Bayesian Inference

CMPUT 261: Introduction to Artificial Intelligence

P&M §10.4, §8.6

Logistics

- Assignment #2 due **Tuesday, Oct 25 at 11:59pm**
- Midterm is **Thursday, Nov 3**
    - Coverage: Everything up to and including lecture 18 (Nov 1: Image Data)

Recap: Linear Models

- **Linear regression** is a simple model for predicting real quantities
  - Can be used for classification too, either based on **sign** of prediction or using **logistic regression**
- **Gradient descent** is a general, widely-used training procedure (with several variants)
  - Linear models can be optimized in **closed form** for certain losses
  - In practice often optimized with gradient descent

Recap: Overfitting

- **Overfitting** is when a learned model fails to **generalize** due to **overconfidence** and/or learning **spurious regularities**
- **Causes of overfitting:**
  - **Bias**: Systematic choice of suboptimal hypotheses
  - **Variance**: Different training sets can yield very different hypotheses
  - **Noise**: Unpredictability that is inherent in the process
    (e.g., coin flips cannot be perfectly predicted, even by the "true" model)
- **Avoiding overfitting:**
  1. **Pseudocounts:** Add **imaginary** observations
  2. **Regularization: Penalize** model complexity

Lecture Outline

1. Recap & Logistics
2. Cross Validation
3. Exact Bayesian Inference
4. Monte Carlo Simulation

After this lecture, you should be able to:

- derive the posterior probability **of a model** using Bayes' rule
- explain how to use the Beta and Bernoulli distributions for Bayesian learning
- demonstrate model averaging
- estimate expectations from a finite sample
- apply Hoeffding's inequality to derive PAC bounds for number of samples, confidence level, and/or error
- implement rejection sampling, importance sampling, and forward sampling

Hyperparameters

- Previous methods for avoiding overfitting require us to choose some numbers:
  - How many **pseudocounts** to add?
  - What should **regularization parameter** λ be?
- These are **hyperparameters**: Parameters that specify the **training process** or **hypothesis space** rather than the hypothesis itself
- Ideally we would like to be able to choose **hyperparameters from the data**
- **Question:** Can we use the **test data** to see which of these work best?
- **Idea:** Use some of the **training data** as an **estimate** of the test data

Cross-Validation Procedure

Cross-validation can be used to estimate most hyperparameters:

For each of a set of candidate hyperparameters:

1. **Randomly remove** some datapoints from the **training set**;
   these examples are the **validation set**

2. **Train** the model on the training set using some values of hyperparameters
   (pseudocounts, polynomial degree, regression parameter, etc.)

3. **Evaluate** the results on the **validation set**

Then, choose whichever hyperparameters had the best performance on the validation set

k-Fold Cross-Validation

- We want our **training set** to be as large as possible, so we get better models
- We want our **validation set** to be as large as possible, so that it is an accurate estimation of test performance
- When one is larger, the other must be **smaller**
- **k-fold cross-validation** lets us use every one of our examples for both validation and training

k-Fold Cross-Validation Procedure

1. **Randomly partition** training data into k approximately equal-sized sets (**folds**)
2. **Train** k times, each time using all the folds but one; **remaining fold** is used for **validation**
3. **Optimize** hyperparameters based on **validation errors**

- Each example is used exactly **once** for **validation** and $k-1$ **times** for **training**
- **Extreme case**: $k = n$ is called **leave-one-out** cross-validation

Learning Point Estimates

- So far, we have considered how to find the best **single** model, e.g.,
  - learn ***a*** classification function
  - optimize ***the*** weights of a linear or logistic regression
- The **predictions** might be a probability distribution, but they are coming out of a single **model**:

$$P(Y \mid X)$$ Probability of target Y given observation X

- We have been learning **point estimates** of our model

Learning Model Probabilities

- Instead, we could learn a distribution over **models**:
  - $\Pr(X, Y \mid \theta)$ Probability of target Y and features X given model θ
  - $\Pr(\theta \mid D)$ Probability of model θ given dataset D
- This is called **Bayesian learning**: we never discard any model, we only weight them differently depending upon their **posterior probability**
- **Question:** Why would we want to do that?

What is a Model?

