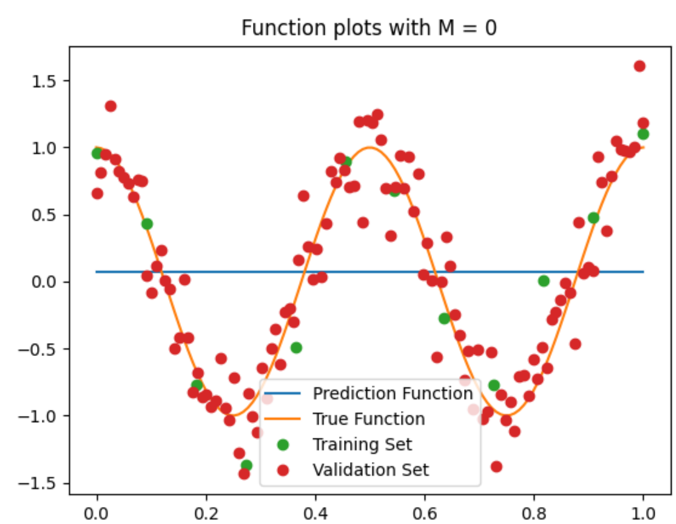
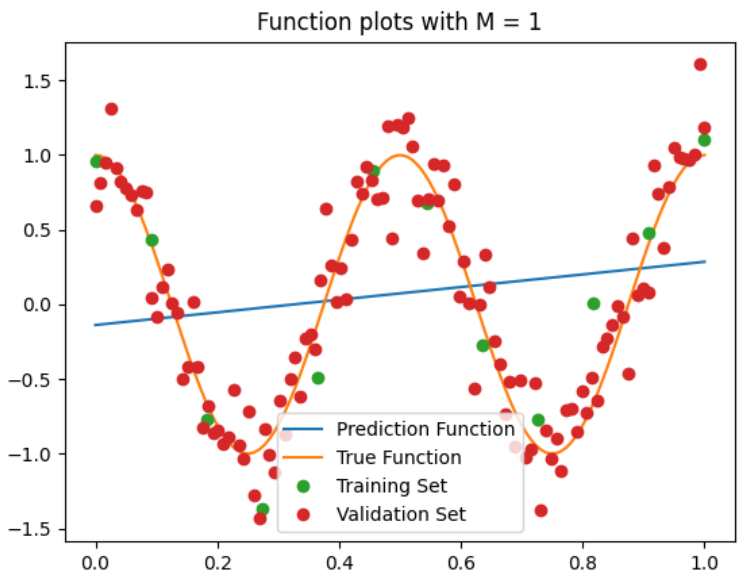
4SL4 REPORT - Assignment 1
Parthav Patel
patep62 - 400251342

Function Plot Discussions:

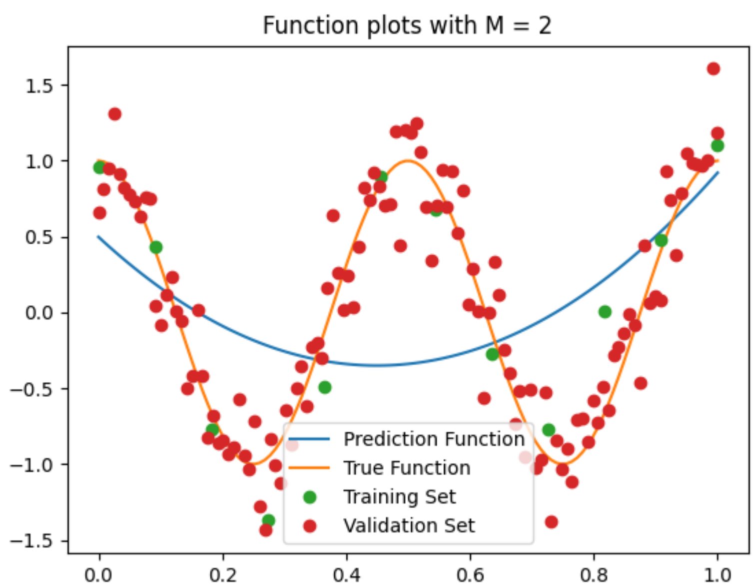
Plot for M = 0:



