Probabilistic Graphical Models

Lecture 13: Learning Graphical Models and Parameter Estimation

Matthew Walter

TTI-Chicago

May 26, 2020

Course Summary

- Graphical models as **representations** of probability distributions
  - Directed graphical models (i.e., Bayesian networks)
  - Undirected graphical models (i.e., Markov random fields, conditional random fields, and factor graphs)
- Algorithms for performing **inference** in these networks
  - Variable elimination (sum-product)
  - Exact inference in clique trees via message passing
  - Exact MAP inference (max-product)
  - Approximate inference via variational methods
  - Approximate inference via sampling
- Techniques for **learning** graphical model parameters and structure
  - Learning Bayesian networks
  - Learning Markov random fields
  - Learning with incomplete data

How Can We Acquire a Model?

- We have several options:
  - Rely on expert knowledge to determine graph structure and parameters
  - Use learning to determine the potentials, i.e., **parameter learning**
  - Use learning to determine the graph structure, i.e., **structure learning**
- Manual design is difficult and can require a lot of time
- Often, we have access to a set of examples (samples) from the distribution that we want to model

How Can We Acquire a Model?

- Assume that the domain is governed by an underlying distribution P^* that is induced by some network model $\mathcal{M}^* = (G^*, \theta^*)$
- We are given a dataset $\mathcal{D}$ of M samples from P^*
- We assume that the samples are **independent and identically distributed (IID)**
- We have a family of models $\{\mathcal{M}\}$ (given or specified by us), and our task is to learn some model $\hat{\mathcal{M}} \in \{\mathcal{M}\}$ that defines a distribution $P_{\hat{\mathcal{M}}}$
- We can learn model parameters for a fixed structure, or both the structure and parameters

Goals 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
- In general, this is not achievable due to
  - Access to a small amount of data relative to the number of random variables, providing a sparse sampling of the true distribution
  - Computational reasons
- Consequently, we need to select $\hat{\mathcal{M}}$ that constructs the "best" approximation to $\mathcal{M}^*$
- Raises the question: How do we define "best"?

How Do We Define "Best"?

- The measure of "best" depends on the specific goals
  1. Density estimation: We want an estimate of the distribution $\hat{P}$ that is as close as possible to P^* (e.g., for general inference)
  2. Specific prediction task: We will use the model to make a particular prediction (e.g., classification, segmentation, or depth estimation)
  3. Knowledge discovery: We are interested in the structure of the model

Density Estimation

- We want to learn the full distribution, so that we can answer *any* probabilistic query
- The learning problem as one of **density estimation**
- We want to construct $\hat{\mathcal{M}}$ such that $\hat{P}$ is as "close" as possible to P^*
- How do we evaluate "closeness"?
- **KL-divergence** (specifically, the M-projection) is one possibility

$$D(P^* \| \hat{P}) = \mathbb{E}_{P^*}\left[\log \frac{P^*(\mathcal{X})}{\hat{P}(\mathcal{X})}\right]$$

(recall that it measures the loss when representing P^* with $\hat{P}$)

Density Estimation: Expected Log-Likelihood

- We can simplify the expression as

$$D(P^*\|\hat{P}) = \mathbb{E}_{P^*}\left[\log \frac{P^*(\mathcal{X})}{\hat{P}(\mathcal{X})}\right] = -H_{P^*}(\mathcal{X}) - \mathbb{E}_{P^*}[\log \hat{P}(\mathcal{X})]$$

- The first term on the RHS does not depend on $\hat{P}$ (and is often unknown since we don't know P^*)
- Thus, finding the *minimal* M-projection is equivalent to *maximizing* the **expected log-likelihood**

$$\mathbb{E}_{P^*}[\log \hat{P}(\mathcal{X})]$$

  - Encourages $\hat{P}$ to assign high probability to instances likely under P^*
  - Because of log scale, $\hat{P}$ will be reasonably large where P^* is not small
- Without $H_{P^*}(\mathcal{X})$, we can compare different models, but we don't know how close they are from the optimum

