Probabilistic Graphical Models

Lecture 15: Learning: Undirected Graphical Models

Matthew Walter

TTI-Chicago

June 2, 2020

ML Estimation in Bayesian Networks (Revisited)

- Objective: Find the parameters $\boldsymbol{\theta} \in \Theta$ that maximize the log-likelihood of the data $\mathcal{D}$
- Assume that structure G is known and let $\boldsymbol{\theta}_{x_i \mid \mathsf{Pa}_{x_i}}$ be the parameters that determine the CPD $P(x_i \mid \mathsf{Pa}_{x_i})$
- Maximum likelihood estimation corresponds to solving:

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

- Gives rise to a closed-form solution:

$$\theta^{ML}_{x_i \mid \mathsf{Pa}_{x_i}} = \frac{\#[x_i, \mathsf{Pa}_{x_i}]}{\sum_{\hat{x}_i} \#[\hat{x}_i, \mathsf{Pa}_{x_i}]}$$

- We can estimate the parameters of each CPD independently because the objective function **decomposes** by variable and parent assignment

Log-Likelihood in Markov Networks

- Can we similarly decompose ML estimation for Markov networks?

$$P(\boldsymbol{X};\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_c \phi_c(\boldsymbol{D}_c;\boldsymbol{\theta})$$

- Consider the log-linear formulation of an MRF with $\phi(\boldsymbol{D}) = \exp\{-\sum_{i=1}^k \theta_i f_i(\boldsymbol{D})\}$

$$P(\boldsymbol{X};\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left\{\sum_{i=1}^k \theta_i f_i(\boldsymbol{D}_i)\right\}$$

  where $f_i(\boldsymbol{D}_i) \in \mathcal{F}$ is a feature defined over variables $\boldsymbol{D}_i$

- Recall (Lecture 4) that log-linear models can represent general Markov networks (e.g., with one indicator function feature per potential entry)

Log-Likelihood in Markov Networks

- For a set $\mathcal{D}$ of M samples, the log-likelihood is

$$\ell(\boldsymbol{\theta} : \mathcal{D}) = \log \left(\frac{1}{Z(\boldsymbol{\theta})^M} \prod_{m=1}^{M} \exp \left\{ \sum_{i=1}^{k} \theta_i f_i(\boldsymbol{D}_i^{(m)}) \right\} \right)$$

Log-Likelihood in Markov Networks

- For a set $\mathcal{D}$ of M samples, the log-likelihood is

$$\ell(\boldsymbol{\theta} : \mathcal{D}) = \log\left(\frac{1}{Z(\boldsymbol{\theta})^M} \prod_{m=1}^{M} \exp\left\{\sum_{i=1}^{k} \theta_i f_i(\boldsymbol{D}_i^{(m)})\right\}\right)$$
$$= \sum_{m=1}^{M} \sum_{i=1}^{k} \theta_i f_i(\boldsymbol{D}_i^{(m)}) - M \log Z(\boldsymbol{\theta})$$

Log-Likelihood in Markov Networks

- For a set $\mathcal{D}$ of M samples, the log-likelihood is

$$\begin{aligned}\ell(\boldsymbol{\theta} : \mathcal{D}) &= \log\left(\frac{1}{Z(\boldsymbol{\theta})^M}\prod_{m=1}^{M}\exp\left\{\sum_{i=1}^{k}\theta_i f_i(\boldsymbol{D}_i^{(m)})\right\}\right)\\ &= \sum_{m=1}^{M}\sum_{i=1}^{k}\theta_i f_i(\boldsymbol{D}_i^{(m)}) - M\log Z(\boldsymbol{\theta})\\ &= \sum_{i=1}^{k}\theta_i\left(\sum_{m=1}^{M} f_i(\boldsymbol{D}_i^{(m)})\right) - M\log Z(\boldsymbol{\theta})\end{aligned}$$

