| CIS399: The Science of Data Ethics | Lectures 1-4 |
|---|---|
| Instructors: Michael Kearns & Ani Nenkova | Scribes: Adel Boyarsky & AJ Hallac |

# 1 Introduction

In the first four lectures of this class, we have discussed class policies and introduced the concepts of machine learning. Class policies can be found on our class page. The aim of these lecture notes are to explore the machine learning concepts discussed in class more thoroughly and provide a standard for the lecture notes we expect.

Before we dive into the notation, let's briefly discuss some concepts of machine learning informally. With modern technology, we can collect and store a variety of data and then use that data to make future decisions. For instance, one could imagine that a potential borrower filling out a loan application may have to answer many questions about salary, education level, and so on. The lender could then store this data. A lender may also want to keep track of which past successful loan applicants did pay their loan back and which borrowers defaulted (did not pay their loan back). This information could be used to make "smarter" lending decisions in the future. By "smarter", we mean that with the use of machine learning techniques, we can learn from prior data and results to make a prediction about the future. Imagine that instead of simply relying on an individual borrowers information to make a decision about lending to them or not, we have access to thousands and thousands of records of people who were granted loans and whether they were successful in paying back them back. Machine learning techniques can allow us to capitalize on the wealth of data available to make "better" decisions. Banks can use the techniques we will discuss in these lecture notes to learn the relationship between the information in the application and the result of whether or not the person paid their loan back. The goal of the bank is to find a relationship that can accurately predict if the person on the **next** loan application will pay it back.

Now that we have informally discussed machine learning and how it can be useful, in the next sections we will formally define these notions.

# 2 The Data

We will start by defining the properties of the data used in machine learning problems in a supervised setting. We say that there is some unknown distribution, $P$, over all the data points. Each point in $P$ consists of a vector of features $\boldsymbol{x}$ and a corresponding label $y$. Each feature vector, $\boldsymbol{X}$, is a $d-$dimensional vector. The goal is to learn the relationship between the features and the label so that given a new vector of features from the population we can accurately predict it's label. However, we must do this only seeing the data in our sample, $S = \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \ldots, (\boldsymbol{x_m}, y_m)\}$, a set of $m$ points from our distribution $P$. This can also be written as $S = (\boldsymbol{X}, \boldsymbol{y})$ where $\boldsymbol{X} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_m})$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$. Going back to the example of banks and loans, we can think of $S$ as the previous loans the bank has given out and the result of the loan. The bank must make decisions based off of this sample as they don't have access to all possible applications and the outcome of the loan.

As we said earlier, our goal isn't to predict the label for an $\boldsymbol{x}$ we have seen before, it is to predict the label for vectors (applicants) that we have not seen before. These such predictions are called **out-of-sample predictions**, or when we are predicting the label for a feature vector that

was not used for training the model. For this reason, we want to be able to predict the accuracy of out-of-sample predictions before using the model in practice. In order to achieve this, we split the $S$ into two sets, a train set and a test set. We use the **train set** to train the model and the **test set** to estimate the accuracy of the model on out-of-sample predictions.

### 2.1 Assumptions about the data

There are many assumptions about the data that we must make moving forward. The first is that the distribution $P$ exists and that our samples are **independently and identically distributed** (*i.i.d.*) from $P$. With this assumption in mind, we must be careful with how we collect our sample. The banks, for example, only have the data for the loans they accepted, which means their data may not be drawn *i.i.d.* from $P$. Remember, banks could not have tracked a result for a loan they did not give (denied applicants).

Along with the assumptions above, some other assumptions that we should make about $P$ are:

- There is some correlation between $\boldsymbol{X}$ and $y$, as this is the relationship we want to find.
- The distribution $P$ is stationary, meaning it doesn't change over time.

While we assume that such a $P$ exists, we should limit the assumptions about what $P$ is. Namely, we shouldn't assume that the features are dependent or independent from each other. We also shouldn't assume anything about the complexity of $P$.

