Outline

• Algorithms for finding MAP structure (18.4)
• MCMC over DAG structure (18.5)
• Dynamic programming + MH
• Stochastic search
Computationally intractable

- There are $O(d!2^{\binom{d}{2}})$ DAGs on $d$ nodes

<table>
<thead>
<tr>
<th>$d$</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>#G($d$)</td>
<td>3</td>
<td>25</td>
<td>543</td>
<td>29,281</td>
<td>3,781,503</td>
<td>1.1e9</td>
<td>7.8e11</td>
<td>1.2e15</td>
</tr>
</tbody>
</table>
Trees

- Can learn optimal tree using MST algo in $O(n^2 \log n + n^2 M)$ time, $n=$#num nodes, $M=$#cases

**Definition 18.4.1:** A network structure $\mathcal{G}$ is called tree-structured if each variable $X$ has at most one parent in $\mathcal{G}$, i.e., $|\text{Pa}_X^\mathcal{G}| \leq 1$.

$$
\Delta(\mathcal{G}) = \text{score}(\mathcal{G} : D) - \text{score}(\mathcal{G}_0 : D)
$$

$$
\Delta(\mathcal{G}) = \sum_{i, \text{Pa}_i^\mathcal{G} \neq \emptyset} (\text{FamScore}(X_i | \text{Pa}_i^\mathcal{G} : D) - \text{FamScore}(X_i : D))
$$

$$
\Delta(\mathcal{G}) = \sum_{X_j \rightarrow X_i \in \mathcal{G}} w_{X_j \rightarrow X_i}
$$

**Undirected max weight spanning tree**

Score equivalence $\Rightarrow$

$$
w_{X_i \rightarrow X_j} = w_{X_j \rightarrow X_i}
$$

where we choose $\text{mhi}$

$$
\text{score}_L(\mathcal{G}_1 : D) - \text{score}_L(\mathcal{G}_0 : D) = M \sum_{x, y} \hat{P}(x, y) \log \frac{\hat{P}(y | x)}{\hat{P}(y)} = M \cdot I_p(X;Y)
$$
TAN classifiers

• Tree-augmented naïve Bayes
• Can learn tree structure for each class conditional density
Mixtures of trees

• We can fit mixtures of trees using EM: just run MST algorithm in M step
• Analogous to mixture of diagonal Gaussians
DAG with known order

- Can find optimal set of parents for each node independently

\[\text{Proposition 18.4.2: Let } \prec \text{ be an ordering over } \mathcal{X}, \text{ and let } \text{score}(\mathcal{G} : \mathcal{D}) \text{ be a decomposable score. If we choose } \mathcal{G} \text{ to be the network where} \]
\[\text{Pa}_i^\mathcal{G} = \arg\max_{U \subseteq \{X_j : X_j \prec X_i\}} \text{FamScore}(X_i \mid U_i : \mathcal{D})\]
for each \(i\), then \(\mathcal{G}\) maximizes the score among the structures consistent with \(\prec\).

- If at most \(d\) parents, last node \(X_n\) must select from
\[X_n \text{ is } 1 + \binom{n-1}{2} + \cdots + \binom{n-1}{d} = O(d^{n-1})\]
- If CPDs are GLMs, can use lasso to find parents
- If order unknown, can search over orders.
Dependency networks

- A depnet is a set of full conditionals \( p(X_i|X(-i)) \) learned independently.
- There may not be any joint which is consistent with these conditionals.
- However, one can define a (non-unique) joint by using an ordered Gibbs sampler.
- If the conditionals are learned from (lots of) data, they are likely to be consistent.
- By performing variable selection at each node independently, we get a sparse graph.
- Provides a fast way to visualize dependencies.
Collaborative filtering