Density Estimation: Maximum Likelihood

- When P^* is unknown, we approximate the expected log-likelihood

$$\mathbb{E}_{P^*}[\log \hat{P}(\mathcal{X})]$$

  with the *empirical* log-likelihood

$$\mathbb{E}_{\mathcal{D}}[\log \hat{P}(\mathcal{X})] = \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{\xi} \in \mathcal{D}} \log \hat{P}(\boldsymbol{\xi})$$

  for some set of samples $\mathcal{D} = \{\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(2)}, \ldots, \boldsymbol{\xi}^{(M)}\}$
- Gives rise to maximum likelihood learning:

$$\max_{\hat{\mathcal{M}}} \; \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{\xi} \in \mathcal{D}} \log \hat{P}(\boldsymbol{\xi})$$

Likelihood, Loss, and Risk

- A **loss function** $\text{loss}(\boldsymbol{x}, \mathcal{M})$ measures the loss that a model $\mathcal{M}$ makes on a particular instance $\boldsymbol{x}$
- When instances are sampled from P^*, our objective is to minimize the **expected loss** (aka the **expected risk**)

$$\mathbb{E}_{P^*}[\text{loss}(\boldsymbol{x}, \mathcal{M})]$$

- In the case of density estimation, we are interested in log-loss

$$\text{loss}(\boldsymbol{x}, \hat{\mathcal{M}}) = -\log \hat{P}(\boldsymbol{x})$$

- Since P^* is unknown, we approximate the expectation using the empirical average, i.e., the **empirical risk**

$$\mathbb{E}_{\mathcal{D}}[\text{loss}(\mathcal{X}, \mathcal{M})] = \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{\xi} \in \mathcal{D}} \text{loss}(\boldsymbol{\xi}, \hat{\mathcal{M}})$$

Prediction Tasks

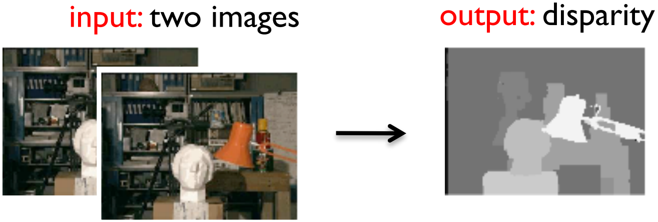
- Density estimation is useful for performing probabilistic queries over an arbitrary subset of the random variables
- However, we are often interested in performing prediction over a subset of variables $\boldsymbol{Y}$ given some others $\boldsymbol{X}$ (e.g., classification tasks, such as image segmentation or document classification)
- Rather than learn the entire joint distribution, we can focus specifically on the conditional distribution $P(\boldsymbol{Y} \mid \boldsymbol{X})$ (recall CRFs)
- We can then consider several different loss functions
  - Classification error (aka $0/1$ loss)
  - Hamming loss (fraction of wrong predictions)
  - Conditional log-loss: $\mathbb{E}_{P^*}\left[\log \hat{P}(\boldsymbol{Y} \mid \boldsymbol{X})\right]$ (analogous to log-loss)

Prediction Tasks: Conditional Log-Likelihood

- We concentrate on predicting the conditional distribution $P(\boldsymbol{Y} \mid \boldsymbol{X})$
- Conditional log-loss measures how well the learned model predicts $\boldsymbol{Y}$ given $\boldsymbol{X}$ in the data, but not the distribution over $\boldsymbol{X}$

$$\text{loss}(\boldsymbol{x}, \boldsymbol{y}, \hat{\mathcal{M}}) = -\log \hat{P}(\boldsymbol{y} \mid \boldsymbol{x})$$

- This is the objective used to train conditional random fields (CRFs), such as the ones that we saw earlier



Prediction Tasks: Structured Prediction

- In **structured prediction**, we predict $\boldsymbol{y}$ given $\boldsymbol{x}$ as

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} P(\boldsymbol{y} \,|\, \boldsymbol{x})$$

- One reasonable choice for the loss function is the **classification error**

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

  (i.e., probability over all $(\boldsymbol{x}, \boldsymbol{y})$ that we predict the wrong assignment)

