COMP4680/8650 Advanced Topics in Statistical Machine Learning

Week 7: Model Structure Learning

Stephen Gould

Parameter versus Structure Learning

So far, during learning, we have assumed that the structure of the Bayesian network is known (or chosen a priori even if incorrect).

In that case we only have to estimate the parameters of the local conditional probability distributions. E.g., with fully observed data we could use maximum likelihood estimation, $\theta^\star = \operatorname{argmax}_\theta L(\theta; D)$.

We now turn to the problem of estimating the **structure** of the Bayesian network from data, i.e. $G^\star = \operatorname{argmax}_G p(G \mid D)$.

Goals of Structured Learning

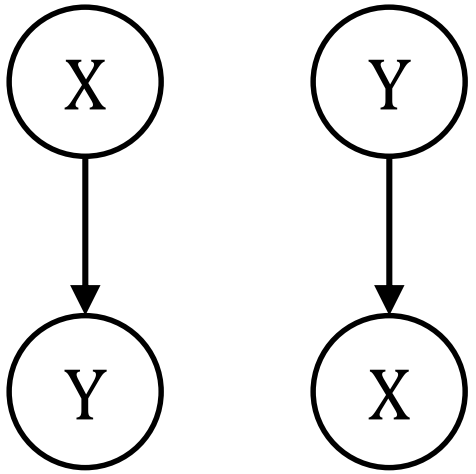
- Knowledge discovery: understanding the dependence structure of variables in a problem.
  - Both direct and indirect (e.g., reasoning over paths)
  - Some structures encode the same independence assumptions so we can only hope to recover the equivalence class.

- Density estimation: estimate a statistical model of the data generation process.
  - The key here is to be able to generalize to new instances.
  - Thus we may prefer sparse structures so as not to overfit.

Identifiability and Markov Equivalence

Two graphs are said to be **Markov equivalent** (or I-equivalent) if they induce the exact same independence assumptions.

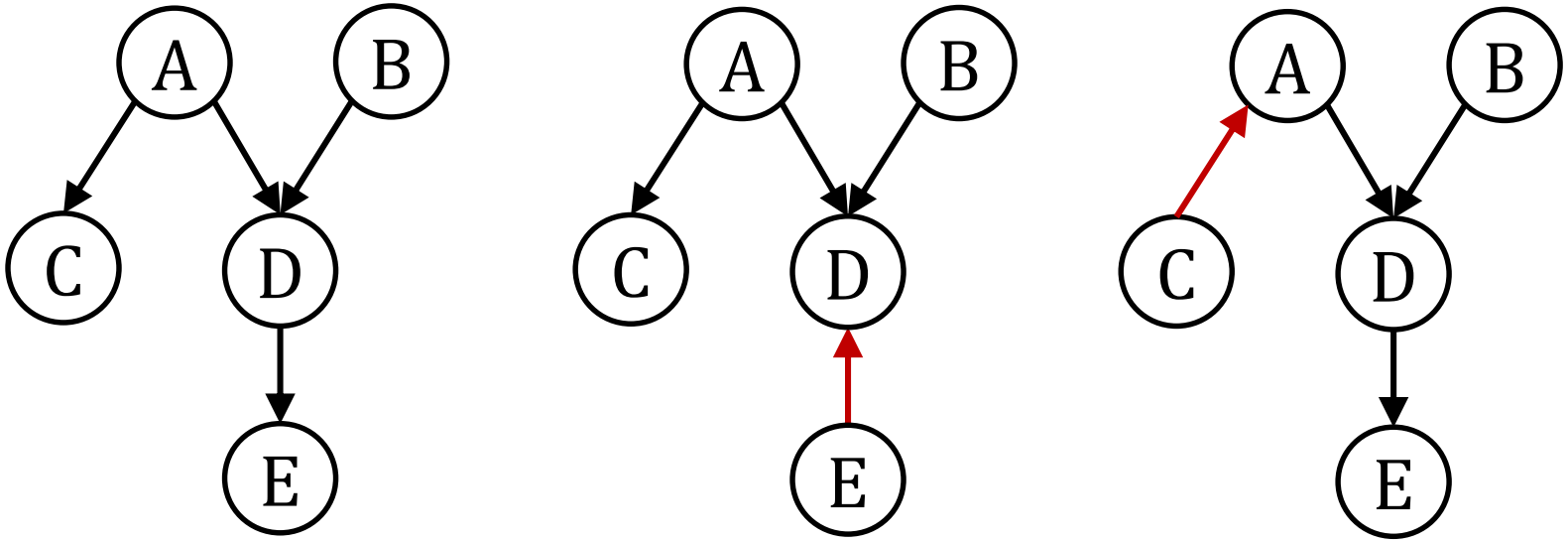
There is no intrinsic property of the data (or distribution) that would allow us to prefer one graph over an equivalent one.



$$P(X,Y) = P(X)P(Y \mid X)$$
or
$$P(X,Y) = P(Y)P(X \mid Y)?$$

Markov Equivalence

Theorem: Two structures are Markov equivalent if and only if they have the same undirected skeleton and the same v-structures.



