulli case

- Toss a coin 250 times, observe $D$: 140 heads and 110 tails.
- Hypothesis $H_0$: the coin is fair ($P(\Theta=0.5) = 1$)
- Hypothesis $H_1$: the coin is biased
- Statistics:
  - The P-value is 7%
  - “suspicious”, but not enough for rejecting the null hypothesis (Dr. Barry Blight, The Guardian, January 4, 2002)
- Bayes:
  - Let’s assume a prior, e.g. Beta($a,a$)
  - Compute the Bayes factor

$$\frac{P(D|H_1)}{P(D|H_0)} = \int P(D|\theta,H_1,a)P(\theta|H_1,a) \, d\theta \frac{1}{2^{250}}$$
Equivalent sample size and the Bayes Factor

Bayes factor in favor of H1

0,37 1 2,7 7,4 20 55 148 403 1096

Equivalent sample size
A slightly modified example

- Toss a coin 250 times, observe $D = 141$ heads and $109$ tails.
- Hypothesis $H_0$: the coin is fair ($P(\Theta=0.5) = 1$)
- Hypothesis $H_1$: the coin is biased
- Statistics:
  - The P-value is 4.97%
  - *Reject the null hypothesis at a significance level of 5%*
- Bayes:
  - Let’s assume a prior, e.g. Beta($a,a$)
  - Compute the Bayes factor

$$\frac{P(D|H_1)}{P(D|H_0)} = \frac{\int P(D|\theta,H_1,a)P(\theta|H_1,a)\,d\theta}{1/2^{250}}$$
Equivalent sample size and the Bayes Factor (modified example)
Lessons learned

• Classical statistics and the Bayesian approach may give contradictory results
  – Using a fixed P-value threshold is problematic as any null hypothesis can be rejected with sufficient amount of data
  – The Bayesian approach compares models and does not aim at an “absolute” estimate of the goodness of the models
• Bayesian model selection depends heavily on the priors selected
  – However, the process is completely transparent and suspicious results can be criticized based on the selected priors
  – Moreover, the impact of the prior can be easily controlled with respect to the amount of available data
• The issue of determining non-informative priors is controversial
  – Reference priors
  – Normalized maximum likelihood & MDL (see www.mdl-research.org)
On Bayes factor and Occam’s razor

- The marginal likelihood (the “evidence”) $P(D \mid H)$ yields a probability distribution (or density) over all the possible data sets $D$.
- Complex models can predict well many different data sets, so they need to spread the probability mass over a wide region of models.
Hyperparameters in more complex cases

- Bad news: the BDeu score seems to be quite sensitive to the equivalent sample size (Silander & Myllymäki, UAI'2007)

Figure 1: Number of arcs in the BDeu optimal network for the Yeast data as a function of $\alpha$.

Figure 2: BDeu scores of different MAP models for the Liver data as a function of $\alpha$. 
So which prior to use?

- An open issue
- One solution: use the "priorless" Normalized Maximum Likelihood approach
- A more Bayesian solution: use the Jeffreys prior
  - Can be formulated in the Bayesian network framework (Kontkanen et al., 2000), but nobody has produced software for computing it in practice (good topic for your thesis!)
  - B-Course: \( \alpha = \frac{1}{2n} \sum_{i=1}^{n} r_i \)
Learning the structure: search
Learning the structure when each node has at most one parent

• The BD score is *decomposable*:

\[
\max_M P(D|M) = \max_M \prod_i P(X_i|Pa_i^M, D)
\]

\[
= \min_M \sum_i f_D(X_i, Pa_i^M),
\]

where \( f_D(X_i, Pa_i^M) = \log P(X_i|Pa_i^M, D)^{-1} \)

• For trees (or forests), can use the *minimum spanning tree* algorithm (see Chow & Liu, 1968)
The General Case

