Formalizing Parameter Estimation

CMPUT 296: Basics of Machine Learning

Textbook §5.1

Logistics

Reminders:

- Assignment 1 due Thursday (**Thursday, September 24**)

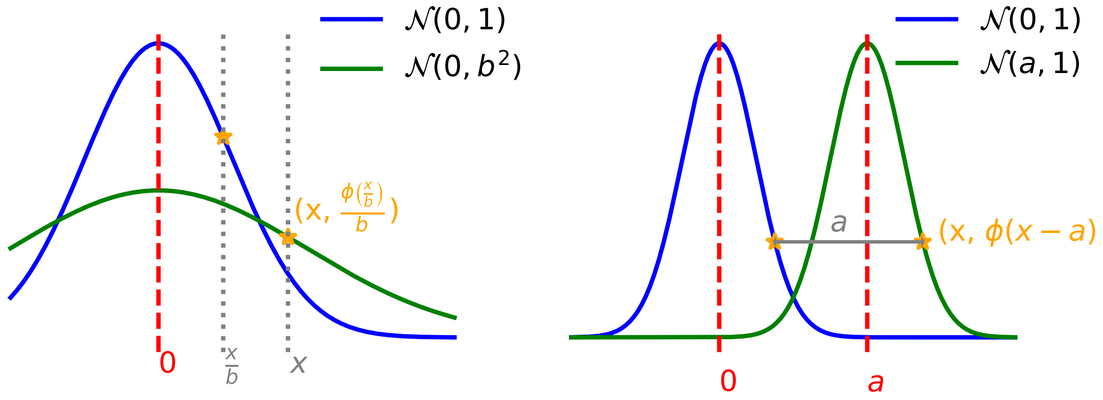
Recap: Optimization

- We often want to find the argument w^* that **minimizes** an **objective function** c:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} c(\mathbf{w})$$

- Every interior minimum is a **stationary point**, so check the stationary points
- Stationary points usually identified **numerically**
  - Typically, by **gradient descent**
- Choosing the **step size** is important for efficiency and correctness
  - Common approach: Adaptive step size
  - E.g., by **line search**

Recap: Properties of Gaussians




```
__main__> import scipy.stats
__main__> mu=4; sigma=2
__main__> f = scipy.stats.norm(loc=mu, scale=sigma)
__main__> f.ppf(0.025)
0.08007203091989101
__main__> f.cdf(0.08007203091989101)
0.024999999999999977
__main__> (0.08007203091989101 - mu) / sigma
-1.9599639845400545
__main__> scipy.stats.norm.ppf(0.025)
-1.9599639845400545
__main__> scipy.stats.norm.cdf(-1.9599639845400545)
0.024999999999999977
```

- Gaussian distributions parameterized by mean μ and variance σ^2: $\mathcal{N}(\mu, \sigma^2)$
- **Standard normal** $\mathcal{N}(0,1)$ has **pdf** $\phi(x)$ and **cdf** $\Phi(x)$
- Let f, F be the pdf and cdf for $\mathcal{N}(\mu, \sigma^2)$

Then: $f(x) = \dfrac{\phi\left(\dfrac{x-\mu}{\sigma}\right)}{\sigma}$

and: $F(x) = \Phi\left(\dfrac{x-\mu}{\sigma}\right)$

Recap: Newton-Raphson as First-Order Gradient Descent?

The **Newton-Raphson** update is:

$$w_{t+1} \leftarrow w_t - \left[c''(w_t)\right]^{-1} c'(w_t)$$

First-order gradient descent for multivariate functions $c : \mathbb{R} \rightarrow \mathbb{R}$ is just:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \nabla c(\mathbf{w}_t)$$

