COE 4SL4 – Fundamentals of Machine Learning

Assignment 1 – Trade-Off Between Overfitting and Underfitting

Nicole Boettcher – 400262726 - boettchn

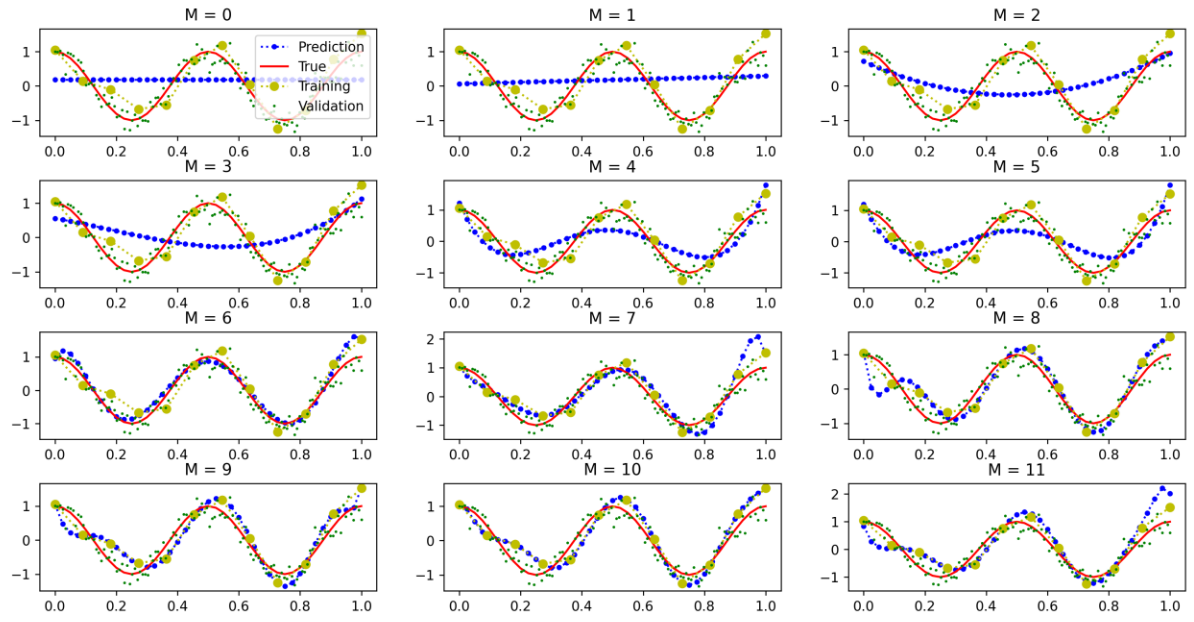
In this experiment, I explored the tradeoffs between overfitting and underfitting. The target data was created from a sinusoidal function with Gaussian noise. A training set of size 12 and a validation set of size 120 were created. I trained 12 least squares regression models ($0 <= M <= 11$) of increasing capacity using the training data. Due to the small size of the training data, the higher order models became sensitive to the noise in the training data. This results in a low training error but a high validation error.

