6.867 Lecture Notes: Section 1: Introduction

Reading

- Lecture 1: Bishop 1.1–1.5
- Lecture 2: Bishop 2.1–2.4; 3.2

Contents

1 Intro

The main focus of machine learning is *making decisions or predictions based on data*. There are a number of other fields with significant overlap in technique, but difference in focus: . in economics and psychology, the goal is to discover underlying causal processes and in statistics it is to find a model that fits a data set well. In those fields, the end product is a model. In machine learning, we often fit models, but as a means to the end of making good predictions.

This story paraphrased from a post on 9/4/12 at `andrewgelman.com`

Generally, this is done in two stages:

1. **Learn** or **estimate** a model from the data
2. **Apply** the model to make predictions or answer queries

Problem of induction: Why do we think that previously seen data will help us predict the future? This is a serious philosophical problem of long standing. We will operationalize it by making assumptions, such as that all training data are IID (independent and identically distributed) and that queries will be drawn from the same distribution as the training data, or that the answer comes from a set of possible answers known in advance.

In general, we need to solve these two problems:

- **estimation:** When we have data that are noisy reflection of some underlying quantity of interest, we have to aggregate the data and make estimates or predictions about the quantity. How do we deal with the fact that, for example, the same treatment may end up with different results on different trials? How can we predict how well an estimate may compare to future results?
- **generalization:** How can we predict results of a situation or experiment that we have never encountered before in our data set?

We can characterize problems and their solutions using six characteristics, three of which characterize the problem and three of which characterize the solution:

1. **Problem class:** What is the nature of the training data and what kinds of queries will be made at testing time?
2. **Assumptions:** What do we know about the source of the data or the form of the solution?
3. **Evaluation criteria:** What is the goal of the system? How will the answers to individual queries be evaluated? How will the overall performance of the system be measured?
4. **Model type:** Will an intermediate model be made? What aspects of the data will be modeled? How will the model be used to make predictions?
5. **Model class:** What particular parametric class of models will be used? What criterion will we use to pick a particular model from the model class?
6. **Algorithm:** What computational process will be used to fit the model to the data and/or to make predictions?

Without making some assumptions about the nature of the process generating the data, we cannot perform generalization.

The introductory section of the course will lay out the spaces of these characteristics and illustrate them with simple (but not always easy!) examples. Then, in later sections, we will refer back to these spaces to characterize new problems and approaches that we introduce.

2 Problem class

There are many different *problem classes* in machine learning. They vary according to what kind of data is provided and what kind of conclusions are to be drawn from it. We will go through several of the standard problem classes here, and establish some notation and terminology.

2.1 Supervised learning

2.1.1 Classification

Training data $\mathcal{D}$ is in the form of a set of pairs $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$ where $x^{(i)}$ represents an object to be classified, most typically a $\mathcal{D}$-dimensional vector of real and/or discrete values, and $y^{(i)}$ is an element of a discrete set of values. The y values are sometimes called *target values*.

A classification problem is *binary* or *two-class* if $y^{(i)}$ is drawn from a set of two possible values; otherwise, it is called *multi-class*.

The goal in a classification problem is ultimately, given a new input value $x^{(n+1)}$, to predict the value of $y^{(n+1)}$.

Classification problems are a kind of *supervised learning*, because the desired output (or class) $y^{(i)}$ is specified for each of the training examples $x^{(i)}$.

We typically assume that the elements of $\mathcal{D}$ are independent and identically distributed according to some distribution $\Pr(X, Y)$.

Our textbook uses x_i and t_i instead of $x^{(i)}$ and $y^{(i)}$. We find that notation somewhat difficult to manage when $x^{(i)}$ is itself a vector and we need to talk about its elements. The notation we are using is standard in some other parts of the machine-learning literature.

2.1.2 Regression

Regression is like classification, except that $y^{(i)} \in \mathbb{R}^k$.

2.2 Unsupervised learning

2.2.1 Density estimation

Given samples $y^{(1)}, \ldots, y^{(n)} \in \mathbb{R}^D$ drawn IID from some distribution $\Pr(Y)$, the goal is to predict the probability $\Pr(y^{(n+1)})$ of an element drawn from the same distribution. Density estimation often plays a role as a "subroutine" in the overall learning method for supervised learning, as well.

This is a type of *unsupervised* learning, because it doesn't involve learning a function from inputs to outputs based on a set of input-output pairs.

2.2.2 Clustering

Given samples $x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^D$, the goal is to find a partition of the samples that groups together samples that are similar. There are many different objectives, depending on the definition of the similarity between samples and exactly what criterion is to be used (e.g., minimize the average distance between elements inside a cluster and maximize the average distance between elements across clusters). Other methods perform a "soft" clustering, in which samples may be assigned 0.9 membership in one cluster and 0.1 in another. Clustering is sometimes used as a step in density estimation, and sometimes to find useful structure in data. This is also unsupervised learning.

2.2.3 Dimensionality reduction

Given samples $x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^D$, the problem is to re-represent them as points in a d-dimensional space, where $d < D$. The goal is typically to retain information in the data set that will, e.g., allow elements of one class to be discriminated from another.

Standard dimensionality reduction is particularly useful for visualizing or understanding high-dimensional data. If the goal is ultimately to perform regression or classification on the data after the dimensionality is reduced, it is usually best to articulate an objective for the overall prediction problem rather than to first do dimensionality reduction without knowing which dimensions will be important for the prediction task.

2.3 Reinforcement learning

In reinforcement learning, the goal is to learn a mapping from input values x to output values y, but without a direct supervision signal to specify which output values y are best for a particular input. There is no training set specified *a priori*. Instead, the learning problem is framed as an agent interacting with an environment, in the following setting:

- The agent observes the current state, $x^{(0)}$.
- It selects an action, $y^{(0)}$.
- It receives a reward, $r^{(0)}$, which depends on $x^{(0)}$ and possibly $y^{(0)}$.
- The environment transitions probabilistically to a new state, $x^{(1)}$, with a distribution that depends only on $x^{(0)}$ and $y^{(0)}$.
- The agent observes the current state, $x^{(1)}$.
- ...

The goal is to find a policy π, mapping x to y, (that is, states to actions) such that some long-term sum or average of rewards r is maximized.

This setting is very different from either supervised learning or unsupervised learning, because the agent's action choices affect both its reward and its ability to observe the environment. It requires careful consideration of the long-term effects of actions, as well as all of the other issues that pertain to supervised learning.

2.4 Other settings

There are many other problem settings. Here are a few.

In *semi-supervised* learning, we have a supervised-learning training set, but there may be an additional set of $x^{(i)}$ values with no known $y^{(i)}$. These values can still be used to improve learning performance if they are drawn from $\Pr(X)$ that is the marginal of $\Pr(X, Y)$ that governs the rest of the data set.

In *active* learning, it is assumed to be expensive to acquire a label $y^{(i)}$ (imagine asking a human to read an x-ray image), so the learning algorithm can sequentially ask for particular inputs $x^{(i)}$ to be labeled, and must carefully select queries in order to learn as effectively as possible while minimizing the cost of labeling.

In *transfer* learning, there are multiple tasks, with data drawn from different, but related, distributions. The goal is for experience with previous tasks to apply to learning a current task in a way that requires decreased experience with the new task.

3 Assumptions

The kinds of assumptions that we can make about the data source or the solution include:

- The data are independent and identically distributed.
- The data are generated by a Markov chain.
- The process generating the data might be adversarial.
- The "true" model that is generating the data can be perfectly described by one of some particular set of hypotheses.

The effect of an assumption is often to reduce the "size" or "expressiveness" of the space of possible hypotheses and therefore reduce the amount of data required to reliably identify an appropriate hypothesis.

4 Evaluation criteria

Once we have specified a problem class, we need to say what makes an output or the answer to a query good, given the training data. We specify evaluation criteria at two levels: how an individual prediction is scored, and how the overall behavior of the system is scored.

The quality of predictions from a learned prediction rule is often expressed in terms of a *loss function*. A loss function $L(g, a)$ tells you how much you will be penalized for making a guess g when the answer is actually a. There are many possible loss functions. Here are some:

- **O-1 Loss** applies to predictions drawn from finite domains.

$$L(g, a) = \begin{cases} 0 & \text{if } g = a \\ 1 & \text{otherwise} \end{cases}$$

If the actual values are drawn from a continuous distribution, the probability they would ever be equal to some predicted g is 0 (except for some weird cases).

- **Squared loss**

$$L(g, a) = (g - a)^2$$

- **Linear loss**

$$L(g, a) = |g - a|$$

- **Asymmetric loss** Consider a situation in which you are trying to predict whether someone is having a heart attack. It might be much worse to predict "no" when the answer is really "yes", than the other way around.

$$L(g, a) = \begin{cases} 1 & \text{if } g = 1 \text{ and } a = 0 \\ 10 & \text{if } g = 0 \text{ and } a = 1 \\ 0 & \text{otherwise} \end{cases}$$

Any given prediction rule will usually be evaluated based on multiple predictions and the loss of each one. At this level, we might be interested in:

- Minimizing expected loss over all the predictions(also known as risk)
- Minimizing maximum loss: the loss of the worst prediction

- Minimizing or bounding regret: how much worse this predictor performs than the best one drawn from some class
- Characterizing asymptotic behavior: how well the predictor will perform in the limit of infinite training data
- Finding algorithms that are probably approximately correct: they probably generate a hypothesis that is right most of the time.

There is a theory of rational agency that argues that you should always select the action that *minimizes the expected loss*. This strategy will, for example, make you the most money in the long run, in a gambling setting. Expected loss is also sometimes called *risk* in the machine-learning literature, but that term means other things in economics or other parts of decision theory, so be careful...it's risky to use it. We will, most of the time, concentrate on this criterion.

Of course, there are other models for action selection and it's clear that people do not always (or maybe even often) select actions that follow this rule.

5 Model type

In this section, we'll examine the role of model-making in machine learning, and frequently refer to a *simple estimation* problem, which we can think of as regression, but without any input value. Assume that there is a single numeric value y to be predicted. We are given training examples $\mathcal{D} = \{y^{(1)}, \ldots, y^{(n)}\}$ and need to predict $y^{(n+1)}$.

5.1 No model

In some simple cases, we can generate answers to queries directly from the training data, without the construction of any intermediate model.

In our simple prediction problem, we might just decide to predict the mean or median of the $y^{(1)}, \ldots, y^{(n)}$ because it seems like it might be a good idea. In regression or classification, we might generate an answer to a new query by averaging answers to recent queries, as in the *nearest neighbor* method.

5.2 Prediction rule

It is more typical to use a two-step process:

1. "Fit" a model to the training data
2. Use the model directly to make predictions

In the *prediction rule* setting of regression or classification, the model will be some hypothesis or prediction rule $y = h(x; \theta)$ for some functional form h. The idea is that θ is a vector of one or more parameter values that will be determined by fitting the model to the training data and then be held fixed. Given a new $x^{(n+1)}$, we would then make the prediction $h(x^{(n+1)}; \theta)$.

The fitting process is usually articulated as an optimization problem: Find a value of θ that maximizes $\textbf{score}(\theta; \mathcal{D})$. As we will explore further in the next section, one optimal strategy, if we knew the actual underlying distribution $\Pr(X, Y)$ would be to predict the value of y that minimizes the *expected loss*, which is also known as the *risk*. If we don't have that actual underlying distribution, or even an estimate of it, we can take the approach of minimizing the *empirical risk*: that is, finding the prediction rule h that minimizes the average loss on our training data set. So, we would seek θ that maximizes

$$\textbf{score}(\theta) = -\frac{1}{n} \sum_{i=1}^{n} L(h(x^{(i)}; \theta), y^{(i)}) \quad .$$

We write $f(a; b)$ to describe a function that is usually applied to a single argument a, but is a member of a parametric family of functions, with the particular function determined by parameter value b. So, for example, we might write $h(x; p) = x^p$ to describe a function of a single argument that is parameterized by p.

In our simple estimation problem, the "hypothesis" would be a single real value, so the distinction between this case and the "no model" case is not as clear as it will be in the case of real regression or classification problems. However, we might still formulate the problem as one of fitting a model by seeking the value, $y_{\textbf{erm}}$ that minimizes empirical risk.

In the case of *squared* loss, we would select $\theta_{\textbf{erm}}$ to maximize

$$\textbf{score}(\theta) = -\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \theta)^2 \quad .$$

The value of θ that maximizes this criterion is

$$\theta_{\textbf{erm}} = \frac{1}{n}\sum_{i=1}^{n} y^{(i)} \quad .$$

Having selected hypothesis $h(\ ;\theta_{\textbf{erm}})$ to minimize the empirical risk on our data, $\theta_{\textbf{erm}}$ will be our predicted value. That is $h(\ ;\theta_{\textbf{erm}}) = \theta_{\textbf{erm}}$.

We will find that minimizing empirical risk is often not a good choice: it is possible to emphasize fitting the current data too strongly and end up with a hypothesis that does not generalize well when presented with new x values.

The prediction-rule approach is also sometimes called a *discriminative* approach, because, when applied to a classification problem, it attempts to directly learn a function that discriminates between the classes.

5.3 Probabilistic model

In the prediction rule approach, we learn a model that is used directly to select a predicted value. In the *probabilistic model* approach, we:

1. "Fit" a probabilistic model to the training data; then
2. Use decision-theoretic reasoning to combine the probabilistic model with a loss function to select the best prediction.

5.3.1 Fitting a probabilistic model

For a regression or classification problem we might choose to fit a *joint distribution* $\Pr(X, Y; \theta)$ or a *conditional distribution*, $\Pr(Y \mid X; \theta)$. The problem of fitting probabilistic models to data is treated at length in the statistics literature. There are many different criteria that one can use to fit a probabilistic model to data.

Probably the common objective is to find the model that maximizes the likelihood of the data from among some parametric class of models; this is known as the *maximum likelihood* model. It is found by maximizing the *data likelihood*,

$$\textbf{score}(\theta) = \prod_{i=1}^{n} \Pr(x^{(i)}, y^{(i)}; \theta) \quad .$$

This is called a *generative model*, because the model describes how the entire data set is generated. For classification problems, when y is an element of a discrete set of possible values, it is often easiest to learn a factored model, of the form

$$\Pr(X, Y; \theta) = \Pr(X \mid Y; \theta_1)\Pr(Y; \theta_2) \quad ,$$

where $\theta = (\theta_1, \theta_2)$.

Another approach is to fit a *discriminative model* by maximizing the *conditional likelihood*,

$$\textbf{score}(\theta) = \prod_{i=1}^{n} \Pr(y^{(i)} \mid x^{(i)}; \theta) \ .$$

We will explore the trade-offs between using generative and discriminative models in more detail in later sections. In general, we find that generative models can be learned somewhat more efficiently and are highly effective if their modeling assumptions match the process that generated the data. Discriminative models generally give the best performance on the prediction task in the limit of a large amount of training data, because they focus their modeling effort based on the problem of making the prediction and not on modeling the distribution over possible query points $x^{(n+1)}$.

We can't take this approach any further until we consider the particular class of models we are going to fit. We'll do some examples in section 6.

5.3.2 Decision theoretic prediction

Now, imagine that you have found some parameters θ, so that you have a probability distribution $\Pr(Y \mid X; \theta)$ or $\Pr(X, Y; \theta)$. How can you combine this with a loss function to make predictions?

The optimal prediction, in terms of minimizing expected loss, g^*, for our simple prediction problem, is

$$g^* = \arg\min_g \int_y \Pr(y; \theta) L(g, y) \ dy \ .$$

For a regression or classification problem with a conditional probabilistic model, it would be a function of a new input x:

$$g^*(x) = \arg\min_g \int_y \Pr(y \mid x; \theta) L(g, y) \ dy \ .$$

In general, we have to know both the form of the probabilistic model $\Pr(y \mid x; \theta)$ and the loss function $L(g, a)$. But we can get some insight into the optimal decision problem in the simple but very prevalent special case of squared error loss. In this case, for the simple prediction problem, we have:

$$g^* = \arg\min_p \int_y \Pr(y; \theta)(g - y)^2 \ dy \ .$$

We can optimize this by taking the derivative of the risk with respect to g, setting it to zero, and solving for g. In this case, we have

$$\begin{aligned}
\frac{d}{dg} \int_y \Pr(y; \theta)(g - y)^2 \ dy &= 0 \\
\int_y \Pr(y; \theta) 2(g - y) \ dy &= 0 \\
\int_y \Pr(y; \theta) g \ dy &= \int_y \Pr(y; \theta) y \ dy \\
g &= E[y; \theta]
\end{aligned}$$

Of course, for different choices of loss function, the risk-minimizing prediction will have a different form.

5.3.3 Benefits of using a probabilistic model

This approach allows us to separate the probability model from the loss function; so we could learn one model and use it in different decision-making contexts. Models that make probabilistic predictions can be easily combined using probabilistic inference: it is much harder to know how to combine multiple direct prediction rules.

5.4 Distribution over models

In the *distribution over models* approach, which we'll often call the *Bayesian approach*, we treat the model parameters θ as random variables, and use probabilistic inference to update a distribution over θ based on the observed data $\mathcal{D}$.

There are a lot of different methods and quantities within machine learning that are called "Bayesian foo." Sometimes people just use it to mean that there is an application of Bayes' rule going on somewhere. We will try to use it to refer to keeping distributions over models or hypotheses.

1. Begin with a prior distribution over possible probabilistic models of the data, $\Pr(\theta)$;
2. Perform Bayesian inference to combine the data with the prior distribution to find a posterior distribution over models of the data, $\Pr(\theta \mid \mathcal{D})$;
3. (Typically), integrate out the posterior distribution over models in the context of decision-theoretic reasoning, using a *loss function* to select the best prediction

In the Bayesian approach, we do not have a single model that we will use to make predictions: instead, we will integrate over possible models to select the minimum-expected-loss prediction. For the simple prediction problem this is:

$$g^* = \arg\min_g \int_\theta \int_y \Pr(y \mid \theta) \Pr(\theta \mid \mathcal{D}) L(g, y) \; dy d\theta \quad .$$

This integral can sometimes be difficult to evaluate. We can evaluate it approximately, using sampling techniques.

For the supervised case, when we are making a prediction for $x^{(n+1)}$, the best prediction is:

$$g^*(x^{(n+1)}) = \arg\min_g \int_\theta \int_y \Pr(y \mid x^{(n+1)}, \theta) \Pr(\theta \mid \mathcal{D}) L(g, y) \; dy d\theta \quad .$$

The Bayesian approach lets us maintain an explicit representation of our uncertainty about the model and take it into account when making predictions. Imagine a situation in which the most likely model would make a prediction g_1, but all of the other models, which are not that much less likely would predict g_2: in such situations g_2 is the choice that is more likely to be correct. Or, it might be that there is a prediction that is not the best, but has relatively small loss in all the models: that might be the choice that minimizes risk, even though it is not the best choice in any of the models.

The Bayesian approach can also provide elegant solutions to the problem of model selection.

6 Model class and parameter fitting

A model *class* $\mathcal{M}$ is a set of possible models, typically parameterized by a vector of parameters θ. What assumptions will we make about the form of the model? When solving a regression problem using a prediction-rule approach, we might try to find a linear function $h(x; w_1, w_0) = w_1 x + w_0$ that fits our data well. In this example, the parameter vector $\theta = (w_1, w_0)$.

For problem types such as discrimination and classification, there are huge numbers of model classes that have been considered...we'll spend much of this course exploring these model classes.

For now, we'll restrict our attention to model classes with a fixed, finite number of parameters. Later in the course, we will relax this assumption and look at "non-parametric" models.

How do we select a model class? In some cases, the machine-learning practitioner will have a good idea of what an appropriate model class is, and will specify it directly. In other cases, we may consider several model classes. In such situations, we are solving a *model selection* problem: model-selection is to pick a model class $\mathcal{M}$ from a (usually finite) set of possible model classes; *model fitting* is to pick a particular model in that class, specified by parameters θ.

6.1 Fitting maximum-likelihood probabilistic models

In the following we will consider some particular probabilistic model classes for the very simple prediction problem. Recall that for a maximum likelihood fit in the simple prediction problem, our goal is to find θ that maximizes

$$\textbf{score}(\theta) = \prod_{i=1}^{n} \Pr(y^{(i)}; \theta) \ .$$

6.1.1 Gaussian with fixed variance

One way to model our data is to assume it was drawn from a Gaussian distribution with a fixed variance, σ_0^2. Our model class, $\mathcal{M}$ is the class of univariate Gaussians with variance, σ_0^2, and the parameter θ will actually be μ, the mean of the Gaussian distribution. So,

$$\begin{aligned} \textbf{score}(\mu) &= \prod_{i=1}^{n} \Pr(y^{(i)}; \theta) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left\{-\frac{1}{2\sigma_0^2}(\mu - y^{(i)})^2\right\} \end{aligned}$$

The product is hard to deal with, but we observe that the value of μ that maximizes this quantity will also maximize its log, because the log is a monotonic function. So, we'll actually seek to maximize

This maneuver will be used a lot, to convert products to sums, throughout this subject.

$$\textbf{score}(\mu) = \sum_{i=1}^{n} \left(\log \frac{1}{\sqrt{2\pi\sigma_0^2}} - \frac{1}{2\sigma_0^2}(\mu - y^{(i)})^2 \right)$$

which, for a fixed σ_0, is the same as maximizing

$$\textbf{score}(\mu) = -\sum_{i=1}^{n} (\mu - y^{(i)})^2$$

Now, to find the value, μ_{ml} that maximizes the score, we take the derivative, set it to 0 and solve for μ.

$$\begin{aligned}
\frac{d}{d\mu_{ml}} \sum_{i=1}^{n} (\mu_{ml} - y^{(i)})^2 &= 0 \\
\sum_{i=1}^{n} 2(\mu_{ml} - yi) &= 0 \\
\sum_{i=1}^{n} \mu_{ml} &= \sum_{i=1}^{n} yi \\
\mu_{ml} &= \frac{1}{n} \sum_{i=1}^{n} yi
\end{aligned}$$

This is cool! We find that the maximum likelihood estimate for a quantity with a Gaussian distribution is the one that minimizes the mean squared error from the data.

This value of μ_{ml} will make the data as likely as possible.

And now, we see that the ML estimate is just the sample mean.

6.1.2 Gaussian

Now, what if σ is not determined? Then $\mathcal{M}$ is the set of all univariate Gaussian distributions and θ is $\langle \mu, \sigma^2 \rangle$.

It seems clear that a very large value of σ will make our data very unlikely; it is also true that a very small value of σ will make the data unlikely.

This problem can be approached in the same way as the previous problem, just solving a system of two equations to find the maximum likelihood estimate.

You could also let θ be $\langle \mu, \sigma \rangle$. You just have to be consistent in your approach.

6.1.3 Uniform over a continuous interval

If we know that our data are drawn uniformly from a finite interval, $[a, b]$, then $\mathcal{M}$ would be the set of all finite intervals in one dimension and θ could be $\langle a, b \rangle$, the lower and upper bounds of the interval.

In maximum likelihood estimation we want to find a_{ml}, b_{ml} to maximize the likelihood of $\mathcal{D}$. The likelihood of each point in the data set, given the parameters, will be $1/(b-a)$, so

$$\textbf{score}(\theta) = \prod_{i=1}^{n} \begin{cases} \frac{1}{b-a} & \text{if } y^{(i)} \in [a, b] \\ 0 & \text{otherwise} \end{cases}$$

It could also be the lower bound or the midpoint and the size.

