Inference and Representation

David Sontag

New York University

Lecture 3, Sept. 15, 2014

How to acquire a model?

- Possible things to do:
  - Use expert knowledge to determine the graph and the potentials.
  - Use learning to determine the potentials, i.e., **parameter learning**.
  - Use learning to determine the graph, i.e., **structure learning**.
- Manual design is difficult to do and can take a long time for an expert.
- We usually have access to a set of examples from the distribution we wish to model, e.g., a set of images segmented by a labeler.

More rigorous definition

- Lets assume that the domain is governed by some underlying distribution p^*, which is induced by some network model $\mathcal{M}^* = (\mathcal{G}^*, \theta^*)$
- We are given a dataset $\mathcal{D}$ of M samples from p^*
- The standard assumption is that the data instances are **independent and identically distributed (IID)**
- We are also given a family of models $\mathcal{M}$, and our task is to learn some model $\hat{\mathcal{M}} \in \mathcal{M}$ (i.e., in this family) that defines a distribution $p_{\hat{\mathcal{M}}}$
- We can learn model parameters for a fixed structure, or both the structure and model parameters

Goal of learning

- The goal of learning is to return a model $\hat{\mathcal{M}}$ that precisely captures the distribution p^* from which our data was sampled
- This is in general not achievable because of
  - computational reasons
  - limited data only provides a rough approximation of the true underlying distribution
- We need to select $\hat{\mathcal{M}}$ to construct the "best" approximation to $\mathcal{M}^*$
- What is "best"?

What is "best"?

This depends on what we want to do

1. Density estimation: we are interested in the full distribution (so later we can compute whatever conditional probabilities we want)
2. Specific prediction tasks: we are using the distribution to make a prediction
3. Structure or knowledge discovery: we are interested in the model itself (often of interest in data science)

1) Learning as density estimation

- We want to learn the full distribution so that later we can answer *any* probabilistic inference query
- In this setting we can view the learning problem as **density estimation**
- We want to construct $\hat{\mathcal{M}}$ as "close" as possible to p^*
- How do we evaluate "closeness"?
- **KL-divergence** (in particular, the M-projection) is one possibility:

$$\mathbf{D}(p^* || p_\theta) = \mathbf{E}_{\mathbf{x} \sim p^*} \left[\log \left(\frac{p^*(\mathbf{x})}{p_\theta(\mathbf{x})} \right) \right]$$

Expected log-likelihood

- We can simplify this somewhat:

$$\mathbf{D}(p^* || p_\theta) = \mathbf{E}_{\mathbf{x} \sim p^*} \left[\log \left(\frac{p^*(\mathbf{x})}{p_\theta(\mathbf{x})} \right) \right] = -\mathbf{H}(p^*) - \mathbf{E}_{\mathbf{x} \sim p^*} \left[\log p_\theta(\mathbf{x}) \right]$$

- The first term does not depend on θ.
- Then, finding the *minimal* M-projection is equivalent to *maximizing* the **expected log-likelihood**

$$\mathbf{E}_{\mathbf{x} \sim p^*} \left[\log p_\theta(\mathbf{x}) \right]$$

  - Asks that p_θ assign high probability to instances sampled from p^*, so as to reflect the true distribution
  - Because of log, samples $\mathbf{x}$ where $p_\theta(\mathbf{x}) \approx 0$ weigh heavily in objective
- Although we can now compare models, since we are not computing $\mathbf{H}(p^*)$, we don't know how close we are to the optimum
- Problem: In general we do not know p^*.

Maximum likelihood

- Approximate the expected log-likelihood

$$\mathbf{E}_{\mathbf{x}\sim p^*}\left[\log p_\theta(\mathbf{x})\right]$$

with the *empirical log-likelihood*:

$$\mathbf{E}_{\mathcal{D}}\left[\log p_\theta(\mathbf{x})\right] = \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \log p_\theta(\mathbf{x})$$

- Maximum likelihood learning is then:

$$\max_\theta \ \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \log p_\theta(\mathbf{x})$$

2) Likelihood, Loss and Risk

- We now generalize this by introducing the concept of a **loss function**
- A **loss function** $loss(\mathbf{x}, \mathcal{M})$ measures the loss that a model $\mathcal{M}$ makes on a particular instance $\mathbf{x}$
- Assuming instances are sampled from some distribution p^*, our goal is to find the model that minimizes the **expected loss** or **risk**,

$$\mathbf{E}_{\mathbf{x}\sim p^*}\left[loss(\mathbf{x}, \mathcal{M})\right]$$

- What is the loss function which corresponds to density estimation? Log-loss,

$$loss(\mathbf{x}, \hat{\mathcal{M}}) = -\log p_\theta(\mathbf{x}) = \log \frac{1}{p_\theta(\mathbf{x})}.$$

