Chapter 2
Assessing Performance; Bias + Variance Tradeoff

2.1 Linear Regression Model

Let us recap what we have learned from Chapter 1 about linear regression. With linear regression we have many these many parts at play:

1. **Data set**: $\{(X_i, y_i)\}_{i=1}^n$ where $X_i \in R^d, y \in R$
2. **Feature Extraction**: $h(X_i) = (h_0(X_i), h_1(X_i), ..., h_D(X_i))$
3. **Linear Regression Model**: $y = w^T h(X_i)$
4. **Quality Metric / Loss function**: $\text{MSE} = \frac{1}{n}\sum_{i=1}^n (y_i - \hat{y}_i)^2$
5. **Predictor**: $\hat{w} = \underset{w}{\text{argmin}}\, \text{MSE}(w)$
6. **Optimization Algorithm**: Optimized using Gradient Descent
7. **Prediction**: $\hat{y_i} = \hat{w}^T h(X_i)$

From our examples we have seen that a linear regression can be quite useful for predicting linear data with high accuracy however, a linear regression has its limitations in that it can only predict linear data well. For instance, take a look at some sample data below.

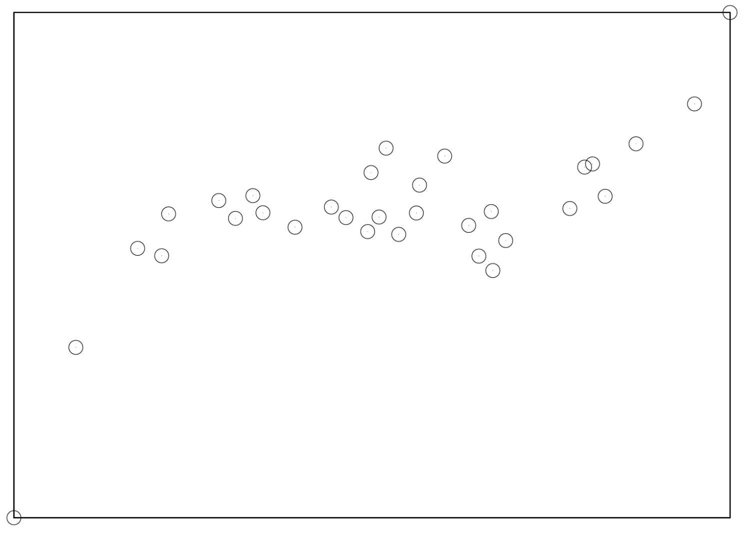

Figure 2.6: Sample data. Note that the true relationship between x and y is nonlinear.

We could attempt to fit a line to this data using the methods we have discussed but such a model would never truly be able to represent the data. To better predict data like this we will use a similar model using more features called polynomial regression.

2.2 Polynomial Regression

From the sample data shown earlier, we see that a linear model would not fit the data well. Polynomial regression is extremely similar to linear regression, which is represented using one input and a trained weight value:

$$y = wx + b \tag{2.8}$$

Now, with polynomial regression we take more input features (weights) to approximate any degree polynomial of our choosing to better fit polynomial like data.

Definition 2.1: Polynomial Regression Model

$$y = w_0 + w_1x + w_2x^2 + ... + w_dx^d$$

Now that we have the option to pick the degree of our model we have encountered our first hyperparameter.

Definition 2.2: Hyperparameter

A **hyperparameter** is a parameter whose value is used to control the learning process, different from the values of other parameters or weights which are derived from training.

Looking again at the sample data from before, the data looks cubic so we would likely choose the model's degree hyperparameter to be 3. However, data does not often look this clean or is easy to visualize, so choosing a polynomial complexity can be difficult. Let's imagine we pick different model complexities for new data and see how they fit.

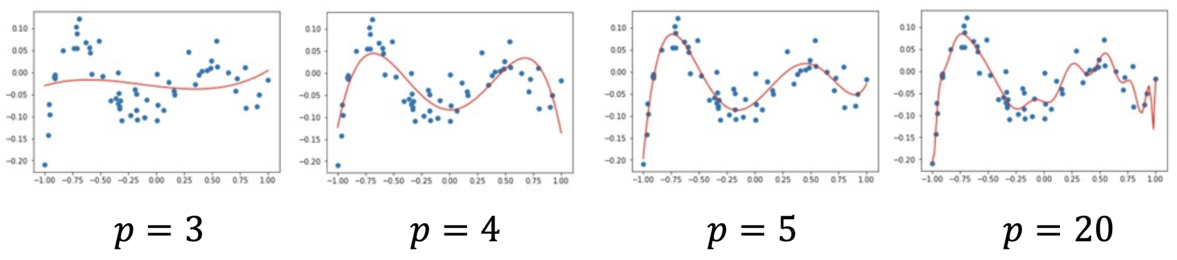

Figure 2.7: Fitting different complexity models to data.

If we choose a linear regression model (degree = 1) or any other model complexity that is too low, our model will perform poorly no matter how well our model is trained (see above). This idea can be referred to as having a model that **underfits** the data. However, it is also possible to choose a model complexity that is too high (see above p = 5, 20) which we call **overfitting**. This introduces the balancing act of machine learning: variance and bias.

2.3 Bias and Variance

Definition 2.3: Bias

Bias is the difference between the average prediction of our model and the expected value which we are trying to predict.

Definition 2.4: Variance

Variance is the variability in the model prediction, meaning how much the predictions will change if a different training data set is used.