- $\Pr(X, Y \mid \theta)$ Probability of target Y and features X given model θ
- $\Pr(\theta \mid D)$ Probability of model θ given dataset D

- We can do Bayesian learning over **finite** sets of models:
  - e.g., { rank by feature $\theta \mid \theta \in$ {height, weight, age} }
- We can do Bayesian learning over **parametric families** of models:
  - e.g., { regression with weights $w_0=\theta_1$, $w_1=\theta_2 \mid \theta \in \mathbb{R}^2$ }
- We can mix the two!
  - θ can encode choice of **model family and parameters**

What is the Dataset?

- $\Pr(X, Y \mid \theta)$ Probability of target Y and features X given model θ
- $\Pr(\theta \mid D)$ Probability of model θ given dataset D

- We have an expression for the probability of a single example given a model: $\Pr(X, Y \mid \theta)$
- **Question:** What is the expression for the probability of a dataset of observations $D = \{(X_1, Y_1), \ldots, (X_m, Y_m)\}$ given a model?
  - Easiest approach: Assume that the dataset independent, identically distributed observations: $(X_i, Y_i) \sim P(X, Y \mid \theta)$

$$\Pr(D \mid \theta) = \Pr(X_1, Y_1 \mid \theta) \times \ldots \times \Pr(X_m, Y_m \mid \theta)$$

$$= \prod_{i=1}^{m} \Pr(X_i, Y_i \mid \theta)$$

What is the Posterior Model Probability?

- $\Pr(X, Y \mid \theta)$ Probability of target Y and features X given model θ
- $\Pr(\theta \mid D)$ Probability of model θ given dataset D

Now we can use **Bayes' Rule** to compute the posterior probability of a model θ:

Prior probability of model θ

$$\Pr(\theta \mid D) = \frac{\Pr(D \mid \theta)\Pr(\theta)}{\Pr(D)}$$

$$= \frac{\prod_i \Pr(X_i, Y_i \mid \theta)\Pr(\theta)}{\Pr(D)}$$

$$= \frac{\prod_i \Pr(X_i, Y_i \mid \theta)\Pr(\theta)}{\sum_{\theta'} \Pr(D \mid \theta')\Pr(\theta')}$$