(write out all the independence assumptions)

Score-based Structure Learning

Score-based methods define a **hypothesis space** of possible models and a **scoring function** that measures how well the model fits the data.

The goal is to find the highest scoring model in the hypothesis space. The space of Bayesian networks, however, is combinatorial.

- For n variables we have $2^{O(n^2)}$ Bayesian network structures.

The problem, in general, is NP-hard and we have to resort to approximate discrete search algorithms.

Discrete (Local) Search

Standard discrete search methods can be thought of a exploring the search space by expanding a tree.

- Each node in the tree represents an element in the search space
- Children of a node are formed by a set of changes (called moves or heuristics) that can be applied to an element to generate another element. Sometimes called expanding a node.

Small state spaces can be explored exhaustively via depth-first or breadth-first search. Larger state spaces lead to approximate solutions using more sophisticated algorithms such heuristic search, stochastic search, tabu search, or branch-and-bound.

Discrete Local Search Assumptions

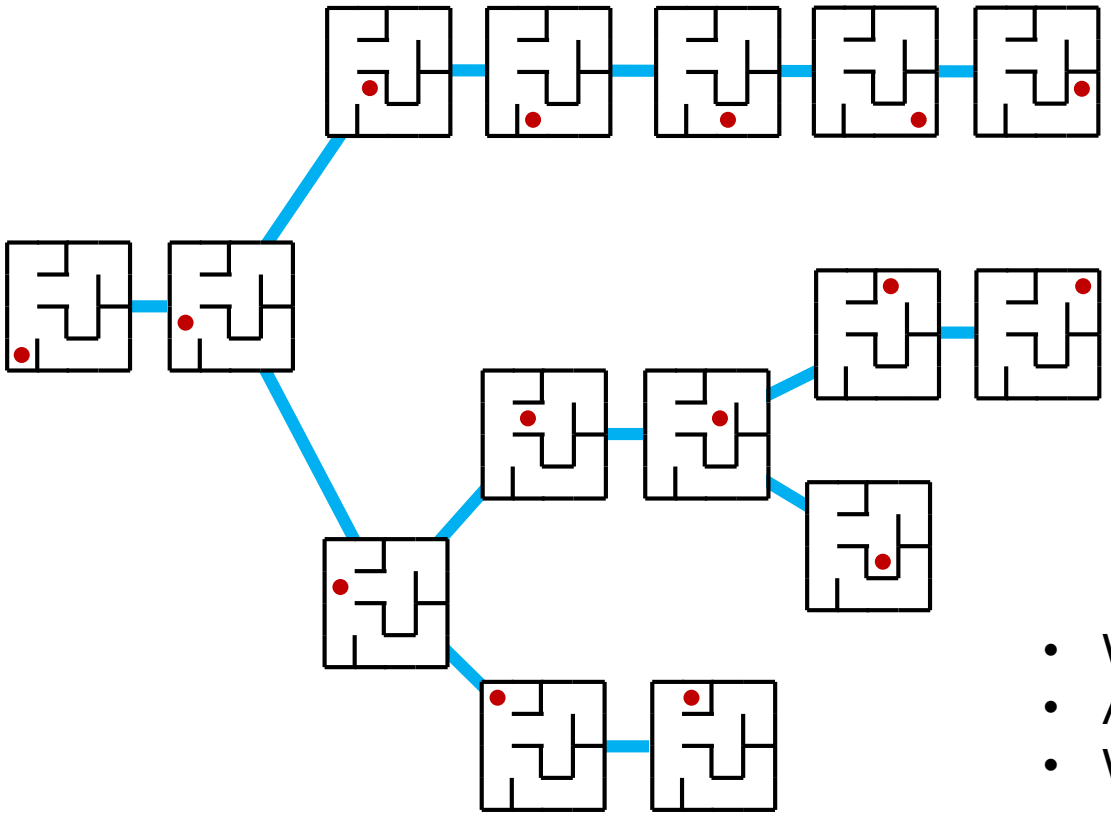
- Every state is reachable from every other state by some sequence of moves.
  - We can think of the states arranged in a graph. We explore the state space by building a spanning tree over the graph.
  - If states are not reachable we may not find the global optimum. Stochastic search can also help.