- p^* is unknown, but we can approximate the expectation using the empirical average, i.e., **empirical risk**

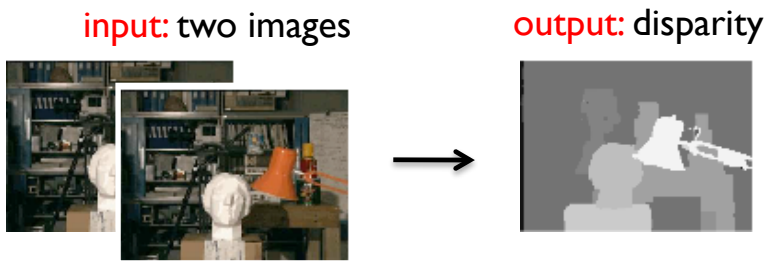
$$\mathbf{E}_{\mathcal{D}}\left[loss(\mathbf{x}, \hat{\mathcal{M}})\right] = \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} loss(\mathbf{x}, \hat{\mathcal{M}})$$

Example: conditional log-likelihood

- Suppose we want to predict a set of variables $\mathbf{Y}$ given some others $\mathbf{X}$, e.g., for segmentation or stereo vision
- We concentrate on predicting $p(\mathbf{Y}|\mathbf{X})$, and use a **conditional** loss function

$$loss(\mathbf{x}, \mathbf{y}, \hat{\mathcal{M}}) = -\log p_\theta(\mathbf{y} \mid \mathbf{x}).$$

- Since the loss function only depends on $p_\theta(\mathbf{y} \mid \mathbf{x})$, suffices to estimate the conditional distribution, not the joint
- This is the objective function we use to train conditional random fields (CRFs), which we discussed in Lecture 2



How to avoid overfitting?

- Hard constraints, e.g. by selecting a less expressive hypothesis class:
  - Bayesian networks with at most d parents
  - Pairwise MRFs (instead of arbitrary higher-order potentials)
- Soft preference for simpler models: **Occam's Razor**.
- Augment the learning objective function with **regularization**:

$$\mathit{objective}(\mathbf{x}, \mathcal{M}) = \mathit{loss}(\mathbf{x}, \mathcal{M}) + R(\mathcal{M})$$

  (often equivalent to MAP estimation where we put a prior over parameters θ and maximize $\log p(\theta \mid \mathbf{x}) = \log p(\mathbf{x}; \theta) + \log p(\theta) - \text{constant}$)

- Can evaluate generalization performance using cross-validation

Summary of how to think about learning

1. Figure out what you care about, e.g. expected loss

$$\mathbf{E}_{\mathbf{x}\sim P^*}\left[loss(\mathbf{x}, \hat{\mathcal{M}})\right]$$

2. Figure out how best to estimate this from what you have, e.g. regularized empirical loss

$$\mathbf{E}_{\mathcal{D}}\left[loss(\mathbf{x}, \hat{\mathcal{M}})\right] + R(\hat{\mathcal{M}})$$

   When used with log-loss, the regularization term can be interpreted as a prior distribution over models, $p(\hat{\mathcal{M}}) \propto \exp(-R(\hat{\mathcal{M}}))$

   (called *maximum a posteriori (MAP) estimation*)

3. Figure out how to optimize over this objective function, e.g. the minimization

$$\min_{\hat{\mathcal{M}}} \ \mathbf{E}_{\mathcal{D}}\left[loss(\mathbf{x}, \hat{\mathcal{M}})\right] + R(\hat{\mathcal{M}})$$

ML estimation in Bayesian networks

- Suppose that we know the Bayesian network structure G
- Let $\theta_{x_i|\mathbf{x}_{pa(i)}}$ be the parameter giving the value of the CPD $p(x_i \mid \mathbf{x}_{pa(i)}; \theta)$
- Maximum likelihood estimation corresponds to solving:

$$\max_\theta \sum_{n=1}^{N} \log p(\mathbf{x}^n; \theta) = \max_\theta \ell(\theta; \mathcal{D})$$

subject to the non-negativity and normalization constraints

- This is equal to:

$$\begin{aligned} \max_\theta \sum_{n=1}^{N} \log p(\mathbf{x}^n; \theta) &= \max_\theta \sum_{n=1}^{N} \sum_{i=1}^{|V|} \log p(x_i^n \mid \mathbf{x}_{pa(i)}^n; \theta) \\ &= \max_\theta \sum_{i=1}^{|V|} \sum_{n=1}^{N} \log p(x_i^n \mid \mathbf{x}_{pa(i)}^n; \theta) \end{aligned}$$

- The optimization problem decomposes into an independent optimization problem for each CPD!

