Inference and Representation

David Sontag

New York University

Lecture 12, Dec. 2, 2014

Acknowledgements: Partially based on slides by Eric Xing at CMU and Andrew McCallum at UMass Amherst

Today: learning undirected graphical models

1. Learning MRFs
   a. Feature-based (log-linear) representation of MRFs
   b. Maximum likelihood estimation
   c. Maximum entropy view
2. Getting around complexity of inference
   a. Using approximate inference (e.g., TRW) within learning
   b. Pseudo-likelihood

Recall: ML estimation in Bayesian networks

- Maximum likelihood estimation: $\max_\theta \ell(\theta; \mathcal{D})$, where

$$\begin{aligned}
\ell(\theta; \mathcal{D}) = \log p(\mathcal{D}; \theta) &= \sum_{\mathbf{x} \in \mathcal{D}} \log p(\mathbf{x}; \theta) \\
&= \sum_i \sum_{\hat{\mathbf{x}}_{pa(i)}} \sum_{\substack{\mathbf{x} \in \mathcal{D}: \\ \mathbf{x}_{pa(i)} = \hat{\mathbf{x}}_{pa(i)}}} \log p(x_i \mid \hat{\mathbf{x}}_{pa(i)})
\end{aligned}$$

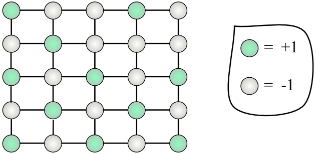
- In Bayesian networks, we have the closed form ML solution:

$$\theta^{ML}_{x_i | \mathbf{x}_{pa(i)}} = \frac{N_{x_i, \mathbf{x}_{pa(i)}}}{\sum_{\hat{x}_i} N_{\hat{x}_i, \mathbf{x}_{pa(i)}}}$$

  where $N_{x_i, \mathbf{x}_{pa(i)}}$ is the number of times that the (partial) assignment $x_i, \mathbf{x}_{pa(i)}$ is observed in the training data

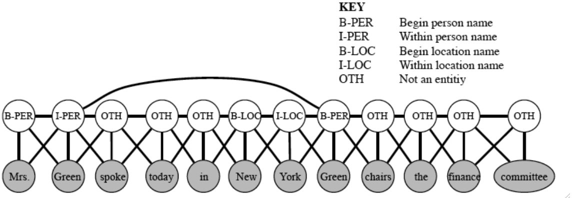
- We were able to estimate each CPD independently because the objective **decomposes** by variable and parent assignment

- How do we learn the parameters of an Ising model?



$$p(x_1, \cdots, x_n) = \frac{1}{Z} \exp\left(\sum_{i<j} w_{i,j} x_i x_j - \sum_i u_i x_i \right)$$

- What about for a skip-chain CRF?



Bad news for Markov networks

- The global normalization constant $Z(\theta)$ kills decomposability:

$$
\begin{aligned}
\theta^{ML} &= \arg\max_{\theta} \log \prod_{\mathbf{x}\in\mathcal{D}} p(\mathbf{x};\theta) \\
&= \arg\max_{\theta} \sum_{\mathbf{x}\in\mathcal{D}} \left(\sum_{c} \log \phi_c(\mathbf{x}_c;\theta) - \log Z(\theta) \right) \\
&= \arg\max_{\theta} \left(\sum_{\mathbf{x}\in\mathcal{D}} \sum_{c} \log \phi_c(\mathbf{x}_c;\theta) \right) - |\mathcal{D}| \log Z(\theta)
\end{aligned}
$$

- The log-partition function prevents us from decomposing the objective into a sum over terms for each potential
- Solving for the parameters becomes much more complicated

What are the parameters?

- Parameterize $\phi_c(\mathbf{x}_c; \theta)$ using a log-linear parameterization:
  - Single weight vector $\mathbf{w} \in \mathbb{R}^d$ that is used globally
  - For each potential c, a vector-valued **feature function** $\mathbf{f}_c(\mathbf{x}_c) \in \mathbb{R}^d$
  - Then, $\phi_c(\mathbf{x}_c; \mathbf{w}) = \exp(\mathbf{w} \cdot \mathbf{f}_c(\mathbf{x}_c))$
- Example: discrete-valued MRF with only edge potentials, where each variable takes k states
  - Let $d = k^2|E|$, and let $w_{i,j,x_i,x_j} = \log \phi_{ij}(x_i, x_j)$
  - Let $f_{i,j}(x_i, x_j)$ have a 1 in the dimension corresponding to (i, j, x_i, x_j) and 0 elsewhere
- The joint distribution is in the *exponential family*!

$$p(\mathbf{x}; \mathbf{w}) = \exp\{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}) - \log Z(\mathbf{w})\},$$

  where $f(\mathbf{x}) = \sum_c f_c(\mathbf{x}_c)$ and $Z(\mathbf{w}) = \sum_{\mathbf{x}} \exp\{\sum_c \mathbf{w} \cdot f_c(\mathbf{x}_c)\}$
- This formulation allows for parameter sharing

Log-likelihood for log-linear models

$$\begin{aligned}
\theta^{ML} &= \arg\max_{\theta} \left(\sum_{\mathbf{x}\in\mathcal{D}} \sum_{c} \log \phi_c(\mathbf{x}_c;\theta) \right) - |\mathcal{D}| \log Z(\theta) \\
&= \arg\max_{\mathbf{w}} \left(\sum_{\mathbf{x}\in\mathcal{D}} \sum_{c} \mathbf{w}\cdot\mathbf{f}_c(\mathbf{x}_c) \right) - |\mathcal{D}| \log Z(\mathbf{w}) \\
&= \arg\max_{\mathbf{w}} \; \mathbf{w}\cdot \left(\sum_{\mathbf{x}\in\mathcal{D}} \sum_{c} \mathbf{f}_c(\mathbf{x}_c) \right) - |\mathcal{D}| \log Z(\mathbf{w})
\end{aligned}$$

- The first term is linear in $\mathbf{w}$
- The second term is also a function of $\mathbf{w}$:

$$\log Z(\mathbf{w}) = \log \sum_{\mathbf{x}} \exp\left(\mathbf{w} \cdot \sum_{c} \mathbf{f}_c(\mathbf{x}_c) \right)$$

Log-likelihood for log-linear models

$$\log Z(\mathbf{w}) = \log \sum_{\mathbf{x}} \exp\left(\mathbf{w} \cdot \sum_c \mathbf{f}_c(\mathbf{x}_c)\right)$$

- $\log Z(\mathbf{w})$ does not decompose
  - No closed form solution; even *computing* likelihood requires inference
- Letting $\mathbf{f}(\mathbf{x}) = \sum_c \mathbf{f}_c(\mathbf{x}_c)$, we showed in Lecture 7 that:

$$\nabla_{\mathbf{w}} \log Z(\mathbf{w}) = \mathbb{E}_{p(\mathbf{x};\mathbf{w})}[\mathbf{f}(\mathbf{x})] = \sum_c \mathbb{E}_{p(\mathbf{x}_c;\mathbf{w})}[\mathbf{f}_c(\mathbf{x}_c)]$$

- Thus, the gradient of the log-partition function can be computed by *inference*, computing marginals with respect to the current parameters $\mathbf{w}$
- Similarly, you can show that 2nd derivative of the log-partition function gives the second-order moments, i.e.

$$\nabla^2 \log Z(\mathbf{w}) = \left(\mathbb{E}_{p(\mathbf{x};\mathbf{w})}[f^i(\mathbf{x}) f^j(\mathbf{x})]\right)_{ij} = \mathrm{cov}[\mathbf{f}(\mathbf{x})]$$

- Since covariance matrices are always positive semi-definite, this proves that $\log Z(\mathbf{w})$ is convex (so $-\log Z(\mathbf{w})$ is concave)

Solving the maximum likelihood problem in MRFs

$$\ell(\mathbf{w};\mathcal{D}) = \frac{1}{|\mathcal{D}|}\mathbf{w}\cdot\left(\sum_{\mathbf{x}\in\mathcal{D}}\sum_c \mathbf{f}_c(\mathbf{x}_c)\right) - \log Z(\mathbf{w})$$

- First, note that the weights $\mathbf{w}$ are unconstrained, i.e. $\mathbf{w}\in\mathbb{R}^d$
- The objective function is jointly concave. Apply any **convex optimization** method to learn!
- Can use gradient ascent, **stochastic gradient ascent**, quasi-Newton methods such as limited memory BFGS (L-BFGS)
- Let's study some properties of the ML solution!

$$\begin{aligned}\frac{d}{dw_k}\ell(\mathbf{w};\mathcal{D}) &= \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}}\sum_c (\mathbf{f}_c(\mathbf{x}_c))_k - \sum_c \mathbb{E}_{p(\mathbf{x}_c;\mathbf{w})}[(\mathbf{f}_c(\mathbf{x}_c))_k] \\ &= \sum_c \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}}(\mathbf{f}_c(\mathbf{x}_c))_k - \sum_c \mathbb{E}_{p(\mathbf{x}_c;\mathbf{w})}[(\mathbf{f}_c(\mathbf{x}_c))_k]\end{aligned}$$

The gradient of the log-likelihood

$$\frac{\partial}{\partial w_k}\ell(\mathbf{w};\mathcal{D}) = \sum_c \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}}(\mathbf{f}_c(\mathbf{x}_c))_k - \sum_c \mathbb{E}_{p(\mathbf{x}_c;\mathbf{w})}[(\mathbf{f}_c(\mathbf{x}_c))_k]$$

- Difference of expectations!
- Consider the earlier pairwise MRF example. This then reduces to:

$$\frac{\partial}{\partial w_{i,j,\hat{x}_i,\hat{x}_j}}\ell(\mathbf{w};\mathcal{D}) = \left(\frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} 1[x_i=\hat{x}_i, x_j=\hat{x}_j]\right) - p(\hat{x}_i,\hat{x}_j;\mathbf{w})$$

- Setting derivative to zero, we see that for the maximum likelihood parameters $\mathbf{w}^{ML}$, we have

$$p(\hat{x}_i,\hat{x}_j;\mathbf{w}^{ML}) = \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} 1[x_i=\hat{x}_i, x_j=\hat{x}_j]$$

  for all edges $ij \in E$ and states $\hat{x}_i, \hat{x}_j$
- Model marginals for ML solution equal the empirical marginals!
- Called **moment matching**, and is a property of maximum likelihood learning in exponential families

Gradient ascent requires repeated marginal inference, which in many models is **hard**!

We will return to this shortly.

Maximum entropy (MaxEnt)

- We can approach the modeling task from an entirely different point of view
- Suppose we know some expectations with respect to a (fully general) distribution $p(\mathbf{x})$:

$$\text{(true)} \ \sum_{\mathbf{x}} p(\mathbf{x}) f_i(\mathbf{x}), \qquad \text{(empirical)} \ \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} f_i(\mathbf{x}) = \alpha_i$$

- Assuming that the expectations are consistent with one another, there may exist **many** distributions which satisfy them. Which one should we select?

  The most uncertain or flexible one, i.e., the one with maximum entropy.

- This yields a new optimization problem:

$$\max_p H(p(\mathbf{x})) = -\sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x})$$

$$\text{s.t.} \qquad \sum_{\mathbf{x}} p(\mathbf{x}) f_i(\mathbf{x}) = \alpha_i$$

$$\sum p(\mathbf{x}) = 1 \quad \text{(strictly concave w.r.t. } p(\mathbf{x})\text{)}$$

What does the MaxEnt solution look like? (c.f. Lec. 9)

- To solve the MaxEnt problem, we form the Lagrangian:

$$L = -\sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x}) - \sum_i \lambda_i \left(\sum_{\mathbf{x}} p(\mathbf{x}) f_i(\mathbf{x}) - \alpha_i \right) - \mu \left(\sum_{\mathbf{x}} p(\mathbf{x}) - 1 \right)$$