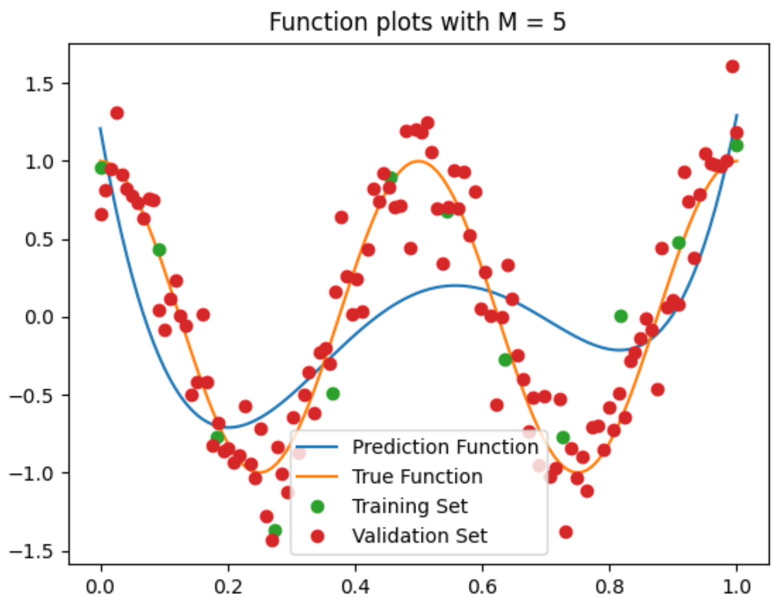
Here we can see that the prediction function is a straight line. This makes sense as with M being 0, the prediction function is simply just w0. So this straight line is at y = w0.



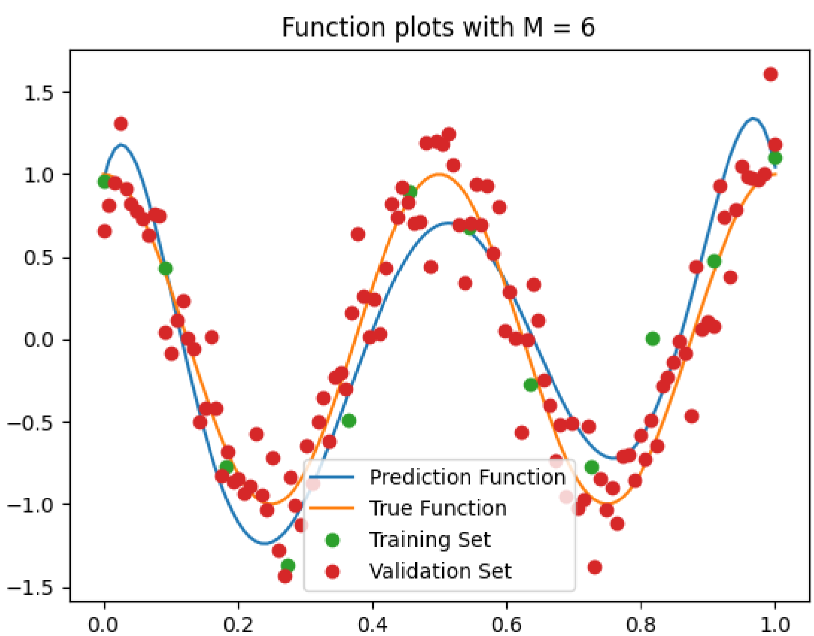
For M = 1, we get a line as well, this time with a positive slope. The prediction function is behaving as we should expect with an increase in polynomial degree.



