Chapter 2 Summary: Statistical Learning

2.1: What is Statistical Learning?

Statistical Learning is a set of tools for estimating a function, f, that connects input variables X and output variables Y. We estimate f because we do not (and often can not) know the true relationship.

Goal: Estimate the function f that relates X and Y:

$$Y = f(X) + \epsilon$$

Given data $(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n)$, where:

- X: Predictors/Independent Variables/features
- Y: Response/Dependent Variable
- ϵ: Irreducible error - measurement errors and other discrepancies

2.1.1: Why Estimate f?

Two primary reasons to estimate f: **prediction** and **inference**

Prediction

Prediction - Using input variables X to predict an output variable Y

$$hatY = \hat{f}(X)$$

- $\hat{Y}$: Estimate for Y
- $\hat{f}$: Estimate of the true function f
- X: Predictor variables

We can represent the error by the average, or *expected value* of the squared difference between the predicted value and the actual value. This will be explained more in the future. We call this the *Expected Prediction Error (EPE)* Just note that this consists of **reducible** and an **irreducible** portion.

$$\begin{aligned} E(Y - \hat{Y})^2 &= E[f(X) + \epsilon + \hat{f}(X)]^2 \\ &= [f(X) - \hat{f}(X)]^2 + \text{Var}(\epsilon) \\ &= \text{reducible error} + \text{irreducible error} \end{aligned}$$

> **Note**
>
> E means expected value, NOT error

- **Reducible** error - The error in our model
  - $[f(X) - \hat{f}(X)]^2$
- **Irreducible** error - The error we cannot get rid of.
  - Even if we knew $f(x)$, we would still make errors in prediction, since for any $X = x$, there is a distribution of possibly Y values
  - $\text{Var}(\epsilon)$

Inference

Inference - Understanding the relationship between X and Y.

2.1.2: How do we estimate f?

Parametric vs. Non-Parametric Models

Parametric

1. Assume some functional form for f. This is choosing the model. One example would be linear:

$$f(x) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

2. Fit or train the model. Use some procedure to estimate the parameters (in this case $\beta_0, \beta_1, \cdots, \beta_p$). That is to say, find values for the parameters such that

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

In this example, we have simplified the problem of estimating f down to fitting parameters $\beta_0, \beta_1, \cdots, \beta_p$

- Too flexible a model leads to underfitting and poor predictions
- Too complex a model leads to overfitting
- Choosing a functional form that is very different from the true form of f will result in poor results

Non-Parametric

- No explicit assumption about the functional form of f
- More flexible since no specific function shape, can fit many shapes
- Requires far more training data to accurately estimate f since the problem isn't reduced to estimating parameters
- Easy to overfit to training data

Supervised vs. Unsupervised Learning

Supervised

- For each observation, we have the input measurements, x_i and the output, y_i is labeled
- Ex: Linear Regression, Classification

Unsupervised

- For each observation, we have the input measurements x_i, but there is no associated response, y_i
- Often seek to understand the relationship between variables or observations
- Ex: Clustering

Regression vs. Classification Problems

Regression

- The response (Y) is quantitative (a number)
- Ex: Predicting income or the value of a house

Classification

- The response (Y) is qualitative (a category or label)
- Attempting to predict what group something is in
- Ex: Is an email spam or not spam?

2.2: Assessing Model Accuracy

2.2.1 Measuring the Quality of Fit

When estimating the function $\hat{f}(x)$ we need to determine how well it predicts the response variable, Y. The most common way of doing this is with the **Mean Squared Error (MSE)**.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\Big(y_i - \hat{f}(x_i)\Big)^2$$

- y_i - actual value
- $\hat{f}(x_i)$ - predicted value
- n - number of data points

If MSE were computed using the training data, we would fit our training points as closely as possible and would surely overfit, so we don't really care how $\text{MSE}_{\text{train}}$ does! We are far more interested in how our model predicts new, unseen data! So what we want to do is split our test data and training data, then test them separately!

Test error is what we really care about, not the training error

- Flexible (complex) models fit the training data very well and thus have low training error
- They may not generalize well to the test data, which leads to poor results. We call this **overfitting**

2.2.2: The Bias-Variance Trade-Off

- **Bias** - the error from wrong assumptions in the model
  - Can think of this as error caused by simplifying a real-life problem into a much simpler model
- **Variance** - error from the model being sensitive to small changes in the training set
  - Can think of this as "how much would $\hat{f}$ change if trained on different data"
- High bias - underfit (model too simple)
- High variance - overfit (model too complex)

$$\text{E[MSE]} = \text{Variance} + \text{Bias}^2 + \text{Irreducible Error}$$
$$\text{E}[y_0 - \hat{f}(x_0)] = \text{Var}(\hat{f}(x_0)) + [\text{Bias}(\hat{f}(x_0))]^2 + \text{Var}(\epsilon)$$

2.2.3: The Classification Setting

The Bayes Classifier

The **Bayes classifier** (sometimes referred to as the Bayes optimal classifier) is simply one which assigns to each x value the class that is most likely, given that $X = x$. That is, whatever class j for which

$$\Pr(Y = j | X = x)$$

is the largest. A mathematical way of writing this could be:

$$C(x) = j \text{ if } p_j(x) = \max\{p_1(x), p_2(x), \ldots, p_k(x)\}$$

An Aside

This can also be represented with a mathematical operation argmax. argmax of $f(x)$ is equal to the x value for which $f(x)$ is biggest. For a formal definition of argmax, please see wikipedia.
Using argmax, we say

$$C_{\text{Bayes}}(x) = \underset{j}{\text{argmax}} \Pr(Y = j | X = x)$$

If you are not familiar with argmax, we can think of $\underset{x}{\text{argmax}} f(x)$ as being equal to whatever value of x maximized $f(x)$.

The Bayes classifier can be thought of as the optimal classifier, since it will minimize error as it always classifies to the most likely class. Unfortunately, we don't know the conditional probability of Y given x, but many methods attempt to estimate this conditional distribution. The most basic one is...

K-Nearest Neighbors

For any test observation, x_0, the K-Nearest Neighbors (KNN) classifier first finds the K points in the training data closest to x_0. It then estimates the conditional probability for class j, given $X = x_0$ as the percent of points in these K points with class j. It then classifies to whichever class has the highest probability.

Using less math lingo, KNN classifies a point x_0 as whichever class is most prevalent (in the training data) of the K points closest to x_0.

Note: While it is simple, for many problems, K-nearest neighbors works great!

Key Takeaways

Note: Key Takeaways

- The goal of statistical learning is to estimate the function f that connects input variable X to output variable Y
- There are trade-offs between:
  - bias and variance
  - interpretability and flexibility
- Supervised learning can be either **regression** (quantitative Y) or **classification** (qualitative Y)
- Model assessment (testing) should be on data that is reserved for testing, not what was used to train