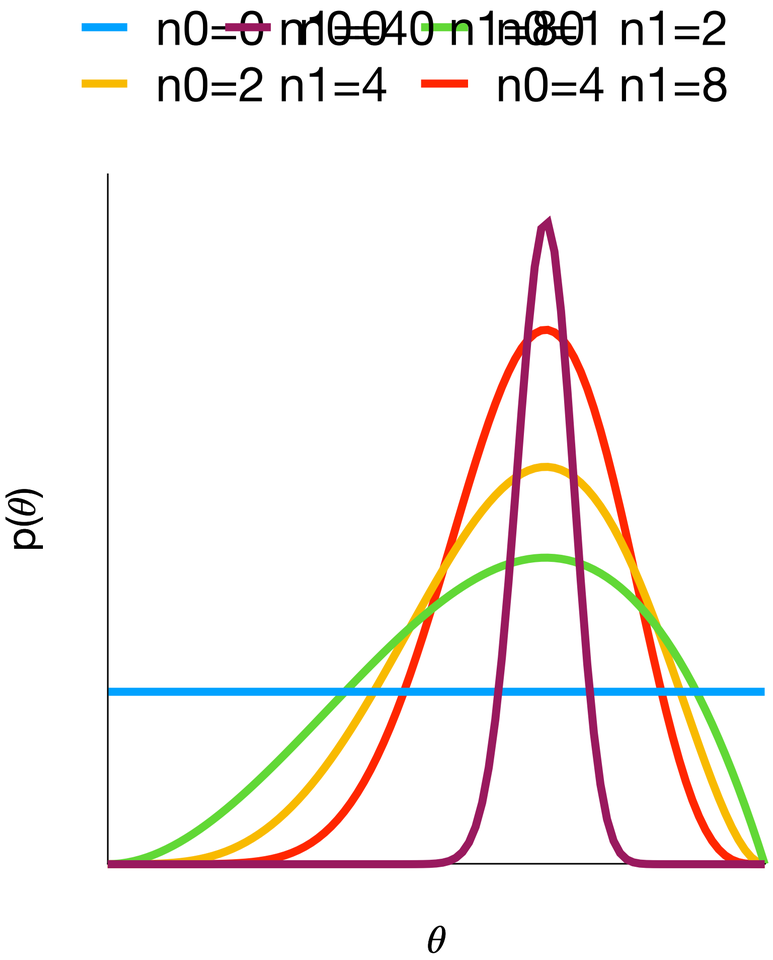
Likelihood of data D given model θ

Example: Biased Coin

- Back to coin flipping! We can flip a coin and observe heads or tails, but we don't know the coin's bias
- Model: **Binomial observations**
  - Observations: $Y \in \{h, t\}$
  - Bias: $\theta \in [0,1]$
  - Likelihood: $\Pr(H \mid \theta) = \theta$
  - **Question:** What should the **prior** $\Pr(\theta)$ be?

Biased Coin: Posterior Probabilities

- Before we see any flips, all biases are equally probable (according to our prior)
- After more and more flips, we become more confident in θ
- θ with **highest probability** is 2/3
  - **Expected** value of θ is less! (**why**?)
  - But with more observations, mode and expected value get **closer**



Beta-Binomial Models

- Likelihood: $P(h \mid \theta) = \theta$
  - aka **Bernoulli**$(h \mid \theta)$
  - Dataset likelihood: $\theta^{n_1} \times (1 - \theta)^{n_0}$
  - aka **Binomial**(n_1, n_0)
- Prior: $P(\theta) \propto 1$
  - aka **Beta**$(1,1)$
- Models of this kind are called **Beta-Binomial models**
- They can be solved analytically: $Pr(\theta \mid D) = \text{Beta}(1 + n_1, 1 + n_0)$

Conjugate Priors

- The beta distribution is a **conjgate prior** for the binomial distribution:
  - Updating a beta prior with a binomial likelihood gives a **beta posterior**
- Other distributions have this property:
  - Gaussian-Gaussian (for means)
  - Dirichlet-Multinomial (generalization of Beta-Binomial for multiple values)

Using Model Probabilities

So we can estimate $\Pr(\theta \mid D)$. What can we do with it?

1. Parameter estimates
2. Target predictions (model averaging)
3. Target predictions (point estimates)

1. Parameter Estimates

- Sometimes, we really want to know the **parameters** of a model itself
- E.g., maybe I don't care about predicting the next coin flip, but I do want to know whether the coin is fair
- Can use $\Pr(\theta \mid D)$ to make statements like

$$\Pr(0.49 \leq \theta \leq 0.51) > 0.9$$

2. Model Averaging

- Sometimes we do want to make **predictions**:

$$\Pr(Y \mid D) = \sum_{\theta} \Pr(Y \mid \theta) \Pr(\theta \mid D)$$

- This is called the **posterior predictive distribution**
- **Question:** How is this different from just learning a point estimate of a model, and then predicting with that model?

3. Maximum A Posteriori

- Sometimes we do want to make predictions, **but...**

$$\Pr(Y \mid D) = \int_0^1 \Pr(Y \mid \theta) \Pr(\theta \mid D) d\theta$$

- the posterior predictive distribution may be **expensive** to compute (or even **intractable**)
- One possible solution is to use the **maximum a posterior** model as a point estimate:

$$\Pr(Y \mid D) \simeq \Pr(Y \mid \hat{\theta}) \qquad \text{where } \hat{\theta} = \arg\max_{\theta} \Pr(\theta \mid D)$$

- **Question:** Why would you do this instead of just using a point estimate that was computed in the usual way?

Prior Distributions as Bias

- Suppose I'm comparing two models, θ_1 and θ_2 such that

$$\Pr(D \mid \theta_1) = Pr(D \mid \theta_2)$$

- **Question:** Which model has higher **posterior probability**?
- Priors are a way of encoding **bias**: they tell use which models to prefer when the data doesn't