Log-Likelihood in Markov Networks

- For a set $\mathcal{D}$ of M samples, the log-likelihood is

$$\begin{aligned}
\ell(\boldsymbol{\theta} : \mathcal{D}) &= \log\left(\frac{1}{Z(\boldsymbol{\theta})^M}\prod_{m=1}^{M}\exp\left\{\sum_{i=1}^{k}\theta_i f_i(\boldsymbol{D}_i^{(m)})\right\}\right)\\
&= \sum_{m=1}^{M}\sum_{i=1}^{k}\theta_i f_i(\boldsymbol{D}_i^{(m)}) - M\log Z(\boldsymbol{\theta})\\
&= \sum_{i=1}^{k}\theta_i\left(\sum_{m=1}^{M} f_i(\boldsymbol{D}_i^{(m)})\right) - M\log Z(\boldsymbol{\theta})
\end{aligned}$$

- The sum of feature values in the data are the *sufficient statistics*

$$\frac{1}{M}\ell(\boldsymbol{\theta} : \mathcal{D}) = \sum_{i=1}^{k}\theta_i\mathbb{E}_{\mathcal{D}}[f_i(\boldsymbol{D}_i)] - \log Z(\boldsymbol{\theta})$$

  where $\mathbb{E}_{\mathcal{D}}[f_i(\boldsymbol{D}_i)]$ is the empirical expectation of f_i
- The first term is linear in the parameters

Log-Likelihood in Markov Networks

- The partition function is also a function of the parameters

$$\log Z(\boldsymbol{\theta}) = \log \sum_{\boldsymbol{x}} \exp \left\{ \sum_i \theta_i f_i(\boldsymbol{x}_i) \right\}$$

- $\log Z(\boldsymbol{\theta})$ does not decompose
- Consider the first and second derivatives of the log-partition:

$$\frac{\partial}{\partial \theta_i} \log Z(\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \sum_{\boldsymbol{x}} \frac{\partial}{\partial \theta_i} \exp \left\{ \sum_i \theta_i f_i(\boldsymbol{x}_i) \right\}$$

Log-Likelihood in Markov Networks

- The partition function is also a function of the parameters

$$\log Z(\boldsymbol{\theta}) = \log \sum_{\boldsymbol{x}} \exp \left\{ \sum_i \theta_i f_i(\boldsymbol{x}_i) \right\}$$

- $\log Z(\boldsymbol{\theta})$ does not decompose
- Consider the first and second derivatives of the log-partition:

$$\begin{aligned}
\frac{\partial}{\partial \theta_i} \log Z(\boldsymbol{\theta}) &= \frac{1}{Z(\boldsymbol{\theta})} \sum_{\boldsymbol{x}} \frac{\partial}{\partial \theta_i} \exp \left\{ \sum_i \theta_i f_i(\boldsymbol{x}_i) \right\} \\
&= \frac{1}{Z(\boldsymbol{\theta})} \sum_{\boldsymbol{x}} f_i(\boldsymbol{x}_i) \exp \left\{ \sum_i \theta_i f_i(\boldsymbol{x}_i) \right\}
\end{aligned}$$

Log-Likelihood in Markov Networks

- The partition function is also a function of the parameters

$$\log Z(\boldsymbol{\theta}) = \log \sum_{\boldsymbol{x}} \exp \left\{ \sum_i \theta_i f_i(\boldsymbol{x}_i) \right\}$$

- $\log Z(\boldsymbol{\theta})$ does not decompose
- Consider the first and second derivatives of the log-partition:

$$\begin{aligned}
\frac{\partial}{\partial \theta_i} \log Z(\boldsymbol{\theta}) &= \frac{1}{Z(\boldsymbol{\theta})} \sum_{\boldsymbol{x}} \frac{\partial}{\partial \theta_i} \exp \left\{ \sum_i \theta_i f_i(\boldsymbol{x}_i) \right\} \\
&= \frac{1}{Z(\boldsymbol{\theta})} \sum_{\boldsymbol{x}} f_i(\boldsymbol{x}_i) \exp \left\{ \sum_i \theta_i f_i(\boldsymbol{x}_i) \right\} \\
&= \mathbb{E}_{p(\boldsymbol{X};\boldsymbol{\theta})}[f_i]
\end{aligned}$$

