Probabilistic Graphical Models

Lecture 14: Learning Directed Graphical Model Structure

Matthew Walter

TTI-Chicago

May 28, 2020

Learning for Graphical Models (Revisited)

- The goal of learning is to learn a model that provides the "best" approximation to the true underlying distribution P^*
- In general, this is difficult due to:
  - Small datasets relative to the number of random variables,
  - Partial observability (e.g., some variables may not be observed) providing a sparse sampling of the true distribution
  - Computational cost
- The definition of "best" depends on the task, where for each we optimize an empirical loss over samples from P^*
  1. Density estimation: Estimate $\hat{P}$ that is as close as possible to P^*, where we often use log-loss (follows from KL-divergence)
  2. Prediction task: Classification error, Hamming loss, or conditional log-loss (e.g., for structured prediction)
  3. Knowledge discovery: Interested in understanding model structure
- Learning involves a trade-off between bias and model variance

Learning for Graphical Models (Revisited)

- We assume input of the form:
  1. Prior knowledge and/or constraints on the model class $\hat{\mathcal{M}}$
  2. A set $\mathcal{D} = \{\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(2)}, \ldots, \boldsymbol{\xi}^{(M)}\}$ of IID samples from P^*
- The output is a model $\hat{\mathcal{M}}$ that may include the structure and/or parameters of the graphical model
- The specifics of a particular learning algorithm vary with
  1. The type of output, i.e., a Bayesian network or Markov random field
  2. The constraints that we place on $\hat{\mathcal{M}}$
  3. The extent to which the training data is fully observed

Learning Procedure (Revisited)

1. Decide on an objective and corresponding loss

$$\mathbb{E}_{P^*}[\text{loss}(\boldsymbol{x}, \mathcal{M})]$$

2. Determine how to best estimate this from what we have, e.g., regularized empirical loss

$$\mathbb{E}_{\mathcal{D}}[\text{loss}(\boldsymbol{x}, \mathcal{M})] + R(\mathcal{M})$$

   When used with log-loss, the regularization term can be interpreted as a prior distribution over models, $P(\mathcal{M}) \propto \exp(-R(\mathcal{M}))$ (called *maximum a posteriori (MAP)* estimation)

3. Determine how to optimize over this objective function

$$\min_{\mathcal{M}} \; \mathbb{E}_{\mathcal{D}}[\text{loss}(\boldsymbol{x}, \mathcal{M})] + R(\mathcal{M})$$

Maximum Likelihood Parameter Estimation (Revisited)

- Use (log-)likelihood of the data $\mathcal{D} = \{x^{(1)}, \ldots, x^{(M)}\}$ as the (log-)loss
- The objective is to maximize the *likelihood function*

$$L(\boldsymbol{\theta} : \mathcal{D}) = \prod_m P(\boldsymbol{x}^{(m)}; \boldsymbol{\theta})$$

- In the case of multinomials, with $\{\#[1], \ldots, \#[K]\}$ being the tuple of counts for each x^k in $\mathcal{D}$, the likelihood function is

$$L(\boldsymbol{\theta} : \mathcal{D}) = \prod_k \theta_k^{\#[k]}$$

- The maximum likelihood estimate for a multinomial is

$$\hat{\theta}_k = \frac{\#[k]}{M}$$

MLE for Bayesian Networks (Revisited)

- Suppose that we know the Bayesian network structure G
- Let $\boldsymbol{\theta}_{X_i \mid \mathsf{Pa}_{X_i}}$ be the parameters that determine the CPD $P(X_i \mid \mathsf{Pa}_{X_i})$
- Assume we have a data set of samples $\mathcal{D} = \left\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(M)}\right\}$
- Maximum likelihood estimation corresponds to maximizing the log-likelihood $\ell(\boldsymbol{\theta} : \mathcal{D})$ (equivalent to maximizing the likelihood):

$$\frac{1}{M} \sum_{m=1}^{M} \log P(\boldsymbol{x}^{(m)}; \boldsymbol{\theta}) = \sum_{i=1}^{N} \frac{1}{M} \sum_{m=1}^{M} \log P(x_i^{(m)} \mid \mathsf{Pa}_{X_i}; \boldsymbol{\theta}_{X_i \mid \mathsf{Pa}_{X_i}})$$

