CS7140: Advanced Machine Learning **Spring 2020**

Lecture 7: Feb 20

Lecturer: Rose Yu *Scribes: Anupam Kumar, Chi Zhang*

7.1 Structure Learning

In the previous lecture, we studied parameter learning of Bayesian networks. The assumption here was that we know the network structure in advance. In this lecture, we will study how to learn structure for Bayesian networks. In addition to the assumption that the data $\mathcal{D}$ are IID generated, we also assumes that $\mathcal{P}^*$ is induced by some Bayesian network $\mathcal{G}^*$ over X.

In particular, our goal is:

- Reconstruct the structure of Bayesian networks $\mathcal{G}^*$.
- Find independencies in variables.

However, there can be equally good structures $\mathcal{G}^*$ for a distribution $\mathcal{P}^*$ based on same data: class of all I-equivalent networks. Hence, $\mathcal{G}^*$ is not identifiable from the data. Here, recall that two Bayesian networks are I-equivalent if they encode precisely the same conditional independence assertions. In particular, Two Bayesian networks are I-equivalent if they have the same skeletons and same V-strutures.

7.1.1 Overview of Methods

There are three approaches to learning without a prespecified structure.

1. **Contstraint-based:** reconstruct a network structure that best captures the independencies.
   These methods try to test for conditional dependence and independence in the data and then to find a network (or more precisely an equivalence class of networks) that best explains these dependencies and independencies. Although quite intuitive, these methods can be sensitive to failures in individual independence test.

2. **Score-based:** find a network structure that best fits the observed data.
   Score-based methods address learning as a *model selection* problem. These all operate on the same principle: define a *hypothesis space* of potential models — the set of possible network structures we are willing to consider — and a scoring function that measures how well the model fits the observed data. The task is to find the highest-scoring network structure. These methods consider the whole structure at once; and therefore are less sensitive to individual failures. The disadvantage is that the problem is $\mathcal{NP}$-hard and we resort to heuristic search techniques.

3. **Bayesian model averaging:** average the prediction of all possible structures.
   Instead of attempting to learn a single structure, this approach generates an ensemble of possible structures and take the average prediction.

7.2 Constraint-based Structure learning

Goal: To learn an I-equivalent class for network structure and find the best minimal I-map for the domain. The algorithms for constraint-based learning are essentially variants of algorithms for building I-maps and P-maps.

Recall that a graph $\mathcal{G}$ is an I-map for a set of independencies $\mathcal{I}$ if $\mathcal{I}(\mathcal{G}) \subset \mathcal{I}$, where $\mathcal{I}(\mathcal{G}) = \{(\mathbf{X} \perp \mathbf{Y}|\mathbf{Z})\}$, set of independencies due to graph $\mathcal{G}$. We want to find a graph that represents a distribution P. The problem with naive approach of taking any graph that is an I-map for P is that there would be many redundant edges. Hence, the need for a minimal I-map. Recall that a graph $\mathcal{G}$ is a minimal $\mathcal{I}$-map for $\mathcal{I}$ if it's an I-map for $\mathcal{I}$ and removal of even a single edge from $\mathcal{G}$ renders it not an I-map.

Let's first discuss algorithms for building minimal I-map. For this, we predetermine a *variable ordering* $\{X_1, \ldots, X_n\}$. Then for each X_i, we pick some minimal subset $\mathbf{U}$ of $\{X_1, \ldots, X_{i-1}\}$ to be X_i's parents in $\mathcal{G}$ satisfying $(X_i \perp \{X_1, \ldots, X_{i-1}\} - \mathbf{U}|\mathbf{U})$ and that no node can be removed from $\mathbf{U}$ without violating it. By construction, $\mathcal{G}$ is a minimal I-map for P. But minimal I-maps can fail to capture some or all of independencies in distribution P, hence we need P-maps.

