Problem 1 (Exploring Bias and Variance, 10 pts)

In this problem, we will explore the bias and variance of a few different model classes when it comes to logistic regression.

Consider the true data generating process $y \sim \text{Bern}(f(x)), f(x) = \sigma(\sin x)$, where $\sigma(z)$ is the sigmoid function $\sigma(z) = (1 + \exp[-z])^{-1}$, $x \in \mathbb{R}$, and $y \in \{0, 1\}$. Recall that for a given x, Bias and Variance are defined in terms of expectations *over randomly drawn datasets* D from this underlying data distribution:

$$\text{Bias}[\hat{f}(x)] = \mathbb{E}_D[\hat{f}(x)] - f(x)$$
$$\text{Variance}[\hat{f}(x)] = \mathbb{E}_D[(\hat{f}(x) - \mathbb{E}_D[\hat{f}(x)])^2]$$

Here, $\hat{f}(x)$ is our estimator (learned through logistic regression on a given dataset D). We will directly explore the Bias-Variance trade-off by drawing multiple such datasets and fitting different logistic regression models to each. Remember that we, the modelers, do not usually the true data distribution. Knowledge of the true $f(x)$ is only exposed in this problem to 1) make possible the simulation of drawing multiple datasets, and 2) to serve as a pedagogical tool in allowing verification of the true Bias.

1. Consider the three bases $\phi_1(x) = [1, x]$, $\phi_2(x) = [1, x, x^2, x^3]$, $\phi_3(x) = [1, x, x^2, x^3, x^4, x^5]$. For each of these bases, generate 10 datasets of size $N = 10$ using the starter code provided, and fit a logistic regression model using $\text{sigmoid}(w^T\phi(x))$ to each dataset by using gradient descent to minimize the negative log likelihood. Note that the classes are represented with 0's and 1's. This means you will be running gradient descent 10 times for each basis, once for each dataset.

   Use random starting values of w, $\eta = 0.001$, take 10,000 update steps for each gradient descent run, and make sure to average the gradient over the data points (for each step). These parameters, while not perfect, will ensure your code runs in a reasonable amount of time. The emphasis of this problem is on capturing the bias-variance trade-off, so don't worry about attaining perfect precision in the gradient descent as long as this trade-off is captured in the final models.

   Note: log(0) is undefined and may cause issues in your implementation. To address these issues, we recommend adding a small constant epsilon (such as ϵ = 1e-7) whenever taking a log. For example, $\log(x) \rightarrow \log(x + \epsilon)$. Separately, overflow RuntimeWarnings due to np.exp should be safe to ignore.

2. Create three plots, one for each basis. Starter code is available which you may modify. By default, each plot displays three types of functions: 1) the true data-generating distribution, 2) all 10 of the prediction functions learned from each randomly drawn dataset, and 3) the mean of the 10 prediction functions. Moreover, each plot also displays 1 of the randomly generated datasets and highlights the corresponding prediction function learned by this dataset.

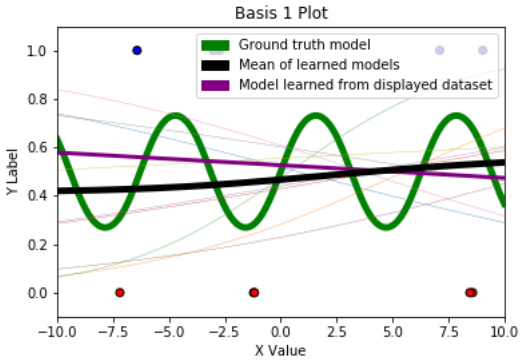
3. Explain what you see in terms of the bias-variance trade-off. How do the fits of the individual and mean prediction functions change? Keeping in mind that none of the model classes match the true generating process exactly, discuss the extent to which each of the bases approximates the true process.

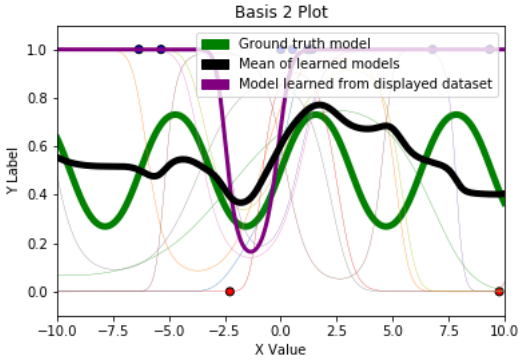
4. If we were to increase the size of each dataset drawn from $N = 10$ to a larger number, how would the variance change? Why might this be the case?