- Then, taking the derivative of the Lagrangian,

$$\frac{\partial L}{\partial p(\mathbf{x})} = -1 - \log p(\mathbf{x}) - \sum_i \lambda_i f_i(\mathbf{x}) - \mu$$

- And setting to zero, we obtain:

$$p^*(\mathbf{x}) = \exp\left(-1 - \mu - \sum_i \lambda_i f_i(\mathbf{x}) \right) = e^{-1-\mu} e^{-\sum_i \lambda_i f_i(\mathbf{x})}$$

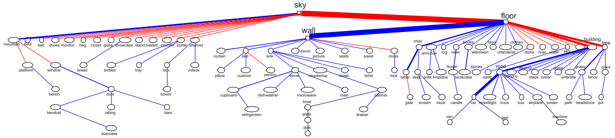
- From the constraint $\sum_{\mathbf{x}} p(\mathbf{x}) = 1$ we obtain $e^{1+\mu} = \sum_{\mathbf{x}} e^{-\sum_i \lambda_i f_i(\mathbf{x})} = Z(\lambda)$
- We conclude that the maximum entropy distribution has the form (substituting $w_i = -\lambda_i$)

$$p^*(\mathbf{x}) = \frac{1}{Z(\mathbf{w})} \exp(\sum_i w_i f_i(\mathbf{x}))$$

Equivalence of maximum likelihood and maximum entropy

- Feature constraints + MaxEnt $\Rightarrow$ exponential family!
- We have seen a case of convex duality:
  - In one case, we assume exponential family and show that ML implies model expectations must match empirical expectations
  - In the other case, we assume model expectations must match empirical feature counts and show that MaxEnt implies exponential family distribution
- Can show that one is the dual of the other, and thus both obtain the same value of the objective at optimality (no duality gap)
- Besides providing insight into the ML solution, this also gives an alternative way to (approximately) solve the learning problem

Chow-Liu algorithm for MRF structure learning



- Recall the PS 3 problem on structure learning of tree-structured MRFs:

$$\max_T \max_{\theta_T} \sum_{\mathbf{x}\in\mathcal{D}} \log p_T(\mathbf{x}; \theta_T).$$

- You used the fact that, for a fixed tree T, the maximum likelihood parameters, i.e.

$$\theta_T^{ML} = \arg\max_{\theta_T} \sum_{\mathbf{x}\in\mathcal{D}} \log p_T(\mathbf{x}; \theta_T).$$

  have $p_T(x_i, x_j; \theta_T^{ML}) = \hat{p}(x_i, x_j)$, the latter computed from the data $\mathcal{D}$

- For the special case of trees, the mapping $\mu \to \theta$ has a simple closed-form solution:

$$p_T(\mathbf{x}) = \prod_{(i,j)\in T} \frac{p_T(x_i, x_j)}{p_T(x_i)p_T(x_j)} \prod_{j\in V} p_T(x_j)$$

Chow-Liu algorithm for MRF structure learning

- This then gave the following optimization problem

$$\max_T \sum_{\mathbf{x}\in\mathcal{D}} \log\left[\prod_{(i,j)\in T} \frac{\hat{p}(x_i, x_j)}{\hat{p}(x_i)\hat{p}(x_j)} \prod_{j\in V} \hat{p}(x_j)\right]$$

  which you solved using a maximum spanning tree algorithm
- For general graphs, solving the maximum entropy problem is itself intractable

How can we get around the complexity of inference during learning?

Monte Carlo methods

- Recall the original learning objective

$$\ell(\mathbf{w}; \mathcal{D}) = \frac{1}{|\mathcal{D}|}\mathbf{w} \cdot \left(\sum_{\mathbf{x} \in \mathcal{D}} \sum_{c} \mathbf{f}_c(\mathbf{x}_c) \right) - \log Z(\mathbf{w})$$

- Use any of the sampling approaches (e.g., Gibbs sampling) that we discussed in Lecture 9
- All we need for learning (i.e., to compute the derivative of $\ell(\mathbf{w}, \mathcal{D})$) are **marginals** of the distribution
- No need to ever estimate $\log Z(\mathbf{w})$