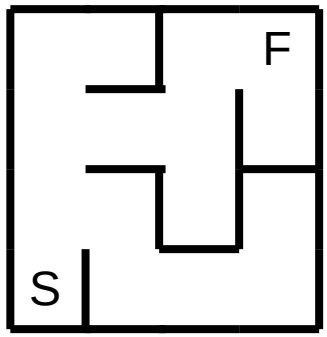
- If the search space is too large to explore exhaustively then we many apply a hill-climbing heuristic (i.e., expand most promising nodes first). This too can get stuck in a local optimum.

Breadth-First and Depth-First Search

```
breadth/depth-first search
set best score to −∞
add first node to queue
while !empty(queue) do
      x = head/tail queue and remove
      if visited(x) then
            continue
      if score(x) > best score then
            set best score to score(x)
      for each unvisited neighbour y of x do
            push y onto queue
```

BFS/DFS Example





- Which is faster BFS or DFS?
- Are any states unvisited?
- What if the maze had a loop?

Beam Search and Tabu Search

- **Beam search** follows breadth-first search but only expands a subset of children (limited by the beam width)
  - How do we choose the nodes to expand?

- Instead of BFS or DFS we can use a priority queue to expand the most promising node first (**best-first search**)
- In **tabu search** we disallow the undoing moves that were made recently.
- This help avoid getting stuck in local optima.

Maximum Likelihood Score

The likelihood function---probability of the data given the model---is a natural choice for the scoring function.

Given graph structure and maximum likelihood parameters $\hat{\theta}_G$ for that structure, the likelihood score is

$$score_L(G; D) = l(\hat{\theta}_G; D)$$

where $l(\cdot)$ is the log-likelihood function and D is the data.

Recall $\hat{\theta}_G = \underset{\theta}{\operatorname{argmax}}\, l(\theta; D)$. Our aim then is to find G that maximizes $score_L(G; D)$.

Structure Learning Example

Consider a model with two random variables X and Y. We observe the following data,

X	Y	Count
0	0	26
0	1	25
1	0	22
1	1	27

We can consider two models, $P(X,Y) = P(X)P(Y)$ and $P(X,Y) = P(X)P(Y \mid X)$. (How would you represent these graphically?)

ML Score Example

X	Y	#
0	0	26
0	1	25
1	0	22
1	1	27

(a) Independent model:

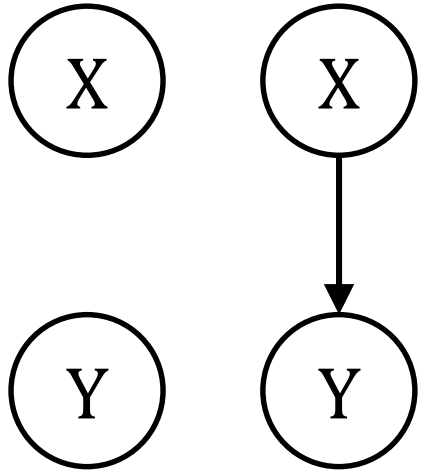
$$\theta_{X=1} = \frac{22+27}{100} \text{ and } \theta_{Y=1} = \frac{25+27}{100}$$

$$score_L(G_a) = 26\log(0.51 \times 0.48) + \cdots$$
$$= -61.1$$

(b) Full model:

$$\theta_{X=1} = \frac{22+27}{100}, \theta_{Y=1|X=1} = \frac{27}{22+27} \text{ and } \theta_{Y=1|X=0} = \frac{25}{26+25}$$

$$score_L(G_b) = 26\log(0.51 \times 0.5098) + \cdots$$
$$= -60.08$$



(a) (b)

Bayesian Score

The likelihood score tends to overfit since adding edges introduces more parameters, which can better fit the data. We end up with a fully connected network.

The Bayesian score allows us to place a prior over graph structures,

$$score_B(G; D) = \log P(D \mid G) + \log P(G)$$

where $P(D \mid G) = \int_\theta P(\, D \mid G, \theta\,) P(\, \theta \mid G\,)\, d\theta$.

Bayesian Information Criterion Score

Assuming a Dirichlet prior on the parameters the Bayesian score can be approximated by

$$score_{BIC}(G;D) = l\left(\hat{\theta}_G;D\right) - \frac{\log m}{2}|\theta_G|$$

where m is the number of training examples and $|\theta|$ is the number of (independent) parameters in the model. The second term **penalizes** more complex models (and relates to minimum description length coding).