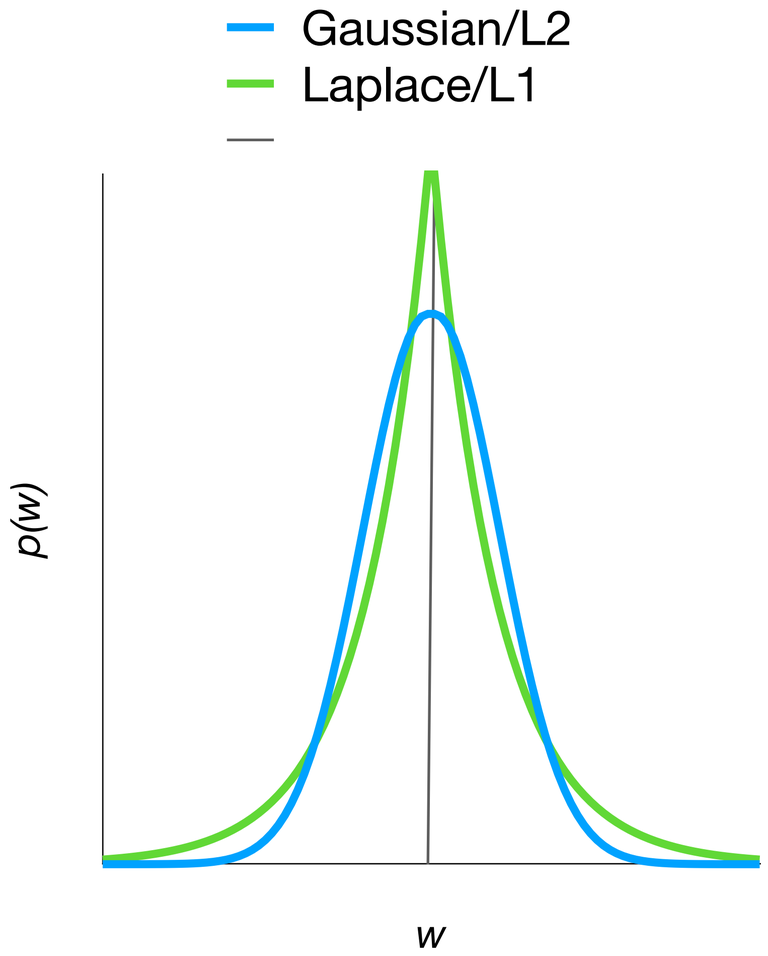
Priors for Pseudocounts

- We can straightforwardly encode pseudocounts as prior information in Beta-Binomial and Dirichlet-Multinomial models
- E.g., for pseudocounts k_1 and k_0,

$$p(\theta) = \text{Beta}(1 + k_1, 1 + k_0)$$

Priors for Regularization

- Some **regularizers** can be encoded as priors also
- **L2 regularization** is equivalent to a **Gaussian** prior on the weights:
  $p(w) = \mathcal{N}(w \mid m, s)$
- **L1 regularization** is equivalent to a **Laplacian** prior on the weights:
  $p(w) = \exp(|w|)/2$



Estimation via Sampling

- Suppose that we are able to generate independent random **samples** from a random variable X
- How can we use those random samples to estimate the **expected value** of X?
  - or some function h of X; but that in general is just a different random variable $Y = h(X)$
- **Question:** But first, why would we *want* to?

Estimation from a Sample

Law of Large Numbers:

As the number n of independent samples $x_1, x_2, \ldots, x_n$ from a random variable X with distribution $f(x)$ approaches infinity, the **sample average** approaches the **expected value** of X.

$$\mathbb{E}[X] = \sum_x f(x)x \approx \frac{1}{n}\sum_{i=1}^{n} x_i$$

Since $Y = h(X)$ is also a random variable, this generalizes to arbitrary **functions** of X:

$$\mathbb{E}[h(X)] = \sum_x f(x)h(x) \approx \frac{1}{n}\sum_{i=1}^{n} h(x_i)$$

Probably Approximately Correct

- We never actually have an **infinite** number of sampled values
- How do we know when we have **enough** samples?

Hoeffding's inequality:
Suppose $0 \leq X \leq 1$, and s is the sample average from n independent samples from X. Then

$$\Pr(|\mathbb{E}[X] - s| > \epsilon) \leq 2e^{-2n\epsilon^2}.$$

- For any given **error margin** ϵ and number of samples n, we can plug into this formula and get a **PAC bound**.
    - Can also go the other way: plug in the acceptable error bound to RHS, and derive the **number of samples** n needed
- This generalizes to arbitrary **bounded** random variables $a \leq X \leq b$.