- Finding the best structure is NP-hard, if max. number of parents > 1 (Chickering)
- New dynamic programming solutions work up to ~30 variables (Silander & Myllymäki, UAI'2006)
- Heuristics:
  - Greedy bottom-up/top-down
  - Stochastic greedy (with restarts)
  - Simulated annealing and other Monte-Carlo approaches
Local Search

initialize structure

score all possible single changes

any changes better?

yes

perform best change

no

return saved structure

extension: multiple restarts

- prior structure
- empty graph
- max spanning tree
- random graph
Simulated Annealing

initialize structure

pick random change and compute:
\[ p = \exp\left(\frac{\text{score}}{T}\right) \]

quit?

perform change with prob \( p \)

return saved structure

- prior structure
- empty graph
- max spanning tree
- random graph
Evaluation Methodology

Measures of utility of learned network:
- Cross Entropy (Gold standard network, learned network)
- Structural difference (e.g. #missing arcs, extra arcs, reversed arcs,...)
Gold Standard

Prior Network

Learned Network

Data
Problems with the Gold standard methodology

- Structural criteria may not properly reflect the quality of the result (e.g., the relevance of an extra arc depends on the parameters)
- Cross-entropy (Kullback-Leibler distance) hard to compute
- With small data samples, what is the "correct" answer? Why should the learned network be like the generating network?
- Are there better evaluation strategies? How about predictive performance?
Learning with Missing Data
Handling Missing Data

- Different types of missing data: missing completely at random, missing at random, not missing at random
- Latent (hidden) variable models, like the finite mixture model, always have to deal with hidden data
- We either are interested in the missing data (e.g., we could be interested in the values of the a hidden variable if it corresponds to a clustering of data), or it is treated as "nuisance" (e.g., if the hidden "class" variable is only used as a modeling tool to produce a joint probability distribution on the observed variables)
- In the latter case, a Bayesian attempts to marginalize over the hidden data
The Finite Mixture Model

With hidden data imposed by $C$, it is computationally infeasible to compute:
- Maximum likelihood parameters
- Expected parameters (or max. posterior)
- Marginal likelihood

Model "structure" learning: how many values for $C$?

$$P(D) = P(X_1^n, ..., X_5^n)$$
$$= \sum_{C^n} P(C^n) P(X_1^n, ..., X_5^n | C^n)$$
$$= \sum_{C^n} P(C^n) \prod_i P(X_i^n | C^n)$$
K-Means

- Normally, a geometric clustering algorithm
- A probabilistic version:
  1. Start with a random initial clustering $c_1,\ldots,c_n$
  2. Build a model $\Theta$ using complete data $(X^n, C^n)$
  3. Using $\Theta$, assign each data vector $X$ independently to its most probable cluster (i.e., find $\max P(C_i | X_i, \Theta)$ for all $i$)
  4. Go to 2.
Expectation Maximization (EM)

- A "soft" version of K-Means
- Intuitively: data vectors are assigned "fractionally" to each cluster (with the fractions determined by the classification probabilities)
- The new model $\Theta$ is computed from semi-complete data (fractional sufficient statistics)
- For HMMs: the Baum-Welch algorithm
K-Means and EM in practice

- Both provably monotonically improve the likelihood (or posterior), so they converge to a local optimum only
- Convergence can be slow
- To get reasonable results, need to repeat several runs from different starting points
- Can be used together: e.g., first run K-means, then continue with EM
- Can be used to find good starting points for other heuristics
Structure learning with FMM's

- Can find models $\Theta$ using different number of values for the hidden variable (different number of parameters)
- Which $\Theta$ to choose? (max. likelihood chooses always the model obtained with the highest number of parameters)
- Computing the marginal likelihood not feasible with the missing data imposed by the hidden variable

$$P(K|D) \propto P(D|K)P(K)$$
$$P(D|K) = \int P(D|K, \theta)P(\theta|K)d\theta$$
$$P(D|K, \theta) = \prod_i \sum_k P(d_i|c_i, \theta)P(c_i|\theta)$$
Approximating the marginal likelihood