- One successful application of depnets is CF.
- $X_i = 1$ if item $i$ has been bought, $X_i = 0$ otherwise.
- Assume $S =$ set of bought items, $Sbar =$ not bought items, $i =$ target item. Compute $p(X_i|S=1,Sbar=0)$.
- In a depnet, this is a simple lookup – all other nodes are observed.
- In a DGM, this is also fairly simple – product of CPDs in the Markov blanket.
- Both techniques have similar predictive accuracy, but depnet is much faster to learn.
- Ships in Microsoft’s ecommerce package.
DAG with unknown order

• Thm 18.4.3 It is NP-hard to find the optimal DAG with $d \geq 2$ parents.
• Standard approach: heuristic local search (e.g., hill climbing), using add/delete/reverse edge ($n^2$ neighbors to each DAG).
• Diameter of space is $O(n^2)$: to get from $G_1$ to $G_2$, delete all edges of $G_1$ then add all edges of $G_2$.
• If too many neighbors, use first best instead of evaluating all of them.
• Often there will be large plateaus of I-equivalent DAGs. Can use tabu search to escape these.
• Multiple restarts or data perturbation.
Data perturbation

- Can be used to escape local minima for many ML algorithms, where score = \( \text{sum}_i \text{score}(D_i) \)
- Idea: use weights \( w_i \), and perturb them at random (or more cleverly – rather like boosting)

```plaintext
1   \( G \leftarrow \text{Search}(G_\emptyset, D, \text{score}, O) \)
2   \( G_{\text{best}} \leftarrow G \)
3   \( t \leftarrow t_0 \)
4   \text{for } i = 1, \ldots \text{ until convergence} \\
5       \( D' \leftarrow \text{Perturb}(D, t) \)
6   \( G \leftarrow \text{Search}(G, D', \text{score}, O) \)
7   \text{if } \text{score}(G : D) > \text{score}(G_{\text{best}} : D) \text{ then} \\
8       \( G_{\text{best}} \leftarrow G \)
9   \( t \leftarrow \gamma \cdot t \)
10  \text{return } G_{\text{best}} 
```
\[ \delta(G : o) = \text{score}(o(G) : D) - \text{score}(G : D) \]

to be the change of score associated with applying \( o \) on \( G \). Using score decomposition, we can compute this quantity relatively efficiently.

**Proposition 18.4.4:** Let \( G \) be a network structure, and \( \text{score} \) be a decomposable score.

- If \( o \) is “Add \( X \rightarrow Y \)”, and \( X \rightarrow Y \notin G \), then
  \[ \delta(G : o) = \text{FamScore}(Y, \text{Pa}_Y^G \cup \{X\} : D) - \text{FamScore}(Y, \text{Pa}_Y^G : D) \]

- If \( o \) is “Delete \( X \rightarrow Y \)” and \( X \rightarrow Y \in G \), then
  \[ \delta(G : o) = \text{FamScore}(Y, \text{Pa}_Y^G - \{X\} : D) - \text{FamScore}(Y, \text{Pa}_Y^G : D) \]

- If \( o \) is “Reverse \( X \rightarrow Y \)” and \( X \rightarrow Y \in G \), then
  \[ \delta(G : o) = \text{FamScore}(X, \text{Pa}_X^G \cup \{Y\} : D) + \text{FamScore}(Y, \text{Pa}_Y^G - \{X\} : D) \]
  \[ - \text{FamScore}(X, \text{Pa}_X^G : D) - \text{FamScore}(Y, \text{Pa}_Y^G : D) \]
Efficient update of cached scores

- After accepting change, only have to update scores of affected families - O(n) operators

Proposition 18.4.5: Let $G$ and $G'$ be two network structures, and score be a decomposable score.

- If $o$ is either “Add $X \rightarrow Y$” or “Delete $X \rightarrow Y$” and $Pa_Y^G = Pa_Y^{G'}$, then $\delta(G : o) = \delta(G' : o)$.

- If $o$ is “Reverse $X \rightarrow Y$”, $Pa_Y^G = Pa_Y^{G'}$, and $Pa_X^G = Pa_X^{G'}$, then $\delta(G : o) = \delta(G' : o)$. 