Target formula: $t = \sin(4\pi x + \pi/2)$ | + noise

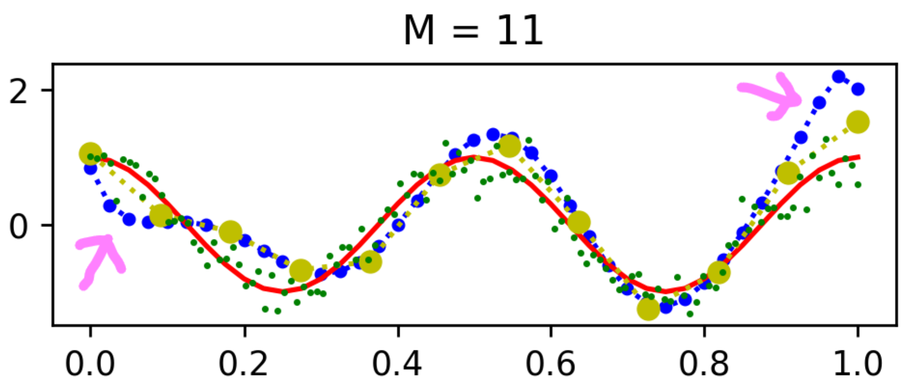
$f\_true = \sin(4\pi x + \pi/2)$

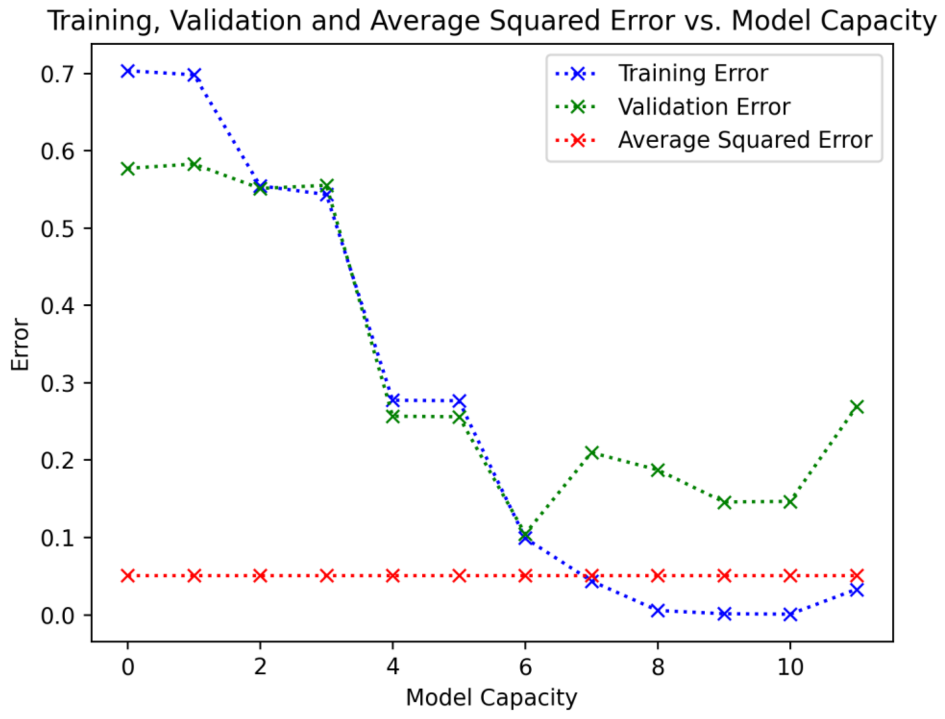
Prediction function: $f_M(x) = w_0 + w_1 * x + w_2 * x^2 + \cdots + w_M * x^M$

Plots of the prediction function, true function, training set and validation set from 0 to 1.0



At small capacities the model does not have enough parameters to properly represent the data, resulting in underfitting. As the M increases, the model becomes better fitted to the training data and validation data. As M increases past 6, we can see signs of overfitting. In the graph of M=11, I highlighted the signs of overfitting with pink arrows below. The model is not learning the general patterns in the training data but instead tuning its weights to the specific samples.





