Bayesian Networks for Regulatory Network Learning

BMI/CS 576
www.biostat.wisc.edu/bmi576/
Colin Dewey
colin.dewey@wisc.edu
Fall 2016
Part of the E. coli Regulatory Network

Figure from Wei et al., *Biochemical Journal* 2004
Gene Regulation Example: the lac Operon

THE LAC OPERON

RNA polymerase

LacI P_i PO LacZ LacY LacA

this protein regulates the transcription of LacZ, LacY, LacA

These proteins metabolize lactose
lactose is absent \( \Rightarrow \) the protein encoded by lacI represses transcription of the lac operon.
Gene Regulation Example: the lac Operon

lactose is present $\Rightarrow$ it binds to the protein encoded by lacI changing its shape; in this state, the protein doesn’t bind upstream from the lac operon; therefore the lac operon can be transcribed
Gene Regulation Example: the lac Operon

• this example provides a simple illustration of how a cell can regulate (turn on/off) certain genes in response to the state of its environment
  – an operon is a sequence of genes transcribed as a unit
  – the lac operon is involved in metabolizing lactose
    • it is “turned on” when lactose is present in the cell
    • the lac operon is regulated at the transcription level
• the depiction here is incomplete; for example, the level of glucose in the cell also influences transcription of the lac operon
The lac Operon: Activation by Glucose

**THE LAC OPERON**

- RNA polymerase promotes binding by CAP protein; increases transcription

In conditions of low glucose:
- CAP + cAMP

**Regulations**
- glucose absent $\Rightarrow$ CAP protein promotes binding by RNA polymerase; increases transcription

**Diagram:***
- **Regions coding for proteins:**
  - Regions coding for proteins
  - Regulatory regions
  - Diffusable regulatory proteins
- **Gene expression:**
  - CAP
  - P
  - O
  - LacZ
  - LacY
  - LacA

**Legend:**
- mRNA + ribosomes
- Z
- Y
- A
Probabilistic Model of *lac* Operon

• suppose we represent the system by the following discrete variables

  - $L$ (lactose) present, absent
  - $G$ (glucose) present, absent
  - $I$ (lacI) present, absent
  - $C$ (CAP) present, absent
  - lacI-unbound true, false
  - CAP-bound true, false
  - $Z$ (lacZ) high, low, absent

• suppose (realistically) the system is not completely deterministic

• the joint distribution of the variables could be specified by $2^6 \times 3 - 1 = 191$ parameters
Motivation for Bayesian Networks

• Explicitly state (conditional) independencies between random variables
  – Provide a more compact model (fewer parameters)

• Use directed graphs to specify model
  – Take advantage of graph algorithms/theory
  – Provide intuitive visualizations of models
A Bayesian Network for the lac System

**Pr** (\( L \))

<table>
<thead>
<tr>
<th></th>
<th>absent</th>
<th>present</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pr</strong> (( L ))</td>
<td>0.9</td>
<td>0.1</td>
</tr>
</tbody>
</table>

**Pr** (lacI-unbound | \( L \), \( I \))

<table>
<thead>
<tr>
<th>( L )</th>
<th>( I )</th>
<th>true</th>
<th>false</th>
</tr>
</thead>
<tbody>
<tr>
<td>absent</td>
<td>absent</td>
<td>0.9</td>
<td>0.1</td>
</tr>
<tr>
<td>absent</td>
<td>present</td>
<td>0.1</td>
<td>0.9</td>
</tr>
<tr>
<td>present</td>
<td>absent</td>
<td>0.9</td>
<td>0.1</td>
</tr>
<tr>
<td>present</td>
<td>present</td>
<td>0.9</td>
<td>0.1</td>
</tr>
</tbody>
</table>

**Pr** (Z | lacI-unbound, CAP-bound)

<table>
<thead>
<tr>
<th>lacI-unbound</th>
<th>CAP-bound</th>
<th>absent</th>
<th>low</th>
<th>high</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
<td>0.1</td>
<td>0.8</td>
<td>0.1</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>0.1</td>
<td>0.1</td>
<td>0.8</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>0.8</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>0.8</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Bayesian Networks

• Also known as Directed Graphical Models
• a BN is a Directed Acyclic Graph (DAG) in which
  – the nodes denote random variables
  – each node $X$ has a *conditional probability distribution* (CPD) representing $P(X \mid \text{Parents}(X))$

• the intuitive meaning of an arc from $X$ to $Y$ is that $X$ *directly influences* $Y$