Log-Likelihood in Markov Networks

- The partition function is also a function of the parameters

$$\log Z(\boldsymbol{\theta}) = \log \sum_{\boldsymbol{x}} \exp \left\{ \sum_i \theta_i f_i(\boldsymbol{x}_i) \right\}$$

- $\log Z(\boldsymbol{\theta})$ does not decompose
- Consider the first and second derivatives of the log-partition:

$$\frac{\partial}{\partial \theta_i} \log Z(\boldsymbol{\theta}) = \mathbb{E}_{p(\boldsymbol{X};\boldsymbol{\theta})}[f_i] \qquad \frac{\partial^2}{\partial \theta_i \partial \theta_j} \log Z(\boldsymbol{\theta}) = \mathsf{Cov}[f_i, f_j]$$

Log-Likelihood in Markov Networks

- The partition function is also a function of the parameters

$$\log Z(\boldsymbol{\theta}) = \log \sum_{\boldsymbol{x}} \exp \left\{ \sum_i \theta_i f_i(\boldsymbol{x}_i) \right\}$$

- $\log Z(\boldsymbol{\theta})$ does not decompose
- Consider the first and second derivatives of the log-partition:

$$\frac{\partial}{\partial \theta_i} \log Z(\boldsymbol{\theta}) = \mathbb{E}_{p(\boldsymbol{X};\boldsymbol{\theta})}[f_i] \qquad \frac{\partial^2}{\partial \theta_i \partial \theta_j} \log Z(\boldsymbol{\theta}) = \mathsf{Cov}[f_i, f_j]$$

- The gradient of the log-partition can be computed by *inference*, by computing the marginals with respect to current parameters $\boldsymbol{\theta}$

Log-Likelihood in Markov Networks

- The partition function is also a function of the parameters

$$\log Z(\boldsymbol{\theta}) = \log \sum_{\boldsymbol{x}} \exp \left\{ \sum_i \theta_i f_i(\boldsymbol{x}_i) \right\}$$

- $\log Z(\boldsymbol{\theta})$ does not decompose
- Consider the first and second derivatives of the log-partition:

$$\frac{\partial}{\partial \theta_i} \log Z(\boldsymbol{\theta}) = \mathbb{E}_{p(\boldsymbol{X};\boldsymbol{\theta})}[f_i] \qquad \frac{\partial^2}{\partial \theta_i \partial \theta_j} \log Z(\boldsymbol{\theta}) = \mathsf{Cov}[f_i, f_j]$$

- The gradient of the log-partition can be computed by *inference*, by computing the marginals with respect to current parameters $\boldsymbol{\theta}$
- Since the covariance matrix is positive semi-definite, $\log Z(\boldsymbol{\theta})$ is convex ($-\log Z(\boldsymbol{\theta})$ is concave)

ML Estimation in Markov Networks

$$\ell(\boldsymbol{\theta} : \mathcal{D}) = \sum_{i=1}^{k} \theta_i \left(\sum_{m=1}^{M} f_i(\boldsymbol{D}_i^{(m)}) \right) - M \log Z(\boldsymbol{\theta})$$

- Consider the gradient of the log-likelihood:

$$\frac{\partial}{\partial \theta_i} \ell(\boldsymbol{\theta} : \mathcal{D}) \overset{M}{\propto} \mathbb{E}_{\mathcal{D}}[f_i(\boldsymbol{D}_i)] - \mathbb{E}_{p(\boldsymbol{X};\boldsymbol{\theta})}[f_i]$$

- Corresponds to the difference in expectations
  - We want expected sufficient statistics in learned distribution to match empirical expectations
  - Equality constraint is an example of *moment matching*
  - ML estimate is *consistent* if model captures data-generating distribution

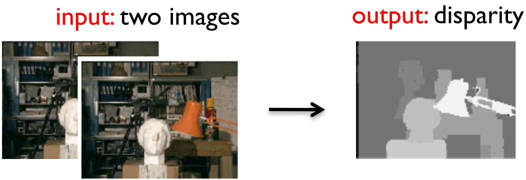
ML Estimation in Markov Networks

$$\ell(\boldsymbol{\theta} : \mathcal{D}) = \sum_{i=1}^{k} \theta_i \left(\sum_{m=1}^{M} f_i(\boldsymbol{D}_i^{(m)}) \right) - M \log Z(\boldsymbol{\theta})$$

- The log-likelihood is unimodal (no local optima), however global optimum may not be unique due to redundancy of parametrization
- No closed-form solution for global optimum
- Since the objective function is jointly concave, we can apply any iterative convex optimization method to learn parameters
- Can use gradient ascent, stochastic gradient ascent, or quasi-Newton methods (e.g., L-BFGS)
- However, gradient ascent **requires marginal inference** (for feature expectations) for every iteration, which may be prohibitive

Estimation for Conditional Likelihoods

- Suppose that we have sets of observed and query variables $\boldsymbol{X}$ and $\boldsymbol{Y}$
- We are interested in the conditional likelihood $P(\boldsymbol{Y} \mid \boldsymbol{X})$ (e.g., CRF)
- We have access to IID samples $\mathcal{D} = \{(\boldsymbol{y}^{(m)}, \boldsymbol{x}^{(m)})\}_{m=1}^{M}$
- We can train this model discriminatively (vs. generatively), since we only care about $P(\boldsymbol{Y} \mid \boldsymbol{X})$ (intuitively: don't waste time with $P(\boldsymbol{X})$)
- The result will tell us nothing about the joint $P(\boldsymbol{X}, \boldsymbol{Y})$



Estimation for Conditional Likelihoods

- The log-conditional-likelihood takes the form:

$$\ell_{\boldsymbol{Y} \mid \boldsymbol{X}}(\boldsymbol{\theta} : \mathcal{D}) = \sum_{m=1}^{M} \log P(\boldsymbol{y}^{(m)} \mid \boldsymbol{x}^{(m)}; \boldsymbol{\theta})$$

- This function is concave (global optimum)
- Each term on left is a log-likelihood of an MRF with different factors (original network reduced by evidence) and *its own partition function*

Estimation for Conditional Likelihoods

- The log-conditional-likelihood takes the form:

$$\ell_{\boldsymbol{Y} \mid \boldsymbol{X}}(\boldsymbol{\theta} : \mathcal{D}) = \sum_{m=1}^{M} \log P(\boldsymbol{y}^{(m)} \mid \boldsymbol{x}^{(m)}; \boldsymbol{\theta})$$

- This function is concave (global optimum)
- Each term on left is a log-likelihood of an MRF with different factors (original network reduced by evidence) and *its own partition function*
- The gradient takes the form:

$$\frac{\partial}{\partial \theta_i} \ell_{\boldsymbol{Y} \mid \boldsymbol{X}}(\boldsymbol{\theta} : \mathcal{D}) = \sum_{m=1}^{M} \left(f_i(\boldsymbol{y}^{(m)}, \boldsymbol{x}^{(m)}) - \mathbb{E}_{P(\boldsymbol{Y} \mid \boldsymbol{X}; \boldsymbol{\theta})}[f_i \mid \boldsymbol{x}^{(m)}] \right)$$

Estimation for Conditional Likelihoods

- The log-conditional-likelihood takes the form:

$$\ell_{\boldsymbol{Y}\,|\,\boldsymbol{X}}(\boldsymbol{\theta} : \mathcal{D}) = \sum_{m=1}^{M} \log P(\boldsymbol{y}^{(m)} \,|\, \boldsymbol{x}^{(m)}; \boldsymbol{\theta})$$

- This function is concave (global optimum)
- Each term on left is a log-likelihood of an MRF with different factors (original network reduced by evidence) and *its own partition function*
- The gradient takes the form:

$$\frac{\partial}{\partial \theta_i} \ell_{\boldsymbol{Y}\,|\,\boldsymbol{X}}(\boldsymbol{\theta} : \mathcal{D}) = \sum_{m=1}^{M} \left(f_i(\boldsymbol{y}^{(m)}, \boldsymbol{x}^{(m)}) - \mathbb{E}_{P(\boldsymbol{Y}\,|\,\boldsymbol{X};\boldsymbol{\theta})}[f_i \,|\, \boldsymbol{x}^{(m)}] \right)$$

- Each expectation on RHS is computed relative to a different model

Estimation for Conditional Likelihoods

- The log-conditional-likelihood takes the form:

$$\ell_{\boldsymbol{Y} \mid \boldsymbol{X}}(\boldsymbol{\theta} : \mathcal{D}) = \sum_{m=1}^{M} \log P(\boldsymbol{y}^{(m)} \mid \boldsymbol{x}^{(m)}; \boldsymbol{\theta})$$

- This function is concave (global optimum)
- Each term on left is a log-likelihood of an MRF with different factors (original network reduced by evidence) and *its own partition function*
- The gradient takes the form:

$$\frac{\partial}{\partial \theta_i} \ell_{\boldsymbol{Y} \mid \boldsymbol{X}}(\boldsymbol{\theta} : \mathcal{D}) = \sum_{m=1}^{M} \Big(f_i(\boldsymbol{y}^{(m)}, \boldsymbol{x}^{(m)}) - \mathbb{E}_{P(\boldsymbol{Y} \mid \boldsymbol{X}; \boldsymbol{\theta})}[f_i \mid \boldsymbol{x}^{(m)}] \Big)$$

- Each expectation on RHS is computed relative to a different model
- Training a CRF requires performing inference *for every single data point* at each iteration (vs. once for unconditional case)

Parameter Priors and Regularization

- ML parameter estimation is prone to overfitting to training data
- We can reduce tendency to overfit via a parameter prior $P(\boldsymbol{\theta})$
- Maximum a posteriori (MAP) estimation:

$$\begin{aligned}\arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta} \mid \mathcal{D}) &= \arg\max_{\boldsymbol{\theta}} P(\mathcal{D} \mid \boldsymbol{\theta}) P(\boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \left(\log P(\mathcal{D} \mid \boldsymbol{\theta}) + \log P(\boldsymbol{\theta})\right)\end{aligned}$$

- Without a closed-form solution, we only care that $P(\boldsymbol{\theta})$ is concave

Parameter Priors and Regularization

- Gaussian prior over parameters:

$$P(\boldsymbol{\theta}) \propto \prod_i \exp\{-\frac{\theta_i^2}{2\sigma^2}\}$$

  - Penalizes parameters with large magnitude
  - Corresponds to L_2-regularization

Parameter Priors and Regularization

- Gaussian prior over parameters:

$$P(\boldsymbol{\theta}) \propto \prod_i \exp\{-\frac{\theta_i^2}{2\sigma^2}\}$$

  - Penalizes parameters with large magnitude
  - Corresponds to L_2-regularization
- Laplacian prior:

$$P(\boldsymbol{\theta}) \propto \prod_i \exp\{-\frac{|\theta_i|}{\beta}\}$$

  - Penalizes parameters with large magnitude and promotes sparsity in θ (models tend to have fewer edges)
  - Corresponds to L_1-regularization

Parameter Priors and Regularization

- Gaussian prior over parameters:

$$P(\boldsymbol{\theta}) \propto \prod_i \exp\{-\frac{\theta_i^2}{2\sigma^2}\}$$

  - Penalizes parameters with large magnitude
  - Corresponds to L_2-regularization
- Laplacian prior:

$$P(\boldsymbol{\theta}) \propto \prod_i \exp\{-\frac{|\theta_i|}{\beta}\}$$

  - Penalizes parameters with large magnitude and promotes sparsity in θ (models tend to have fewer edges)
  - Corresponds to L_1-regularization
- Hyperparameters σ and β are important — tune via cross-validation

Learning with Approximate Inference

- ML parameter estimation for MRFs requires full inference at each iteration
- Recall the form of the gradient

$$\frac{\partial}{\partial \theta_i} \ell(\boldsymbol{\theta} : \mathcal{D}) \propto \mathbb{E}_{\mathcal{D}}[f_i(\boldsymbol{D}_i)] - \mathbb{E}_{p(\boldsymbol{X};\boldsymbol{\theta})}[f_i]$$

- Learning (i.e., computing the gradients) requires **marginals**

Learning with Approximate Inference

- ML parameter estimation for MRFs requires full inference at each iteration
- Recall the form of the gradient

$$\frac{\partial}{\partial \theta_i} \ell(\boldsymbol{\theta} : \mathcal{D}) \propto \mathbb{E}_{\mathcal{D}}[f_i(\boldsymbol{D}_i)] - \mathbb{E}_{p(\boldsymbol{X};\boldsymbol{\theta})}[f_i]$$

- Learning (i.e., computing the gradients) requires **marginals**
- We can use any of the approximate inference methods that we've learned to estimate marginals/expectations

Learning with Approximate Inference

- ML parameter estimation for MRFs requires full inference at each iteration
- Recall the form of the gradient

$$\frac{\partial}{\partial \theta_i} \ell(\boldsymbol{\theta} : \mathcal{D}) \propto \mathbb{E}_{\mathcal{D}}[f_i(\boldsymbol{D}_i)] - \mathbb{E}_{p(\boldsymbol{X};\boldsymbol{\theta})}[f_i]$$

- Learning (i.e., computing the gradients) requires **marginals**
- We can use any of the approximate inference methods that we've learned to estimate marginals/expectations
- *However*, nonconvergence of approximate inference (or convergence to approximate value) can lead to inaccurate gradients
  - Using loopy BP to compute marginals (f_i must be a subset of a cluster C) may yield unstable (oscillating) gradients

Optimizing an Approximate Objective

- Approximately optimizing true objective with approximate inference can be formulated as exact optimization of approximate objective
- Consider the log-likelihood for a single instance $\boldsymbol{x}^{(m)}$

$$
\begin{aligned}
\ell(\boldsymbol{\theta} : \boldsymbol{x}^{(m)}) &= \log \tilde{P}(\boldsymbol{x}^{(m)}; \boldsymbol{\theta}) - \log Z(\boldsymbol{\theta}) \\
&= \log \tilde{P}(\boldsymbol{x}^{(m)}; \boldsymbol{\theta}) - \log \left(\sum_{\boldsymbol{x}'} \tilde{P}(\boldsymbol{x}'; \boldsymbol{\theta}) \right)
\end{aligned}
$$

Optimizing an Approximate Objective

- Approximately optimizing true objective with approximate inference can be formulated as exact optimization of approximate objective
- Consider the log-likelihood for a single instance $\boldsymbol{x}^{(m)}$

$$
\begin{aligned}
\ell(\boldsymbol{\theta} : \boldsymbol{x}^{(m)}) &= \log \tilde{P}(\boldsymbol{x}^{(m)}; \boldsymbol{\theta}) - \log Z(\boldsymbol{\theta}) \\
&= \log \tilde{P}(\boldsymbol{x}^{(m)}; \boldsymbol{\theta}) - \log \left(\sum_{\boldsymbol{x}'} \tilde{P}(\boldsymbol{x}'; \boldsymbol{\theta}) \right)
\end{aligned}
$$

- Intuitively, $\boldsymbol{\theta}$ should increase the first term while respecting the second (which involves summing over all Val$(\mathcal{X})$)

Optimizing an Approximate Objective

- Approximately optimizing true objective with approximate inference can be formulated as exact optimization of approximate objective
- Consider the log-likelihood for a single instance $\boldsymbol{x}^{(m)}$

$$\begin{aligned}\ell(\boldsymbol{\theta} : \boldsymbol{x}^{(m)}) &= \log \tilde{P}(\boldsymbol{x}^{(m)}; \boldsymbol{\theta}) - \log Z(\boldsymbol{\theta}) \\ &= \log \tilde{P}(\boldsymbol{x}^{(m)}; \boldsymbol{\theta}) - \log \left(\sum_{\boldsymbol{x}'} \tilde{P}(\boldsymbol{x}'; \boldsymbol{\theta}) \right)\end{aligned}$$

- Intuitively, $\boldsymbol{\theta}$ should increase the first term while respecting the second (which involves summing over all $\text{Val}(\mathcal{X})$)
- Log-likelihood amounts to increasing distance between log-measure of $\boldsymbol{x}^{(m)}$ and the aggregate measure of all instances

Optimizing an Approximate Objective

- Approximately optimizing true objective with approximate inference can be formulated as exact optimization of approximate objective
- Consider the log-likelihood for a single instance $\boldsymbol{x}^{(m)}$

$$\begin{aligned}\ell(\boldsymbol{\theta} : \boldsymbol{x}^{(m)}) &= \log \tilde{P}(\boldsymbol{x}^{(m)}; \boldsymbol{\theta}) - \log Z(\boldsymbol{\theta}) \\ &= \log \tilde{P}(\boldsymbol{x}^{(m)}; \boldsymbol{\theta}) - \log \left(\sum_{\boldsymbol{x}'} \tilde{P}(\boldsymbol{x}'; \boldsymbol{\theta}) \right)\end{aligned}$$

- Intuitively, $\boldsymbol{\theta}$ should increase the first term while respecting the second (which involves summing over all Val$(\mathcal{X})$)
- Log-likelihood amounts to increasing distance between log-measure of $\boldsymbol{x}^{(m)}$ and the aggregate measure of all instances
- Can we come up with an approximate objective that increases distance relative to a more tractable set?

Optimizing an Approximate Objective

- We can approximate the original learning objective

$$\ell(\boldsymbol{\theta} : \mathcal{D}) = \sum_{i=1}^{k} \theta_i \left(\sum_{m=1}^{M} f_i(\boldsymbol{D}_i^{(m)}) \right) - M \log Z(\boldsymbol{\theta})$$

  with one using a tractable approximation to the log-partition function

$$\tilde{l}(\boldsymbol{\theta} : \mathcal{D}) = \sum_{i=1}^{k} \theta_i \left(\sum_{m=1}^{M} f_i(\boldsymbol{D}_i^{(m)}) \right) - M \log \tilde{Z}(\boldsymbol{\theta})$$

- Recall from Lecture 10 that we can compute a bound on the log-partition function
- Thus, we can bound the learning objective

Pseudo-Likelihood

- Alternatively, we can consider an altogether different objective function for which learning doesn't require full inference
- Consider the likelihood of a single instance $\boldsymbol{x}^{(m)}$. Via chain rule:

$$P(\boldsymbol{x}^{(m)}) = \prod_{j=1}^{n} P(x_j^{(m)} \mid x_1^{(m)}, \ldots, x_{j-1}^{(m)})$$

Pseudo-Likelihood

- Alternatively, we can consider an altogether different objective function for which learning doesn't require full inference
- Consider the likelihood of a single instance $\boldsymbol{x}^{(m)}$. Via chain rule:

$$P(\boldsymbol{x}^{(m)}) = \prod_{j=1}^{n} P(x_j^{(m)} \mid x_1^{(m)}, \ldots, x_{j-1}^{(m)})$$

- Approximate $P(x_j^{(m)} \mid x_1^{(m)}, \ldots, x_{j-1}^{(m)})$ by $P(x_j^{(m)} \mid \boldsymbol{x}_{-j}^{(m)})$ (i.e., condition over all other variables)

Pseudo-Likelihood

- Alternatively, we can consider an altogether different objective function for which learning doesn't require full inference
- Consider the likelihood of a single instance $\boldsymbol{x}^{(m)}$. Via chain rule:

$$P(\boldsymbol{x}^{(m)}) = \prod_{j=1}^{n} P(x_j^{(m)} \,|\, x_1^{(m)}, \ldots, x_{j-1}^{(m)})$$

- Approximate $P(x_j^{(m)} \,|\, x_1^{(m)}, \ldots, x_{j-1}^{(m)})$ by $P(x_j^{(m)} \,|\, \boldsymbol{x}_{-j}^{(m)})$ (i.e., condition over all other variables)
- Gives rise to the following approximation

$$\begin{aligned} P(\boldsymbol{x}^{(m)}) &\approx \prod_{j=1}^{n} P(x_j^{(m)} \,|\, x_1^{(m)}, \ldots, x_{j-1}^{(m)}, x_{j+1}^{(m)}, \ldots x_n^{(m)}) \\ &= \prod_{j=1}^{n} P(x_j^{(m)} \,|\, \boldsymbol{x}_{-j}^{(m)}) \end{aligned}$$

Pseudo-Likelihood

- Results in the *pseudolikelihood* approximation to original objective:

$$\begin{aligned}\ell_{\mathsf{PL}}(\boldsymbol{\theta} : \mathcal{D}) &= \frac{1}{M}\sum_{m}\sum_{j}\log P(x_j^{(m)} \mid \boldsymbol{x}_{-j}^{(m)}; \boldsymbol{\theta})\\ &= \frac{1}{M}\sum_{m}\sum_{j}\log P(x_j^{(m)} \mid \boldsymbol{x}_{\mathsf{MB}(j)}^{(m)}; \boldsymbol{\theta})\end{aligned}$$

  where $\boldsymbol{x}_{\mathsf{MB}(j)}$ is the Markov blanket for x_j

- The pseudolikelihood objective for a single instance becomes (via Bayes' rule):

$$\sum_{j}\log P(x_j^{(m)} \mid \boldsymbol{x}_{-j}^{(m)}; \boldsymbol{\theta}) = \sum_{j}\left(\log \tilde{P}(x^{(m)}) - \log \sum_{x'_j} \tilde{P}(x'_j, \boldsymbol{x}_{\mathsf{MB}(j)}^{(m)})\right)$$

Pseudo-Likelihood

$$\sum_j \log P(x_j^{(m)} \mid \boldsymbol{x}_{-j}^{(m)}; \boldsymbol{\theta}) = \sum_j \left(\log \tilde{P}(x^{(m)}) - \log \sum_{x'_j} \tilde{P}(x'_j, \boldsymbol{x}_{\mathsf{MB}(j)}^{(m)}) \right)$$

- The aggregate involves summation only over x_j (tractable)
- Optimization increases distance between each example and local neighborhood
- Has many partition functions, one for each variable and each setting of its neighbors, rather than one big one
- Objective is still concave in $\boldsymbol{\theta}$ (global optima)
- Assuming that data is drawn from an MRF with parameters $\boldsymbol{\theta}^*$, one can show that $\boldsymbol{\theta}^{\mathsf{PL}} \rightarrow \boldsymbol{\theta}^*$ as $M \rightarrow \infty$