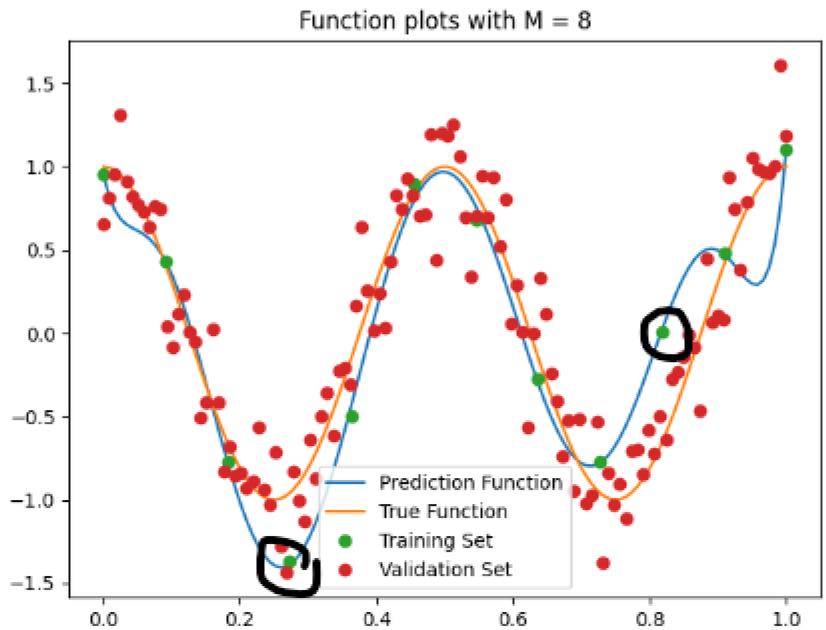
For M = 2 we start to see a convex shape. This aligns with the 2nd degree polynomial we should see with a capacity 2 model.



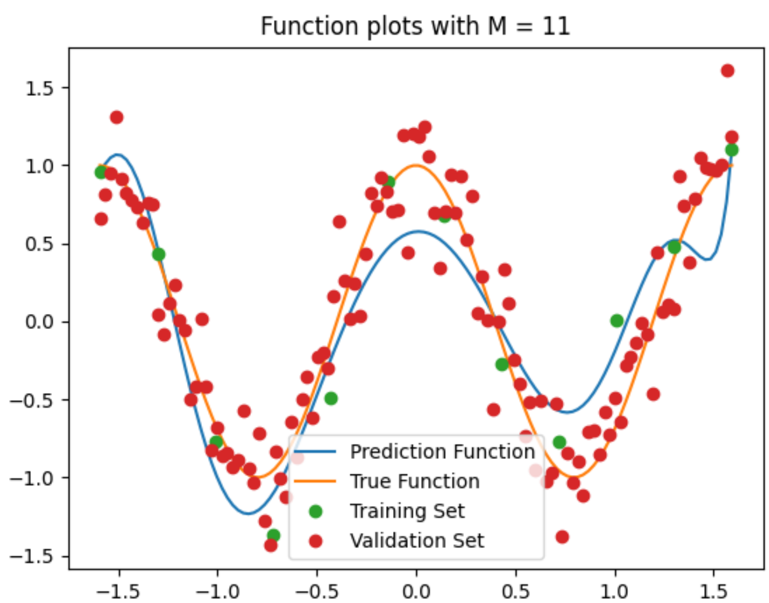
Skipping ahead to M = 5, we start to see a resemblance to the true function since we have enough capacity now.



With M = 6, we see a very solid prediction of the true function. More and more training set points are being run through by our predictor as well, which is expected. We can also see that the predictor is still unable to avoid the validation set noise in the top left and right corners of the plots.