• formally: each variable $X$ is independent of its non-descendants given its parents
Bayesian Networks

• a BN provides a factored representation of the joint probability distribution

\[
\Pr(L,I,LU,G,C,CB,Z) = \\
\Pr(L) \times \Pr(I) \times \Pr(LU \mid L,I) \times \Pr(G) \times \Pr(C) \times \Pr(CB \mid G,C) \times \Pr(Z \mid LU,CB)
\]

• this representation of the joint distribution can be specified with 20 parameters (vs. 191 for the unfactored representation)
Representing CPDs for Discrete Variables

- CPDs can be represented using tables or trees
- consider the following case with Boolean variables $A, B, C, D$

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
</tr>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>$F$</td>
<td>$T$</td>
</tr>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>$F$</td>
<td>$F$</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
<td>$F$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
</tr>
</tbody>
</table>
Representing CPDs for Continuous Variables

- we can also model the distribution of continuous variables in Bayesian networks
- one approach: linear Gaussian models

\[ \Pr(X \mid u_1, \ldots, u_k) \sim N(a_0 + \sum_{i} a_i \times u_i, \sigma^2) \]

- \( X \) normally distributed around a mean that depends linearly on values of its parents \( u_i \)
The Inference Task in Bayesian Networks

**Given:** values for some variables in the network (*evidence*), and a set of *query* variables

<table>
<thead>
<tr>
<th>L</th>
<th>G</th>
<th>I</th>
<th>C</th>
<th>lacI-unbound</th>
<th>CAP-bound</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>present</td>
<td>present</td>
<td>present</td>
<td>present</td>
<td>?</td>
<td>?</td>
<td>low</td>
</tr>
</tbody>
</table>

**Do:** compute the posterior distribution over the query variables

\[ P(CAP = \text{bound} = \text{true} \mid L = \text{present}, G = \text{present}, I = \text{present}, C = \text{present}, Z = \text{low}) \]

- variables that are neither evidence variables nor query variables are *hidden* variables
- the BN representation is flexible enough that any set can be the evidence variables and any set can be the query variables
The Parameter Learning Task

- **Given**: a set of training instances, the graph structure of a BN

<table>
<thead>
<tr>
<th>L</th>
<th>G</th>
<th>I</th>
<th>C</th>
<th>lacI-unbound</th>
<th>CAP-bound</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>present</td>
<td>present</td>
<td>present</td>
<td>present</td>
<td>true</td>
<td>false</td>
<td>low</td>
</tr>
<tr>
<td>present</td>
<td>present</td>
<td>present</td>
<td>present</td>
<td>true</td>
<td>false</td>
<td>absent</td>
</tr>
<tr>
<td>absent</td>
<td>present</td>
<td>present</td>
<td>present</td>
<td>false</td>
<td>false</td>
<td>high</td>
</tr>
</tbody>
</table>

- **Do**: infer the parameters of the CPDs

- this is straightforward when there aren’t missing values, hidden variables
The Structure Learning Task

• **Given**: a set of training instances

<table>
<thead>
<tr>
<th>L</th>
<th>G</th>
<th>I</th>
<th>C</th>
<th>lacI-unbound</th>
<th>CAP-bound</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>present</td>
<td>present</td>
<td>present</td>
<td>present</td>
<td>true</td>
<td>false</td>
<td>low</td>
</tr>
<tr>
<td>present</td>
<td>present</td>
<td>present</td>
<td>present</td>
<td>true</td>
<td>false</td>
<td>absent</td>
</tr>
<tr>
<td>absent</td>
<td>present</td>
<td>present</td>
<td>present</td>
<td>false</td>
<td>false</td>
<td>high</td>
</tr>
</tbody>
</table>

• **Do**: infer the graph structure (and perhaps the parameters of the CPDs too)
Bayes Net Structure Learning Case Study: Friedman et al., *JCB* 2000

- expression levels in populations of yeast cells
- 800 genes
- 76 experimental conditions
Learning Bayesian Network Structure

- given a function for scoring network structures, we can cast the structure-learning task as a search problem

figure from Friedman et al., *Journal of Computational Biology*, 2000
Structure Search Operators

add an edge

A
B
C
D

delete an edge

reverse an edge

A
B
C
D
Bayesian Network Structure Learning

- we need a scoring function to evaluate candidate networks; Friedman et al. use one with the form

\[
\text{score}(G : D) = \log \Pr(G \mid D) = \log \Pr(D \mid G) + \log \Pr(G) + C
\]

- where they take a Bayesian approach to computing \( \Pr(D \mid G) \)

\[
\Pr(D \mid G) = \int \Pr(D \mid G, \Theta) \Pr(\Theta \mid G) d\Theta
\]

i.e. don’t commit to particular parameters in the Bayes net
The Bayesian Approach to Structure Learning

• Friedman et al. take a Bayesian approach:

$$\Pr(D \mid G) = \int \Pr(D \mid G, \Theta) \Pr(\Theta \mid G) d\Theta$$

• How can we calculate the probability of the data without using specific parameters (i.e. probabilities in the CPDs)?