- **Global decomposability**: Likelihood decomposes into a product of independent terms, one for each set of parameters
- We can optimize each of the local likelihoods separately (e.g., $\hat{\theta}_{x \mid \boldsymbol{u}} = \frac{\#[x, \boldsymbol{u}]}{\#[\boldsymbol{u}]}$ for the tabular case)

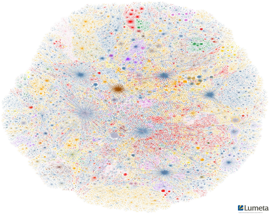
Limitations of ML Estimation (Revisited)

- Maximum likelihood estimation is purely data-driven and does not consider any a priori knowledge of the parameters (i.e., a prior over the parameters)
- Maximum likelihood estimation doesn't provide a measure of confidence in the resulting estimates

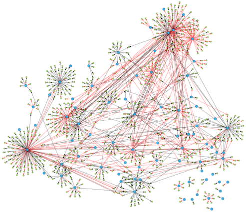
Knowledge Discovery

Structure ⇒ Knowledge Significant attention has been paid to learning graph structure (directed and undirected) from data

Social network graph



Internet graph



Gene regulatory network

Knowledge Discovery

- Objective is to learn higher-level properties about P^* (v.s. densities)
  - Nature of the dependencies, e.g., positive or negative correlation
  - Direct and indirect dependencies
- Learning the network structure provides more information, e.g., conditional independencies, and causal relationships
- Statistical methods can be used to identify dependencies, but can not differentiate between direct and indirect dependencies
- We care about discovering the correct model $\mathcal{M}^*$ rather than a different model $\hat{\mathcal{M}}$ that induces a similar distribution
- Metric is in terms of the differences between $\mathcal{M}^*$ and $\hat{\mathcal{M}}$

Knowledge Discovery

- However, the true model may not be **identifiable**
  - Bayesian network may have several I-equivalent structures
  - In this case, our best hope is to discover an I-equivalent graph structure
  - Problem is worse when the amount of data is limited and the relationships are weak
- When the number of variables is large relative to the amount of training data, pairs of variables can appear strongly correlated simply by chance
- In which of the following would you say that there is correlation?
  - Consider 100 trials of two coin flips:
    $\{(H, H) : 27; (H, T) : 22; (T, H) : 25; (T, T) : 26\}$
  - Consider a student newspaper article each day for 100 days and recording whether the words "snow" and "closed" exist:
    $\{(T, T) : 27; (T, F) : 22; (F, T) : 25; (F, F) : 26\}$

Structure Learning in Bayesian Networks

- The space of Bayesian networks is combinatorial, with $2^{\mathcal{O}(n^2)}$ different structures
- As the data is limited and noisy, it is difficult to detect which independencies are present in the distribution
- We need to decide whether or not to keep edges that we are unsure about: accept having spurious edges vs. unmodeled dependencies
- Intuition might suggest spurious edges to avoid invalid Independencies
- However, adding more parents to a variable results in **data fragmentation** as the data is spread across more bins
- If the objective is density estimation, it is generally better to favor sparser graphs

Structure Learning in Bayesian Networks

There are roughly three approaches to structure learning:

1. **Constraint-based structure learning** view Bayesian networks as independency representations and test for conditional dependencies and independencies in the data
2. **Score-based methods** treat learning as a *model selection* problem, finding the Bayesian network among a hypothesis class that achieves the highest score
3. **Bayesian model averaging** employ Bayesian reasoning to average the prediction of all possible structures