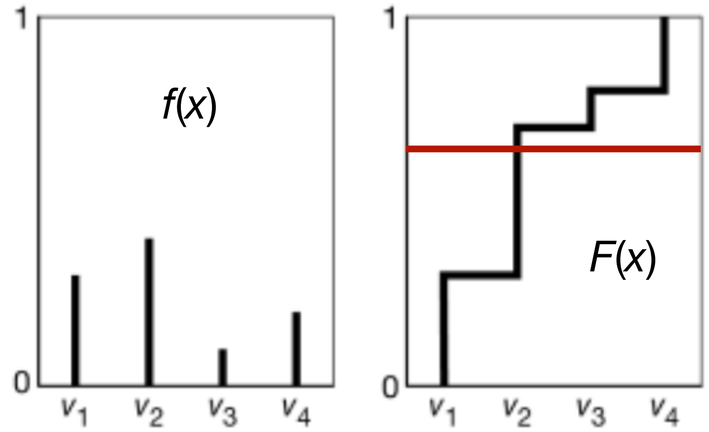
Generating Samples from a Single Variable

How can we generate samples from a distribution?

1. **Totally order** the domain of the variable
   (can be arbitrary for categorical variables)

2. **Cumulative distribution**: $F(x) = \Pr(X \leq x)$

$$F(x) = \int_{-\infty}^{x} f(z)dz \qquad F(x) = \sum_{x' \leq x} f(x')$$



3. Select a **uniform** random number $y \in [0,1]$

4. Return $x_i = F^{-1}(y)$

Hard-To-Sample Distributions

Often, we want to sample from distributions that are **hard** to sample from, especially large **joint distributions**

Question: Why might a distribution be hard to sample from?

1. Use samples from easier distributions:
    - Rejection Sampling
    - Importance Sampling
2. Go piece by piece through the joint distribution
    - Forward Sampling in a Belief Network
    - Particle Filtering

Proposal Distributions

- Can we use an **easy-to-sample** distribution $g(x)$ to help us sample from $f(x)$?
  - Very common: We know an **unnormalized** $f^*(x)$, but not the properly normalized distribution $f(x)$:

$$f(x) = \frac{f^*(x)}{\int_{-\infty}^{\infty} f^*(z)dz}$$

- $f(x)$ is the **target distribution**
  - $f^*(x)$ is the **unnormalized target distribution**
- $g(x)$ is the **proposal distribution**

Importance Sampling

- Rejection sampling works, but it can be **wasteful**
  - Lots of samples get rejected when proposal and target distributions are very **different**
- What if we took a **weighted average** instead?
  1. Sample $x_1, x_2, \ldots, x_n$ from $g(x)$
  2. **Weight** each sample x_i by $w_i = \frac{Mf^*(x_i)}{g(x_i)}$
  3. **Estimate** is $\frac{1}{\sum_j w_j} \sum_{x_i \sim g} w_i x_i$

$$
\begin{aligned}
\mathbb{E}[X] &= \sum_x f(x)x \\
&= \sum_x \frac{g(x)}{g(x)} f(x)x \\
&= \sum_x g(x) \frac{f(x)}{g(x)} x \\
&\approx \frac{1}{n} \sum_{x_i \sim g} \frac{f(x_i)}{g(x_i)} x_i
\end{aligned}
$$

Forward Sampling in a Belief Network

- Sometimes we know how to sample **parts** of a large joint distribution in terms of other parts
  - E.g., belief networks: $P(X, Y, Z) = P(X)P(Y)P(Z \mid X, Y)$
  - We might be able to **directly** sample from each **conditional distribution** but not from the **joint distribution**
- Forward sampling:
  1. **Select** an ordering of variables consistent with the factoring
  2. **Repeat** until enough samples generated:
     For each variable X in the ordering:
     Sample $x_i \sim P(X \mid pa(X))$

Summary

- **Cross-validation** is a powerful technique for selecting hyperparameters based on data
- In Bayesian Learning, we learn a **distribution** over models instead of a **single model**
- When the model is **conjugate**, posterior probabilities can be computed **analytically**
- We can make predictions by **model averaging** to compute the **posterior predictive distribution**
- The **prior** can encode **bias over models**, much the same as **regularization**
- Often we **cannot directly estimate** probabilities or expectations from our model
- **Monte Carlo estimates**: Use a **random sample** from the distribution to estimate expectations by sample averages
- Two families of techniques for hard to sample distributions:
  1. Use an easier-to-sample **proposal** distribution instead
  2. Sample parts of the model **sequentially**