Using approximations of the log-partition function

- We can substitute the original learning objective

$$\ell(\mathbf{w}; \mathcal{D}) = \frac{1}{|\mathcal{D}|}\mathbf{w} \cdot \Big(\sum_{\mathbf{x} \in \mathcal{D}} \sum_{c} \mathbf{f}_c(\mathbf{x}_c) \Big) - \log Z(\mathbf{w})$$

  with one that uses a tractable approximation of the log-partition function:

$$\tilde{\ell}(\mathbf{w}; \mathcal{D}) = \frac{1}{|\mathcal{D}|}\mathbf{w} \cdot \Big(\sum_{\mathbf{x} \in \mathcal{D}} \sum_{c} \mathbf{f}_c(\mathbf{x}_c) \Big) - \log \tilde{Z}(\mathbf{w})$$

- Recall from Lecture 9 that we came up with a *convex relaxation* that provided an upper bound on the log-partition function,

$$\log Z(\mathbf{w}) \leq \log \tilde{Z}(\mathbf{w})$$

  (e.g., tree-reweighted belief propagation, log-determinant relaxation)

- Using this, we obtain a *lower bound* on the learning objective

$$\ell(\mathbf{w}; \mathcal{D}) \geq \tilde{\ell}(\mathbf{w}; \mathcal{D})$$

- Again, to compute the derivatives we only need *pseudo-marginals* from the variational inference algorithm

Pseudo-likelihood

- Alternatively, can we come up with a *different* objective function (i.e., a different *estimator*) which succeeds at learning while avoiding inference altogether?
- Pseudo-likelihood method (Besag 1971) yields an exact solution if the data is generated by a model in our model family $p(\mathbf{x}; \theta^*)$ and $|\mathcal{D}| \to \infty$ (i.e., it is **consistent**)
- Note that, via the chain rule,

$$p(\mathbf{x}; \mathbf{w}) = \prod_i p(x_i | x_1, \ldots, x_{i-1}; \mathbf{w})$$

- We consider the following approximation:

$$p(\mathbf{x}; \mathbf{w}) \approx \prod_i p(x_i | x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n; \mathbf{w}) = \prod_i p(x_i | x_{-i}; \mathbf{w})$$

where we have added conditioning over additional variables

Pseudo-likelihood

- The pseudo-likelihood method replaces the likelihood,

$$\ell(\theta; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \log p(\mathcal{D}; \theta) = \frac{1}{|\mathcal{D}|} \sum_{m=1}^{|\mathcal{D}|} \log p(\mathbf{x}^m; \theta)$$

  with the following approximation:

$$\ell_{PL}(\mathbf{w}; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{m=1}^{|\mathcal{D}|} \sum_{i=1}^{n} \log p(x_i^m \mid x_{N(i)}^m; \mathbf{w})$$

  (we replaced x_{-i} with $x_{N(i)}$, i's Markov blanket)

- For example, suppose we have a pairwise MRF. Then,

$$p(x_i^m \mid x_{N(i)}^m; \mathbf{w}) = \frac{1}{Z(x_{N(i)}^m; \mathbf{w})} e^{\sum_{j \in N(i)} \theta_{ij}(x_i^m, x_j^m)}, \;\; Z(x_{N(i)}^m; \mathbf{w}) = \sum_{\hat{x}_i} e^{\sum_{j \in N(i)} \theta_{ij}(\hat{x}_i, x_j^m)}$$

- More generally, and using the log-linear parameterization, we have:

$$\log p(x_i^m \mid x_{N(i)}^m; \mathbf{w}) = \mathbf{w} \cdot \sum_{c : i \in c} f_c(x_c^m) - \log Z(x_{N(i)}^m; \mathbf{w})$$

Pseudo-likelihood

- This objective only involves summation over x_i and is tractable
- Has many small partition functions (one for each variable and each setting of its neighbors) instead of one big one
- It is still concave in $\mathbf{w}$ and thus has no local maxima
- Assuming the data is drawn from a MRF with parameters $\mathbf{w}^*$, can show that as the number of data points gets large, $\mathbf{w}^{PL} \rightarrow \mathbf{w}^*$