Constraint-based Structure Learning

Algorithm 3.2 Procedure to build a minimal I-map given an ordering

Procedure Build-Minimal-I-Map (
  $X_1, \ldots, X_n$ // an ordering of random variables in $\mathcal{X}$
  $\mathcal{I}$ // Set of independencies
)
  Set $\mathcal{G}$ to an empty graph over $\mathcal{X}$
  for $i = 1, \ldots, n$
    $\boldsymbol{U} \leftarrow \{X_1, \ldots, X_{i-1}\}$ // $\boldsymbol{U}$ is the current candidate for parents of X_i
    for $\boldsymbol{U}' \subseteq \{X_1, \ldots, X_{i-1}\}$
      if $\boldsymbol{U}' \subset \boldsymbol{U}$ and $(X_i \perp \{X_1, \ldots, X_{i-1}\} - \boldsymbol{U}' \mid \boldsymbol{U}') \in \mathcal{I}$ **then**
        $\boldsymbol{U} \leftarrow \boldsymbol{U}'$
      // At this stage $\boldsymbol{U}$ is a minimal set satisfying $(X_i \perp \{X_1, \ldots, X_{i-1}\} - \boldsymbol{U} \mid \boldsymbol{U})$
      // Now set $\boldsymbol{U}$ to be the parents of X_i
    for $X_j \in \boldsymbol{U}$
      Add $X_j \to X_i$ to $\mathcal{G}$
  return $\mathcal{G}$

- Assume access to a function that, given an arbitrary independence relation, returns True if it holds under P (e.g., $\mathcal{X}^2$ test)
- One approach: learn minimal I-map (e.g., BUILD-MINIMAL-I-MAP)
  - Sensitive to the ordering
  - Independence queries may involve a large number of variables (2^{i-1} for X_i)

Constraint-based Structure Learning

- Alternatively, we can learn an I-equivalence *class* of networks rather than a single network
- Requires that we make the following assumptions:
  - G^* has bounded indegree d: $|\text{Pa}_{X_i}^{G^*}| \leq d$ for all i
  - Independence test exactly answers any query involving $\leq 2d+2$ variables
  - The underlying distribution P^* is faithful to G^* (i.e., any independence in P^* is captured by G^*)
- Tends to be brittle: If we say that $X_i \perp X_j | X_k$ and they are not, the resulting structure may be very off
- Irrespective of the approach, there are several independence tests that can be used (e.g., hypothesis tests, $\mathcal{X}^2$ tests, mutual information-based tests, etc.)

Score-based Structure Learning

$$\max_{G,\boldsymbol{\theta}_G} \log P_{G,\boldsymbol{\theta}_G}(\mathcal{D}; G, \boldsymbol{\theta}_G) = \max_{G} \max_{\boldsymbol{\theta}_G} \log P_{G,\boldsymbol{\theta}_G}(\mathcal{D}; G, \boldsymbol{\theta}_G)$$
$$= \max_{G} \log P_{G,\hat{\boldsymbol{\theta}}_G}(\mathcal{D}; G, \hat{\boldsymbol{\theta}}_G)$$

- We define the score as $\text{score}_L(\mathcal{D}; G) = \log P_{G,\hat{\boldsymbol{\theta}}_G}(\mathcal{D}; G, \hat{\boldsymbol{\theta}}_G)$
- Consider each possible graph structure in terms of the best (i.e., MLE) parameters
- This is "optimistic", but still correct when the objective is maximum likelihood estimation

Score-based Structure Learning: Example

- Suppose that we have two binary random variables X and Y
- If we consider G_0 such that X and Y are independent:

$$\text{score}_L(\mathcal{D}; G_0) = \sum_m \log \hat{\theta}_{x^{(m)}} + \log \hat{\theta}_{y^{(m)}}$$

- If we consider a graph $G_1 : X \rightarrow Y$, then

$$\text{score}_L(\mathcal{D}; G_1) = \sum_m \log \hat{\theta}_{x^{(m)}} + \log \hat{\theta}_{y^{(m)} \mid x^{(m)}}$$

where $\hat{\theta}_x$ and $\hat{\theta}_{y \mid x}$ are the ML estimates for $P(X)$ and $P(Y \mid X)$