• let’s consider a simple case of estimating the parameter of a weighted coin…
The Beta Distribution

- suppose we’re taking a Bayesian approach to estimating the parameter $\theta$ of a weighted coin
- the Beta distribution provides a convenient prior

$$P(\theta) = \frac{\Gamma(\alpha_h + \alpha_t)}{\Gamma(\alpha_h)\Gamma(\alpha_t)} \theta^{\alpha_h-1}(1-\theta)^{\alpha_t-1}$$

where

$\alpha_h$ # of “imaginary” heads we have seen already

$\alpha_t$ # of “imaginary” tails we have seen already

$\Gamma$ continuous generalization of factorial function
The Beta Distribution

- suppose now we’re given a data set $D$ in which we observe $M_h$ heads and $M_t$ tails

$$P(\theta \mid D) = \frac{\Gamma(\alpha + M_h + M_t)}{\Gamma(\alpha_h + M_h)\Gamma(\alpha_t + M_t)} \theta^{\alpha_h + M_h - 1} (1 - \theta)^{\alpha_t + M_t - 1}$$

$$= \text{Beta}(\alpha_h + M_h, \alpha_t + M_t)$$

- the posterior distribution is also Beta: we say that the set of Beta distributions is a *conjugate* family for binomial sampling
The Beta Distribution

• assume we have a distribution $P(\theta)$ that is Beta($\alpha_h$, $\alpha_t$)

• what’s the marginal probability (i.e. over all $\theta$) that our next coin flip would be heads?

$$P(X = \text{heads}) = \int_0^1 P(X = \text{heads} \mid \theta) \ P(\theta) \ d\theta$$

$$= \int_0^1 \theta \ P(\theta) \ d\theta = \frac{\alpha_h}{\alpha_h + \alpha_t}$$

• what if we ask the same question after we’ve seen $M$ actual coin flips?

$$P(X_{M+1} = \text{heads} \mid x_1, \ldots, x_M) = \frac{\alpha_h + M_h}{\alpha_h + \alpha_t + M}$$
The Dirichlet Distribution

- for discrete variables with more than two possible values, we can use *Dirichlet* priors

- Dirichlet priors are a *conjugate* family for multinomial data

\[
P(\theta) = \frac{\Gamma\left(\sum_i \alpha_i\right)}{\prod_i \Gamma(\alpha_i)} \prod_{i=1}^{K} \theta_i^{\alpha_i-1}
\]

- if \(P(\theta)\) is Dirichlet(\(\alpha_1, \ldots, \alpha_K\)), then \(P(\theta|D)\) is Dirichlet(\(\alpha_1+M_1, \ldots, \alpha_K+M_K\)), where \(M_i\) is the # occurrences of the \(i^{th}\) value
The Bayesian Approach to Scoring BN Network Structures

\[ \Pr(D \mid G) = \int \Pr(D \mid G, \Theta) \Pr(\Theta \mid G) d\Theta \]

- we can evaluate this type of expression fairly easily because
  - *parameter independence*: the integral can be decomposed into a product of terms: one per variable
  - Beta/Dirichlet are conjugate families (i.e. if we start with Beta priors, we still have Beta distributions after updating with data)
  - the integrals have closed-form solutions
Scoring Bayesian Network Structures

• when the appropriate priors are used, and all instances in $D$ are complete, the scoring function can be decomposed as follows

$$
\text{score}(G : D) = \sum_i \text{Score}(X_i, \text{Parents}(X_i) : D)
$$

• thus we can
  – score a network by summing terms (computed as just discussed) over the nodes in the network
  – efficiently score changes in a local search procedure
Bayesian Network Search: The Sparse Candidate Algorithm
[Friedman et al., UAI 1999]

Given: data set $D$, initial network $B_0$, parameter $k$

Loop for $n = 1, 2, \ldots$ until convergence

Restrict

Based on $D$ and $B_{n-1}$, select for each variable $X_i$ a set $C^n_i$ ($|C^n_i| \leq k$) of candidate parents.
This defines a directed graph $H_n = (\mathcal{X}, E)$, where
$E = \{X_j \rightarrow X_i | \forall i, j, X_j \in C^n_i\}$.
(Note that $H_n$ is usually cyclic.)