Sufficient statistics

• Need to walk over M rows for all the columns in a given family
• If we need to update n operators, this is $O(nM)$ time
• Can use AD-trees for discrete data, or KD-trees for cts data, to do this more efficiently (possibly subject to approximation error)
Heaps

• Need to search over $O(n^2)$ operators to find best at each step
• Can use a heap to find the best in $O(1)$ time if we do $O(n \log n)$ time to update it when scores change
Learning params vs structure

- ICU Alarm network
Dynamic programming (DP)

- Can analytically marginalize over $d!$ orderings and all possible subsets in $O(d \cdot 2^d)$ time/space using DP and fast Mobius transform.
- Since order of parents does not matter, e.g., $p(X_1|X_2,X_3) = p(X_1|X_3,X_2)$, we can share work.
- Can find exact global MAP DAG

Ordered permutation tree

Unordered permutation lattice

Koivisto & Sood, JMLR, 2004
Equivalence classes

• Can search through PDAG space - smaller than DAG space, and fewer (if any) plateau
• To evaluate score of a PDAG, convert to a DAG then use score for DAG
• To find neighbors: convert PDAG to DAG, add or delete edge; this changes skeleton hence moves to a new PDAG
• Greedy Equivalence Search: start with empty PDAG, add best edge until local max, then delete best edge till local max. If M→infty, this will provably find optimal PDAG given any consistent scoring fn.
• Performing local updates to score of a PDAG is harder.
Bayes model averaging

• When the sample size is small, the posterior $p(G|D)$ gives support to multiple (non equivalent) models

• We should perform BMA when performing prediction

\[
\]

• And when computing $E[f(G)|D]$, where $f(G)$ is some feature, eg $f(G)=$there is an edge X->Y in G, average path length in G

\[
E_{P(G|D)}[f(G)] = \sum_G f(G) P(G | D).
\]
MC3

• Markov Chain Monte Carlo Model Composition
• Use MH in space of DAGs, with proposal = uniform over neighbors (add/delete/reverse edge)
• Does not mix well in more than ~10 dimensions. Also, posterior gets more peaky as sample size increases (can use parallel tempering).
MH on Alarm

Init empty

N=500

(a)

(b)

(c)

N=1000

(a)

(b)

Edge(marginals)
MH in order space

• Given a known order, we can integrate over all possible graphs by summing over all parents sets.
• Hence use MH to sample over orders, using traveling-salesman like moves.

\[
(X_{t_1} \ldots X_{t_j} \ldots X_{t_d} \ldots X_{t_n}) \leftrightarrow (X_{t_1} \ldots X_{t_d} \ldots X_{t_j} \ldots X_{t_n}).
\]

\[
\min \left[ 1, \frac{P(\zeta', D) T^Q(\zeta' \rightarrow \zeta)}{P(\zeta, D) T^Q(\zeta \rightarrow \zeta')} \right]
\]

Target distribution

\[
P(D | \zeta) = \sum_{g \in \mathcal{G}_{d, \zeta}} \prod_i \exp\{\text{FamScore}_B(X_i | \text{Pa}_{X_i}^g : D)\}
\]

\[
= \prod_i \sum_{U_i \in \mathcal{U}_{i, \zeta}} \exp\{\text{FamScore}_B(X_i | U_i : D)\}
\]

\[
\mathcal{U}_{i, \zeta} = \{U : U \prec X_i, |U| \leq k \}.
\]
Posterior features

• Given samples from $p(<|D)$ we compute

$$P(f \mid D) \approx \frac{1}{T} \sum_{t=1}^{T} P(f \mid D, <_t).$$

• Parent features

\[ P(P_{\alpha_i} = U \mid D, <) = \frac{\exp\{\text{FamScore}_B(X_i \mid U : D)\}}{\sum_{U' \in u_{i,<}} \exp\{\text{FamScore}_B(X_i \mid U' : D)\}}. \]

• Edge features

\[ P(X_j \in P_{\alpha_i} \mid <, D) = \frac{\sum_{U \in u_{i,<}} : x_j \in U} {\sum_{U \in u_{i,<}} \exp\{\text{FamScore}_B(X_i \mid U : D)\}}. \]

• General features: sample $G$ given $<$, then use

\[ P(f \mid <, D) = \frac{P(f, D \mid <)}{P(D \mid <)}. \]

\[ P(f, D \mid <) = \sum_{G \in G_{d,<}} f(G)P(G \mid <)P(D \mid G). \]
RB MH on Alarm
Dynamic programming

• Koivisto & Sood showed how to compute all edge marginals $p(G_{ij}=1|D)$ exactly in $O(n \ 2^n)$ time.
• Requires special (“modular”) prior $p(G)$ which can be unnatural (see later).
Comparison of approaches

\[ \sum_{ij} |\hat{P}(G_{ij} = 1|D) - P(G_{ij} = 1|D)| \]

5 sec (d=10) to 5 mins (d=20)
Error floor due to wrong $p(G)$

Error due to wrong $p(G)$
p(G) needed by DP and MH+order

- Joint (“modular”) prior on G and ≺

\[
p(G, \prec) = \frac{1}{Z} \prod_{i=1}^{d} \rho_i(G_i) q_i(\prec_i) I(G, \prec \text{ is valid})
\]

 unordered set of parents
ordering of predecessors

- Induced prior on p(G)

\[
p(G') = \sum_{\prec} p(G, \prec)
\]
Graphs consistent with more orderings are more probable

\[ P \left( \frac{X_1 \leftarrow X_2 \downarrow X_3} {X_2 \leftarrow X_3} \right) > P \left( X_1 \rightarrow X_2 \rightarrow X_3 \right) \]

- Effect will not get erased even with infinite data, since both models are likelihood equivalent
Problems with induced $p(G)$

- Prior is highly non-uniform
- Effect will not get erased even with infinite data
- Cannot encode arbitrary prior knowledge in $p(G)$

$$p(G) = \sum_{\prec} \frac{1}{Z} \prod_{i=1}^{d} \rho_i(G_i) q_i(\prec_i) I(G, \prec \text{ is valid})$$

$\rho_i(G_i) = 1$

$$\rho_i(G_i) = \left(\frac{d - 1}{|G_i|}\right)^{-1}$$
Solutions to p(G) problem

- Importance sampling -- Ellis & Wong ’06
  - Use MH+order as proposal
  - #P-hard to compute exact IS weights

$$w(G) = \frac{p^*(G)}{p(G)} = \frac{\frac{1}{Z} \prod_i \rho_i^*(G_i)}{\sum_\prec \frac{1}{Z} \prod_{i=1}^d \rho_i(G_i) q_i(\prec_i) I(G, \prec \text{ is valid})}$$

$$= \frac{1}{\sum_\prec I(\text{consistent}(\prec, G))}$$
Solutions to $p(G)$ problem

- Importance sampling -- Ellis & Wong ’06
- Metropolis Hastings -- this paper
  - Use DP marginals as proposal for MH
MH with DP+local proposal

- Compute $p_{ij} = p(G_{ij} = 1 | D)$ offline using DP
- $\text{wp } \beta$, we use a standard local move
- $\text{wp } 1-\beta$, sample a new graph $\sim p_{ij}$
- If $\beta=0$ (global) independence sampler
- If $\beta=1$ (local) standard proposal
Why MH?

• DP alone has 3 problems
  1. Modular prior $p(G)$
  2. Cannot compute prob. of “long range” features (e.g., path from i to j), only edge features.
  3. Very slow to compute predictive density
     $p(x|D) = \sum_G p(x|G) p(G|D)$
MH allows any $p(G)$

- Propose using $q(G'|G)$
- Accept with probability $\alpha$

$$\alpha = \min \left(1, \frac{p(D|G')p(G') q(G|G')}{p(D|G)p(G) q(G'|G)}\right)$$
Modular vs uniform $p(G)$