Score-based Structure Learning: Example (Continued)

- We can write the difference in scores as

$$\begin{aligned}\text{score}_L(\mathcal{D};G_1) - \text{score}_L(\mathcal{D};G_0) &= \sum_m \log \hat{\theta}_{y^{(m)}\,|\,x^{(m)}} - \log \hat{\theta}_{y^{(m)}} \\ &= \sum_{x,y} M[x,y] \log \hat{\theta}_{y\,|\,x} - \sum_y M[y] \log \hat{\theta}_y\end{aligned}$$

- Letting $\hat{P}$ be the empirical distribution, $M[x,y] = M \cdot \hat{P}(x,y)$ and $M[y] = M \cdot \hat{P}(y)$, and $\hat{\theta}_{y\,|\,x} = \hat{P}(y\,|\,x)$ and $\hat{\theta}_y = \hat{P}(y)$, the relative score becomes

$$\begin{aligned}\text{score}_L(\mathcal{D};G_1) - \text{score}_L(\mathcal{D};G_0) &= M \sum_{x,y} \hat{P}(x,y) \log \frac{\hat{P}(y\,|\,x)}{\hat{P}(y)} \\ &= M \cdot \mathbb{I}_{\hat{P}}(X;Y)\end{aligned}$$

  where $\mathbb{I}_{\hat{P}}(X;Y)$ is the *mutual information* between X and Y in $\hat{P}$
- Intuitively, higher mutual information implies a stronger dependency between X and Y, hence a bias towards $G_1 : X \rightarrow Y$

Score-based Structure Learning

- More generally, the likelihood score decomposes as

$$\mathsf{score}_L(\mathcal{D}; G) = M \sum_{i=1}^{n} \mathbb{I}_{\hat{P}}(X_i; \mathsf{Pa}_{X_i}^{G}) - M \sum_{i=1}^{n} \mathbb{H}_{\hat{P}}(X_i)$$

- The second term does not depend on the network structure (we can ignore it when comparing models)
- The likelihood of a graph measures the strength of the dependencies between variables and their parents, i.e., favor networks for which parents are informative about their children

Score-based Structure Learning

- However, mutual information is always nonnegative

$$\text{score}_L(\mathcal{D}; G_{X \to Y}) \geq \text{score}_L(\mathcal{D}; G_0)$$

- The maximum likelihood score *never* favors simpler networks

$$\mathbb{I}(\boldsymbol{X}; \boldsymbol{Y} \cup \boldsymbol{Z}) \geq \mathbb{I}(\boldsymbol{X}; \boldsymbol{Y})$$

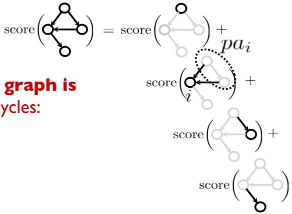
  (equality only holds if $\boldsymbol{X} \perp \boldsymbol{Z} \mid \boldsymbol{Y}$)

- Unless conditional independencies hold *exactly* in the data (very rare, e.g., due to statistical noise), more connections are always better!

Score-based Structure Learning

- Given G, assume prior distribution for CPD parameters $\boldsymbol{\theta}_{x_i \mid \mathrm{Pa}_{x_i}}$ is Dirichlet
- Choose G that maximizes the posterior $P(G \mid \mathcal{D}) \propto P(\mathcal{D} \mid G) P(G)$ (this is the *Bayesian score*)
- In order to compute the first term (the *marginal likelihood*), use the chain rule
- Obtain a combinatorial optimization problem over acyclic graphs

$$\mathrm{score}(G; D) = \sum_{i=1}^{n} \mathrm{score}(i \mid pa_i, D)$$



Finding highest scoring graph is NP-hard – must disallow cycles: