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
The goal of learning is to return a model $\hat{M}$ 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 $\hat{M}$ to construct the "best" approximation to $M^*$

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 $\hat{M}$ as "close" as possible to $p^*$.
- How do we evaluate "closeness"?
- **KL-divergence** (in particular, the M-projection) is one possibility:

$$D(p^* \| p_\theta) = E_{x \sim p^*} \left[ \log \left( \frac{p^*(x)}{p_\theta(x)} \right) \right]$$
Expected log-likelihood

- We can simplify this somewhat:

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

- The first term does not depend on \( \theta \).

- Then, finding the minimal M-projection is equivalent to maximizing the expected log-likelihood

\[
\mathbb{E}_{x \sim p^*} [\log p_\theta(x)]
\]

- Asks that \( p_\theta \) assign high probability to instances sampled from \( p^* \), so as to reflect the true distribution

- Because of log, samples \( x \) where \( p_\theta(x) \approx 0 \) weigh heavily in objective

- Although we can now compare models, since we are not computing \( H(p^*) \), 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

\[ E_{x \sim p^*} \left[ \log p_\theta(x) \right] \]

with the *empirical log-likelihood*:

\[ E_D \left[ \log p_\theta(x) \right] = \frac{1}{|D|} \sum_{x \in D} \log p_\theta(x) \]

- Maximum likelihood learning is then:

\[ \max_\theta \frac{1}{|D|} \sum_{x \in D} \log p_\theta(x) \]
2) Likelihood, Loss and Risk

- We now generalize this by introducing the concept of a **loss function**

- A **loss function** $\text{loss}(x, M)$ measures the loss that a model $M$ makes on a particular instance $x$

- Assuming instances are sampled from some distribution $p^*$, our goal is to find the model that minimizes the **expected loss** or **risk**,

$$E_{x \sim p^*} [\text{loss}(x, M)]$$

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

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

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

$$E_{\mathcal{D}} \left[ \text{loss}(x, \hat{M}) \right] = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \text{loss}(x, \hat{M})$$
Example: conditional log-likelihood

- Suppose we want to predict a set of variables $Y$ given some others $X$, e.g., for segmentation or stereo vision.

- We concentrate on predicting $p(Y|X)$, and use a **conditional** loss function

$$\text{loss}(x, y, \hat{M}) = -\log p_\theta(y | x).$$

- Since the loss function only depends on $p_\theta(y | x)$, 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:

  \[ \text{objective}(x, M) = \text{loss}(x, M) + R(M) \]

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

\[ E_{x \sim P^*} \left[ \text{loss}(x, \hat{M}) \right] \]

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

\[ E_{\mathcal{D}} \left[ \text{loss}(x, \hat{M}) \right] + R(\hat{M}) \]

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

(called maximum a posteriori (MAP) estimation)

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

\[ \min_{\hat{M}} E_{\mathcal{D}} \left[ \text{loss}(x, \hat{M}) \right] + R(\hat{M}) \]
ML estimation in Bayesian networks

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

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

subject to the non-negativity and normalization constraints

- This is equal to:

$$\max_{\theta} \sum_{n=1}^{N} \log p(x^n; \theta) = \max_{\theta} \sum_{n=1}^{N} \sum_{i=1}^{\mid V \mid} \log p(x^n_i \mid x^n_{pa(i)}; \theta)$$

$$= \max_{\theta} \sum_{i=1}^{\mid V \mid} \sum_{n=1}^{N} \log p(x^n_i \mid x^n_{pa(i)}; \theta)$$

- The optimization problem decomposes into an independent optimization problem for each CPD!
ML estimation in Bayesian networks

\[ \ell(\theta; D) = \log p(D; \theta) = \sum_{i=1}^{N} \sum_{n=1}^{V} \log p(x_i^n | x_{pa(i)}^n; \theta) \]

\[ = \sum_{i=1}^{V} \sum_{x_{pa(i)}} \sum_{x_i} \sum_{\hat{x} \in D: \hat{x}_{i,\hat{x}_{pa(i)}} = x_i, x_{pa(i)}} \log p(x_i | x_{pa(i)}; \theta) \]

\[ = \sum_{i=1}^{V} \sum_{x_{pa(i)}} \sum_{x_i} N_{x_i,x_{pa(i)}} \log \theta_{x_i|x_{pa(i)}} \]