Empirical Risk and Overfitting

- Given data $\mathcal{D}$ sampled from P^*, define the *empirical distrbution* as
$$\bar{P}_{\mathcal{D}}(A) = \frac{1}{M} \sum_m \mathbb{1}(\xi^{(m)} \in A)$$
- If $\mathcal{D} = \{\xi^{(1)}, \xi^{(2)}, \ldots, \xi^{(M)}\}$ is a sequence of IID samples $\xi^{(m)} \sim P^*$,
$$\lim_{M \to \infty} \bar{P}_{\mathcal{D}}(A) = P^*(A)$$
- $\bar{P}_{\mathcal{D}}$ is the distribution that maximizes the log-likelihood of the data
- Optimizing empirical risk can be sensitive to **overfitting** to the data
- Consider a domain with 100 binary variables (2^{100} assignments):
  - Suppose that $\mathcal{D}$ contains 1000 instances (likely distinct)
  - We will estimate $\hat{P}$ that assigns 0.001 to each of the 1000 assignments, and 0 to the $2^{100} - 1000$ others
- Alternatively, consider a Bayesian network with some random variables having a large number k of parents
  - Number of parameters for each CPD is exponential in k
  - It is unlikely that $\mathcal{D}$ will span all instantiations of parents
- In order to improve generalization, we restrict the **hypothesis space**

Bias-Variance Trade-Off

- If the hypothesis space is very limited, we may not be able to learn P^*, even with unlimited data
- This introduces a **bias** in how close learning can approximate the true distribution
- If we select a highly expressive hypothesis space, we can better represent the data
- When we have a small amount of data, multiple models may fit well, possibly even better than the true model
- Small perturbations of the data will result in very different estimates, i.e., high **variance**
- There is an inherent **bias-variance trade-off** when selecting hypothesis class
- Error in learning due to both bias and variance

How Can We Avoid Overfitting?

- Impose hard constraints, e.g., limiting hypothesis class
  - Bayesian networks with at most d parents per node
  - Pairwise MRFs (vs. higher-order potentials)
- Update objective to include soft preference for simpler models via **regularization**

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

- Can evaluate generalization performance via cross-validation

Learning Procedure

- We assume input of the form:
  1. Prior knowledge and/or constraints on the model class $\hat{\mathcal{M}}$
  2. A set $\mathcal{D} = \{\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(2)}, \ldots, \boldsymbol{\xi}^{(M)}\}$ of IID samples from P^*
- The output is a model $\hat{\mathcal{M}}$ that may include the structure and/or parameters of the graphical model
- The specifics of a particular learning algorithm vary with
  1. The type of output, i.e., a Bayesian network or Markov random field
  2. The constraints that we place on $\hat{\mathcal{M}}$
  3. The extent to which the training data is fully observed

Model Constraints

- Constraints on the model define the hypothesis space that specifies the class of admissible models
- These constraints may be over the structure, the parameters, or both
  1. We may be given the graph structure and tasked with learning some or all of the parameters
  2. We may not know the structure or parameters, and need to learn both
  3. We may not even know the complete set of variables over which the distribution P^* is defined
- As we discussed, there is a trade-off between over- and under-constraining the model space
  - If the hypothesis space is too restricted, it may not contain P^*
  - On the other hand, if the hypothesis space is too large, we may overfit, learning a model that assigns high likelihood to the data, but is a poor approximation to P^*

Data Observability

- Different learning problems exhibit different amounts of observability with respect to the data:
  1. Each datapoint $\boldsymbol{\xi}^{(i)}$ provides an instantiation to every random variable
  2. Each datapoint $\boldsymbol{\xi}^{(i)}$ provides an instantiation for a subset of the random variables, though every variable is observed at least once
  3. There are some *hidden* variables that are never observed in the training data (i.e., we don't know the full set of random variables)
- Partial observability is encountered in many practical problems (observing all random variables as part of each sample may be difficult or impossible)
- Hidden variables are often usefull
  - They help us better understand the nature of the problem (i.e., knowledge discovery)
  - They simplify the distribution and graphical model (e.g., consider a naive Bayes model)

Summary of Learning[1]

1. Decide on an objective and corresponding loss

$$\mathbb{E}_{P^*}[\text{loss}(\boldsymbol{x}, \mathcal{M})]$$

2. Determine how to best estimate this from what we have, e.g., regularized empirical loss

$$\mathbb{E}_{\mathcal{D}}[\text{loss}(\boldsymbol{x}, \mathcal{M})] + R(\mathcal{M})$$

   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. Determine how to optimize over this objective function

$$\min_{\mathcal{M}} \mathbb{E}_{\mathcal{D}}[\text{loss}(\boldsymbol{x}, \mathcal{M})] + R(\mathcal{M})$$

(We will start by assuming **complete** (**fully observable**) data)

[1]Obviously, learning deserves an entire course (Greg's ML class is excellent.)