### 2.2 Important Vocabulary

- $(\boldsymbol{x_i}, y_i)$: $\boldsymbol{x_i}$ is a $d$-dimensional vector that represents the features and $y_i$ is the corresponding label.
- $P$: The unknown distribution of all $\boldsymbol{x_i}$ and their corresponding $y_i$.
- $S$: The subset of the population that we have access to for training and evaluation. $S \triangleq \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \ldots, (\boldsymbol{x_m}, y_m)\}$, or $S \triangleq \{(\boldsymbol{X}, \boldsymbol{y})\}$
- **In-sample predictions**: predicting the label for a feature vector that was used for training the model.
- **Out-of-sample predictions**: predicting the label for a feature vector that was not used for training the model. The goal is to have good accuracy for out-of-sample predictions.
- **Train set**: The set of vectors and their labels from our sample $S$ used in training the model.
- **Test set**: The set of vectors and their labels from our sample $S$ used to evaluate the model rather than training the model. In practice, this is normally 10% of the sample.
- **Independently and identically distributed** (*iid*): Each point in the sample share the same probability distribution and are independently drawn.

# 3 The Model

The next step of the process is to choose a model class, also known as a hypothesis class, $H$. This model class defines the type of relationship you expect to find. Examples of model classes are linear models, decision trees, neural nets, linear threshold functions, and SVMs. In these lectures, we will focus on decision trees and linear threshold functions. A class, $H$, contains all possible models in that class. Our goal, using these terms, is to learn the "best" $h \in H$. However, what do we mean by the "best"?

## 3.1 Evaluating a model

We define $\epsilon(h) \triangleq Pr[h(\boldsymbol{x}) \neq y]$ where $(\boldsymbol{x}, y)$ is drawn randomly from $P$. In other words, if we took a $(\boldsymbol{x}, y)$ from $P$, $\epsilon(h)$ is the probability that we would make a mistake when labeling that point with $h$ as our function. We call this our **true error**. Our goal, then would be to find the function in our model class with the lowest true error. We call this ideal function $h^*$ and it's associated error $\epsilon^*$. Formally, those are defined as:

$$h^* \triangleq \text{argmin}_{h \in H} \epsilon(h)$$

$$\epsilon^* \triangleq \epsilon(h^*)$$

However, it is impossible for us to calculate $h^*$ as we do not know the true distribution $P$. We must do our best with our sample $S$. So, rather than seeking to find the best function over the true distribution, let's find the best function over our sample and hope that the errors are close, which we will prove in Section 4. First, let's define $\widehat{\epsilon}(h)$ to be the percentage of vectors in our sample that $h$ would label incorrectly. Similarly, we define $\widehat{h}^*$ to be the function in our model class with the lowest error in our sample and $\widehat{\epsilon}^*$ to be the error of that function. Formally, these are:

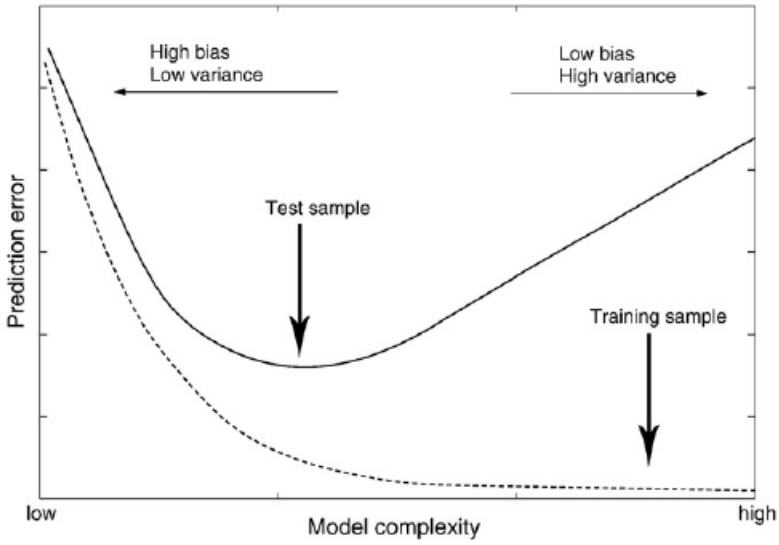
$$\widehat{h} \triangleq \text{argmin}_{h \in H} \widehat{\epsilon}(h)$$

$$\widehat{\epsilon}^* \triangleq \widehat{\epsilon}(\widehat{h})$$

To recap all this notation, we would like to achieve an error of $\epsilon^*$. We approach this by finding $\widehat{h}$, which achieves a $\epsilon(\widehat{h})$ true error and a $\widehat{\epsilon}(\widehat{h})$ in-sample error. We only know the value of $\widehat{\epsilon}(\widehat{h})$ when evaluating our model, so ideally all three values will be close together. In Section 4, we will show why this is the case.

## 3.2 Model complexity and overfitting

You might think that using more complex models allows you to recognize more complex relationships and thus should be used more. However, there is an important trade-off to know. As your model complexity increases, your lowest possible true error does decrease as you would expect but at the same time, the difference between training error and true error may become larger. This phenomenon is called **overfitting** the data. It occurs when our model is complex enough to simply memorize the data rather than learn from it. We can see this relationship in the following graph.




### 3.3 Important Vocabulary

- $H$: A class of models.
- $\epsilon(h) \triangleq Pr[h(\boldsymbol{x}) \neq y]$ for $(\boldsymbol{x}, y) \sim P$. The true error of function $h$ over the distribution $P$.
- $h^* \triangleq \text{argmin}_{h \in H} \epsilon(h)$. The function in our model class that has the lowest true error.
- $\epsilon^* \triangleq \epsilon(h^*) = \min_{h \in H} \epsilon(h)$. The lowest possible true error for a function in our model class.
- $\widehat{\epsilon}(h) \triangleq Pr[h(\boldsymbol{x}) \neq y]$ for $(\boldsymbol{x}, y) \sim S$. The error of function $h$ over the sample, $S$.
- $\widehat{h} \triangleq \text{argmin}_{h \in H} \widehat{\epsilon}(h)$. The function in our model class that has the lowest error in our sample.
- **Overfitting**: The phenomenon when the relationship learned by the model is too specific to the data in the sample, causing the model not to generalize well to the true population. This often occurs when the model is too complex or when the data in the sample is bad.

## 4 Analysis of $\widehat{\epsilon}(\widehat{h})$

As mentioned in Section 3.1, we will now take a closer look into the relationship between $\epsilon^*$, $\epsilon(\widehat{h})$ and $\widehat{\epsilon}(\widehat{h})$.

### 4.1 What do we expect?

First, let's consider what we would expect the relationship between those three variables to be. Focusing on only two at a time, we see:

1. $\epsilon^* \leq \epsilon(\widehat{h})$: The true error for our model must at least as large as the best possible error for our model class.
2. $\widehat{\epsilon}(\widehat{h}) \leq \epsilon(\widehat{h})$: We expect that training error will be an under-estimate of true error. Therefore, since we are are trying to minimize the training error, we expect the true error to be larger.

3. $\epsilon^* ? \widehat{\epsilon}(\widehat{h})$: There is no consensus here. We could overfit the data such that there is no training error, however generally this wouldn't be the case.

### 4.2 What is the true relation

We will see that if $H$ is sufficiently "simple" and $m$ is sufficiently large, $\epsilon^* \approx \epsilon(\widehat{h}) \approx \widehat{\epsilon}(\widehat{h})$.

#### 4.2.1 Showing $\epsilon(\widehat{h}) \approx \widehat{\epsilon}(\widehat{h})$

We are helped by a helpful by the following theorem: If $H$ is finite then for almost every sample, $S$, of size $m$ and for every $h \in H$:

$$|\widehat{\epsilon}(\widehat{h}) - \epsilon(\widehat{h})| < \sqrt{\frac{\log(|H|)}{m}}$$

This means that the training error and the true error of $\widehat{h}$ are close to each other if the $|H|$ (the number of possible models in $H$) is "small" or if there is enough data. We did a thought experiment about the proof of this theorem, however the formal proof uses math that is out of the scope of this course. One interesting takeaway is that with infinite data, the difference would approach zero. In addition, we note that this is fact is explored in Section 3.2.

#### 4.2.2 Showing $\epsilon^* \approx \epsilon(\widehat{h})$

There are two cases to explore here. The first is if after training on our data, we have $h^* = \widehat{h}$. In this case, we see that $\epsilon^* = \epsilon(\widehat{h})$ as they are both calculating the true error of the same function. However, consider if we have $h^* \neq \widehat{h}$. This means that $\widehat{\epsilon}(\widehat{h}) \leq \widehat{\epsilon}(h^*)$, or that the training error of $\widehat{h}$ is less than or equal to the training error of our optimal model. Now, however, we can utilize the bound from the Section 4.2.1 to show that the two values are still close together:

$$|\epsilon(h^*) - \epsilon(\widehat{h})| < 2 \times \sqrt{\frac{\log(|H|)}{m}}$$

$$|\epsilon^* - \epsilon(\widehat{h})| < 2 \times \sqrt{\frac{\log(|H|)}{m}}$$

## 5 An Empirical Case Study

To see many of these concepts in a real-life example, review the following study which claims to show that an algorithm is better at "detecting sexual orientation from facial images" than a human is.

**Link:**
*Research Paper*
**Related Links:**
*Vox Article*
*Medium Article*