

CS 583: PROBABILISTIC GRAPHICAL MODELS

TOPIC: STRUCTURE LEARNING
CHAPTERS: 18, 20



Mustafa Bilgic



<http://www.cs.iit.edu/~mbilgic>



<https://twitter.com/bilgicm>

TASK

- We are given fully observed data
- Our task is to learn the structure (and the parameters)

GOAL: KNOWLEDGE DISCOVERY

- There are at least two problems
 - Unidentifiability
 - Given observational data, the best we can hope to learn is an I-Equivalent structure to G^*
 - Weak connections
 - Due to noise, two variables will almost never be independent
 - If a connection is weak, how do we determine if it is due to noise or it is a real weak connection?

GOAL: DENSITY ESTIMATION

- Adding an unnecessary edge is very costly for parameter estimation
 - Each unnecessary binary parent divides the data into half – data fragmentation
 - The estimated parameters have high variance
- For better generalization, it is preferable to have a sparse structure
- *In fact, if the data is limited, it might be even better to have sparser structure than finding G^**

BAYESIAN NETWORKS

THREE APPROACHES

1. *Constraint-based structure learning*

- View Bayesian network as a representation of independencies
- Test for conditional dependence and independence in the data
- Sensitive to failure in individual independence tests

2. *Score-based structure learning*

- Hypothesis space of possible models
- Find the highest scoring structure
- The hypothesis space is super exponential $2^{(O(n^2))}$

3. *Bayesian model averaging*

- Rather than learning a structure, learn an ensemble of structures
- Again, the number of structures is immense
- It is efficient for some while it has to be approximate for others

1. CONSTRAINT-BASED APPROACHES

- In chapter 3, we have seen two procedures
 1. Find minimal I-MAP
 2. Find all I-equivalent structures
- See chapter 3 slides 46 to 49 for details
- In Chapter 3, we assumed that, given a distribution P , we could answer any independence question perfectly
- In this chapter, we'll assume that we don't have access to P but instead we have access to a dataset D

INDEPENDENCE IN THE DATA?

- Assume we toss two coins independently
- We first toss one coin, record it, and then toss the next coin, and record the outcome
- We repeat this process 100 times
- Assume we get 25 H/H, 23 H/T, 27 T/H, 25 T/T
- Based on the data, can we say that the two coin tosses are independent?

INDEPENDENCE TESTS

- χ^2 statistic

$$d_{\chi^2}(D) = \sum_{x,y} \frac{\left(M[x,y] - M \times \hat{P}(x) \times \hat{P}(y)\right)^2}{M \times \hat{P}(x) \times \hat{P}(y)}$$

- Mutual information

$$d_I(D) = \sum_{x,y} \hat{P}(x,y) \log \frac{\hat{P}(x,y)}{\hat{P}(x)\hat{P}(y)}$$

- χ^2 and MI are zero when X and Y are independent
- Extension of χ^2 statistic to conditional independence queries

$$d_{\chi^2}(D) = \sum_{x,y,z} \frac{\left(M[x,y,z] - M \times \hat{P}(z) \times \hat{P}(x|z) \times \hat{P}(y|z)\right)^2}{M \times \hat{P}(z) \times \hat{P}(x|z) \times \hat{P}(y|z)}$$

THREE APPROACHES

1. *Constraint-based structure learning*

- View Bayesian network as a representation of independencies
- Test for conditional dependence and independence in the data
- Sensitive to failure in individual independence tests

2. *Score-based structure learning*

- Hypothesis space of possible models
- Find the highest scoring structure
- The hypothesis space is super exponential $2^{(O(n^2))}$

3. *Bayesian model averaging*

- Rather than learning a structure, learn an ensemble of structures
- Again, the number of structures is immense
- It is efficient for some while it has to be approximate for others