Definition 7.1 *We say that a graph $\mathcal{G}$ is a perfect map (P-map) for a set of independencies $\mathcal{I}$ if we have that $\mathcal{I}(\mathcal{G}) = \mathcal{I}$. We say that $\mathcal{G}$ is a perfect map for P if $\mathcal{I}(\mathcal{G}) = \mathcal{I}(\mathcal{P})$.*

We want to identify a DAG $\mathcal{G}^*$ that is a P-map for P. For ease of understanding, we will divide our algorithm into three parts: Build-PMap-Skeleton, Identifying immortalities (v-structures), Build-PDAG which outputs $\mathcal{G}^*$. We call a set $\mathbf{U}$, a witness of independence of X and Y if $P \models (X \perp Y|\mathbf{U})$.

Jointly the algorithm would be:

Algorithm 1: Build graph $\mathcal{G}^*$, a P-map for P

```
Input: $\{X_1, \ldots, X_n\}$
Output: $\mathcal{G}^*$
1    Build a complete graph for $\{X_1, \ldots, X_n\}$
2    For every pair $X_i, X_j$, find witness set $\mathbf{U}$ such that $X_i \perp X_j|\mathbf{U}$.
3    Remove edge $X_i - X_j$ if $\mathbf{U}$ is not empty.
4    Identify triplets that are immortalities.
5    Orient edges by propagating a set of constraints.
```

Build-PDAG procedure algorithm reconstructs the network that best matches the domain without a pre-specified order and uses only a polynomial number of *independence tests* that involve a bounded number of variables.

To achieve these performance guarantees, we must make some assumptions:

- The network $\mathcal{G}^*$ has bounded indegree, that is, for all i, $|\mathrm{Pa}_{X_i}^{\mathcal{G}^*}| \leq d$ for some constant d.
- The independence procedure can perfectly answer any independence query that involves up to $2d + 2$ variables.
- The underlying distribution P^* is faithful to $\mathcal{G}^*$.

In the next three algorithms, we will just expand on the three parts: Build-PMap-Skeleton, Identifying immortalities (v-structures), and Build-PDAG.

The first part is: algorithm for recovering skeleton for $\mathcal{G}^*$:

Algorithm 2: Recovering the undirected skeleton for P that has P-map

```
Procedure Build-PMap-Skeleton
Input: $\mathcal{X} = \{X_1, \ldots, X_n\}, P$
1  Let $\mathcal{H}$ be the complete undirected graph $\mathcal{X}$
2  for $X_i, X_j \in \mathcal{X}$ do
3      $\mathbf{U}_{X_i,X_j} \leftarrow \emptyset$
4      for $\mathbf{U} \in$ Witnesses$(X_i, X_j, \mathcal{H})$ do
           // Consider $\mathbf{U}$ as a witness set for $X_i, X_j$
5          if $P \models (X_i \perp X_j | \mathbf{U})$ then
6              $U_{X_i,X_j} \leftarrow \mathbf{U}$
7              Remove $X_i - X_j$ from $\mathcal{H}$
8              break
9          end
10     end
11     return $(\mathcal{H}, \{\mathbf{U}_{X_i,X_j} : i, j \in \{1, \ldots, n\}\})$
12 end
```

The second part: algorithm for identifying immortalities (v-strutures):

Algorithm 3: Identify immortalities (v-structures) in the construction of P-map

```
Procedure Mark-immortalities
Input: $\mathcal{X} = \{X_1, \ldots, X_n\}, S$ (Skeleton), $\{\mathbf{U}_{X_i,X_j:1 \le i,j \le n}\}$ (Witnesses found by Build-PMap-Skeleton)
1  $\mathcal{G} \leftarrow S$
2  for $X_i, X_j, X_k$ such that $X_i - X_j - X_k \in S$ and $X_i - X_k \notin S$ do
       // $X_i - X_j - X_k$ if $X_j \notin \mathbf{U}_{X_i,X_k}$ then
3          Add the orientations $X_i \rightarrow X_j$ and $X_j \leftarrow X_k$ to $\mathcal{G}$
4      end
5  end
6  return $\mathcal{G}$
```

Now what we are left with is to orient rest of the edges in the returned graph $\mathcal{G}$ from above algorithm. Our goal here is to orient edges while preserving I-equivalence (hence v-strutures) and DAG (acyclicity). We will be following three rules for orienting edges: R1, R2, R3 (R3 can be thought of being implied from R1 and R2).