The BIC score is **consistent**. As $m \to \infty$ the true Bayesian network structure maximizes the score (and all graphs that are not Markov equivalent have strictly lower score).

BIC Score Example

X	Y	#
0	0	26
0	1	25
1	0	22
1	1	27

(a) Independent model:

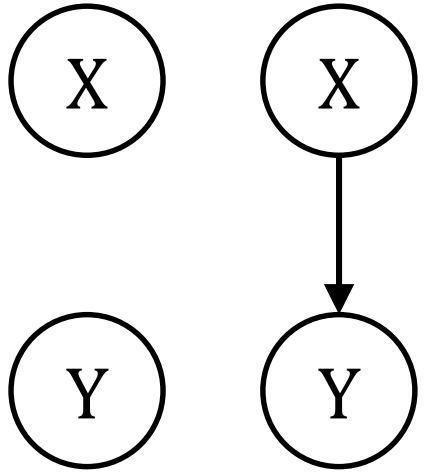
$$\theta_{X=1} = \frac{22+27}{100} \text{ and } \theta_{Y=1} = \frac{25+27}{100}$$

$$score_{BIC}(G_a) = 26\log(0.51 \times 0.48) + \cdots$$
$$= -61.1 - \frac{\log(100)}{2}2$$
$$= -63.1$$

(b) Full model:

$$\theta_{X=1} = \frac{22+27}{100}, \theta_{Y=1|X=1} = \frac{27}{22+27} \text{ and } \theta_{Y=1|X=0} = \frac{25}{26+25}$$

$$score_{BIC}(G_b) = 26\log(0.51 \times 0.5098) + \cdots$$
$$= -60.08 - \frac{\log(100)}{2}3 = -63.08$$



(a) (b)

Decomposable Scores

A structure score is said to be **decomposable** if the score of a structure can be written as

$$score(G; D) = \sum_i LocalScore(X_i \mid Pa_i^G; D)$$

where $LocalScore(X_i \mid U; D)$ measures how well U serve as the parents of X_i given the data D.

Decomposable scores make structure search much more efficient (since local changes do not change the score of other parts of the model).

Bayesian Network Search Moves

Standard search moves in Bayesian network structure learning include:

- edge addition (with restrictions)
- edge deletion
- edge reversal (with restrictions)

When the score is decomposable the effect of these moves on the score can be evaluated very efficiently.

Delta Score

Define the change in score after applying a move on a graph as

$$\delta(G: o) = score(o(G); D) - score(G; D)$$

Then

- $\delta(G: \text{add } X \to Y) = LocalScore\left(Y, Pa_Y^G \cup \{X\}; D\right) - LocalScore(Y, Pa_Y^G; D)$
- $\delta(G: \text{delete } X \to Y) = LocalScore\left(Y, Pa_Y^G \setminus \{X\}; D\right) - LocalScore(Y, Pa_Y^G; D)$
- $\delta(G: \text{reverse } X \to Y) = \delta(G : \text{delete } X \to Y) + \delta(G: \text{add } X \to Y)$

And

- If $Pa_Y^G = Pa_Y^{G'}$ then $\delta(G: \text{add } X \to Y) = \delta(G': \text{add } X \to Y)$. Similarly for $\text{delete } X \to Y$.
- If $Pa_Y^G = Pa_Y^{G'}$ and $Pa_X^G = Pa_X^{G'}$ then $\delta(G: \text{rev } X \to Y) = \delta(G': \text{rev } \to Y)$

Chow-Liu Algorithm for Trees

The goal of the **Chow-Liu algorithm** is to find a tree structured model that maximizes the likelihood of the data.

The algorithm proceeds as follows:

`for each pair` (X_i, X_j) `of variables`

$$\texttt{compute } w_{ij} = I(X_i, X_j) = \sum_{x_i, x_j} p(x_i, x_j) \log \left(\frac{p(x_i, x_j)}{p(x_i) p(x_j)} \right)$$

`find a maximum spanning tree (MST)`

`assign directions to the edges in the MST`

Use the empirical distribution for $p(x_i, x_j)$.

Finding a Maximum Spanning Tree

Minimum and maximum spanning trees can be found efficiently using a greedy algorithm (e.g., Prim's algorithm and Kruskal's algorithm).

The basic algorithm proceeds as follows:

initialize $V = \{i\}$ for arbitrary i and $E = \emptyset$
until $|V| = n$ do
    find $(i, j) \in V \times \bar{V}$ with maximum weight w_{ij}
    add (i, j) to E and add i to V

Why Mutual Information?