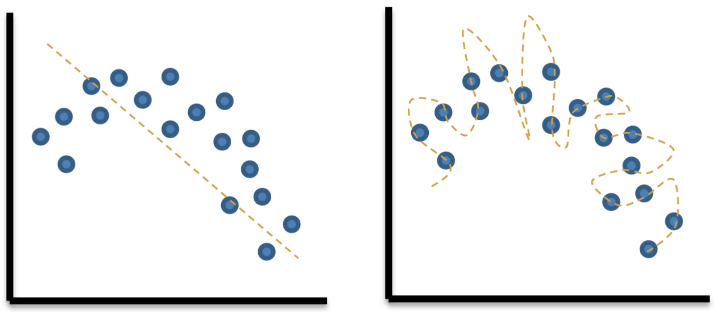

Figure 2.8: Often low complexity or simple (on the left) models tend to have high bias and high complexity models (on the right) tend to have high variance.

Variance and bias form the two main sources of error in a model that we work to control, but there is also irreducible error, which is the error we can't avoid or possibly eliminate.

Definition 2.5: Irreducible Error

Irreducible error is unavoidable error caused by elements outside of our control, such as noise from observations.

In general for ML models, simple models have high bias while overly complex models have low bias but high variance. We can identify our total error as this:

$$\text{Error} = \text{Biased squared} + \text{Variance} + \text{Irreducible Error} \tag{2.9}$$

Or more formally as:

$$E[(y - \hat{f}(x))^2)] = \text{bias}[\hat{f}(x)]^2 + \text{var}[\hat{f}(x)] + \sigma_\epsilon^2 \tag{2.10}$$

When it comes to best fitting a model, choosing hyperparameters like the degree in a polynomial regression can have a major impact on the results of the model. In almost all real world data there is some noise. The goal for creating a model is to be able to make a prediction about the given training data while also generalizing well enough to predict the test data accurately. In Figure 1.1.3, we see that the simple linear regression on the left is not fitting the data well and will then have poor training and test set performance. The model on the right fits the training set with zero training error...but do we have a perfect model? The answer is no, this model is overfitting our training data, meaning that once we give the model more data it will likely predict the new data with low accuracy as the model snakes around to hit every training point perfectly. While it is our goal in ML to reduce error from the model, 0 training error is often not ideal. The high complexity model in Figure 1.1.3 has high variance, relating to its definition, because it fits the training data well but will perform poorly on new test set data, showing the differing results from one set to another.

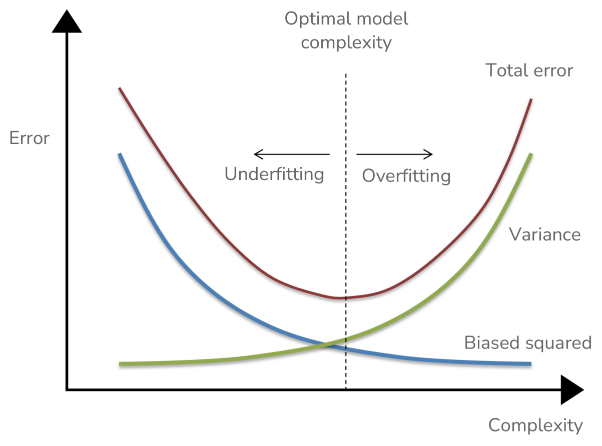

Figure 2.9: Bias and variance tradeoff

Our goal is to find a model that fits the data most accurately. This middle ground often lies where there is a good balance between bias and variance, as shown in Figure 1.1.4. This means that we are truly understanding the main function at play and not the noise within the data set.

Another factor to consider is the amount of data. With a low volume of training data, being able to generalize well is difficult. Imagine trying to fit a potentially highly complex model to only 10 data points. This would make understanding the true relationship difficult. With more data, our model will inherently begin to generalize better as outliers in the data have a decreasing impact on the overall model. This is because increasing the number of data points to consider decreases the values placed on each data point individually.

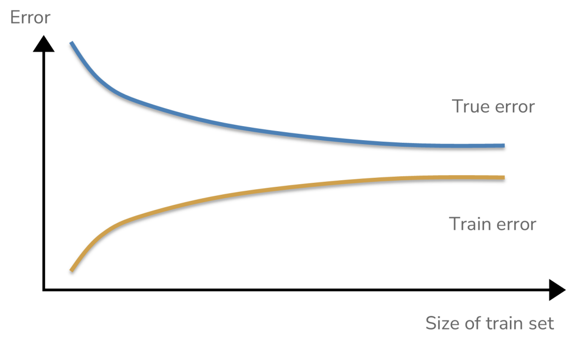

Figure 2.10: Error versus the size of the training

The figure above represents how the decreased weight on individual points can help us generalize better and decrease the model's true error. Because the model is generalizing better (reducing variance), train error slightly increases as it converges towards the decreasing true error.

In summary for choosing complexity we have learned that:

1. Choosing hyperparameters are essential for a well fitting model.
2. Overfitting and Underfitting
3. Simple models tend to have high bias and low variance and complex models ten to have low bias and high variance.
4. Choosing a model which has 0 training error is often not a good idea due to its high variance.
5. Error as a function of train set size

From these ideas it is easy to see that picking the model which has the best performance for unseen test data set is the best indicator of true error. By choosing a model based on test error we will minimize variance as we compare different model complexities. However, this situation would create a biased approach. Our goal for a test set is to have a pure, ”untouched” representation for our model. If we chose the best model based solely on test data performance we are choosing a model for that set. The test set would no longer represent “the unknown” since we probed it many times to figure out which model would be best.

When choosing what type of model will perform the best, we must perform another split in our data. Between a training and test set there is usually an 80/20 split, however this can vary depending on data set sizes. To now choose a model's hyperparameters without biasing towards the test set, we must create a third split of the data called a **validation set**. This will allow us to make decisions about the model architecture and complexity without ever touching the final testing set. We will learn more about validation sets in the following chapter.