Causal Inference
CS 477-677

Structure Learning With Scoring Methods

Ilya Shpitser

Outline

Structure Learning Algorithms

- Assume there is a graph $\mathcal{G}(\vec{V})$ with k vertices, and a distribution $p(\vec{V})$ factorizing relative to $\mathcal{G}$.
- INPUT: a dataset $\vec{X}_{n\times k}$ (assumed sampled independently from $p(\vec{V})$).
- OUTPUT: a set of graphs consistent with what we know about $\vec{X}_{n\times k}$ (hopefully including $\mathcal{G}$).

Structure Learning Algorithms

- Assume there is a graph $\mathcal{G}(\vec{V})$ with k vertices, and a distribution $p(\vec{V})$ factorizing relative to $\mathcal{G}$.
- INPUT: a dataset $\vec{X}_{n\times k}$ (assumed sampled independently from $p(\vec{V})$).
- OUTPUT: a set of graphs consistent with what we know about $\vec{X}_{n\times k}$ (hopefully including $\mathcal{G}$).
- This is an unsupervised learning problem.
- We want to find a sensible causal description of the data.
- Lots of ways of doing this!

Types Of Structure Learning

- Constraint based learning (today):
  - Find constraints that hold in $\vec{X}_{n \times k}$.
  - Rule out graphs inconsistent with constraints we found.
  - Return what's left.
- Score based learning (next time):
  - Assign a score to any graphical model.
  - Scores typically reward fit, but also regularize (want a concise description of the data).
  - Do search (model selection) for high scoring models given $\vec{X}_{n \times k}$.
- These are asymptotically equivalent, but behave differently in finite samples.
- "Parametric" structure learning:
  - Make strong additional assumptions on $p(\vec{V})$.
  - Orient edges using those assumptions.
  - Examples: additive noise models, structure learning as classification, etc.

Intuition Behind Score Based Learning

- Assume the true graph is on the left, proposed graph is on the right.

- In the true graph, $A \perp\!\!\!\perp C$ but (under faithfulness) $A \not\perp\!\!\!\perp C \mid B$.
- In the proposed graph $A \perp\!\!\!\perp C \mid B$, but (under faithfulness) $A \not\perp\!\!\!\perp C$.
- In the data, will we likely see $A \perp\!\!\!\perp C$, and (under faithfulness) $A \not\perp\!\!\!\perp C \mid B$.
- Conclusion: proposed graph does not fit the data well.
- How do we assess data fit?

Intuition Behind Score Based Learning

- Assume the true graph is on the left, proposed graph is on the right.

- In the true graph, $A \perp\!\!\!\perp C$ but (under faithfulness) $A \not\perp\!\!\!\perp C \mid B$.
- In the proposed graph $A \perp\!\!\!\perp C \mid B$, but (under faithfulness) $A \not\perp\!\!\!\perp C$.
- In the data, will we likely see $A \perp\!\!\!\perp C$, and (under faithfulness) $A \not\perp\!\!\!\perp C \mid B$.
- Conclusion: proposed graph does not fit the data well.
- How do we assess data fit? The (log) likelihood function.

Log Likelihood For DAG Models

- Recall the DAG factorization:

$$p(\vec{V}) = \prod_{V_i \in \vec{V}} p(V_i \mid \text{pa}_{\mathcal{G}}(V_i)).$$

- Assume (discriminative) models $p(V_i \mid \text{pa}_{\mathcal{G}}(V_i); \alpha_i)$ (regression, etc.) for each $V_i \in \vec{V}$.
- For a dataset $\vec{D}_{n \times k}$, and $\vec{\alpha} = \{\alpha_i \mid V_i \in \vec{V}\}$, the log likelihood is

$$\log \mathcal{L}_{\vec{V}}(\vec{\alpha}; \vec{D}) = \sum_{j=1}^{n} \sum_{i=1}^{k} \log p(V_i^j \mid \text{pa}_{\mathcal{G}}^j(V_i); \alpha_i).$$

- Can fit each α_i separately on part of data that involves $p(V_i \mid \text{pa}_{\mathcal{G}}(V_i))$.
- This is called a decomposable likelihood. Already saw this with sequential g-formula.

Likelihood For Model Fit

- In our running example, the true and proposed graph log likelihoods are

$$\mathcal{L}^{\text{left}}_{\vec{V}}(\alpha; \vec{D}) = \sum_{j=1}^{n} \log p(A^j; \beta_A) + \log p(B^j \mid A^j, C^j; \beta_B) + \log p(C^j; \beta_C)$$

$$\mathcal{L}^{\text{right}}_{\vec{V}}(\alpha; \vec{D}) = \sum_{j=1}^{n} \log p(A^j \mid B^j; \alpha_A) + \log p(B^j \mid C^j; \alpha_B) + \log p(C^j; \alpha_C)$$

- To compare these two models, pick $\vec{\alpha}, \vec{\beta}$ to "minimize surprise" (maximize log likelihood), compare results.
- True graph will have a higher likelihood value.