ML estimation in Bayesian networks

$$\begin{aligned}
\ell(\theta; \mathcal{D}) = \log p(\mathcal{D}; \theta) &= \sum_{i=1}^{|V|} \sum_{n=1}^{N} \log p(x_i^n \mid \mathbf{x}_{pa(i)}^n; \theta) \\
&= \sum_{i=1}^{|V|} \sum_{\mathbf{x}_{pa(i)}} \sum_{\mathbf{x}_i} \sum_{\substack{\hat{\mathbf{x}} \in \mathcal{D}: \\ \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_{pa(i)} = \mathbf{x}_i, \mathbf{x}_{pa(i)}}} \log p(x_i \mid \mathbf{x}_{pa(i)}; \theta) \\
&= \sum_{i=1}^{|V|} \sum_{\mathbf{x}_{pa(i)}} \sum_{\mathbf{x}_i} N_{x_i, \mathbf{x}_{pa(i)}} \log \theta_{x_i | \mathbf{x}_{pa(i)}},
\end{aligned}$$

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

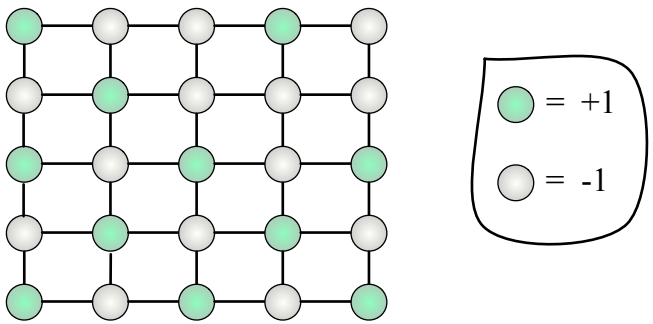
- We have the closed form ML solution:

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

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

ML estimation in Markov networks

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



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

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

3) Knowledge Discovery

- We hope that looking at the learned model we can discover something about p^*, e.g.
  - Nature of the dependencies, e.g., positive or negative correlation
  - What are the direct and indirect dependencies
- Simple statistical models (e.g., looking at correlations) can be used for the first
- But the learned network gives us much more information, e.g. conditional independencies, causal relationships
- In this setting we care about discovering the correct model $\mathcal{M}^*$, rather than a different model $\hat{\mathcal{M}}$ that induces a distribution similar to $\mathcal{M}^*$.
- Metric is in terms of the differences between $\mathcal{M}^*$ and $\hat{\mathcal{M}}$.

This is not always achievable

- The true model might not be identifiable
  - e.g., Bayesian network with several I-equivalent structures.
  - In this case the best we can hope is to discover an I-equivalent structure.
  - Problem is worse when the amount of data is limited and the relationships are weak.

Structure learning using maximum likelihood

- Recall that for Bayesian networks we have decomposability of the likelihood:

$$\log p(\mathcal{D};\theta) = \sum_{i=1}^{|V|} \sum_{\mathbf{x}_{pa(i)}} \sum_{\mathbf{x}_i} N_{x_i,\mathbf{x}_{pa(i)}} \log p(x_i \mid \mathbf{x}_{pa(i)};\theta)$$

- Given a candidate structure $G = (V, E)$, the maximum likelihood parameters are given by: $\theta^{ML}_{x_i|\mathbf{x}_{pa(i)}} = \frac{N_{x_i,\mathbf{x}_{pa(i)}}}{\sum_{\hat{x}_i} N_{\hat{x}_i,\mathbf{x}_{pa(i)}}} = \hat{p}(x_i \mid \mathbf{x}_{pa(i)})$
- Putting these together, maximum likelihood structure learning reduces to:

$$\max_G \sum_{i=1}^{|V|} \text{score}(i \mid pa_i, \mathcal{D}), \qquad \text{where}$$

$$\begin{aligned}
\text{score}(i \mid pa_i, \mathcal{D}) &= \sum_{\mathbf{x}_{pa(i)}} \sum_{\mathbf{x}_i} N_{x_i,\mathbf{x}_{pa(i)}} \log p(x_i \mid \mathbf{x}_{pa(i)}; \theta^{ML}_{x_i|\mathbf{x}_{pa(i)}}) \\
&= N \sum_{\mathbf{x}_{pa(i)}} \frac{N_{\mathbf{x}_{pa(i)}}}{N} \sum_{\mathbf{x}_i} \frac{N_{x_i,\mathbf{x}_{pa(i)}}}{N_{\mathbf{x}_{pa(i)}}} \log \hat{p}(x_i \mid \mathbf{x}_{pa(i)})
\end{aligned}$$