- For **univariate** functions, Newton-Raphson can be understood as first-order gradient descent that chooses a step size of $\eta_t = \dfrac{1}{c''(w_t)}$
- For multivariate functions, $\left[c''(w_t)\right]^{-1}$ is the inverse of an $n \times n$ **matrix** of partial derivatives called the **Hessian**
  - So it doesn't perform a simple scalar multiplication

$$\begin{bmatrix} \frac{\partial^2 c}{\partial x_1^2}(\mathbf{x}) & \frac{\partial^2 c}{\partial x_1 \partial x_2}(\mathbf{x}) & \cdots & \frac{\partial^2 c}{\partial x_1 \partial x_n}(\mathbf{x}) \\ \frac{\partial^2 c}{\partial x_2 \partial x_1}(\mathbf{x}) & \frac{\partial^2 c}{\partial x_2^2}(\mathbf{x}) & \cdots & \frac{\partial^2 c}{\partial x_2 \partial x_n}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 c}{\partial x_n \partial x_1}(\mathbf{x}) & \frac{\partial^2 c}{\partial x_n \partial x_2}(\mathbf{x}) & \cdots & \frac{\partial^2 c}{\partial x_n^2}(\mathbf{x}) \end{bmatrix}$$

Outline

1. Recap & Logistics
2. Prediction
3. Modeling Problem
4. MAP and MLE

Prediction

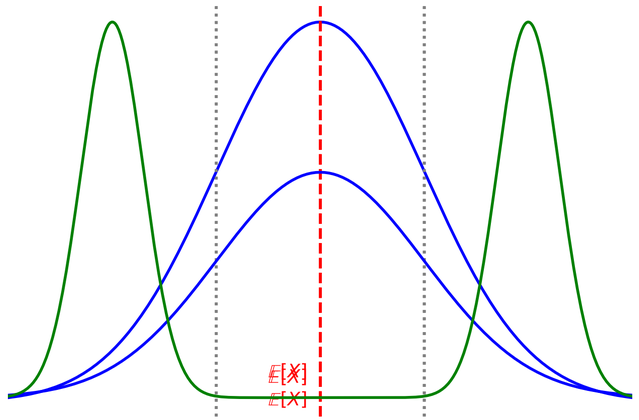
- **Previously:** Given an i.i.d. dataset $X_1, \ldots, X_n$, we wanted to estimate some property of the distribution that generated them (usually μ)
- Concentration inequalities (Hoeffding, Chebyshev) let us bound the probability of our estimate $\bar{X}$ being within $\pm\epsilon$ of the true value:

$$\Pr\left(|\bar{X} - \mu| \leq \epsilon\right) \geq (1 - \delta)$$

Now suppose that we want to **predict** the value of the **next** datapoint X_{n+1} based on our estimate from $X_1, \ldots, X_n$.

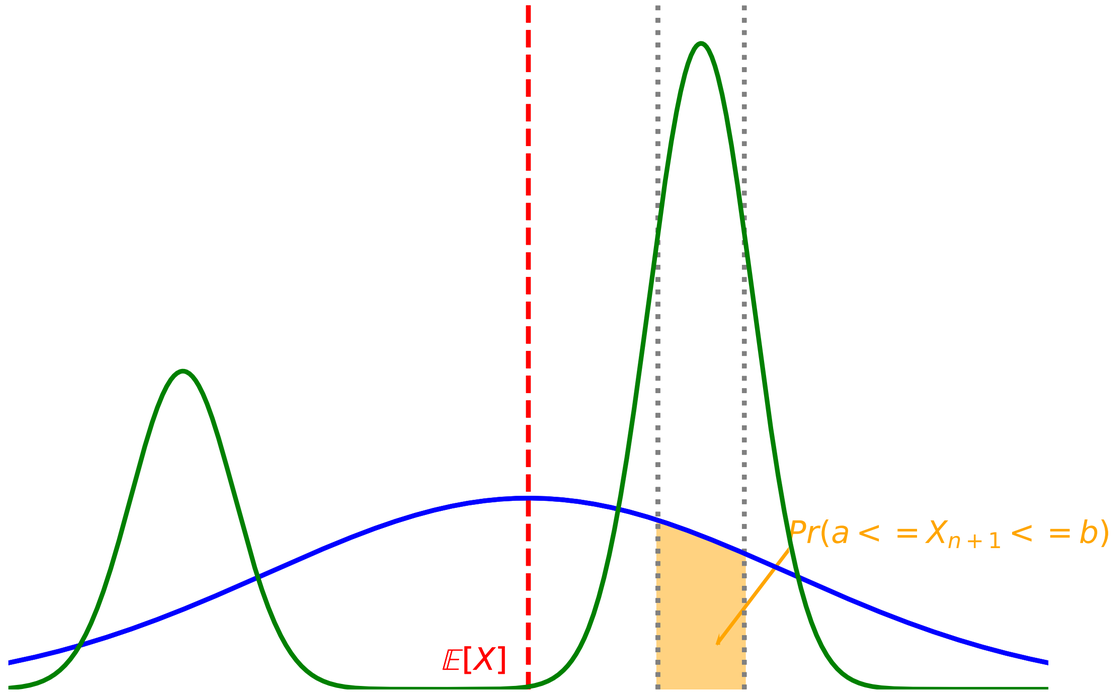
Questions:

1. What number should we predict to minimize MSE?
2. What is the probability that we will be within ϵ of the true value?



Prediction:
Mean and Variance Are Not Enough

- If we know σ^2, we can bound the probability of X_{n+1} being within ϵ of μ
- What if we want to know the probability of X_{n+1} lying in some other range $[a, b]$?
- If we know the full distribution, then we can compute $F(b) - F(a)$
- But many very different distributions share the same μ and σ



The Modeling Problem

- For prediction, we will want to find a **model**
  - A function $\hat{f}$ that approximates the distribution f that generates our data
- A good modeling procedure should:
  1. **Generalize:** Model should perform well on **unseen** data
  2. **Incorporate prior knowledge/assumptions:** E.g., we should be able to take advantage of knowing that the true distribution is bounded, etc.
  3. **Scale:** Compute a solution in a reasonable amount of time for large sets of training data

Parametric Models

- Our goal is to select $\hat{f} \in \mathcal{F}$ based on a dataset $\mathcal{D} = \{x_i\}_{i=1}^n$
  - The data is drawn from some unknown "true" distribution f^*
  - $\mathcal{F}$ is a family of possible distributions (the **hypothesis space** or **function class**)
- It is often convenient to consider **parametric hypothesis spaces**
  - E.g., univariate Gaussians $\mathcal{F} = \{\mathcal{N}(\mu, \sigma^*) \mid \mu \in \mathbb{R}, \sigma \in \mathbb{R}^+\}$
  - Picking $\hat{f}$ is then equivalent to picking a particular set of **parameters**

Questions:

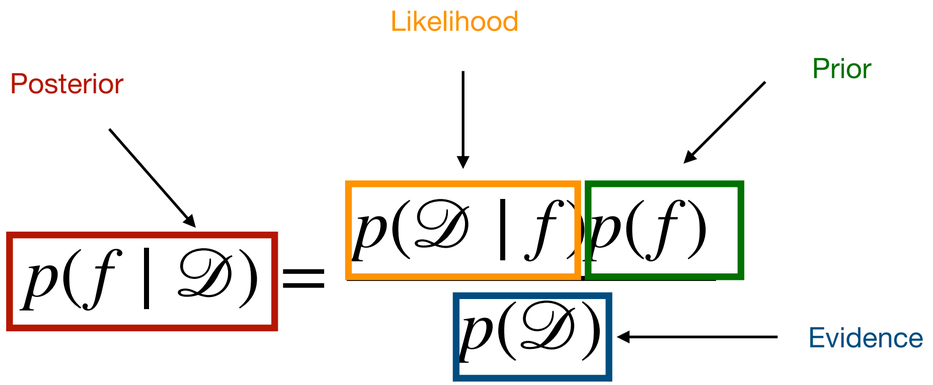
1. What is a **good** model?
2. How should we **choose** a model from $\mathcal{F}$?

Maximum A Posteriori Estimation

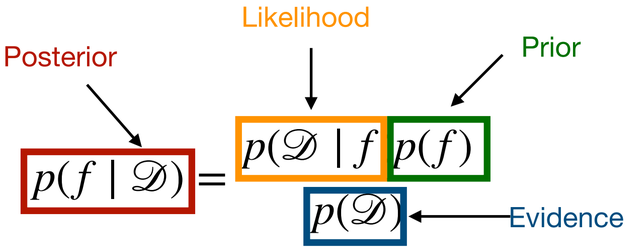
Maximum a Posteriori estimate:
Choose the model that is **most probable** given the data

$$f_{\text{MAP}} = \arg\max_{f \in \mathcal{F}} p(f \mid \mathcal{D})$$

Question: How are we supposed to compute the probability of a model?



Likelihood



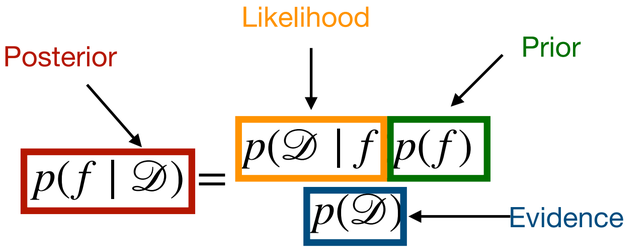
When $\mathscr{D} = \{x_1, \ldots, x_n\}$ are assumed to be distributed **i.i.d.**:

$$p(\mathscr{D} \mid f) = p(x_1, x_2, \ldots, x_n \mid f) = \prod_{i=1}^{n} p(x_i \mid f)$$

But $p(x_i \mid f) = f(x_i)$, so the **likelihood** is

$$p(\mathscr{D} \mid f) = \prod_{i=1}^{n} f(x_i)$$

Prior

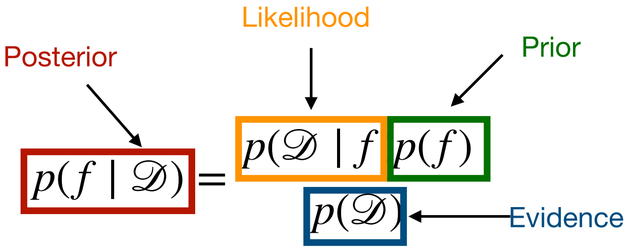


- The **prior** $p(f)$ allows us to express our beliefs about which models are more probable
- E.g.:
  - No model is more probable than another: uniform prior
  - Preference for models with small-magnitude means:

$$p(\mu) \propto \left| \frac{1}{\mu} \right|$$

  - Preference for "simple" models: smaller coefficients more probable
- The key point is that these are reasons to prefer given models that **don't depend on the data** (i.e., they are "prior" to the dataset).

Model Evidence and Constants



The **model evidence** (or **marginal likelihood**) $p(\mathcal{D})$ is the expected probability of the dataset, marginalizing over all models:

$$p(\mathcal{D}) = \underbrace{\mathbb{E}}_{\text{expectation with respect to } p(f)}\left[p(\mathcal{D} \mid f)\right] = \begin{cases} \sum_{f \in \mathcal{F}} p(\mathcal{D} \mid f)p(f) & \text{for discrete } f \\ \int_{\mathcal{F}} p(\mathcal{D} \mid f)p(f)\,df & \text{for continuous } f \end{cases}$$

$$p(x) = \int_{\mathcal{F}} p(x, y)\,dy$$

$$p(x, y) = p(x \mid y)p(y)$$

Note that $p(\mathcal{D})$ is **constant** with respect to the model f

So

$$f_{\text{MAP}} = \arg\max_{f \in \mathcal{F}} p(f \mid \mathcal{D}) = \arg\max_{f \in \mathcal{F}} \frac{p(\mathcal{D} \mid f)p(f)}{p(\mathcal{D})} = \arg\max_{f \in \mathcal{F}} p(\mathcal{D} \mid f)p(f)$$

Maximum Likelihood Estimation

- Sometimes we have no reason to prefer one model over another!
  - Then $p(f) = k$ for some constant k
- Then $p(f)$ is also constant with respect to f, and we have

Likelihood

$$f_{\text{MAP}} = \arg\max_{f \in \mathcal{F}} p(\mathcal{D} \mid f)p(f) = \arg\max_{f \in \mathcal{F}} p(\mathcal{D} \mid f)k = \arg\max_{f \in \mathcal{F}} p(\mathcal{D} \mid f)$$

MAP estimates with a uniform prior are also called **maximum likelihood estimates**

$$f_{\text{MLE}} = \arg\max_{f \in \mathcal{F}} p(\mathcal{D} \mid f)$$

Example: Poisson Data

Example: Suppose dataset $\mathscr{D} = \{2,5,9,5,4,8\}$ is drawn i.i.d. from an unknown Poisson distribution, with parameter λ_0.

We will maximize

$$\lambda_{\text{MLE}} = \arg\max_{\lambda \in (0,\infty)} p(\mathscr{D} \mid \lambda)$$

$$= \arg\max_{\lambda \in (0,\infty)} \ln p(\mathscr{D} \mid \lambda) \longleftarrow \textbf{Why?}$$

$$= \arg\max_{\lambda \in (0,\infty)} \sum_{i=1}^{n} \ln p(x_i \mid \lambda)$$

1. Log is an increasing function, so
$$\arg\max_{x>0} x = \arg\max_{x>0} \ln x$$
2. $$p(10 \text{ coin tosses }) = 2^{-10}$$
$$p(1000 \text{ coin tosses }) = 2^{-1000}$$
$$\cdots$$
3. $$\ln(a \times b) = \ln a + \ln b$$

Inserting pmf for Poisson distribution, taking derivative, and solving for 0 yields:

$$\lambda_{\text{MLE}} = \frac{1}{n} \sum_{i=1}^{n} x_i = 5.5 \text{ for dataset } \mathscr{D}$$

MAP vs MLE for Infinite Data

Suppose instead we want to use a **Gamma prior** with parameters $k = 3$ and $\theta = 1$:

$$p(\lambda) = \frac{\lambda^{k-1} e^{-\frac{\lambda}{\theta}}}{\theta^k \Gamma(k)}$$

Then MAP estimate is $\lambda_{\text{MAP}} = \arg \max_{\lambda \in (0,\infty)} p(\mathscr{D} \mid \lambda, k, \theta) p(\lambda \mid k, \theta)$

$$= \arg \max_{\lambda \in (0,\infty)} \ln p(\mathscr{D} \mid \lambda, k, \theta) + \ln p(\lambda \mid k, \theta)$$

$$= \frac{k - 1 + \sum_{i=1}^{n} x_i}{n + \frac{1}{\theta}} = 5 \text{ for dataset } \mathscr{D}$$

Question: What happens as the size of the dataset grows to infinity?

Summary

- We are usually interested in predicting the value of unseen data X_{n+1} based on **training data** $\mathscr{D} = \{x_1, \ldots, x_n\}$
- Just estimating mean, variance etc. are not good enough
- Instead, we will want to choose a **model** $\hat{f}$ from a **hypothesis space** $\mathscr{F}$
  - Where the data are generated according to some "true" model f^*
  - $\mathscr{F}$ is often **parametric**: its members identified by **parameter** values
- Two approaches to parameter estimation (in this lecture):

$$f_{\text{MAP}} = \arg\max_{f \in \mathscr{F}} p(f \mid \mathscr{D}) = \arg\max_{f \in \mathscr{F}} p(\mathscr{D} \mid f) p(f)$$

$$f_{\text{MLE}} = \arg\max_{f \in \mathscr{F}} p(\mathscr{D} \mid f) p(f)$$