Maximum Likelihood Parameter Estimation

- A function $P(\boldsymbol{X}; \boldsymbol{\theta})$ is a *parametric model* (parametric family) specified in terms of a set of parameters $\boldsymbol{\theta} \in \Theta$ (parameter space)
- If x is a multinomial taking on K values, the parameters $\boldsymbol{\theta} \in \mathbb{R}^K$ define the distribution as:

$$P(x = x^k; \boldsymbol{\theta}) = \theta_k \qquad \text{where} \qquad \Theta = \left\{ \boldsymbol{\theta} \in [0,1]^K : \sum_i \theta_i = 1 \right\}$$

Maximum Likelihood Parameter Estimation

- The *likelihood function* is the probability of the data

$$L(\boldsymbol{\theta} : \mathcal{D}) = \prod_m P(\boldsymbol{x}^{(m)}; \boldsymbol{\theta})$$

- Consider M IID tosses of a possibly biased coin $\mathcal{D} = \{x^{(1)}, x^{(2)}, \dots, x^{(M)}\}$, where $x^{(i)} \in \{H, T\}$
- The likelihood of the data as a function of the parameter vector $\boldsymbol{\theta} = [\theta, \theta - 1]$, where $\theta = P(H)$

$$L(\boldsymbol{\theta} : \mathcal{D}) = \theta^{\#[H]}(1 - \theta)^{\#[T]}$$

where $\#[H]$ and $\#[T]$ denote the number of heads and tails in the training data, respectively

Maximum Likelihood Parameter Estimation

$$L(\boldsymbol{\theta} : \mathcal{D}) = \theta^{\#[H]}(1-\theta)^{\#[T]}$$

- In practice, it is convenient to maximize the log-likelihood

$$\begin{aligned}\ell(\boldsymbol{\theta} : \mathcal{D}) &= \log\left(\theta^{\#[H]}(1-\theta)^{\#[T]}\right)\\ &= \#[H]\log\theta + \#[T]\log(1-\theta)\end{aligned}$$

- Taking the gradient and setting it to zero yields

$$\frac{\#[H]}{\theta} - \frac{\#[T]}{1-\theta} \Longrightarrow \theta = \frac{\#[H]}{\#[H]+\#[T]}$$

which is simply the fraction of heads

Maximum Likelihood Parameter Estimation

- The *likelihood function* is the probability of the data

$$L(\boldsymbol{\theta} : \mathcal{D}) = \prod_m P(\boldsymbol{x}^{(m)}; \boldsymbol{\theta})$$

- A function $\tau(x)$ is a *sufficient statistic* if, for any $\mathcal{D}$ and $\mathcal{D}'$ and $\boldsymbol{\theta} \in \Theta$

$$\sum_m \tau(x^{(m)}) = \sum_{\bar{m}} \tau({x'}^{(\bar{m})}) \Rightarrow L(\boldsymbol{\theta} : \mathcal{D}) = L(\boldsymbol{\theta} : \mathcal{D}')$$

- The tuple of counts $\{\#[1], \ldots, \#[K]\}$, where $\#[k]$ is the number of occurrences of x^k in $\mathcal{D}$, is a sufficient statistic for multinomials

$$L(\boldsymbol{\theta} : \mathcal{D}) = \prod_k \theta_k^{\#[k]}$$

- The maximum likelihood estimate for a multinomial is

$$\hat{\theta}_k = \frac{\#[k]}{M}$$

ML Parameter Estimation: Sufficient Statistics

- Why do we refer to $\tau(\boldsymbol{X})$ as a *sufficient statistic*?
- Sufficiency characterizes what is essential in a dataset
- A *statistic* is any function on the sample space that isn't a function of the parameter
- We say that $\tau(\boldsymbol{X})$ is *sufficient* if there is no information in $\boldsymbol{X}$ about θ that isn't available in $\tau(\boldsymbol{X})$
- Consider θ to be a random variable
- In the Bayesian sense, $\tau(\boldsymbol{X})$ is sufficient if

$$\theta \perp \boldsymbol{X} \mid \tau(\boldsymbol{X})$$

ML Parameter Estimation: Sufficient Statistics

- Gaussian Distribution

$$P_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(\boldsymbol{t}(\boldsymbol{\theta})^\top \boldsymbol{\tau}(\boldsymbol{x})\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

where

$$\begin{aligned} \boldsymbol{\tau}(x) &= \begin{bmatrix} x & x^2 \end{bmatrix} \\ \boldsymbol{\theta} &= \begin{bmatrix} \mu & \sigma^2 \end{bmatrix} \in \mathbb{R} \times \mathbb{R}^+ \\ \boldsymbol{t}(\mu, \sigma^2) &= \begin{bmatrix} \frac{\mu}{\sigma^2} & -\frac{1}{2\sigma^2} \end{bmatrix} \end{aligned}$$

$$\implies \mu_{\mathsf{MLE}} = \frac{1}{M} \sum_m \tau_1(x^{(m)}) = \frac{1}{M} \sum_m x^{(m)}$$