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

- We have the closed form ML solution:
  \[ \theta_{x_i|x_{pa(i)}}^{ML} = \frac{N_{x_i,x_{pa(i)}}}{\sum_{\hat{x}_i} N_{\hat{x}_i,x_{pa(i)}}} \]

- 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?

\[ p(x_1, \cdots, x_n) = \frac{1}{Z} \exp \left( \sum_{i<j} w_{i,j} x_i x_j - \sum_i u_i x_i \right) \]
Bad news for Markov networks

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

$$\theta^{ML} = \arg \max_{\theta} \log \prod_{x \in D} p(x; \theta)$$

$$= \arg \max_{\theta} \sum_{x \in D} \left( \sum_c \log \phi_c(x_c; \theta) - \log Z(\theta) \right)$$

$$= \arg \max_{\theta} \left( \sum_{x \in D} \sum_c \log \phi_c(x_c; \theta) \right) - |D| \log Z(\theta)$$

- 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 \( M^* \), rather than a different model \( \hat{M} \) that induces a distribution similar to \( M^* \).

- Metric is in terms of the differences between \( M^* \) and \( \hat{M} \).
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.
Recall that for Bayesian networks we have decomposability of the likelihood:

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

Given a candidate structure $G = (V, E)$, the maximum likelihood parameters are given by:

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

Putting these together, maximum likelihood structure learning reduces to:

$$\max_G \sum_{i=1}^{\left| V \right|} \text{score}(i \mid \mathbf{pa}_i, D), \quad \text{where}$$

$$\text{score}(i \mid \mathbf{pa}_i, D) = \sum_{\mathbf{x}_{pa(i)}} \sum_{\mathbf{x}_i} N_{\mathbf{x}_i, \mathbf{x}_{pa(i)}} \log p(x_i | \mathbf{x}_{pa(i)}; \theta_{x_i|x_{pa(i)}}^{ML})$$

$$= N \sum_{\mathbf{x}_{pa(i)}} \frac{N_{\mathbf{x}_{pa(i)}}}{N} \sum_{\mathbf{x}_i} \frac{N_{\mathbf{x}_i, \mathbf{x}_{pa(i)}}}{N_{\mathbf{x}_{pa(i)}}} \log \hat{p}(x_i | \mathbf{x}_{pa(i)})$$
Simplifying further, we get:

\[
\text{score}(i \mid pa_i, \mathcal{D}) = N \sum_{x_{pa(i)}} \left( \frac{N_{x_{pa(i)}}}{N} \sum_{x_i} \frac{N_{x_i, x_{pa(i)}}}{N_{x_{pa(i)}}} \log \hat{p}(x_i \mid x_{pa(i)}) \right)
\]

\[
= N \sum_{x_{pa(i)}} \hat{p}(x_{pa(i)}) \sum_{x_i} \hat{p}(x_i \mid x_{pa(i)}) \log \hat{p}(x_i \mid x_{pa(i)})
\]

\[
= -N \sum_{x_{pa(i)}} \hat{p}(x_{pa(i)}) \sum_{x_i} \hat{p}(x_i \mid x_{pa(i)}) \log \frac{1}{\hat{p}(x_i \mid x_{pa(i)})}
\]

\[
= -N \cdot \hat{H}(X_i \mid X_{pa(i)}).
\]

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.
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:

$$\text{score}(G; D) = \sum_{i=1}^{n} \text{score}(i \mid \text{pa}_i, D)$$

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 \( X_i \perp X_j \mid X_v \) and they in fact are not, the resulting structure can be very off