Probabilistic Graphical Models

David Sontag

New York University

Lecture 10, April 3, 2012

Summary so far

- Representation of directed and undirected networks
- Inference in these networks:
  - Variable elimination
  - Exact inference in trees via message passing
  - MAP inference via dual decomposition
  - Marginal inference via variational methods
  - Marginal inference via Monte Carlo methods
- The rest of this course:
  - Learning Bayesian networks (today)
  - Learning Markov random fields
  - Structured prediction
  - Decision-making under uncertainty
  - Advanced topics (if time)
- Today we will refresh your memory about what learning is

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.
- We call this task of constructing a model from a set of instances **model learning**.

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
- We might be interested in returning a single model, a set of hypothesis that are likely, a probability distribution over models, or even a confidence of the model we return

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

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:

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

Expected log-likelihood

- We can simplify this somewhat:

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

- The first term does not depend on $\hat{p}$.
- Then, finding the *minimal* M-projection is equivalent to *maximizing* the **expected log-likelihood**

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

  - Asks that $\hat{p}$ assign high probability to instances sampled from p^*, so as to reflect the true distribution
  - Because of log, samples $\mathbf{x}$ where $\hat{p}(\mathbf{x}) \approx 0$ 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

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

with the *empirical log-likelihood*:

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

- Maximum likelihood learning is then:

$$\max_{\hat{\mathcal{M}}} \;\; \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \log \hat{p}(\mathbf{x})$$

2) Likelihood, Loss and Risk

- We now generalize this by introducing the concept of a **loss function**
- A **loss function** $\mathit{loss}(\mathbf{x}, \mathcal{M})$ 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**,

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

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

$$\mathit{loss}(\mathbf{x}, \hat{\mathcal{M}}) = -\log \hat{p}(\mathbf{x}).$$

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

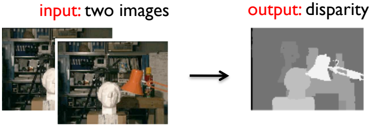
$$\mathbf{E}_{\mathcal{D}}\left[\mathit{loss}(\mathbf{x}, \hat{\mathcal{M}})\right] = \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \mathit{loss}(\mathbf{x}, \hat{\mathcal{M}})$$

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

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

- Since the loss function only depends on $\hat{p}(\mathbf{y} \mid \mathbf{x})$, 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 4



Example: structured prediction

- In **structured prediction**, given $\mathbf{x}$ we predict $\mathbf{y}$ by:

$$\operatorname*{argmax}_{\mathbf{y}} \hat{p}(\mathbf{y}|\mathbf{x})$$

- What loss function should we use to measure error in this setting?
- One reasonable choice would be the **classification error**:

$$\mathbf{E}_{(\mathbf{x},\mathbf{y})\sim p^*}\left[\mathbb{1}\{\ \exists \mathbf{y}' \neq \mathbf{y} \text{ s.t. } \hat{p}(\mathbf{y}'|\mathbf{x}) \geq \hat{p}(\mathbf{y}|\mathbf{x})\ \}\right]$$

  which is the probability over all $(\mathbf{x}, \mathbf{y})$ pairs sampled from p^* that our classifier selects the right labels

- We will go into much more detail on this in two lectures

Consistency

- To summarize, our learning **goal** is to choose a model $\hat{\mathcal{M}}$ that minimizes the risk (expected loss)

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

- We don't know p^*, so we instead minimize the empirical risk

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

- For many reasonable loss functions (including log-loss), one can show the following **consistency** property: as $|\mathcal{D}| \to \infty$,

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

- In particular, if $\mathcal{M}^* \in \mathcal{M}$, then given a sufficiently large training set, we will find it by minimizing the empirical risk

Empirical Risk and Overfitting

- Empirical risk minimization can easily **overfit** the data
- For example, consider the case of N random binary variables, and M number of training examples, e.g., $N = 100, M = 1000$
- Thus, we typically restrict the **hypothesis space** of distributions that we search over

Bias-Variance trade off

- If the hypothesis space is very limited, it might not be able to represent p^*, even with unlimited data
- This type of limitation is called **bias**, as the learning is limited on how close it can approximate the target distribution
- If we select a highly expressive hypothesis class, we might represent better the data
- When we have small amount of data, multiple models can fit well, or even better than the true model
- Moreover, small perturbations on $\mathcal{D}$ will result in very different estimates
- This limitation is call the **variance**.
- There is an inherent **bias-variance trade off** when selecting the hypothesis class
- Error in learning due to both things: bias and variance.

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 Razor**.
- Augment the objective function with **regularization**:

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

- Can evaluate generalization performance using cross-validation

Learning theory

- We hope that a model that achieves low training loss also achieves low expected loss (risk).
- We cannot guarantee with certainty the quality of our learned model.
- This is because the data is sample stochastically from P^*, and it might be unlucky sample.
- The goal is to prove that the model is approximately correct: for most $\mathcal{D}$, the learning procedure returns a model whose error is low
- This question – the study of generalization – is at the core of learning theory

Summary of how to think about learning

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

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

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

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

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

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

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 $p(x_i \mid \mathbf{x}_{pa(i)})$
- Maximum likelihood estimation corresponds to solving:

$$\max_\theta \frac{1}{M}\sum_{m=1}^{M} \log p(\mathbf{x}^M;\theta)$$

  subject to the non-negativity and normalization constraints

- This is equal to:

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

- The optimization problem decomposes into an independent optimization problem for each CPD! Has a simple closed-form solution.

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.
- When the number of variables is large relative to the amount of training data, pairs of variables can appear strongly correlated just by chance

Structure learning in Bayesian networks

- Score-based approaches
  - Given G, assume prior distribution for CPD parameters $\theta_{x_i|x_{pa(i)}}$ is Dirichlet (this is called the *Bayesian score*)
  - Choose G which maximizes the posterior

$$p(G \mid \mathcal{D}) \propto p(\mathcal{D} \mid G)p(G)$$

  - To compute the first term (called the *marginal likelihood*), use the chain rule together with your solution to problem 5 of PS 2
  - Obtain a combinatorial optimization problem over acyclic graphs – extremely difficult to solve optimally
- Hypothesis testing based on conditional independence
  - 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
  - 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

Bayesian prediction

- Rather than choose 1 graph structure, learn the full posterior

$$p(G \mid \mathcal{D})$$

- Then, compute expectations with respect to this, e.g.

$$p(x_1 = 1 \mid \mathcal{D}) = \sum_G p(G \mid \mathcal{D}) p(x_1 = 1 \mid G, \mathcal{D})$$

- This inference task is very difficult to approximate – typically done using MCMC, but very slow