2. SCORE-BASED APPROACHES

- 2.1 Maximum likelihood score
- 2.2. Bayesian score

2.1 MAXIMUM LIKELIHOOD SCORE

- Find a model, $\langle \mathcal{G}, \theta_{\mathcal{G}} \rangle$ that would make the data D as probable as possible

$$\begin{aligned}\max_{\mathcal{G}, \theta_{\mathcal{G}}} L(\langle \mathcal{G}, \theta_{\mathcal{G}} \rangle : D) &= \max_{\mathcal{G}} \left[\max_{\theta_{\mathcal{G}}} L(\langle \mathcal{G}, \theta_{\mathcal{G}} \rangle : D) \right] \\ &= \max_{\mathcal{G}} L(\langle \mathcal{G}, \hat{\theta}_{\mathcal{G}} \rangle : D)\end{aligned}$$

- In order to find the maximum likelihood pair $\langle \mathcal{G}, \theta_{\mathcal{G}} \rangle$
 - We should find the structure \mathcal{G} that achieves the highest likelihood when we use the MLE parameters for \mathcal{G}

$$score_L(\mathcal{G} : D) = l(\hat{\theta}_{\mathcal{G}} : D)$$

TWO VARIABLE CASE

- Given two variables X and Y
- Consider two structures: G_{\emptyset} and $G_{X \rightarrow Y}$.
- What is $\text{score}_L(G_{X \rightarrow Y} : D) - \text{score}_L(G_{\emptyset} : D)$?
- The general equation for $\text{score}_L(G : D)$ is as follows. **Proof is left as an exercise.**

$$\text{score}_L(G : D) = M \sum_{i=1}^n I_{\hat{P}}(X_i; Pa_{X_i}^G) - M \sum_{i=1}^n H_{\hat{P}}(X_i)$$

LIMITATIONS OF ML SCORE

- Can $\text{score}_L(G_{X \rightarrow Y} : D)$ be smaller than $\text{score}_L(G_{\emptyset} : D)$ ever?
- They are equal only when X and Y are truly independent and otherwise $\text{score}_L(G_{X \rightarrow Y} : D)$ is always bigger
- Exercise: Show that $I_P(X; Y \cup Z) \geq I_P(X; Y)$
- Adding a parent can never decrease the maximum likelihood score
- The highest scoring structure is the complete graph
- One remedy is to bound the indegree of the graph

2.2 BAYESIAN SCORE

- Bayesian approach puts a distribution on anything we are uncertain about
 - In parameter estimation, we put a distribution on θ
- In this case, we have the following distributions
 - $P(\mathcal{G})$
 - $P(\theta_{\mathcal{G}} | \mathcal{G})$
- Given data D , the posterior $P(\mathcal{G} | D)$ is
 - $P(\mathcal{G} | D) = P(D | \mathcal{G}) P(\mathcal{G}) / P(D)$
- $P(D)$ is a normalizing constant
- $\text{score}_B(\mathcal{G} : D) = \log P(D | \mathcal{G}) + \log P(\mathcal{G})$

$P(D | \mathcal{G})$

$$P(D | \mathcal{G}) = \int_{\Theta_{\mathcal{G}}} P(D | \theta_{\mathcal{G}}, \mathcal{G}) P(\theta_{\mathcal{G}} | \mathcal{G}) d\theta_{\mathcal{G}}$$

- There is a closed form solution on page 801 of the book
- If we use a Dirichlet priors for all parameters in the network, as $M \rightarrow \infty$

$$\log P(D | \mathcal{G}) = l(\hat{\theta}_{\mathcal{G}} : D) - \frac{\log M}{2} \text{Dim}[\mathcal{G}] + O(1)$$

- $l(\theta:D)$ is the loglikelihood and $\text{Dim}[\mathcal{G}]$ is the model dimension, i.e., the number of independent parameters in \mathcal{G}

BIC SCORE

$$\begin{aligned} \text{score}_{BIC}(\mathcal{G} : D) &= l(\hat{\theta}_{\mathcal{G}} : D) - \frac{\log M}{2} \text{Dim}[\mathcal{G}] \\ &= M \sum_{i=1}^n \mathbf{I}_{\hat{P}}(X_i; Pa_{X_i}^{\mathcal{G}}) - M \sum_{i=1}^n \mathbf{H}_{\hat{P}}(X_i) - \frac{\log M}{2} \text{Dim}[\mathcal{G}] \end{aligned}$$

- A trade-off between the fit to the data and the model complexity
- The mutual information grows linearly in M whereas the model complexity grows logarithmically
 - The larger the M the more emphasis on fit to the data

SCORE EQUIVALENCE

- A score satisfies score equivalence if for all I-Equivalent structures G and G' , $\text{score}(G : D) = \text{score}(G' : D)$
- The likelihood score satisfies score equivalence
- The Bayesian score satisfies score equivalence if and only if we use BDe priors

DECOMPOSABLE SCORE

- A structure score $\text{score}(\mathcal{G} : D)$ is decomposable if it can be written as

$$\text{score}(\mathcal{G} : D) = \sum_i \text{FamScore}(X_i | \text{Pa}_{X_i}^{\mathcal{G}} : D)$$

where $\text{FamScore}(X | U : D)$ measures how well a set of variables U serves as parents of X in D .

- The likelihood score is decomposable
 - $\text{FamScore}(X | U : D) = M^*[I(X;U) - H(X)]$
- The Bayesian score is decomposable under certain natural assumptions

WHY IS DECOMPOSABILITY IMPORTANT?

- When searching for structures, small changes to the structure change the score of the structure, but
- The score of the structure can be updated efficiently by recomputing the scores for only the affected families
- For example
 - Adding or deleting an edge changes the score for only one variable
 - Reversing an edge changes the score for only two variables

STRUCTURE SEARCH

- So far, we have discussed computing a score for a given structure
- How can we find the highest scoring structure?
- Three scenarios
 1. A tree-structured network
 2. Known variable order
 3. General graphs

TREE-STRUCTURED NETWORKS

- Each variable has at most one parent
- Can be learned efficiently – in polynomial time
- They capture the most important dependencies and they serve as a good baseline
- Define $\Delta(\mathcal{G}) = \text{score}(\mathcal{G} : D) - \text{score}(\mathcal{G}_{\emptyset} : D)$
- We know $\text{score}(\mathcal{G}_{\emptyset} : D) = \sum_i \text{FamScore}(X_i : D)$
- $\Delta(\mathcal{G}) = \sum_j \text{FamScore}(X_j \mid \text{Pa}(X_j) : D) - \text{FamScore}(X_j : D)$
- Algorithm
 - Define a fully connected undirected graph where the edge weight between X_i and X_j is $\text{FamScore}(X_i \mid X_j : D) - \text{FamScore}(X_i : D)$
 - Find the maximum spanning forest
- Complexity: $O(n^2M + n^2 \log n)$

KNOWN VARIABLE ORDER

- A variable order is given
 - Assuming a variable order is problematic, but in some cases, the order is natural, such as a temporal one
- A variable's parents can be only the variables that precede it

$$Pa_{X_i}^G = \arg \max_{\mathbf{U}_i \subseteq \{X_j : X_j \prec X_i\}} \text{FamScore}(X_i \mid \mathbf{U}_i : D)$$

- To find the parent of the n^{th} node, we need to consider 2^{n-1} potential sets
- We can bound the indegree to be d ; then the algorithm is exponential in d but polynomial in n

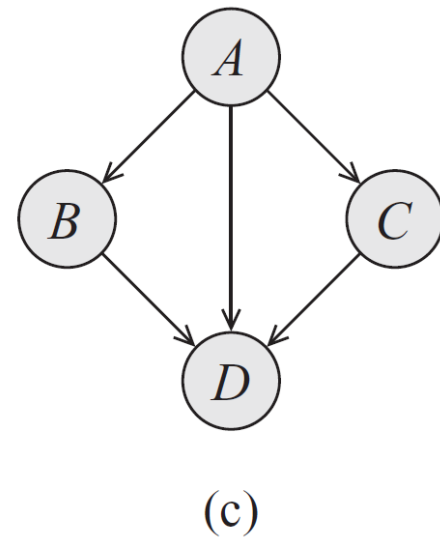
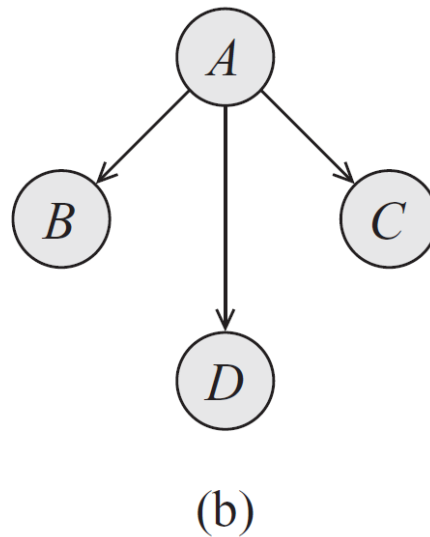
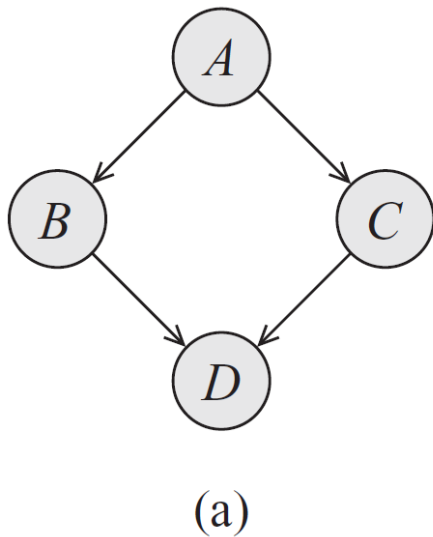
GENERAL GRAPHS

- Finding the best scoring G is NP-hard
- Even if we limit the indegree to be d , finding the best scoring G is NP-hard when $d \geq 2$
- Heuristic algorithms that might find the best one but not guaranteed
- This is a combinatorial optimization problem
 - Search the search space to find a good scoring G
 - Use local search
- Need three components
 1. The search space
 2. Scoring function
 3. A search procedure

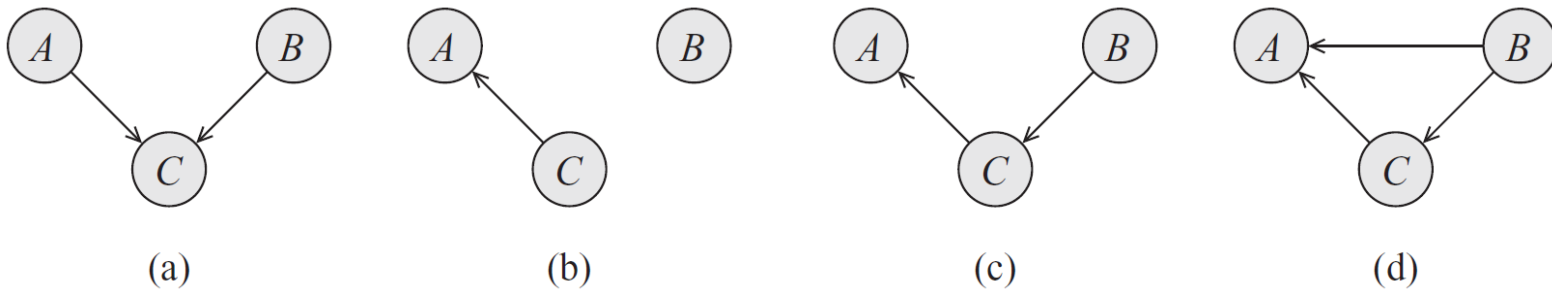
SEARCH SPACE

- The search space is a graph
 - Whose nodes are potential solutions,
 - Connected by search operators that allow us to move from one candidate solution to another
- We choose the following search operators
 - Edge addition
 - Edge deletion
 - Edge reversal
- Why can't we just start from G_\emptyset and use only edge addition operators? Why do we need edge deletion and reversal?
- When is edge reversal useful? Can't we just do it in two steps by deleting an edge and adding back it in the reverse direction?

EDGE DELETION EXAMPLE – FIG 18.5



EDGE REVERSAL EXAMPLE – FIG 18.6



SEARCH PROCEDURE

- Start with an initial graph
 - It can be the empty graph, a random graph, the best tree, or a network that is hand constructed
- Apply the search operators to generate new candidate structures, score them, and move to the one that has the highest score
- The runtime is roughly $O(K * n^2 * (M + n * d))$
 - K is the number of iterations,
 - n is the number of variables,
 - M is the number of data instances,
 - d is the max indegree

LOCAL MAXIMA

- What happens if none of the operators find a better scoring structure than the current one?
 - We hit either a *local maxima* or a *plateau*
- Unfortunately, plateaus are pretty common
 - When the scoring function satisfies score equivalence, then all the I-equivalent structures have equal score
 - The edge reversal operator will generate many I-equivalent structures

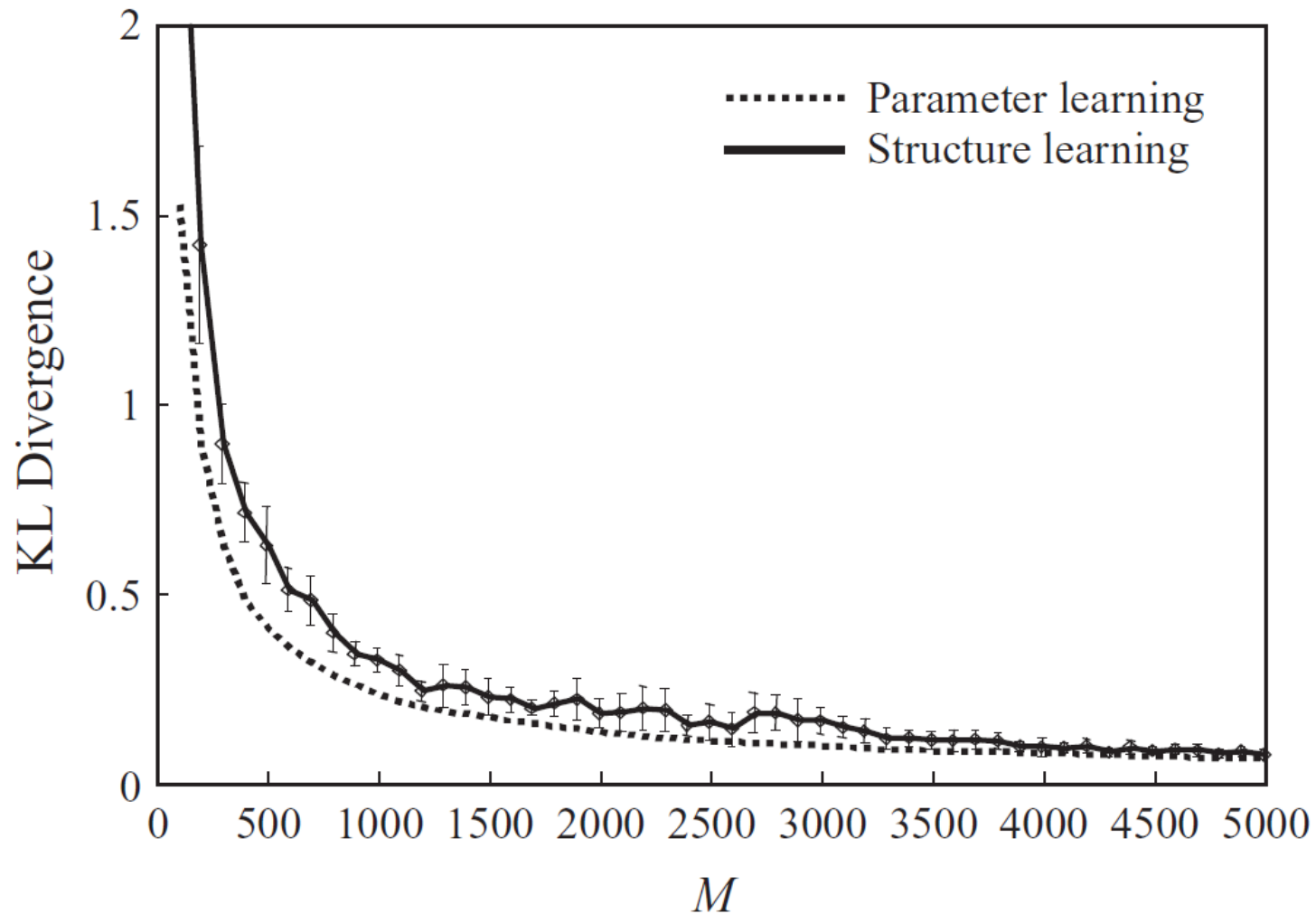
HOW TO DEAL WITH LOCAL MAXIMA AND PLATEAUS?