Rule R1 (preserve the v-structures) corresponds to creating edges such that the procedure does not create addition v-structures than what are already present. An example would be that given $X \rightarrow Y — Z$, we must orient the edge $Y \leftarrow Z$, for otherwise we would create an immortality $X \rightarrow Y \leftarrow Z$.

Rule R2 (acyclic graph) is derived from the standard acyclicity constraint: If we have the directed path $X \rightarrow Y \rightarrow Z$, and an undirected edge $X — Z$, we cannot direct the edge $X \leftarrow Z$ without creating a cycle.

For the third rule R3 (Allow undirected edges to have both directions), consider the example in the next figure. Assume, by contradiction, that we direct the edge $Z \rightarrow X$. In this case, we cannot direct the edge $X — Y_1$ as $X \rightarrow Y_1$ without creating a cycle; thus, we must have $Y_1 \rightarrow X$. Similarly, we must have $Y_2 \rightarrow X$. But, in this case, $Y_1 \rightarrow X \leftarrow Y_2$ forms an immorality (as there is no edge between Y_1 and Y_2), which contradicts the fact that the edges $X — Y_1$ and $X — Y_2$ are undirected in the original PDAG.

Now the third part: algorithm for finding PDAG class characterizing P-map:

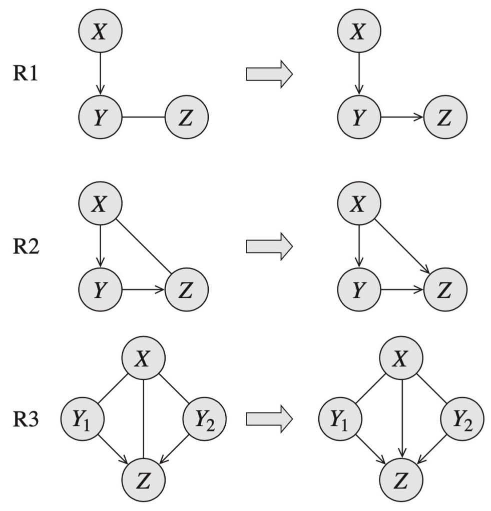

Figure 3.12 Rules for orienting edges in PDAG. Each rule lists a configuration of edges before and after an application of the rule.

Algorithm 4: Finding the class PDAG characterizing the P-map of a distribution P

```
Procedure: Build-PDAG
Input: $\mathcal{X}, P$
1 $S, \{U_{X_i,X_j}\} \leftarrow$ Build-PMap-Skeleton $(\mathcal{X}, P)$
2 $\mathcal{G} \leftarrow$ Find-Immortalities $(\mathcal{X}, S, \{U_{X_i,X_j}\})$
3 while not converged do
4     Find a subgraph in $\mathcal{G}$ matching the left-hand side of a rule R1-R3
5     Replace the subgraph with the right-hand side of the rule.
6 end
```

7.3 Score-based Structure learning

The score-based methods approach the problem of structure learning as an optimization problem. We define a score function that can score each candidate structure with respect to the training data, and then search for a high-scoring structure. One of the most important decisions we must make in this framework is the choice of scoring function. We will discuss two choices for scoring function: Maximum likelihood scores and Bayesian score. The high level algorithm would be:

Algorithm 5: Optimize the network structure score that best fits the data

```
Input: $\{\mathcal{X} = \{X_1, \ldots, X_n\}\}$
Output: $\mathcal{G}^*$, a scoring function
1     Generate a set of possible network structures with bounded indegree.
2     Search the space of graphs.
3     Return the high-scoring one.
```