Structure learning using maximum likelihood

- Simplifying further, we get:

$$\begin{aligned}
\text{score}(i \mid pa_i, \mathcal{D}) &= N \sum_{\mathbf{x}_{pa(i)}} \frac{N_{\mathbf{x}_{pa(i)}}}{N} \sum_{\mathbf{x}_i} \frac{N_{x_i, \mathbf{x}_{pa(i)}}}{N_{\mathbf{x}_{pa(i)}}} \log \hat{p}(x_i \mid \mathbf{x}_{pa(i)}) \\
&= N \sum_{\mathbf{x}_{pa(i)}} \hat{p}(\mathbf{x}_{pa(i)}) \sum_{\mathbf{x}_i} \hat{p}(x_i \mid \mathbf{x}_{pa(i)}) \log \hat{p}(x_i \mid \mathbf{x}_{pa(i)}) \\
&= -N \sum_{\mathbf{x}_{pa(i)}} \hat{p}(\mathbf{x}_{pa(i)}) \sum_{\mathbf{x}_i} \hat{p}(x_i \mid \mathbf{x}_{pa(i)}) \log \frac{1}{\hat{p}(x_i \mid \mathbf{x}_{pa(i)})} \\
&= -N \cdot \hat{H}(X_i \mid X_{pa(i)}).
\end{aligned}$$

- We see that the maximum likelihood structure problem is equivalent to

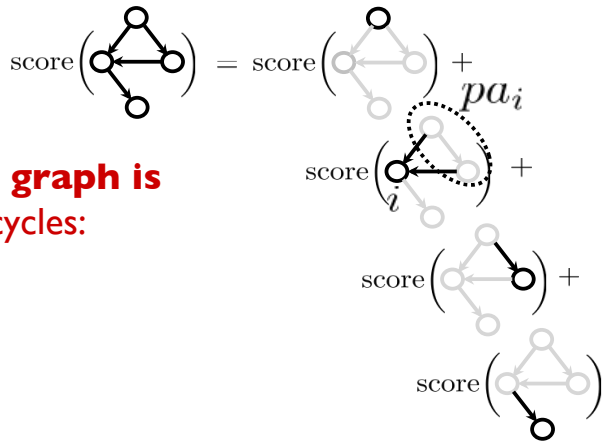
$$\min_G \sum_{i=1}^{N} \hat{H}(X_i \mid X_{pa(i)}),$$

  i.e. choose a graph structure which minimizes the entropy of each individual variable.

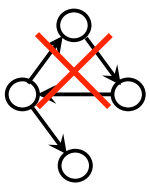
Structure learning: score-based approaches

- Q: What is the maximum likelihood graph?
  - A: The complete graph! Because $H(X \mid Y) \leq H(X)$ **always**.
  - Must *regularize* to recover a sparse graph and have any hope of recovering true structure (called *consistency*)
  - Common approaches such as BIC and BDe (Bayesian Dirichlet score) are also decomposable
- Obtain a combinatorial optimization problem over acyclic graphs:

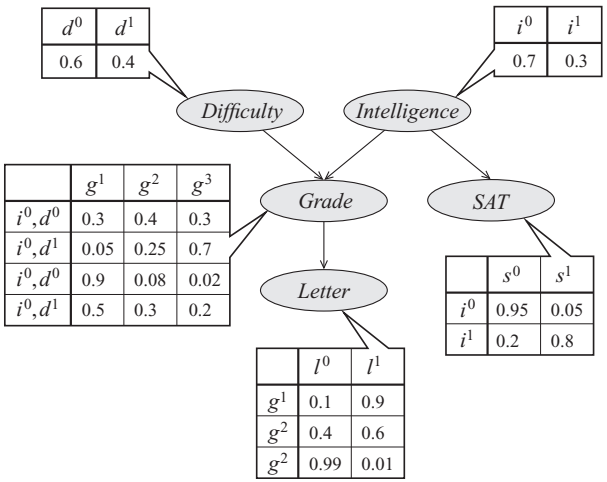
$$\text{score}(G; D) = \sum_{i=1}^{n} \text{score}(i | pa_i, D)$$



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



Independence tests



The network structure implies several conditional independence statements:

$$D \perp I$$
$$G \perp S \mid I$$
$$D \perp L \mid G$$
$$L \perp S \mid G$$
$$L \perp S \mid I$$
$$D \perp S$$

If two variables are (conditionally) independent, structure has no edge between them

- Must make assumption that data is drawn from an I-map of the graph
- Possible to learn structure with polynomial number of data points and polynomial computation time (e.g., the SGS algorithm from Spirtes, Glymour, & Scheines '01)
- Very brittle: if we say that $X_i \perp X_j | X_{\mathbf{v}}$ and they in fact are not, the resulting structure can be very off