- *Basin flooding*
 - Keep track of visited states
- *Tabu search*
 - Keep a list of recently performed operations and do not reverse their effect in the next L steps
- *Data perturbation*
 - Make small changes to data D through replicas and removals.
 - The new data will cause similar networks to differ, but the big trends will be preserved

HOW SCORE DECOMPOSITION COMES IN HANDY

- Adding and deleting an edge changes the family of one variable and reversal changes the family of two variables
- We keep track of how much the score would change for each candidate operation
- Let's say we are considering to add the edge $X \rightarrow Y$.
 - We compute the score change, but we decide to do a better operation which does not change Y 's family.
 - In the next step, we do not need to compute the score change for $X \rightarrow Y$; we can use the previous computation.

HOW GOOD IS LOCAL SEARCH?



SUMMARY

- Not all the dependencies seen in the data are real
- Determining the edge directionality is sometimes impossible
- We have seen two structure learning approaches
 - Constraint-based
 - Sensitive to independence failures, which are common in real-world datasets
 - Score-based
 - Maximum likelihood score
 - Bayesian Information Criterion (BIC) score
- Tree-structured networks can be learned in polynomial time
- General structure learning is NP-hard but there are many local search algorithms that perform fairly well in practice

MARKOV NETWORKS

STRUCTURE LEARNING

- Two approaches
 - Constraint-based
 - Score-based

CONSTRAINT-BASED

- Assume \mathcal{H}^* is a P-Map for P^*
- We already know the algorithm
 - Use the first step of constraint-based structure learning for Bayesian networks

SCORE-BASED

- To compute the score for a candidate structure, we need to estimate the parameters
 - Unlike Bayesian networks, parameter estimation cannot be done independently for each variable
 - Unlike Bayesian networks, parameter estimation requires running inference
- Unlike Bayesian networks, the score for a structure does not decompose over variables
 - There is no clean and efficient way of reusing computations

IN PRACTICE

- Log-linear models are learned
- Search is performed over candidate features (over candidate cliques)
- L_1 -regularization is used so that the weights of the useless features become zero
 - If all the weights for all features for a candidate clique become zero, that clique is gone
 - If there is a single feature with a non-zero weight, the clique is retained
 - To achieve sparse solutions, *block L_1 -regularization* is used where a block of features for a candidate clique are penalized together