- Laplace (Gaussian) approximation
- Bayesian Information Criterion (BIC)
- Akaike Information Criterion (AIC)
- Missing data completion
- Stochastic methods (MCMC etc.)
- Variational methods
Laplace's method / Gaussian approximation

• Based on Taylor approximation at the maximum likelihood parameters:

\[-\log P(D|M) \approx -\log P(D|M, \hat{\theta}) - \log P(\hat{\theta}|M) + \frac{d}{2} \log \frac{n}{2\pi} + \log \sqrt{|I(\hat{\theta})|}\]

• Here "d" is the number of parameters, "n" is the size of the data, and \(|I(\Theta)|\) is the determinant of the Fisher information matrix at \(\Theta\)

• A "penalized log-likelihood" criterion: likelihood grows with more complex models, but it compensated by the penalizing factors

• Jeffreys' prior: \(P(\theta|M) = \frac{\sqrt{|I(\theta)|}}{\int \sqrt{|I(\theta)|} \, d\theta}\)
BIC and AIC

- **BIC:** \[-\log P(D|M) \approx -\log P(D|M, \hat{\theta}) + \frac{d}{2} \log n\]

- **AIC:** \[-\log P(D|M) \approx -\log P(D|M, \hat{\theta}) + d\]

- Both converge *asymptotically* to the marginal likelihood (minus a constant)

- Hence marginal likelihood is also in a sense a penalized maximum likelihood criterion!

- It is a non-trivial problem to determine the "correct" value of $d$
Missing data completion

- Direct marginalization not feasible:
  \[ P(\mathbf{X}^n|\mathbf{M}) = \sum_{\mathbf{C}^n} P(\mathbf{X}^n, \mathbf{C}^n|\mathbf{M}) = \sum_{\mathbf{C}^n} P(\mathbf{X}^n|\mathbf{C}^n, \mathbf{M}) P(\mathbf{C}^n|\mathbf{M}) \]

- \( \mathbf{C}^n \) is like an unknown "parameter"

- If you cannot marginalize over a parameter, you can try to maximize it
  \[ P(\mathbf{X}^n|\mathbf{M}) \propto \max_{\mathbf{C}^n} P(\mathbf{X}^n|\mathbf{C}^n, \mathbf{M}) P(\mathbf{C}^n|\mathbf{M}) \]

- As the "parameter" \( \mathbf{C}^n \) is actually data, it is easy to think of reasonable "priors" \( P(\mathbf{C}^n|\mathbf{M}) \)

- With fixed \( \mathbf{M} \), \( \mathbf{C}^n \) can be optimized with K-means, EM, or whatever...
Supervised BN Learning

- **Parameter learning**
  - Generative modeling: \( \text{Find } \arg \max_{\theta} P(X^n, C^n| M, \theta) \)
  - Discriminative modeling: \( \text{Find } \arg \max_{\theta} P(C^n|X^n, M, \theta) \)
  - In general, the result is not the same!

- **Structure learning**
  - Generative modeling: \( \text{Find } \arg \max_{M} P(X^n, C^n| M) \)
  - Discriminative modeling: \( \text{Find } \arg \max_{M} P(C^n|X^n, M) \)
  - In general, the result is not the same!
  - Marginal conditional likelihood not feasible
    - Kontkanen et al. (UAI 1999): approximations, connection to cross-validation
Optimizing the conditional likelihood

• Bad news: even for the Naive Bayes model, the maximum of the conditional likelihood cannot be presented in closed form

• Good news: For some Bayesian networks (e.g., NB and TAN), the conditional log-likelihood space is concave (Roos et al., MLJ 2005) → it has a single global optimum

• "Supervised" Naive Bayes = logistic regression

• For model structure learning: marginal conditional likelihood not feasible (Kontkanen et al., UAI 1999)
Models with many hidden nodes