Maximize

Find network $B_n = \langle G_n, \Theta_n \rangle$ maximizing $\text{Score}(B_n \mid D)$ among networks that satisfy $G_n \subset H_n$ (i.e., $\forall X_i$, $\text{Pa}^{G_n}(X_i) \subseteq C^n_i$).

Return $B_n$
The Restrict Step In Sparse Candidate

- to identify candidate parents in the first iteration, can compute the \textit{mutual information} between pairs of variables

\[ I(X, Y) = \sum_{x, y} \hat{P}(x, y) \log \frac{\hat{P}(x, y)}{\hat{P}(x)\hat{P}(y)} \]

- where \( \hat{P} \) denotes the probabilities estimated from the data set
The Restrict Step In Sparse Candidate

• suppose true network structure is:

• we’re selecting two candidate parents for $A$ and
  $I(A;C) > I(A;D) > I(A;B)$

• the candidate parents for $A$ would then be $C$ and $D$; how could we get $B$ as a candidate parent on the next iteration?
The Restrict Step In Sparse Candidate

- **Kullback-Leibler (KL) divergence** provides a distance measure between two distributions, $P$ and $Q$

  \[ D_{KL}(P(X) \| Q(X)) = \sum_x P(x) \log \frac{P(x)}{Q(x)} \]

- Mutual information can be thought of as the KL divergence between the distributions
  \[ \hat{P}(X, Y) \]
  \[ \hat{P}(X)\hat{P}(Y) \] (assumes $X$ and $Y$ are independent)
The Restrict Step In Sparse Candidate

• we can use KL to assess the discrepancy between the network’s estimate $P_{net}(X, Y)$ and the empirical estimate

$$M(X,Y) = D_{KL}(\hat{P}(X,Y)) \| P_{net}(X,Y)$$

true distribution

```
  D
   \downarrow
    C
     \downarrow
      B
```

```
  B
  \downarrow
  A
```

current Bayes net

```
  B
  \downarrow
  D
    \downarrow
    C
```

```
  A
```

$$D_{KL}(\hat{P}(A,B)) \| P_{net}(A,B)$$
The Restrict Step in Sparse Candidate

Input:

- Data set $D = \{x^1, \ldots, x^N\}$,
- A network $B_n$,
- a score
- parameter $k$.

Output: For each variable $X_i$, a set of candidate parents $C'_i$ of size $k$.

Loop for each $X_i$, $i = 1, \ldots, n$

- Calculate $M(X_i, X_j)$ for all $X_j \neq X_i$ such that $X_j \notin \text{Pa}(X_i)$
- Choose $x_1, \ldots, x_{k-l}$ with highest ranking, where $l = |\text{Pa}(X_i)|$
- Set $C'_i = \text{Pa}(X_i) \cup \{x_1, \ldots, x_{k-l}\}$

Return $\{C'_i\}$

important to ensure monotonic improvement
The Maximize Step in Sparse Candidate

- hill-climbing search with *add-edge*, *delete-edge*, *reverse-edge* operators
- test to ensure that cycles aren’t introduced into the graph
# Efficiency of Sparse Candidate

<table>
<thead>
<tr>
<th></th>
<th>possible parent sets for each node</th>
<th>changes scored on first iteration of search</th>
<th>changes scored on subsequent iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>ordinary greedy search</td>
<td>$O(2^n)$</td>
<td>$O(n^2)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>greedy search w/at most $k$ parents</td>
<td>$O\left(\binom{n}{k}\right)$</td>
<td>$O(n^2)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Sparse Candidate</td>
<td>$O(2^k)$</td>
<td>$O(kn)$</td>
<td>$O(k)$</td>
</tr>
</tbody>
</table>
Bayes Net Structure Learning Case Study: Friedman et al., *JCB* 2000

- expression levels in populations of yeast cells
- 800 genes
- 76 experimental conditions
- used two representations of the data
  - discrete representation (underexpressed, normal, overexpressed) with CPTs in the models
  - continuous representation with linear Gaussians
Bayes Net Structure Learning Case Study: Two Key Issues

• Since there are many variables but data is sparse, there is not enough information to determine the “right” model. Instead, can we consider many of the high-scoring networks?