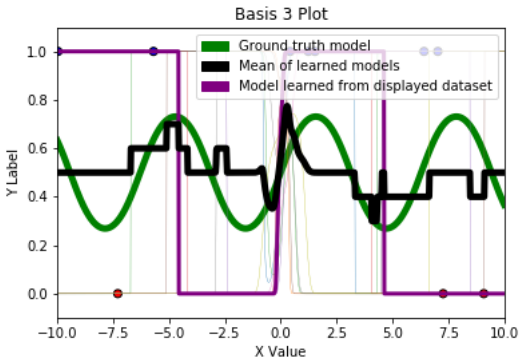
Solution

1.

2. The three plots for the three bases are shown below:





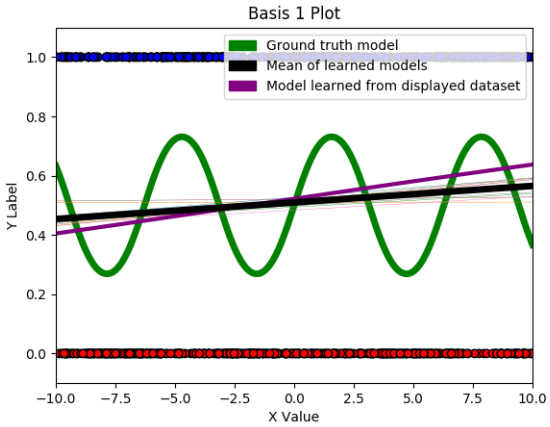


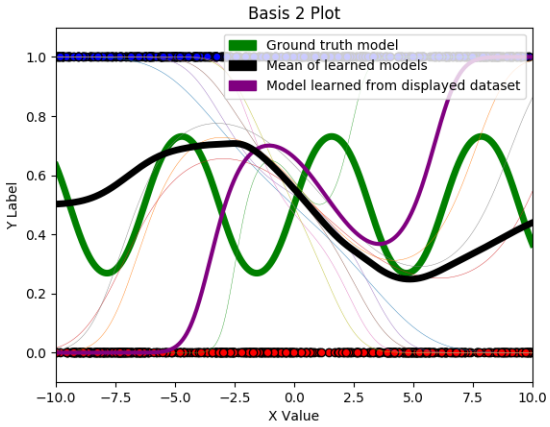
3. We see that for Basis 1 there is little to no variance (as it predicts a linear model), but there is higher bias (predicts points further away from the true distribution) than our other bases because the linear model is simple and inflexible. Since our model is a non-linear model, Basis 1 is not able to accurately predict many of the points of the true model.

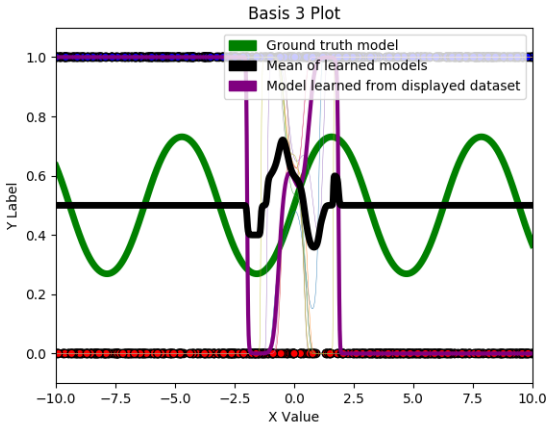
   For Basis 2 and Basis 3, as we add more polynomial terms we find that the variance of our models increase but the bias decreases. That is, Basis 3 has the highest variance and lowest bias of our models while Basis 1 has the lowest variance and highest bias. Basis 2 is between the two other bases, having more variance than Basis 1 but less than Basis 3, and less bias than Basis 1 but more than Basis 3.

This makes intuitive sense, because our bases that have more polynomial terms are more flexible and complex (since more polynomial terms allow more "bends" in our model). That flexibility allows our model to more accurately fit the true data (lower bias) but also means there are more bends in our model (higher variance) from the higher order polynomial terms. The more complicated models have higher variances because they become too specific to our data (overfitting the data) as random noise and outliers in sampling the data will more drastically affect the fitting of our model, whereas the simple linear model (Basis 1) is so simple that it is underfitting the data and it is not a complex enough basis to fully capture the true distribution of the true non-linear model.

4. Below we recreate our basis plots using 1000 generated datasets ($N = 1000$).







We can clearly see from the plots that increasing the size of the dataset (from 10 to 1000 as is shown) reduces both the bias and variance of our fitted model. This makes intuitive sense because having more data allows our theoretical model to fit the true distribution more accurately, since we have more samples from the true distribution when we have more data points. In fact, having more data is one of the only ways to reduce both the bias and variance of our model simultaneously.