- Is it sensible to first learn a Bayesian network (NP-hard) and then try to transform it to a simpler representation for probabilistic inference (NP-hard)?
- How about learning directly structures where inference is easy?
Dynamic Bayesian networks

- Complex Markov models involving temporal dependencies
Undirected Graphical Models
Definitions of independence

- Following definitions equivalent for $X_1 \perp X_2 \mid Z$:
  - $p(X_1, X_2 \mid Z) = p(X_1 \mid Z)p(X_2 \mid Z)$ whenever $p(Z) > 0$
  - $p(X_1 \mid X_2, Z) = p(X_1 \mid Z)$ whenever $p(X_2, Z) > 0$
  - $p(X_2 \mid X_1, Z) = p(X_2 \mid Z)$ whenever $p(X_1, Z) > 0$
  - $p(X_1, X_2, Z) = f(X_1, Z)g(X_2, Z)$ for non-negative functions $f(\cdot), g(\cdot)$

- Definitions symmetric in $X_1$ and $X_2$
Image models

- The graph on the right says that each pixel is influenced only by its neighbors.
Undirected graphical models

- **Local Markov property:**
  - \( X \perp (G\text{-nbrs}(X)-\{X\}) \mid \text{nbrs}(X) \)
  - Minimal independence properties to uniquely determine a graph

- **Global Markov property:**
  - For all \( X_1, X_2, Z: X_1 \perp X_2 \mid Z \) iff \( X_1 \) is separated in the graph from \( X_2 \) by \( Z \).
  - How to test for independence

- **Functional form:**
  - \( P(X_1, \ldots, X_n) = \prod_C f_C(X_C) \)
  - Product over cliques \( C \) (\( X_C \) denoting the members of the clique)
  - Definition for purposes of computation
For example...

- **Local Markov property:**
  - E.g.: $B \perp E,F \mid A,C,D$; $C \perp A,F \mid B,D,E$;

- **Global Markov property:**
  - E.g.: $A,B \perp E,F \mid C,D$.

- **Functional form:**
  - $P(A,B,C,D,E) = e(A,B)f(B,C,D)g(C,D,E)h(E,F)$
The three properties are equivalent

- Global Markov property implies the local
- Functional form implies the global Markov property
- Hammersley-Clifford theorem: Local Markov property implies the functional form (for discrete variables)
Markov Random Fields

- Undirected graphical models, a.k.a. Markov networks

- Typically use alternative functional form:
  \[
  P(X) = \frac{1}{Z} \exp \left( \sum_C \alpha_C f_C(X_C) \right)
  \]

- Sometimes also called the Gibbs distribution

- The cliquewise functions \( f_C \) are called *clique potentials*

- The normalizer \( Z \) is called the *partition function*
Mapping a DAG to a MRF is possible...

- Mapping is straightforward if a node and its parents in a DAG belong to the same clique in the MRF

$$\prod_i P(X_i | Pa_i) \rightarrow \prod_C f_C(X_C)$$

- This means that to get the corresponding MRF, we need to "marry" nodes with common children (this is called *moralizing* the graph)

- It follows that inference in undirected graphs is NP-hard too...
...but DAGs and MRFs are not equivalent independence models

- $A \perp D \mid B, C$ and $B \perp C \mid A, D$

- $A \perp B$ and $A \not\perp\!\!\!\perp B \mid C$
Final remarks

- The Bayesian framework offers an elegant, consistent formalism for uncertain reasoning

- The basic principle is simple: compute the probability of what you want to know while marginalizing over the other unknown factors

- We have focused on the discrete Dirichlet-multinomial case and directed acyclic graphs (Bayesian networks), but the same principles apply with other probabilistic model families as well

- Graphical models offer a unifying framework where many popular methods are easily understood
  - E.g. Factor analysis, PCA, ICA, mPCA, HMM, Kalman filter, switching Kalman filter, AR models,...
  - See: http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html