The MST algorithm finds a tree that maximizes

$$\sum_{ij \in E} I(X_i, X_j)$$

Writing out the likelihood of the data given the tree we have

$$\log p(D; T, \theta) = \sum_{k=1}^{m} \sum_{i} \log p\left(x_i^{(k)} \mid pa_i^{(k)} \right)$$

$$= m \sum_{i} I(X_i, Pa_i) - H(X_i)$$

We care about maximizing the likelihood

$$\text{argmax}_{T,\theta} \log p(D; T, \theta) = \text{argmax}_{T} \sum_{i} I(X_i, Pa_i)$$

Chow-Liu With Other Scores

Any decomposable score can be used within the Chow-Liu algorithm to give the best tree structured graph under that score.

- Replace w_{ij} with $LocalScore(X_i \mid X_j; D)$
- Stop maximum spanning tree search when best remaining edges have negative weight

Thin Junction Trees

Learning a tree leads to a model where inference is very efficient but the model may not be sufficiently powerful.

When learning a graph via the discrete search method discussed earlier we have no control over the complexity of inference in the resulting graph. Recall, complexity of exact inference is a function of treewidth.

Thin junction trees solve this problem (approximately) by greedily adding edges ensuring that a given treewidth is never exceeded:

- sort edges by their weight then iterate through edges in order skipping the edge if it results in a treewidth greater than *k* and adding it to the graph otherwise

Learning with Known Ordering

Assume we restrict our search to the hypothesis space of graphs with some known ordering $X_1 \prec X_2 \prec \cdots \prec X_n$. That is, if $X_i \in Pa_j$ then $i < j$.

Then, assuming we have a decomposable score, we can learn the parent set for each variable independently,

$$Pa_i = \operatorname{argmax}_U LocalScore(X_i \mid U; D)$$

where U is restricted to $\{X_1, \ldots, X_{i-1}\}$.

Note that for X_n we need to consider 2^{n-1} possible parent sets. But this is better than the $2^{O(n^2)}$ structures for a general Bayesian network.

Gaussian Markov Random Fields

Recall the density of a zero-mean Gaussian is

$$p(x_1, \ldots, x_n) \propto \exp\left\{-\frac{1}{2}\boldsymbol{x}^T \Sigma^{-1} \boldsymbol{x}\right\}$$

Define the precision matrix $Q = \Sigma^{-1}$. Then

- $X_i \perp X_j \mid \{X_k | k \neq i, j\}$ if and only if $Q_{ij} = 0$

Proof (sketch): Consider

$$p(x_1, x_2) \propto \exp\left\{-\frac{1}{2}Q_{11}x_1^2 - Q_{12}x_1x_2 - \frac{1}{2}Q_{22}x_2^2\right\}$$

Gaussian Log-Likelihood

The log-likelihood of *m* i.i.d. samples drawn from $N(\mu;\Sigma)$ is

$$\sum_{i=1}^{m}\log p\left(x^{(i)}\right) = \sum_{i=1}^{m} -\frac{1}{2}\log|\Sigma| - \frac{1}{2}\left(x^{(i)}-\mu\right)^{T}\Sigma^{-1}\left(x^{(i)}-\mu\right) + k$$

$$= \frac{m}{2}\log|Q| - \frac{1}{2}\sum_{i=1}^{m}\mathrm{tr}\left(\left(x^{(i)}-\mu\right)^{T}Q\left(x^{(i)}-\mu\right)\right) + k$$

$$= \frac{m}{2}\log|Q| - \frac{1}{2}\sum_{i=1}^{m}\mathrm{tr}\left(\left(x^{(i)}-\mu\right)\left(x^{(i)}-\mu\right)^{T}Q\right) + k$$

$$= \frac{m}{2}\log|Q| - \frac{1}{2}tr\left(\sum_{i=1}^{m}\left(x^{(i)}-\mu\right)\left(x^{(i)}-\mu\right)^{T}Q\right) + k$$

$$= \frac{m}{2}\log|Q| - \frac{1}{2}tr\left(\hat{\Sigma}Q\right) + k$$

Learning Gaussian Random Fields

Maximum likelihood learning can then be expressed as

$$\text{maximize } \log|Q| - \text{tr}\left(\hat{\Sigma}Q\right)$$

Taking derivatives we get $Q^{-1} - \hat{\Sigma} = 0$. Therefore $\Sigma = \hat{\Sigma}$.

In order to induce a sparse structure we can introduce a penalty on the values of Q,

$$\text{maximize } \log|Q| - \text{tr}\left(\hat{\Sigma}Q\right) - \lambda \sum_{ij} |Q_{ij}|$$

This technique is also known as **graphical lasso** and can be solved via various convex optimization techniques.