ML Estimation for Bayesian Networks: Example

- Consider a Bayesian network $X \to Y$, where X and Y are binary
- The parameters defining the CPDs are

$$\begin{aligned} P(X) : \boldsymbol{\theta}_X &= [\theta_{x^0}, \theta_{x^1}] \\ P(Y \mid X) : \boldsymbol{\theta}_{Y \mid x^0} &= [\theta_{y^0 \mid x^0}, \theta_{y^1 \mid x^0}], \boldsymbol{\theta}_{Y \mid x^1} = [\theta_{y^0 \mid x^1}, \theta_{y^1 \mid x^1}] \end{aligned}$$

- Assume we have a data set of samples $\mathcal{D} = \left\{ \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(M)} \right\}$
- The likelihood function follows as

$$\begin{aligned} L(\boldsymbol{\theta} : \mathcal{D}) &= \prod_m P(y^{(m)} \mid x^{(m)}; \boldsymbol{\theta}) P(x^{(m)}; \boldsymbol{\theta}) \\ &= \left(\prod_m P(y^{(m)} \mid x^{(m)}; \boldsymbol{\theta}_{Y \mid X}) \right) \left(\prod_m P(x^{(m)}; \boldsymbol{\theta}_X) \right) \end{aligned}$$

- Each term *decomposes* into separate *local likelihoods* for each variable, and is a function of the variable's CPD

ML Estimation for Bayesian Networks: Example

$$L(\boldsymbol{\theta} : \mathcal{D}) = \left(\prod_m P(y^{(m)} \mid x^{(m)}; \boldsymbol{\theta}_{Y \mid X}) \right) \left(\prod_m P(x^{(m)}; \boldsymbol{\theta}_X) \right)$$

- The term $\prod_m P(x^{(m)}; \boldsymbol{\theta}_X)$ is the likelihood for a multinomial
- The likelihood decomposes into a product of terms for each group of parameters

$$\prod_m P(y^{(m)} \mid x^{(m)}; \boldsymbol{\theta}_{Y \mid X}) = \prod_{m : x^{(m)} = x^0} P(y^{(m)} \mid x^{(m)}; \boldsymbol{\theta}_{Y \mid x^0}) \cdot \prod_{m : x^{(m)} = x^1} P(y^{(m)} \mid x^{(m)}; \boldsymbol{\theta}_{Y \mid x^1})$$

- As with the multinomial, we can write each as

$$\prod_{m : x^{(m)} = x^0} P(y^{(m)} \mid x^{(m)}; \boldsymbol{\theta}_{Y \mid x^0}) = \theta_{y^0 \mid x^0}^{\#[x^0, y^0]} \cdot \theta_{y^1 \mid x^0}^{\#[x^0, y^1]}$$

- The maximum likelihood parameters follow as

$$\theta_{y^0 \mid x^0} = \frac{\#[x^0, y^0]}{\#[x^0]}$$

ML Estimation for Bayesian Networks

- Suppose that we know the Bayesian network structure G
- Let $\boldsymbol{\theta}_{X_i \mid \mathsf{Pa}_{X_i}}$ be the parameters that determine the CPD $P(X_i \mid \mathsf{Pa}_{X_i})$
- Assume we have a data set of samples $\mathcal{D} = \left\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(M)}\right\}$
- Maximum likelihood estimation corresponds to maximizing the log-likelihood $\ell(\boldsymbol{\theta} : \mathcal{D})$ (equivalent to maximizing the likelihood):

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

  (subject to non-negativity and normalization constraints)
- **Global decomposability**: Likelihood decomposes into a product of independent terms, one for each set of parameters
- Results in an independent optimization problem for each CPD with a simple closed-form solution (objective is concave)