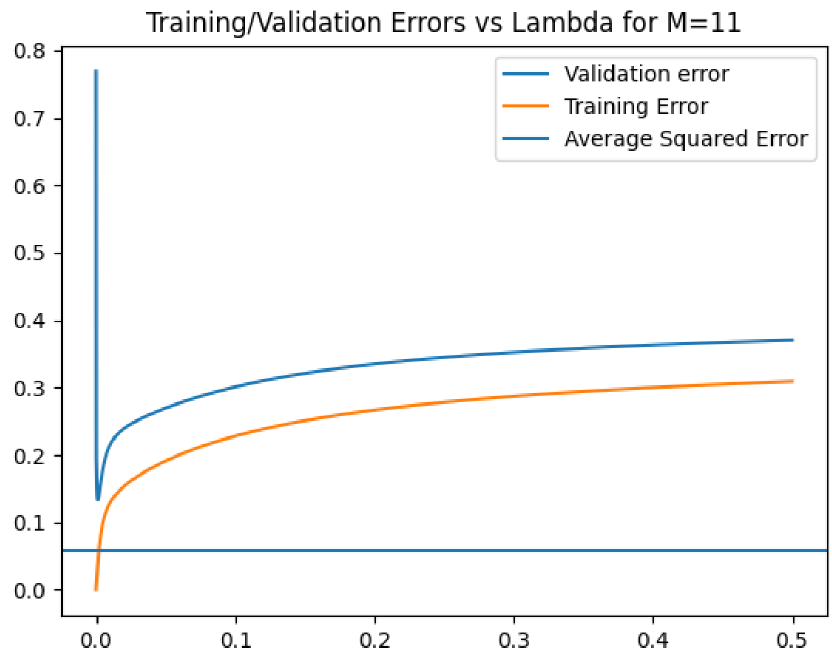


With M = 8, we can see that most of the validation set noise is now being avoided. However, as the capacity has become too high, the predictor has adapted to noise within the training set, causing inaccurate plottings. As seen in the graph, there are 2 main green outliers that are passed through by the prediction function,



Here we can see that due to regularization, overfitting has not fully occurred. The outliers mentioned earlier are no longer being accounted for as closely. In theory, with a capacity of 11, and a training set consisting of 12 points, we should see some very obvious overfitting as the predictor is able to pass through almost all training points.

In this plot, we can see various expected behaviors. Training error consistently decreases with capacity increase, approaching zero. Validation error starts very high but rapidly decreases, and due to regularization, does not shoot back up at higher capacities. Additionally, the average squared error is very low at just above zero. This value is pretty much a representation of the noise, since it is the average difference between the target validation set and the true set. The only difference between these sets in reality, should be the noise we put in. Therefore, if we increase the number of samples for the validation set infinitely, the validation error should eventually reach the variance, which in this case was set to 0.0625. In other words, the average squared error is the lowest possible value for validation error we can get, assuming a perfect model and infinite samples.