7.3.1 Maximum likelihood score

Objective function:

$$\max_{G,\theta_G} L(D|G,\theta_G)$$

Decomposition:

$$score_L(G|D) = M\sum_{i=1}^{n} I_{\hat{P}}(X_i; Pa_{X_i}^G) - M\sum_{i=1}^{n} H_{\hat{P}}(X_i)$$

Comment: The likelihood of a network measures the strength of the dependencies between variables and their parents. In other words, we prefer networks where the parents of each variable are informative about it.
Limitation of maximum likelihood score:

- Overfit
- Never prefers simpler network

7.3.2 Bayesian score

Score function:

$$score_B(G|D) = \log P(D|G) + \log P(G)$$

Intuition:

$$\frac{1}{M}\log P(G|D) \approx E_P[\log P(X|G,D)]$$

Comment: The Bayesian score seems to be biased toward simpler structures, but as it gets more data, it is willing to recognize that a more complex structure is necessary. In other words, it appears to trade o fit to data with model complexity, thereby reducing overfitting the extent of overfitting.
BIC score (Bayesian information criterion):

$$score_{BIC}(G|D) = l(\hat{\theta}_G|D) - \frac{\log M}{2}Dim[G]$$

7.3.3 Score Decomposition

- Specify prior over structure: $P(G) \propto \prod_i P(Pa_{X_i} = Pa_{X_i}^G)$
- Decomposable score: $score(G|D) = \sum_i score(X_i|Pa_{X_i} : D)$
- A local change in the structure (such as adding an edge) does not change the score of other parts of the structure that remained the same.
- Reduce dramatically the computational overhead of evaluating different structures during search.

7.3.4 Score Equivalence

Definition 7.2 *Let $score(G : D)$ be some scoring rule. We say that it satisfies score equivalence if for all score equivalence I-equivalent networks G and $G\ 0$ we have $score(G : D) = score(G0 : D)$ for all data sets D.*

Theorem 7.3 *The likelihood score and the BIC score satisfy score equivalence.*

Theorem 7.4 *Let P(G) be a structure prior that assigns I-equivalent networks identical prior. Let $P(\theta_G|G)$ be a prior over parameters for networks with table-CPDs that satisfies global and local parameter independence and where for each X_i and $u_i Val(Pa^G_{X_i})$, we have that $P(_{X_i|u_i}|G)$ is a Dirichlet prior. The Bayesian score with this prior satisfies score equivalence if and only if the prior is a BDe prior for some choice of α and P'.*

7.4 Structure search

State: a network structure of variables
Action: edge addition, edge deletion, edge reversal

7.4.1 Tree Structure

Definition 7.5 *A network structure G is called tree-structured if each variable X has at most one parent in G, tree network that is, $|PaGX| \leq 1$.*

7.4.2 General graph

Theorem 7.6 *The following problem is N P-hard for any d 2:*
Given a data set D and a decomposable score function score, find

$$G^* = \arg\max_{G \in G_d} score(G : D)$$

7.5 Bayesian Model Averaging

Compute the average prediction over candidate network structures:

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

Algorithm input X1,, Xn, output G , a prediction function

- Find a set g of high scoring structures
- Estimate the probability of the structure in g
- Compute average prediction: $P(f|D) = \frac{\sum_{G \in g} P(G|D)f(G)}{\sum_{G \in g} P(G|D)}$

7.5.1 Find Candidate Structures

- Large data: a single high-scoring structure gives a good approximation of the prediction
- Small data: a large number of high-scoring models
- Alternative: MCMC over structures

7.5.2 MCMC over structure

Approximate:

$$p(f|D) \approx \frac{1}{K} \sum_k f(G_k)$$

Then sample using MCMC.

- Use Metropolis-Hasting as an example.
- The current state is G, sample G' from the proposal distribution
- Accept the transition with probability: $\min\left(1, \frac{P(G',D)T^Q(G'->G)}{P(G,D)T^Q(G->G')}\right)$

References

[KF09] D. Koller and N. Friedman, "Probabilistic Graphical Models: Principles and Techniques," *MIT Press*, 2009, pp. 85–89, 783–787.