Likelihood For Model Fit

- In our running example, the true and proposed graph log likelihoods are

$$\mathcal{L}^{\text{left}}_{\vec{V}}(\alpha; \vec{D}) = \sum_{j=1}^{n} \log p(A^j; \beta_A) + \log p(B^j \mid A^j, C^j; \beta_B) + \log p(C^j; \beta_C)$$

$$\mathcal{L}^{\text{right}}_{\vec{V}}(\alpha; \vec{D}) = \sum_{j=1}^{n} \log p(A^j \mid B^j; \alpha_A) + \log p(B^j \mid C^j; \alpha_B) + \log p(C^j; \alpha_C)$$

- To compare these two models, pick $\vec{\alpha}, \vec{\beta}$ to "minimize surprise" (maximize log likelihood), compare results.
- True graph will have a higher likelihood value.
- Any problems with doing this?

Problems With Likelihood As A Score

- True graph on the left, a different proposed graph on the right.

- Respective likelihoods are:

$$\mathcal{L}^{\text{left}}_{\vec{V}}(\alpha; \vec{D}) = \sum_{j=1}^{n} \log p(A^j; \beta_A) + \log p(B^j \mid A^j, C^j; \beta_B) + \log p(C^j; \beta_C)$$

$$\mathcal{L}^{\text{right}}_{\vec{V}}(\alpha; \vec{D}) = \sum_{j=1}^{n} \log p(A^j \mid B^j, C^j; \alpha_A) + \log p(B^j \mid C^j; \alpha_B) + \log p(C^j; \alpha_C)$$

- Which likelihood fits the data better?

Problems With Likelihood As A Score

- True graph on the left, a different proposed graph on the right.

- Respective likelihoods are:

$$\mathcal{L}^{\text{left}}_{\vec{V}}(\alpha; \vec{D}) = \sum_{j=1}^{n} \log p(A^j; \beta_A) + \log p(B^j \mid A^j, C^j; \beta_B) + \log p(C^j; \beta_C)$$

$$\mathcal{L}^{\text{right}}_{\vec{V}}(\alpha; \vec{D}) = \sum_{j=1}^{n} \log p(A^j \mid B^j, C^j; \alpha_A) + \log p(B^j \mid C^j; \alpha_B) + \log p(C^j; \alpha_C)$$

- Which likelihood fits the data better?
- $\mathcal{L}^{\text{right}}_{\vec{V}}(\alpha; \vec{D})$. Why? The left model is $\{p(A, B, C) \mid A \perp\!\!\!\perp C\}$. The right model is $\{$ all $p(A, B, C)\}$.
- Left model is a submodel (subset) of the right. Subsets are strictly more restrictive, so will always fit data less well.

Likelihood And Overfitting

- If we use the likelihood to score models, the highest scoring model is always a complete graph.
- This is silly!
- Related to overfitting in machine learning.
- If the true graph is sparse, a complete graph has too many parameters.
- Fits very well, but does not capture the true structure of the problem.

Likelihood And Overfitting

- If we use the likelihood to score models, the highest scoring model is always a complete graph.
- This is silly!
- Related to overfitting in machine learning.
- If the true graph is sparse, a complete graph has too many parameters.
- Fits very well, but does not capture the true structure of the problem.
- How to address this?
- Want to reward fit, but also punish too many parameters.
- Lots of ways to do this.

The Bayesian Information Criterion (BIC) Score

- Will discuss one particular score – the Bayesian Information Criterion (BIC).
- A number assigned to a DAG + parameterization $\vec{\alpha} = \{\alpha_i \mid V_i \in \vec{V}\}$ for all Markov factors:

$$\text{BIC} = -2 \cdot \log \mathcal{L}_{\vec{V}}(\vec{\alpha}, \vec{D}_{n \times k}) + m \cdot log(n),$$

  where m is the number of parameters.
- Minus sign means "low is good." Prefer models with low scores.
- Can be viewed as a kind of regularization (over graph structures).

Consistency of BIC

- Why use BIC to select models?
- In the limit of $n \to \infty$, BIC does "sensible things." What does this mean?
- Say $\mathcal{G}^*$ is the true DAG.
- Property 1
  - Say $\mathcal{G}_1$ is an edge supergraph of $\mathcal{G}^*$ and $\mathcal{G}_2$ is not.
  - Then $\mathcal{G}^*$ (true model) is a submodel of $\mathcal{G}_1$ model, but not a submodel of $\mathcal{G}_2$ model. (Why?)
  - Then BIC gives a better score to $\mathcal{G}_1$ than $\mathcal{G}_2$.
- Property 2
  - Say $\mathcal{G}_1, \mathcal{G}_2$ are both edges supergraphs of $\mathcal{G}^*$. But $\mathcal{G}_1$ has fewer parameter than $\mathcal{G}_2$.
  - Then BIC gives a better score to $\mathcal{G}_1$ than $\mathcal{G}_2$.
- So if we choose correct parameter families for $p(V_i \mid \text{pa}_{\mathcal{G}}(V_i); \alpha_i)$, BIC will find the true graph (up to equivalence!) eventually.