• How can we tell if the structure learning procedure is finding real relationships in the data? Is it doing better than chance?
Representing Partial Models

- How can we consider many high-scoring models? Use the bootstrap method to identify high-confidence features of interest.

- Friedman et al. focus on finding two types of “features” common to lots of models that could explain the data
  - Markov relations: is $Y$ in the Markov blanket of $X$?
  - order relations: is $X$ an ancestor of $Y$
Markov Blankets

- every other node $Y$ in the network is conditionally independent of $X$ when conditioned on $X$’s Markov blanket $\text{MB}(X)$

\[
\Pr(X \mid \text{MB}(X), Y) = \Pr(X \mid \text{MB}(X))
\]

- the Markov blanket for node $X$ consists of its parents, its children, and its children’s parents
Markov Blankets

• why are parents of $X$’s children in its Markov blanket?
• suppose we’re using the following network to infer the probability that it rained last night

we observe the grass is wet; is the Sprinkler-on variable now irrelevant?

no – if we observe that the sprinkler is on, this helps to “explain away” the grass being wet
Estimating Confidence in Features: The *Bootstrap* Method

- for $i = 1$ to $m$
  - randomly draw sample $S_i$ of $N$ expression experiments from the original $N$ expression experiments with replacement
  - learn a Bayesian network $B_i$ from $S_i$

- some expression experiments will be included multiple times in a given sample, some will be left out.
- the confidence in a feature is the fraction of the $m$ models in which it was represented
Permutation Testing: Do the Networks Represent Real Relationships

• how can we tell if the high-confidence features are meaningful?

• compare against confidence values for randomized data – genes should then be independent and we shouldn’t find “real” features

\[
\begin{array}{ccc}
  g_{1,1} & \cdots & g_{1,n} \\
  g_{2,1} & \cdots & g_{2,n} \\
  g_{m,1} & \cdots & g_{m,n}
\end{array}
\]

randomize each row independently
Confidence Levels of Features:
Real vs. Randomized Data

Markov features

order features

figure from Friedman et al., *Journal of Computational Biology, 2000*
Bayes Net Structure Learning Case Study: Sachs et al., Science 2005

- measured
  - levels of key molecules in (thousands of) single cells, discretized to low, medium, high
  - 11 phosphoproteins and phospholipids

Figure from Sachs et al., Science, 2005
A Signaling Network

Figure from Sachs et al., Science 2005
Causality

• given only *observational* data, there are many cases in which we can’t determine causality

<table>
<thead>
<tr>
<th>Smokes</th>
<th>Lung-Cancer</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

• these two networks explain the observations in the table equally well
How to Get at Causality?

- Observing how events are related in time can provide information about causality.
How to Get at Causality?

• Interventions -- manipulating variables of interest -- can provide information about causality.
Sachs et al. Computational Experiments

- Simulated-annealing search with *add-edge*, *delete-edge*, *reverse-edge* operators
- repeated 500 times with different initial random graphs
- final model includes edges with confidence > 85%

- to evaluate importance of large data set of single-cell measurements with interventions, constructed control data sets
  1. small
  2. observation-only
  3. population-average
Interventions
Evaluating the Model Learned by Sachs et al.
The Value of Interventions, Data Set Size and Single-Cell Data

<table>
<thead>
<tr>
<th></th>
<th>A: Lacking Intervention data</th>
<th>B: Truncated Dataset</th>
<th>C: Averaged Dataset</th>
<th>Complete Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expected</td>
<td>7/10</td>
<td>7/14</td>
<td>6/16</td>
<td>14/17</td>
</tr>
<tr>
<td>Reported</td>
<td>1/10</td>
<td>1/14</td>
<td>1/16</td>
<td>2/17</td>
</tr>
<tr>
<td>Reversed</td>
<td>N/A</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Unexplained</td>
<td>2</td>
<td>6</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Missed</td>
<td>11</td>
<td>10</td>
<td>12</td>
<td>4</td>
</tr>
</tbody>
</table>
Summary of BN Structure Learning

- structure learning often cast as a search problem
- the *sparse candidate* algorithm is more efficient than a naïve greedy search -- takes advantage of assumption that networks in this domain are sparsely connected
- we can score candidate networks efficiently because of *parameter independence* (overall score is a sum of local terms)
- we can score candidate networks efficiently with a Bayesian approach if we use conjugate priors
- high scoring network structures can be characterized using a *bootstrapping* approach that considers *order* and *Markov-blanket* features
- the significance of such features can be calculated using *permutation testing*
- we can gain more information about causality via time-series and *interventions*