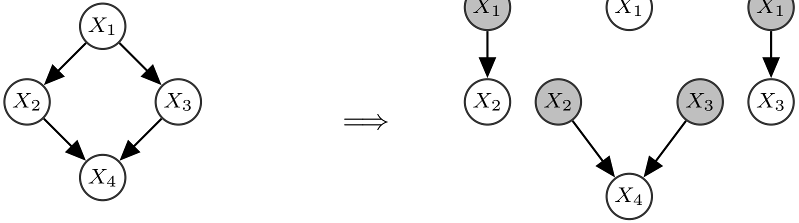
ML Estimation for Bayesian Networks: Decomposability

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

- Consider the four-node Bayesian network

$$P(\boldsymbol{X};\boldsymbol{\theta}) = P(X_1;\theta_1)P(X_2 \mid X_1;\theta_2)P(X_3 \mid X_1;\theta_3)P(X_4 \mid X_2, X_3;\theta_4)$$

- Global decomposability allows us to break this up into four small Bayesian networks



ML Estimation for Bayesian Networks

- Maximum likelihood estimation corresponds to maximizing:

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

- Consider a tabular CPD and a random variable X with parents $\boldsymbol{U}$

$$L_X(\boldsymbol{\theta}_{X \mid \boldsymbol{U}} : \mathcal{D}) = \prod_m \theta_{x^{(m)} \mid \boldsymbol{u}^{(m)}} = \prod_{\boldsymbol{u}\in Val(\boldsymbol{U})}\prod_{x\in Val(X)} \theta_{x \mid \boldsymbol{u}}^{\#[x,\boldsymbol{u}]}$$

- We can optimize each of the local likelihoods separately for each value of $\boldsymbol{u}$, which correspond to multinomial likelihoods

$$\hat{\theta}_{x \mid \boldsymbol{u}} = \frac{\#[x,\boldsymbol{u}]}{\#[\boldsymbol{u}]}$$

- The number of expected assignments for a particular parent decreases exponentially in the number of parents (known as *data fragmentation*)

Limitations of ML Estimation

- Maximum likelihood estimation is purely data-driven
  - Consider estimating $P(H)$ for a coin and a rubber duck
  - Suppose that we toss each 10 times and get 3 heads for each
  - MLE would assign $P(H) = 0.3$ to both the coin and rubber duck
  - However, our intuition suggests that $P(H) \approx 0.5$ for the coin
- MLE does not consider any prior knowledge that we might have about the parameters (i.e., in the form of a prior over the parameters)
- MLE does not provide a measure of confidence
  - Suppose that we had tossed the coin/duck 1000000 times and they had come up heads 300000 times each
  - Whether we use the experiment involving 10 tosses or the experiment involving 1000000 tosses, MLE estimates $P(H) = 0.3$
  - MLE can't capture the fact that we would intuitively trust the larger experiment much more

Bayesian Parameter Estimation

- A prior over the parameters $\boldsymbol{\theta}$ allows us to incorporate knowledge of the parameters (i.e., the parameters are now random variables)
- This gives rise to a joint model over the data and parameters

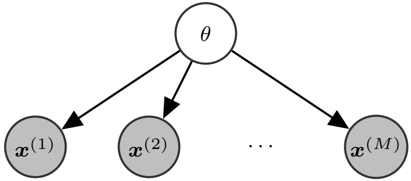
$$P(\mathcal{D}, \boldsymbol{\theta}) = P(\mathcal{D} \,|\, \boldsymbol{\theta}) P(\boldsymbol{\theta})$$

  where $P(\mathcal{D} \,|\, \boldsymbol{\theta})$ is the likelihood function from MLE and $P(\boldsymbol{\theta})$ is the *parameter prior*
- Bayes' rule results in the *parameter posterior*

$$P(\boldsymbol{\theta} \,|\, \mathcal{D}) = \frac{P(\mathcal{D} \,|\, \boldsymbol{\theta}) P(\boldsymbol{\theta})}{P(\mathcal{D})}$$

  where $P(\mathcal{D}) = \int P(\mathcal{D} \,|\, \boldsymbol{\theta}) P(\boldsymbol{\theta}) d\boldsymbol{\theta}$ is the *marginal likelihood*

Bayesian Parameter Estimation: Example



- Let's return to the coin flip example, where the objective is to estimate $\theta = P(H)$
- For MLE, where θ is fixed, we assumed that the samples were marginally independent
- However, since θ is a random variable, the samples are no longer marginally independent, since they provide information regarding θ
- Given θ, the samples are conditionally independent