5 node “cancer” network

Markov equivalence class

Modular prior biases posterior even as $|D| \rightarrow \infty$

MH fixes bias
T-cell signaling network

Protein phosphorylation (d=11, N=5400)

Ground truth DAG

Exact $P(G_{ij}=1|D)$

“Causal Protein-Signaling Networks derived from Multiparameter Single-Cell Data”, Sachs, Perez, Pe’er, Lauffenberger, Nolan, Science 2005

ROC

$P(G|\text{unif})$
Informative \( p(G) \)

Protein phosphorylation (\( d=11, N=5400 \))

Ground truth DAG

ROC

\[ p(G|\text{KEGG})? \]

"Reconstructing Gene Regulatory Networks with Bayesian Networks by Combining Expression Data with Multiple Sources of Prior Knowledge", Werhli & Husmeier, 2007
Sampling G allows any features

- DP can only compute posterior of features that are functions of a local family topology

**Possible**
- Specific parents
- ∃ Edge A→C
- A in Markov blanket of C

**Not possible**
- ∃ Causal path A→B, mediated by C
- ∃ Directed path A → B

- By sampling DAGs, we can compute E[f(G)] for arbitrary features f
Posterior features

- We sampled $N=10k$ data from $d=20$ node graph with random CPTs
- Compute $p(\text{edge } i \rightarrow j|D)$ and $p(\text{path } i \rightarrow j|D)$
AUC for $p(\text{feature}=1|D)$

Area under the ROC curve after 200 seconds of wall clock time*

All algorithms were implemented in Matlab/C and run on a standard desktop
Sampling G allows fast prediction

- DP can compute the marginal likelihood of data $p(D)$
- Hence can compute the predictive likelihood of a test point $x$:
  $$p(x|D) = \frac{p(x,D)}{p(D)}$$

- Since DP integrates out $G$, we have to keep $D$, and re-run algorithm for each $x$, which is very slow

- Our approach: keep a sample of $G^s \sim p(G|D)$ and compute posterior mean parameters $\overline{\theta}^s$ for each $G^s$

$$p(x|D) = \sum_G \int_{\theta} p(x|G, \theta)p(\theta|G, D)p(G|D) \approx \frac{1}{M} \sum_{s=1}^{M} p(x|G^s, \overline{\theta}^s)$$
US census data (d=15, N=49k)

1. Exact BMA (but takes 350h!)
2. MH-DP hybrid $\beta=0.1$
3. Plug-in MAP-optimal DAG, MH-DP global $\beta=0$, MH-order
4. MH-local $\beta=1$
5. Gibbs
Why MH+DP?

- MH + DP mixes faster than MH + other
\[ \sum_{i,j} |p(G_{ij} = 1| D) - \hat{p}_t(G_{ij} = 1| D)| \]

d=5 cancer network
Traceplots of log p(G,D)

US census (d=14, N=49k)
Repeatability

MH+DP (hybrid)  MH-order  MH-local

US census (d=15, N=49k)

We plot edge marginals after two runs from different random starting points
Stochastic search

- MCMC approximates $p(M=m|D)$ by counting how many samples are equal to $m$.
- Since we can compute $p(m,D)$ exactly, we don’t need to visit $m$ more than once. We can approximate

$$p(m|D) \approx \frac{p(m, D)}{\sum_{m' \in S} p(m', D)}$$

- It is better to rapidly move through model space, covering as much posterior mass as possible.
- Shotgun stochastic search (SSS), mode oriented stochastic search (MOSS)
Occam’s window

• Goal: compute level set of the posterior

\[ C(\alpha) = \{ m : p(m|D) \geq \alpha p(m^*|D) \} \]

• \( M^* \) is unknown, so approximate this by

\[ \hat{C}(\alpha) = \{ m : p(m|D) \geq \alpha p(\hat{m}^*|D) \} \]

\[ \hat{m}^* = \arg \max_{m \in S} p(m|D) \]

• Can find this by beam search, throwing out models that are worse than \( \alpha \) time the current best (Raftery, Dobra)