Score Based Search In Asymptopia

- With infinite samples, and infinite computing resources, and the right Markov factor models, life is good!
- List all DAGs on k vertices.
- Compute BIC score for each DAG.
- Pick the best scoring DAGs – these will be the equivalence class of the truth, by consistency of BIC.
- Issues with this?

Score Based Search In Asymptopia

- With infinite samples, and infinite computing resources, and the right Markov factor models, life is good!
- List all DAGs on k vertices.
- Compute BIC score for each DAG.
- Pick the best scoring DAGs – these will be the equivalence class of the truth, by consistency of BIC.
- Issues with this?
- Exponentially many DAGs. Possibly intractable likelihoods for larger DAGs.

Score Based Search In Asymptopia

- With infinite samples, and infinite computing resources, and the right Markov factor models, life is good!
- List all DAGs on k vertices.
- Compute BIC score for each DAG.
- Pick the best scoring DAGs – these will be the equivalence class of the truth, by consistency of BIC.
- Issues with this?
- Exponentially many DAGs. Possibly intractable likelihoods for larger DAGs.
- Problem is NP-hard in general, so must make assumptions.
- As with PC, assume the true DAG $\mathcal{G}^*$ is sparse.
- Search space for all sparse DAGs is still large... so try local search.

How Does Local Search Work?

- Local search arose in making game-playing programs (Samuel's checkers program is an early example).
- You have a big undirected graph (search space)
- Vertices are states (game positions)
- Edges are neighboring states (game position reachable from current one by one move).
- Each state has a value. Want explore space, and find a good one.

How Does Local Search Work?

- Local search arose in making game-playing programs (Samuel's checkers program is an early example).
- You have a big undirected graph (search space)
- Vertices are states (game positions)
- Edges are neighboring states (game position reachable from current one by one move).
- Each state has a value. Want explore space, and find a good one.
- Computers now play chess better than any human due to
  - Discrete search (plus tricks...)
  - Big opening database.
  - Big endgame database.
  - Clever ways of assigning value to positions.
  - Powerful computers to search quickly.

Local Search Of DAG Structures

- For us, search space states are all DAGs.
- Want to define "local moves" to be able to reach any DAG from any other DAG.
- Want a clever algorithm to find the best state (DAG) using a (polynomial) sequence of moves.
- Can we think of a problem with this set up?

Local Search Of DAG Structures

- For us, search space states are all DAGs.
- Want to define "local moves" to be able to reach any DAG from any other DAG.
- Want a clever algorithm to find the best state (DAG) using a (polynomial) sequence of moves.
- Can we think of a problem with this set up?
- DAGs can be observationally equivalent.
- Many equivalent DAGs differ by only one edge.
- Wasteful to traverse state spaces we know are the same.
- Can we define states and "local moves" on equivalence classes?

Local Search Of DAG Structures

- For us, search space states are all DAGs.
- Want to define "local moves" to be able to reach any DAG from any other DAG.
- Want a clever algorithm to find the best state (DAG) using a (polynomial) sequence of moves.
- Can we think of a problem with this set up?
- DAGs can be observationally equivalent.
- Many equivalent DAGs differ by only one edge.
- Wasteful to traverse state spaces we know are the same.
- Can we define states and "local moves" on equivalence classes?
- Yes! Chickering's GES (Greedy Equivalence Search) algorithm.

The Greedy Equivalence Search (GES) Algorithm

- Start with a graph with no edges.
- Repeat until no score improvement
  - Pick the best new class after adding a single edge to current class.
  - Note: adding an edge to a class moves to different classes depending on starting DAG.
  - Example: if we add $C \rightarrow B$ to $\{A \rightarrow B \quad C; A \leftarrow B \quad C\}$, we either move to $\{A \rightarrow B \leftarrow C\}$ or $\{A \leftarrow B \leftarrow C; A \leftarrow B \rightarrow C; A \rightarrow B \rightarrow C\}$.
- Repeat until no score improvement
  - Pick the best new class after removing a single edge to current class.
  - Note: removing an edge to a class moves to different classes depending on starting DAG.
- Return current class.

GES Completeness

Surprisingly, this works (in the limit):

Theorem (Chickering, 2002)

Under faithfulness, and in the limit of $n \to \infty$, GES correctly identifies the true equivalence class.

Remaining Issues

- BIC and GES results are large sample results.
- With finite data, all bets are off.
- Local search still intractable if the score cannot be evaluated in polynomial time.
- If we get models $p(V_i \mid \text{pa}_{\mathcal{G}}(V_i); \alpha_i)$ wrong, all bets are off.
- Cannot use this in $p >> n$ problems (5000 variables, 100 samples).
- In practice, structure learning is often useful in genomics and social networks, which are often $p >> n$ settings.
- Need sparsity ideas to make progress. Lots of literature, leaving aside for lack of time.
- Assumed no hidden variables (which is silly).
- We will talk about the hidden variable case in the Spring class!

Next time: Missing Data.