Bayesian Parameter Estimation: Example

- Consider the joint distribution $P(\mathcal{D}, \theta)$

$$\begin{aligned} P(x^{(1)}, \dots, x^{(M)}, \theta) &= P(x^{(1)}, \dots, x^{(M)} \,|\, \theta) P(\theta) \\ &= \left(\theta^{\#[H]}(1-\theta)^{\#[T]}\right) P(\theta) \end{aligned}$$

- For Bayesian estimation, we are interested in the parameter posterior

$$P(\theta \,|\, x^{(1)}, \dots, x^{(M)}) \propto P(x^{(1)}, \dots, x^{(M)} \,|\, \theta) P(\theta)$$

  which is a product of a Bernoulli distribution and the parameter prior
- We would like the prior and posterior to be of the same family, ideally with a closed-form update to the distribution's hyperparameters (we want a family that is the *conjugate prior* for the likelihood function)

Bayesian Parameter Estimation: Example

- A **Beta distribution** is parameterized by two parameters $\alpha_1, \alpha_0 \in \mathbb{R}^+$

$$\theta \sim \text{Beta}(\alpha_1, \alpha_0) \text{ with } p(\theta) = \gamma \theta^{\alpha_1 - 1}(1-\theta)^{\alpha_0 - 1}$$

  where $\gamma = \frac{\Gamma(\alpha_1 + \alpha_0)}{\Gamma(\alpha_1)\Gamma(\alpha_0)}$ and $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$ is the *Gamma function*

- Consider the marginal distribution over the first coin toss

$$\begin{aligned} P(x^{(1)} = H) &= \int_0^1 P(x^{(1)} = H \mid \theta) P(\theta) d\theta \\ &= \int_0^1 \theta \; P(\theta) d\theta = \frac{\alpha_1}{\alpha_1 + \alpha_0} \end{aligned}$$

  α_1 and α_0 act like the number of imaginary Heads and Tails that we have thrown prior to sampling

Bayesian Parameter Estimation: Example

- Let's consider the posterior after drawing M samples

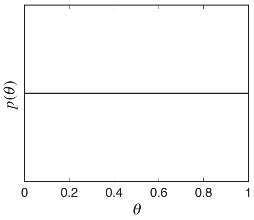
$$\begin{aligned} P(\theta \mid x^{(1)}, \ldots, x^{(M)}) &\propto P(x^{(1)}, \ldots, x^{(M)} \mid \theta) P(\theta) \\ &\propto \theta^{\#[H]} \cdot (1-\theta)^{\#[T]} \cdot \theta^{\alpha_1 = 1} (1-\theta)^{\alpha_0 - 1} \\ &= \theta^{\#[H]-1} (1-\theta)^{\alpha_0 + \#[T] - 1} \\ &= \theta^{\alpha_1 + \#[H]-1} (1-\theta)^{\alpha_0 + \#[T] - 1} \\ &= \text{Beta}(\alpha_1 + \#[H], \alpha_0 + \#[T]) \end{aligned}$$

- The Beta distribution is conjugate to the Bernoulli distribution
- The hyperparameter update is straightforward

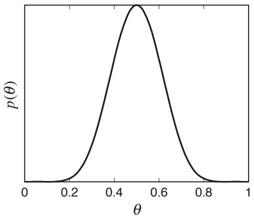
$$\alpha_1 \leftarrow \alpha_1 + \#[H] \qquad \alpha_0 \leftarrow \alpha_0 + \#[T]$$

i.e., we are essentially updating the counts for Heads and Tails

Bayesian Parameter Estimation: Example



Beta$(1,1)$



Beta$(10,10)$

$$\alpha_1 \leftarrow \alpha_1 + \#[H] \qquad \alpha_0 \leftarrow \alpha_0 + \#[T]$$

- Suppose that $\mathcal{D}$ contains 3 Heads and 7 Tails
  - $P(\theta) = \text{Beta}(1,1)$: $P(\theta \mid \mathcal{D}) = \text{Beta}(4,8)$
  - $P(\theta) = \text{Beta}(10,10)$: $P(\theta \mid \mathcal{D}) = \text{Beta}(13,17)$
- Consider the likelihood of the next toss $P(X^{(M+1)} = H \mid \mathcal{D}) = \frac{\alpha_1}{\alpha_1 + \alpha_0}$
  - $P(\theta) = \text{Beta}(1,1)$: $P(X^{(M+1)} = H \mid \mathcal{D}) = 4/12 \approx 0.33$
  - $P(\theta) = \text{Beta}(10,10)$: $P(X^{(M+1)} = H \mid \mathcal{D}) = 13/30 \approx 0.43$