Exercise: What are the maximum likelihood estimators of a and b? In what ways might you expect the ML estimates result in a poor fit to the underlying distribution from which the data was drawn?

6.1.4 Uniform over a finite set of points

Another model class $\mathcal{M}$ is a uniform distribution over k points on the real line. It would be parameterized by $\theta = \langle v_1, \ldots, v_k \rangle$. In this case,

$$\textbf{score}(\theta) = \prod_{i=1}^{n} \begin{cases} \frac{1}{k} & \text{if } y^{(i)} \in \theta \\ 0 & \text{otherwise} \end{cases}$$

What values of θ make the data most likely? $\theta_{ml} = D$. This is sometimes known as the *empirical distribution*.

We can see that, for values of k that are less than the number of distinct values in the training set, the likelihood of the data will be 0, even in the best model in the class. When k is equal to the size of the training set, the data likelihood ($\Pr(\mathcal{D} \mid \theta_{ml})$ will be maximized. As k increases past the size of the training data, $\Pr(\mathcal{D} \mid \theta_{ml})$ will decrease. This should

give us a hint that the model that maximizes likelihood is not necessarily the best one for characterizing future data.

6.1.5 Mixture of Gaussians

A Gaussian distribution might not represent a data set very well if the data set is not *unimodal*, that is, if it has multiple peaks of density.

A *mixture of Gaussians* model is a probability distribution whose PDF is the normalized sum of multiple Gaussian PDFs, which intuitively describes a process of generating data in which one first selects which *mixture component* the data item is to be drawn from and then selects a data item from that component. So, if we were measuring heights of students in school, and we believed that the distribution in each classroom was Gaussian, but that there was a difference in the means and/or variances between classrooms, we might describe that distribution using a mixture of Gaussians.

Formally, a mixture of k Gaussians in one dimension is parameterized by

$$\theta = (p_1, \ldots, p_k, \mu_1, \ldots, \mu_k, \sigma_1, \ldots, \sigma_k),$$

with the constraints that $\sum_{i=1}^{k} p_k = 1$ and $\sigma_i > 0$ for all i. Then, the likelihood of a sample y is given by

$$\Pr(y; \theta) = \sum_{j=1}^{k} p_j \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left\{ -\frac{1}{2\sigma_j^2}(\mu_j - y)^2 \right\}$$

Finding the maximum likelihood parameter estimates for this model is difficult. Taking the derivative and setting to 0 gives something that can't be solved analytically. In future homework assignments, we will explore local optimization methods for this problem.

6.1.6 Maximum likelihood and binary data

It's also interesting to consider a data set in which $y \in \{0, 1\}$. If we assume the $y^{(i)}$ are drawn from a Bernoulli distribution,

$$Y \sim \text{Bernoulli}(p) \ ,$$

then our only parameter θ is the probability p and

$$\textbf{score}(p) = \Pr(D; p) = \prod_{i=1}^{n} \begin{cases} p & \text{if } y^{(i)} = 1 \\ 1 - p & \text{otherwise} \end{cases}$$

There is a sneaky way to write this that makes things much easier:

$$\textbf{score}(p) = \prod_{i=1}^{n} p^{y^{(i)}} (1 - p)^{1 - y^{(i)}} \ .$$

We begin by taking the log, so

$$\arg\max_p \textbf{score}(p) \quad = \quad \arg\max_p \sum_{i=1}^{n} y^{(i)} \log p + (1 - y^{(i)}) \log(1 - p)$$

Now we can take the derivative wrt p and set to 0:

$$\begin{aligned}\sum_{i=1}^{n}\left(\frac{y^{(i)}}{p_{\mathbf{ml}}}-\frac{1-y^{(i)}}{1-p_{\mathbf{ml}}}\right) &= 0\\ \sum_{i=1}^{n}(y^{(i)}-p_{\mathbf{ml}}) &= 0\\ p_{\mathbf{ml}}=\frac{1}{n}\sum_{i=1}^{n}y^{(i)}\end{aligned}$$

So, the maximum likelihood estimate of p is the fraction of elements in the data set that have value 1.

6.2 Model selection over classes of probabilistic models

Given a model class, it seems like it's reasonable to select the model that makes the observed data as likely as possible. Why? Because we think that model will do a good job of describing the distribution of future data that we will see. But this connection is not as clear as it might seem. In sections 6.1.4 and 6.1.5 we explored classes of models whose complexity varied as a function of the number of components k.

The empirical distribution assigns all the probability mass in the distribution to the data points we've seen so far. We are unhappy with such a model, though, because it *over-fits* our data: it seems highly unlikely to be a good description of data that we will see in the future, drawn from the same distribution as D. We can see a way to deal with overfitting using Bayesian reasoning in the next section. But we can also approach it in this setting.

6.2.1 Bias-Variance decomposition

We start with some preliminaries from *frequentist* statistics, which is one of the two main schools of thought about the foundations of probability. The idea is that probabilities are long-run frequencies of events: the probability that this coin will come up heads is the long-term limit of the proportion of times it comes up heads.

When we do statistics in this framework, we can estimate underlying quantities, such as the probability that the coin will come up heads. An *estimator* is a function from a data set to an estimate of some parameter that characterizes the distribution of that data set. An *estimate* is the output of an estimator.

When we make an estimate, $\widehat{\theta}$, based on a finite amount of data $\mathcal{D}$, we don't expect it to be exactly right. We consider the uncertainty we have about our estimate by

- Imagining that we know the value of the parameter of interest, θ;
- Thinking of all the different data sets of the same size as the one that we have, that might have arisen if θ were the true parameter;
- Considering the distribution of estimates $\widehat{\theta}$ that would have arisen from those alternative data sets.

Our estimate $\widehat{\theta}$ is a function of the data set $\mathcal{D}$. So, given a distribution over $\mathcal{D}$, there is an induced distribution over $\widehat{\theta}$. Now, we can say things about the distribution of $\widehat{\theta}$, where, we emphasize again, there is a fixed underlying θ and the uncertainty is with respect to the data set.

With this understanding, we can define two useful ways to measure the quality of an estimator: its *bias* and its *variance*.

We will assume we are using some particular estimator f_θ that maps data sets into parameter estimates. Let:

- θ be the fixed unknown true parameter defining $\Pr(Y;\theta)$;
- D be a data set with n samples, drawn iid from the distribution $\Pr(Y;\theta)$. Data set D is a random variable;
- $\widehat{\theta}_n = f_\theta(D)$: be the estimate of θ after n samples; this is a random variable that depends on the training data (which is, itself, a random variable).

Now, we can define the *bias* of the estimate as:

$$\textbf{bias}(\widehat{\theta}_n) = E_D(\widehat{\theta}_n) - \theta$$

The expectation is the average of $\widehat{\theta}_n$ over possible data sets of size n, drawn from distribution $\Pr(Y;\theta)$. The bias is a measure of the systematic error between the estimated and the true values. An estimator for which the bias is 0 is called *unbiased*. An estimator for which the bias tends to 0 as n tends to infinity is called *asymptotically unbiased*.

We define the *variance* of the estimate as:

$$\textbf{var}(\widehat{\theta}_n) = E_D[(\widehat{\theta}_n - E_D(\widehat{\theta}_n))^2] \ .$$

This is the usual definition of the variance of a random variable. It measures how different, on average, the actual estimator value is from the mean of the distribution of estimator values. Another way to think of it is that it measures how much the variation in the data set influences the resulting estimate.

Ultimately, we are interested in measuring the quality of our whole learning algorithm, which is a combination of estimating the underlying probability distribution and then using it to make predictions. Let's assume that:

- We will use some estimator (we don't have to make a commitment to which one) f_θ to compute an estimated $\widehat{\theta}$ from data $\mathcal{D}$.
- We will make a guess g that minimizes the expected squared loss.

There is a bias-variance trade-off in play for other loss functions, but the particular form in the derivation here applies only to squared loss.

So, our *prediction process*, π, is defined by:

$$\pi(\mathcal{D}) = \arg\min_g \int_a (g-a)^2 \Pr(a; f_\theta(\mathcal{D}))da \ .$$

We have already shown that the optimizing prediction with squared loss is the expected value, so:

$$\pi(\mathcal{D}) = E[a; f_\theta(\mathcal{D})] \ .$$

Notice that we will be making our predictions using the distributional parameters we estimated from the data, not the true ones (which we don't have access to). We have now characterized what prediction we will make, given data set $\mathcal{D}$.

Now, we are interested in knowing how good that prediction is, with respect to the *actual* distribution $\Pr(a;\theta)$. We will develop a way of thinking of the expected loss of our prediction which gives us theoretical insight into the choices we might make for estimators.

The actual risk of making guess g is

$$R(g) = \int_a (g-a)^2 \Pr(a;\theta)da \ .$$

We can rewrite this as follows:

Exercise: Prove this for yourself, to be sure you know what all these quantities mean. The critical step is to rewrite $(a-g)^2$ as $((g - E[a;\theta]) + (E[a;\theta] - a))^2$.

$$R(g) = (g - E[a;\theta])^2 + \int_a (E[a;\theta] - a)^2 \Pr(a;\theta) da \ .$$

Our goal in analyzing the effectiveness of the estimator is not to know how good a particular guess is, but now good a *way of making guesses* that is, a prediction process, is. We will be interested in the *expected risk*, now with the expectation taken with respect to data sets $\mathcal{D}$ drawn according to $\Pr(a;\theta)$.

$$\begin{aligned} ER(\pi) &= E_{\mathcal{D}}[R(\pi(\mathcal{D}))] & (1) \\ &= E_{\mathcal{D}}[(\pi(\mathcal{D}) - E[a;\theta])^2] + \int_a (E[a;\theta] - a)^2 \Pr(a;\theta) da \ . & (2) \end{aligned}$$

The second term is independent of our prediction rule, and characterizes the variability in the *actual* values, so we don't have too much more to say about it. We can expand the first term using the same trick as before:

$$\begin{aligned} E_{\mathcal{D}}[(\pi(\mathcal{D}) - E[a;\theta])^2] &= E_{\mathcal{D}}[(\pi(\mathcal{D}) - E_{\mathcal{D}}[\pi(\mathcal{D})] + E_{\mathcal{D}}[\pi(\mathcal{D})] - E[a;\theta])^2] \\ &= (E_{\mathcal{D}}[\pi(\mathcal{D}) - E[a;\theta]])^2 + E_{\mathcal{D}}[(\pi(\mathcal{D}) - E_{\mathcal{D}}[\pi(\mathcal{D})])^2] \end{aligned}$$

The first term of this expression is $\textbf{bias}(\pi(\mathcal{D}))^2$ and the second is $\textbf{var}(\pi(\mathcal{D}))$.

So, finally, we can write the expected risk of the prediction process π as:

$$ER(\pi) = \underbrace{\textbf{bias}(\pi(\mathcal{D}))^2}_{\text{systematic error}} + \underbrace{\textbf{var}(\pi(\mathcal{D}))}_{\text{sensitivity to data variability}} + \underbrace{\int_a (E[a;\theta] - a)^2 \Pr(a;\theta) da}_{\text{noise in true distribution}} \ .$$

This is sometimes known as the *bias-variance decomposition theorem*.

This is a critical idea in machine learning: generally, more complex models have lower bias (they can represent a wide variety of hypotheses) but higher variance (small changes in the data set can generate big changes in the estimate). The variance of an estimator decreases with n; so, with more data you can (and should) use a more complex model.

Here is an example of the bias and variance of two simple estimators. Imagine that we have y_i, drawn from a Bernoulli distribution,

$$Y \sim \text{Bernoulli}(p) \ .$$

Because they are exchangeable (the order doesn't matter) we really need to keep track of the number of 1's, h, and the number of 0's, t, in the data set. Then, the maximum likelihood estimator is

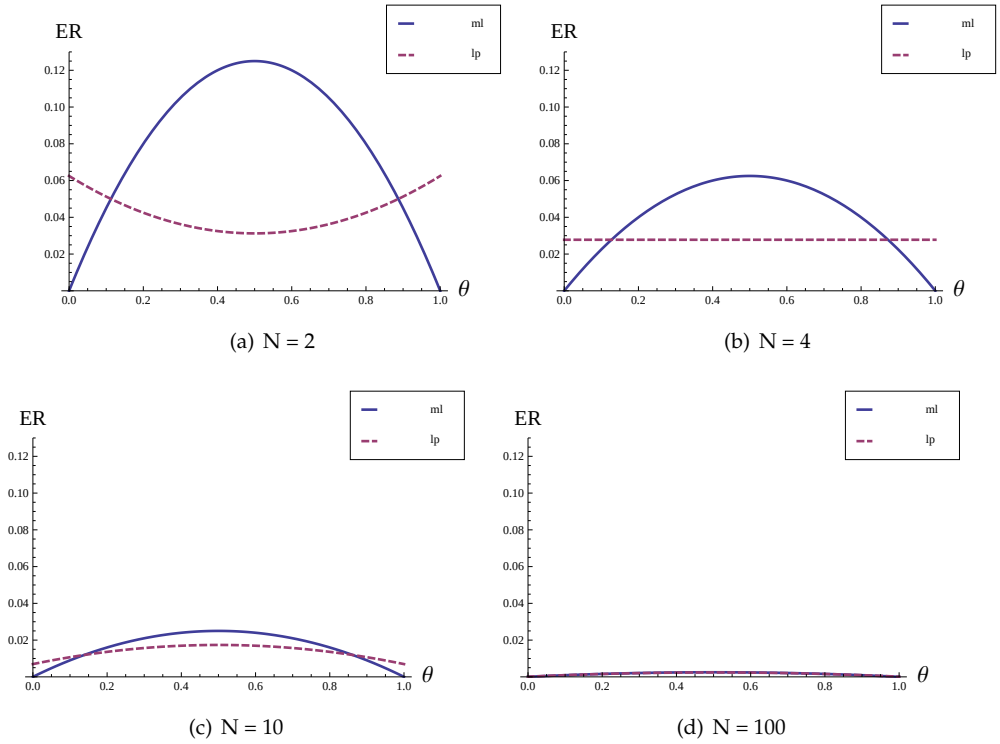
$$\widehat{\theta}_{ml} = \frac{h}{h+t} \ .$$

A frequently used alternative estimator uses the *Laplace correction:*

$$\widehat{\theta}_{lp} = \frac{h+1}{h+t+2} \ .$$

The reason for using the Laplace correction is illustrated by a small data set: $\{0, 0\}$. In this case, $\widehat{\theta}_{ml} = 0$. On one hand, this seems reasonable, but on the other hand, it seems a very strong conclusion to draw from only two data points. For that same data, $\widehat{\theta}_{lp} = 0.25$. In fact, the maximum likelihood estimator is unbiased, but on small data sets, it has high variance. The estimator with the Laplace correction is biased, but has lower variance.

In this very simple case, we can actually compute the bias and variance, by taking an expectation over all possible data sets, since all that matters about a data set is the number of 1's it contains. Here are plots of the bias, variance, and MSE of the two estimators (ML in blue and LP in purple) as a function of n, the size of the data set. We can see that, for



(a) N = 2 (b) N = 4

(c) N = 10 (d) N = 100

Figure 1: Expected risk as a function of θ, for data sets of size 2, 4, 10, and 100. Maximum likelihood is blue solid line; Laplace correction is purple dashed line.

small data sets, ML has significantly higher bias and MSE, but the difference disappears in data sets of size about 20 or higher. Asymptotically, the estimators are equivalent.

Intuitively, the increased bias and decreased variance in the Laplace estimator comes from the fact that data has a smaller effect on the resulting estimate in the Laplace case than in the ML case.

6.2.2 Regularization and model selection

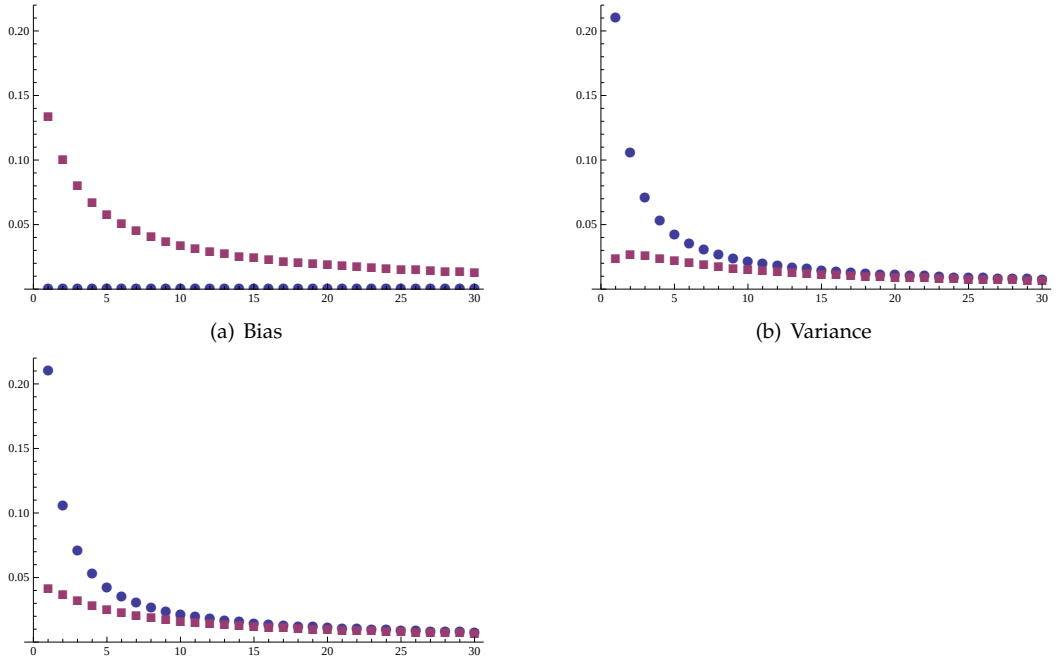
We were able to compute the MSE in the previous case because the data sets were very small and simple, and only with respect to a particular assumed distribution for the data. In general, it is intractable to do so, so we must turn to other methods for both selecting parameters of a given model class (*regularization*) and choosing between model classes (*model selection*).

We will discuss these strategies in lots of detail for particular model classes in the rest of the course, but just give a short introduction here.

Regularization is a method for decreasing variance by preventing parameter estimates from being extremely effected by data in the phenomenon is known as overfitting. Simply maximizing the likelihood of the data is tends to result in solutions that are highly dependent on the training data, and hence high variance.

Some methods of preventing overfitting include

- Using the Laplace correction when estimating a probability; or, even more extremely, using something like $(h + 5)/(h + t + 10)$. This prevents the ratio from taking on extreme values, until the number of actual data points is very large.
- Adding $(c/n)I$, where I is the identity matrix, to the estimate of the covariance matrix



(a) Bias

(b) Variance

(c) Expected Risk

Figure 2: Bias, variance, and expected risk as a function of training set size, with a true underlying $\theta = 0.3$. Blue circles are maximum likelihood estimate and purple squares are estimate with the Laplace correction.

for a multivariate Gaussian. This keeps the covariance from becoming extremely non-circular or extremely rotated, until the number of actual data points is very large.

- In curve-fitting problems, keeping the magnitudes of the parameters from getting too large.

In many cases, we can construct regularized estimators that have the form of maximizing a *penalized likelihood*:

$$\widehat{\theta} = \arg\max_{\theta} \log \Pr(\mathcal{D}; \theta) - \alpha |\theta|_c \quad ,$$

for some complexity measure of the parameters.

Model selection is the problem of deciding, for example, how many mixture components to use in a mixture of Gaussians, or whether to use a line or a 3rd-order polynomial to fit a function to 1-D inputs.

Ultimately, the best value of the regularization parameter α or the most appropriate model class is governed by generalization properties: we are using these mechanisms to help generate a predictor that will perform well on data we have yet to see. One thing we can do is use a proxy: we can divide our training data into two parts, run several estimators on one part of the data, and then see how well it performs on the other part of the data (often called a *hold-out set* or *validation set*). If the total amount of data available is too small to have a large validation set, then it may be appropriate to do k-fold *cross validation*, in which we divide the training set into k subsets, and then repeatedly:

- Train a model on $k - 1$ of the subsets;
- Test it on the remaining subset

This process will yield k estimates of the *testing error* of the data. It can be applied to different model classes or to estimators with different values of a complexity penalty, to select the model class or penalization function that generates the best generalization performance in the domain, and for the amount of data available. Once the best model class or complexity penalty is selected, it is typical to run the estimator one more time, for that model class, on all the data, and use the resulting model.

The cross-validation approach is widely used, but can be computationally very expensive and it still has problems with generating biased estimates.

HTF 7.10 has a good discussion of this.

There are other strategies for penalizing model complexity based on an assessment of the number of free parameters of the model. These include the Akaike information criterion, which attempts to maximize

$$\log \Pr(\mathcal{D}; \theta) - |M| \quad .$$

where $|M|$ is the number of parameters in the model; the Bayesian information criterion is similar in form, but subtracts $(1/2) \log n |M|$ from the log likelihood. In general, counting parameters is a blunt instrument for measuring the complexity of a model.

See HTF 7.5–7.6 for a more nuanced discussion of the effective number of parameters.

6.3 Bayesian inference from prior to posterior over models

The Bayesian approach to statistics is very different from the frequentist approach at the foundational and philosophical levels, but most of the mathematical manipulations are shared. The philosophical difference is that we treat the model M and its parameters θ that generated the data as a random variable, and use probability to model our uncertainty about it. Probability is subjective: you and I could assign a different probabilities to the same event, and not be wrong. The data are not treated as a random variable: they are the observations that we have in hand, that give us evidence about the underlying model.

We will use a prior distribution $\Pr(\theta)$ to encode what we believe, before seeing any data, about the distribution over processes that might have generated our data. In many cases, we can put a uniform distribution over the parameters, indicating no prior preference among models; or, we can adopt a strong prior when we know something about the model in advance.

The *posterior distribution* is $\Pr(\theta \mid \mathcal{D})$, which is the result of conditioning on the data we have actually observed. If we need a point estimate of θ we sometimes compute the *maximum a posteriori probability* (MAP) estimate:

$$\arg\max_{\theta} \Pr(\theta \mid \mathcal{D}) \quad .$$

Using Bayes' rule, we can see a relationship between the MAP and ML estimates:

$$\begin{aligned} \arg\max_{\theta} \Pr(\theta \mid \mathcal{D}) &= \arg\max_{\theta} \frac{\Pr(\mathcal{D} \mid \theta) \Pr(\theta)}{\Pr(\mathcal{D})} \\ &= \arg\max_{\theta} \Pr(\mathcal{D} \mid \theta) \Pr(\theta) \end{aligned}$$

So, if $\Pr(\theta)$ is uniform, then the MAP estimate is the same as the ML estimate.

In the following, we will show three examples of Bayesian estimation for different combinations of prior distributions and data distributions. In each case, we must consider:

- A distribution $\Pr(\mathcal{D} \mid \theta)$ that governs the generation of data, characterized by parameters θ.
- A prior distribution $\Pr(\theta; \eta)$ which is characterized by some fixed, known parameters η.

Bayesian inference is at its most beautiful and convenient when $\Pr(\mathcal{D} \mid \theta)$ and $\Pr(\theta; \eta)$ have a relationship called *conjugacy*. It means that if the PDF of the prior has some parametric form $f(\eta)$, then the PDF of the posterior has the same parametric form, with new parameter values that depend on the parameters of the prior and on the data; that is the PDF of the posterior has the form $f(g(\eta, \mathcal{D}))$ and we write

$$\begin{aligned} Y \mid \Theta &\sim \text{Family1}(\Theta) \\ \Theta &\sim \text{Family2}(\eta) \\ \Theta \mid \mathcal{D} &\sim \text{Family2}(g(\eta, \mathcal{D})) \end{aligned}$$

Here Family1 and Family2 are classes of distributions, such as the normal or uniform. In each of the cases below, we describe which families we are considering and determine the function g that maps prior parameter values and data into posterior parameter values.

6.3.1 Bernoulli/Bernoulli

Imagine that there is a hidden binary variable θ that governs a distribution of binary observations Y. If $\theta = 0$, then $Y = 1$ with probability ϕ_0; otherwise, if $\theta = 1$, then $Y = 1$ with probability ϕ_1. Furthermore, we have a prior distribution over whether θ is 0 or 1: $\Pr(\theta = 1) = \eta_0$.

Here is a formal description of the model:

- $\theta \in \{0, 1\}$
- $y^{(i)} \in \{0, 1\}$
- $y^{(i)} \mid \theta = k \sim \text{Bernoulli}(\phi_k)$
- $\theta \sim \text{Bernoulli}(\eta_0)$

A situation fitting this model might be that we have a coin, and we know it came either from factory 0 or factory 1. Factory 0 produces coins that come up heads with probability ϕ_0. Factory 1 produces coins that come up heads with probability ϕ_1. We have a coin and we're not sure which factory it came from. But we believe, with probability η_0 that it came from factory 1.

The parameter we want to estimate is θ, but rather than getting a point estimate for θ, we will seek a distribution over values of θ.

Exercise: Without seeing any flips of this coin, what is $\Pr(Y = \text{heads})$?

If we flip the coin and get outcome y, what can we infer about where the coin was manufactured?

$$\begin{aligned} \Pr(\theta = 1 \mid y) &= \frac{\Pr(y \mid \theta = 1)\Pr(\theta = 1)}{\Pr(y)} \\ &= \frac{\phi_1^y (1 - \phi_1)^{1-y} \eta_0}{\phi_0^y (1 - \phi_0)^{1-y} (1 - \eta_0) + \phi_1^y (1 - \phi_1)^{1-y} \eta_0} \\ \eta_1 &= g(\eta_0, y) \end{aligned}$$

We'll use that exponentiation trick again, so that we can write it in a generic form, though it doesn't always come out so cleanly.

So, η_1 are the parameters of the posterior.

So, for example, let:

- $\phi_0 = 1$: coins from factory 0 have heads on both sides (never come up tails)
- $\phi_1 = 0.5$: coins from factory 1 are fair
- $\eta_0 = 0.7$: we think with probability 0.7 that this coin came from factory 1

Now we flip. First flip is heads.

$$\eta_1 = \frac{0.5 \cdot 0.7}{0.5 \cdot 0.7 + 1 \cdot 0.3} \approx 0.54$$

What if flip 2 is heads? The update is the same, but starting from the posterior we had before.

$$\eta_2 = \frac{0.5 \cdot 0.54}{0.5 \cdot 0.54 + 1 \cdot 0.46} \approx 0.46$$

Instead, what if flip 2 is tails? We know for sure where this coin came from!

$$\eta_2 = \frac{0.5 \cdot 0.54}{0.5 \cdot 0.54 + 0 \cdot 0.46} = 1 \ .$$

Exercise: Prove to yourself that the order of the sequence of observations doesn't matter.

6.3.2 Beta/Binomial

Here is a more interesting case. Imagine that, again, we have a coin that we don't understand very well, but in this case, instead of having only two possible underlying θ values, with associated probabilities, now θ ranges over continuous probability values from 0 to 1. We will use a *Beta* distribution as a prior on θ; we will find that it is conjugate with Bernoulli observations, yielding a Beta as a posterior as well.

- $\theta \in [0, 1]$
- $y^{(i)} \in \{0, 1\}$
- $y^{(i)} \mid \theta \sim \text{Bernoulli}(\theta)$
- $\theta \sim \text{Beta}(\alpha, \beta)$

Beta distribution
Parameters are two positive numbers, α and β. The PDF is

$$f(y; \alpha, \beta) = \begin{cases} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} y^{\alpha-1}(1-y)^{\beta-1} & \text{if } 0 < y < 1 \\ 0 & \text{otherwise} \end{cases}$$

The gamma function is a generalization of factorial. If n is an integer, then

$$\Gamma(n) = (n-1)! \ .$$

Here are some important facts about the beta distribution. If $X \sim \text{Beta}(\alpha, \beta)$, then

$$E(X) = \frac{\alpha}{\alpha + \beta}$$

If $\alpha, \beta > 1$, then

$$\text{Mode}(X) = \frac{\alpha - 1}{\alpha + \beta - 2}$$

Exercise: What is the predictive distribution, $\Pr(X = 1; \alpha, \beta)$?

Let's go ahead and see how to do an evidence update, conditioned seeing m heads and l tails (it would work out the same way if we did it one at a time, but this is pretty cool).

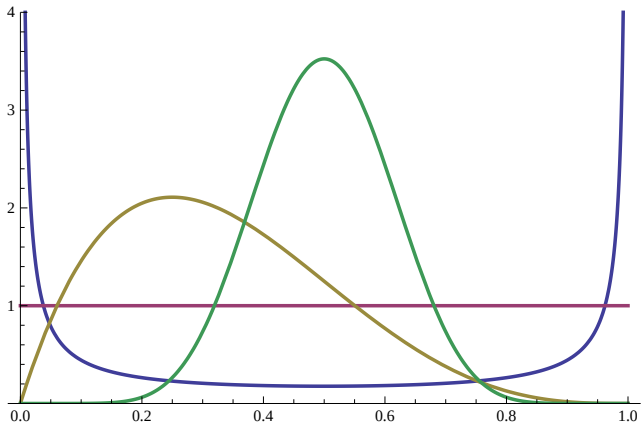


Figure 3: Several examples of the Beta distribution: Beta(.1, .1) (goes up at the ends), Beta(1, 1) (uniform), Beta(2, 4) (asymmetric), Beta(10, 10) (peaked at 0.5).

Remembering that the probability, under the Binomial distribution, of seeing m heads and l tails is

$$\Pr(\mathcal{D} \mid \theta) = \binom{l+m}{m} \theta^m (1-\theta)^l$$

Now, we can compute the posterior

$$\begin{aligned} \Pr(\theta \mid \mathcal{D}) &\propto \Pr(\mathcal{D} \mid \theta) \Pr(\theta) \\ &\propto \theta^m (1-\theta)^l \theta^{\alpha-1} (1-\theta)^{\beta-1} \\ &\propto \theta^{m+\alpha-1} (1-\theta)^{l+\beta-1} \end{aligned}$$

So,

$$\theta \mid \mathcal{D} \sim \text{Beta}(m+\alpha, l+\beta) \ .$$

In order to make this be a real probability density, it has to integrate to 1, which means we need to divide through by normalizing constant $Z = \int_\theta \theta^{m+\alpha-1}(1-\theta)^{l+\beta-1}$.

Sometimes we call α, β *pseudocounts* because they act just like actual observations.

Examples

Let's compare two inference scenarios. Assume that we have a coin with unknown θ of coming up heads. We get $\mathcal{D}$ consisting of 2 heads and 4 tails. We need to predict the next result, with the following loss function:

$$L(g, a) = \begin{cases} l_{10} & \text{if } g = 1 \text{ and } a = 0 \\ l_{01} & \text{if } g = 0 \text{ and } a = 1 \\ 0 & \text{otherwise} \end{cases}$$

In the first case, we begin with a "uniform" prior:

$$\theta \sim \text{Beta}(1, 1) \ .$$

So,

$$\theta \mid \mathcal{D} \sim \text{Beta}(3, 5) \ .$$

The risk of predicting heads is $l_{10} \cdot (1 - \Pr(y^{(n+1)} = 1 \mid \mathcal{D})) = .625 \cdot l_{10}$. The risk of predicting tails is $l_{01} \cdot \Pr(y^{(n+1)} = 1 \mid \mathcal{D}) = .375 \cdot l_{01}$. So, we should predict heads if $.625 \cdot l_{10} < .375 \cdot l_{01}$.

If, instead, we had a prior of $\text{Beta}(0.1, 0.1)$, the posterior would be $\text{Beta}(2.1, 4.1)$, and we should predict heads if $.661 \cdot l_{10} < .339 \cdot l_{01}$.

The prior affects the resulting decision-making.

6.3.3 Gaussian with fixed variance

Now, let's consider a case where the samples in our data set are real values, and that they are drawn from a Gaussian distribution with known variance, σ_D^2, but unknown mean. We will also assume a prior distribution on the mean, which is a Gaussian with parameters μ_0, σ_0^2. So:

- $\theta \in \mathbb{R}$
- $y^{(i)} \in \mathbb{R}$
- $y^{(i)} \mid \theta \sim \text{Normal}(\theta, \sigma_D^2)$
- $\theta \sim \text{Normal}(\mu_0, \sigma_0^2)$

Assume we make a single observation $y^{(1)}$. What is the posterior?

$$\begin{aligned}
\Pr(\theta \mid y^{(1)}) &\propto \Pr(y^{(1)} \mid \theta; \sigma_D^2) \Pr(\theta; \mu_0, \sigma_0^2) \\
&\propto \exp\left(-\frac{(y^{(1)} - \theta)^2}{2\sigma_D^2}\right) \exp\left(-\frac{(\theta - \mu_0)^2}{2\sigma_0^2}\right) \\
&\propto \exp\left(-\frac{(\theta - \mu_1)^2}{2\sigma_1^2}\right)
\end{aligned}$$

where

$$\mu_1 = \frac{\sigma_D^2 \mu_0 + \sigma_0^2 y^{(1)}}{\sigma_D^2 + \sigma_0^2} \ ,$$

which is a weighted average of the prior mean and the data, and

$$\sigma_1^2 = \frac{\sigma_0^2 \sigma_D^2}{\sigma_0^2 + \sigma_D^2} \ .$$

Note that the new variance is less than the prior variance and less than the variance of the observation. So, we can conclude that

$$\theta \mid y^{(1)} \sim \text{Normal}(\mu_1, \sigma_1) \ .$$

When $\mathcal{D}$ contains N observations, whose values have mean $\mu_{\mathbf{ml}}$, then we have

$$\theta \mid \mathcal{D} \sim \text{Normal}(\mu_n, \sigma_n) \ ,$$

where

$$\mu_N = \frac{\mu_0 \sigma_D^2 + N \mu_{\mathbf{ml}} \sigma_0^2}{N\sigma_0^2 + \sigma_D^2}$$

and

$$\sigma_N^2 = \frac{N\sigma_0^2 \sigma_D^2}{N\sigma_0^2 + \sigma_D^2} \ .$$

In this case, we have incorporated a whole batch of data at once. You should be convinced that this is equivalent to incorporating each data element at a time, using the posterior from the previous step as the prior for the next, and that the order of presentation of the data doesn't matter.

Another important question in this case is, what is the *posterior predictive distribution?*? It is

$$\begin{aligned}\Pr(y^{(n+1)} \mid \mathcal{D}) &= \int_\theta \Pr(y^{(n+1)} \mid \theta) \Pr(\theta \mid \mathcal{D}) d\theta \\ &= \int_\theta \Pr(y^{(n+1)} \mid \theta) \Pr(\theta \mid \mu_n, \sigma_n) d\theta \\ &= \mathcal{N}(y^{(n+1)}; \mu_n, \sigma_n^2 + \sigma_D^2)\end{aligned}$$

One way to derive this is with a lot of hassling with the integral and the quadratic stuff in the exponent. Another (thanks to a paper by Murphy) is to make the following observations:

- $\theta \mid \mathcal{D} \sim \text{Normal}(\mu_n, \sigma_n^2)$
- $y^{(n+1)} \mid \theta \sim \text{Normal}(\theta, \sigma_D^2)$
- $y^{(n+1)} - \theta \sim \text{Normal}(0, \sigma_D^2)$

Now, we can see $y^{(n+1)} \mid \mathcal{D}$ as a sum of $(y^{(n+1)} - \theta) \mid \mathcal{D}$ and $\theta \mid \mathcal{D}$. The sum of two Gaussian random variables is also a Gaussian, where the new mean is the sum of the means and the new variance is the sum of the variances. So,

$$y^{(n+1)} \mid \mathcal{D} \sim \text{Normal}(\mu_1, \sigma_D^2 + \sigma_1^2)$$

6.3.4 Finding conjugate families

Finding a prior/observation distribution pair that is conjugate can be kind of tricky. For distributions in the *exponential family*, it is reasonably straightforward to find appropriate pairs, because we end up with products of densities which have linear functions in the exponent; such products are also from that same class.

For example, if you have a Gaussian with uncertainty over both the mean and standard deviation, you can construct a prior on the joint distribution of those parameters that has the same parametric form as the data likelihood. Bishop works through this in section 2.3.6.

6.4 Bayesian model selection

In the Bayesian framework, we can put a prior on anything! Could be on the parameters of a polynomial fit to data, or even on class of models. But, even without a strong prior on the model class, taking the Bayesian view of parameter estimation actually helps control complexity. Here's a sketch of how it works, and we'll see it in context a couple of times later in the course.

- You have a choice between two different model classes, one of which is drawn from a much larger space than the other.
- Conditioned on the data you have distributions over the parameters for each model. Generally, the distribution over the parameters from the smaller model class will be tighter.
- If you then evaluate $\Pr(\mathcal{D} \mid M_i)$, which is the *marginal likelihood* assigned to the data (often also called the *evidence*), integrating out the model parameters according to the posterior distribution, you will find that the model class with bigger variance in the parameter distributions will assign lower likelihood to the data.

- As you get more data, if the smaller model class is actually a poor fit, then the larger model class will still have larger variance in its parameter estimates, but the likelihood it assigns to the data will be larger.
- As a result, you can either just pick the best model class and use that model...or be really Bayesian and average over all your model classes to make a prediction.

Bishop, in section 3.4, gives a nice discussion of this.

7 Algorithm

How can we find the best model, according to our criterion, given the data that we have?

Note that we will often have to approximate. There are two reasons. First, there may not be any closed-form representation of the result, so we have to project it into a different, hopefully close, form. Second, it might be intractable to find the optimum.

Homework 1 will explore simple optimization strategies applied to several machine learning problems.

8 Recap

We have shown how to think about learning problems and solutions according to these characteristics.

1. **Problem class:** What is the nature of the training data and what kinds of queries will be made at testing time?
2. **Assumptions:** What do we know about the source of the data or the form of the solution?
3. **Evaluation criteria:** What is the goal of the system? How will the answers to individual queries be evaluated? How will the overall performance of the system be measured?
4. **Model type:** Will an intermediate model be made? What aspects of the data will be modeled? How will the model be used to make predictions?
5. **Model class:** What particular parametric class of models will be used? What criterion will we use to pick a particular model from the model class?
6. **Algorithm:** What computational process will be used to fit the model to the data and/or to make predictions?

As we go forward, we will explore different model classes, algorithms for fitting those model classes, and theoretical approaches to selecting appropriate models, making good predictions, and providing formal analyses of the quality of the predictions and models.

We will try to characterize each of the methods we study in terms of their problem class and model type, placing it into this space:

	No model	Prediction rule	Prob model	Dist over models
Regression				
Classification				
Density estimation				
Reinforcement learning				
Clustering				
Dimensionality reduction				
Other				