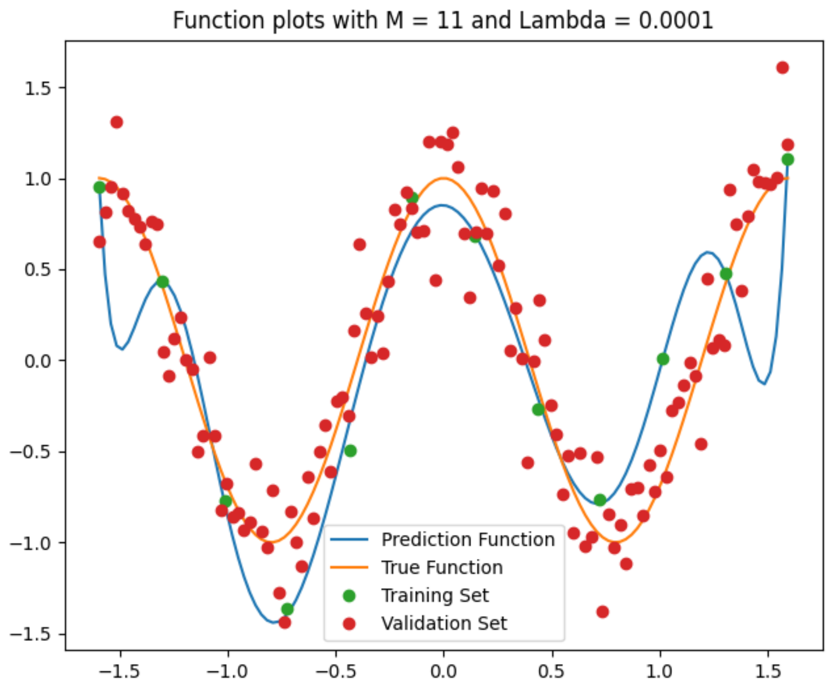


Here I select a very large number of lambdas to test within a range of 0 to 0.5. As lambda is very small, close to zero, validation error is very high. You can also see a dip in the training error at low lambda values. This is most likely where overfitting would occur, as training error is very low, but validation error is high. Moving a little bit forwards, we can see our ideal lambda value that gives us the lowest validation error. It is hard to tell where underfitting might occur, as the training error seems to remain stagnant, only slightly increasing. Therefore, it can be assumed that underfitting will occur towards the higher end of the lambda range as both validation and training error seem to steadily increase. Therefore, I will choose 0.001 for the overfitting lambda and 0.5 for the underfitting lambda. Additionally, the idea that the average squared error line is the absolute minimum for our validation error is nicely displayed here. With overfitting, the training error can approach zero, however the validation error cannot go below the ASE line due to the noise we applied to the test set.

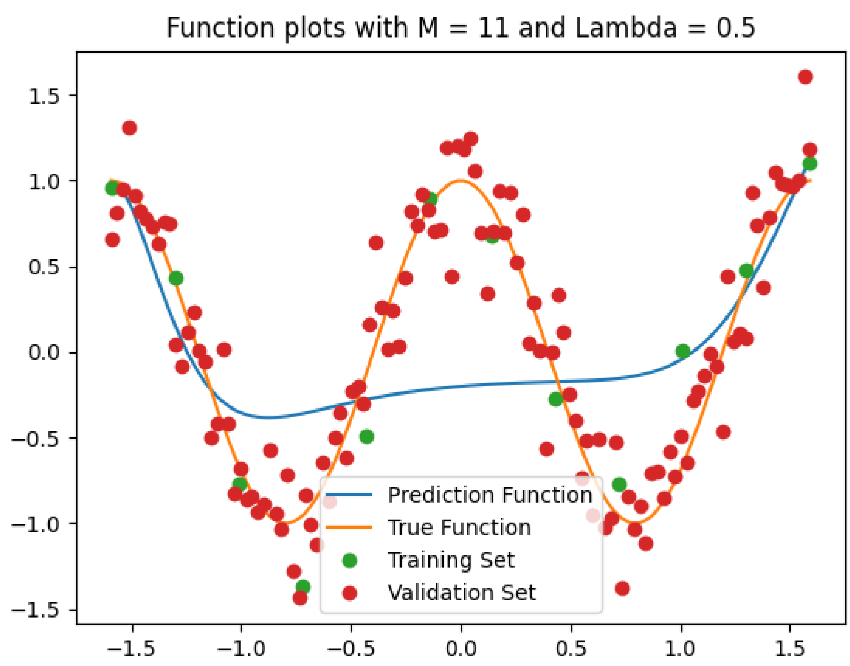
Also, the following parameters were obtained for lambdas 0.0001 and 0.5, respectively.

```
[[ 0.85111769]
 [-0.09591257]
 [-7.19025281]
 [ 1.55185534]
 [ 6.38632074]
 [-0.95943943]
 [ 1.99296697]
 [-0.42199557]
 [-3.24996478]
 [ 0.38791364]
 [ 0.75545258]
 [-0.06468405]]
```

```
[[-0.20079161]
 [ 0.10734977]
 [-0.13407979]
 [ 0.0636935 ]
 [-0.01392757]
 [ 0.02324533]
 [ 0.0759835 ]
 [-0.01582897]
 [ 0.11005477]
 [-0.03948687]
 [-0.03914673]
 [ 0.01444415]]
```



With lambda at 0.0001, we can see very clear overfitting (as opposed to the earlier plot at M = 11). Every single training set point is passed through very closely by the graph (with the exception of 1). This causes inaccuracies in our model as some of these training set points are thrown off by the noise.



With lambda at 0.5, the underfitting is very obvious as the prediction curve has almost entirely lost its shape.