Bayesian Parameter Estimation: Example

- Recall the example of estimating $P(H)$ for a coin and a rubber duck
- We tossed each 10 times and got 3 heads for each
- MLE assigned $P(H) = 0.3$ to both the coin and rubber duck, which is inconsistent with intuition
- Instead we might use a Beta prior with $\alpha_1 = \alpha_0 = 100$ for the coin and $\alpha_1 = \alpha_0 = 1$ for the duck

Bayesian Parameter Estimation: Prior Distribution

- We would like to express the posterior in terms of sufficient statistics just as we can the likelihood function, e.g., for a multinomial

$$L(\boldsymbol{\theta} : \mathcal{D}) = \prod_k \theta_k^{\#[k]}$$

- This depends upon the nature of the prior over parameters
- One effective prior is the *Dirichlet* distribution

$$\boldsymbol{\theta} \sim \text{Dirichlet}(\alpha_1, \ldots, \alpha_K) \text{ s.t. } P(\boldsymbol{\theta}) \propto \prod_k \theta_k^{\alpha_k - 1}$$

  where $\{\alpha_1, \alpha_2, \ldots, \alpha_K\}$ is a set of hyperparameters
- The Dirichlet is a generalization of the Beta distribution

Bayesian Parameter Estimation: Conjugate Priors

- The Dirichlet is the conjugate prior to the multinomial, i.e., if the prior is Dirichlet (i.e., $P(\boldsymbol{\theta}) = \text{Dirichlet}(\alpha_1, \ldots, \alpha_K)$), then the posterior is also Dirichlet, $P(\boldsymbol{\theta} \mid \mathcal{D}) = \text{Dirichlet}(\alpha_1 + \#[1], \ldots, \alpha_K + \#[K])$
- A conjugate prior makes it easy to update our distribution over the parameters, i.e., from

$$P(\boldsymbol{\theta}) = \text{Dirichlet}(\alpha_1, \ldots, \alpha_K)$$

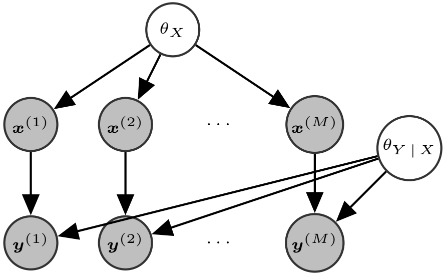
to

$$P(\boldsymbol{\theta} \mid \mathcal{D}) = \text{Dirichlet}(\alpha_1 + \#[1], \ldots, \alpha_K + \#[K])$$

- Beyond the Dirichlet distribution, there are other conjugate priors for other distributions (e.g., Gaussians are self-conjugate)

Bayesian Parameter Estimation for Bayesian Networks

- Consider estimating the parameters for a Bayesian network $X \rightarrow Y$
- The associated parameters are $\boldsymbol{\theta}_X$ and $\boldsymbol{\theta}_{Y \mid X}$
- Since the parameters are now random variables, we can formulate the joint likelihood $P(\mathcal{D}, \boldsymbol{\theta})$ as a Bayesian network



- Data instances $(x^{(i)}, y^{(i)})$ and $(x^{(j)}, y^{(j)})$ are independent given the latent parameters

Bayesian Parameter Estimation for Bayesian Networks

- Consider again the parameter posterior

$$P(\boldsymbol{\theta} \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid \boldsymbol{\theta}) P(\boldsymbol{\theta})}{P(\mathcal{D})}$$

- The likelihood function decomposes into local likelihoods

$$P(\mathcal{D} \mid \boldsymbol{\theta}) = \prod L_i(\boldsymbol{\theta}_{X_i \mid \mathsf{Pa}_{X_i}} : \mathcal{D})$$

- If *global parameter independence* holds (i.e., the prior decomposes),

$$P(\boldsymbol{\theta}) = \prod_i P(\boldsymbol{\theta}_{X_i \mid \mathsf{Pa}_{X_i}})$$

- The posterior becomes

$$P(\boldsymbol{\theta} \mid \mathcal{D}) = \prod_i P(\boldsymbol{\theta}_{X_i \mid \mathsf{Pa}_{X_i}} \mid \mathcal{D})$$