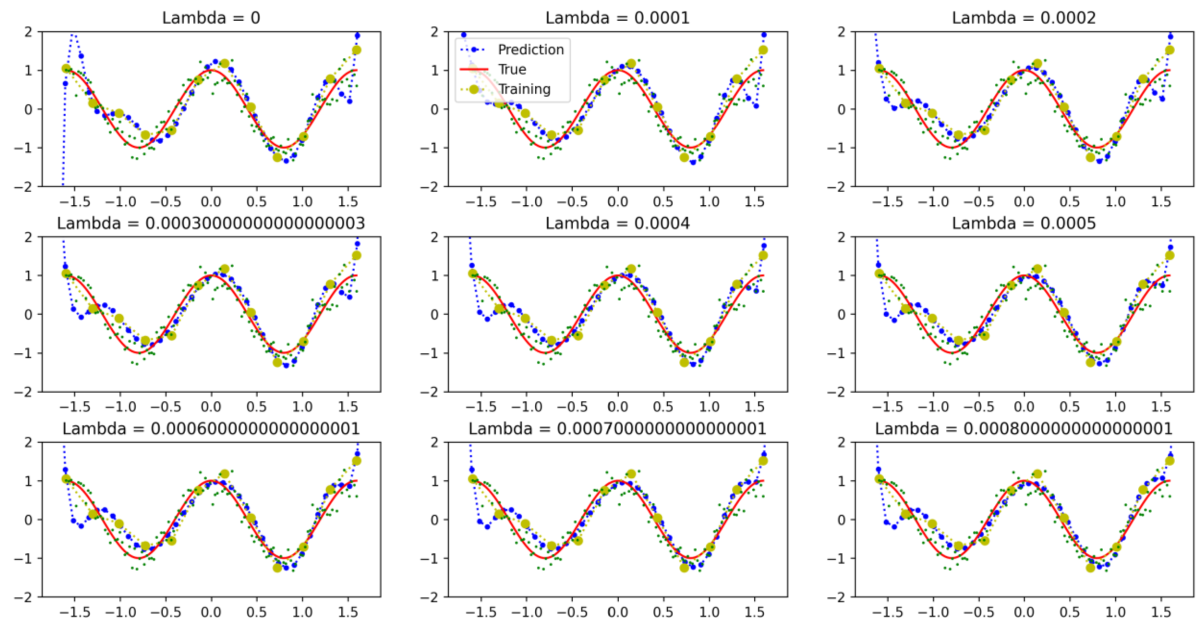
The plot above illustrates the training, validation and average squared error against the model capacity (M). The red line is the average squared error between the validation examples and the true function. If we were asked to take the mean squared error of the prediction's vs the validation data, this line would represent how good the fit is. Smaller value meaning more accurate. However, the mean squared error of the validation samples would represent the noise, as the only difference between the two sets of data is the randomly generated Gaussian noise. The line is horizonal because the validation set does not change as M is increased.

As expected, the training error and validation error decrease as M increases from 0 to 6. The parameters in the prediction function are being varied closer and closer to more accurate coefficients representing the target sin function. Beyond M=6, the training error continues to decrease as the model has more parameters to tune each sample too. Meanwhile the validation error begins to increase due to the prediction function following the noise and not the general pattern.

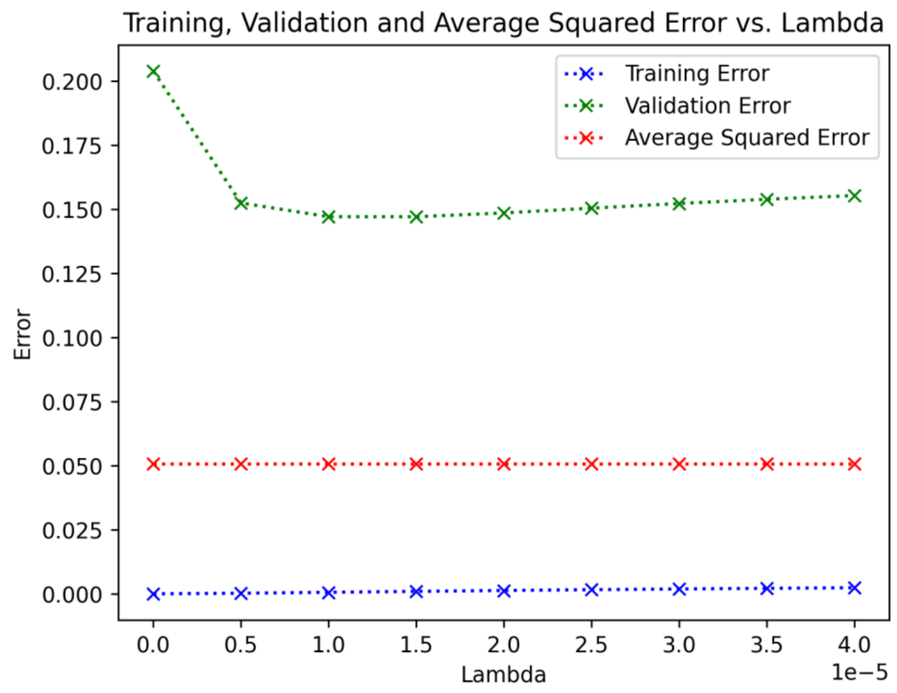
At M=11, the training error drops to 0 because there are 12 coefficients and 12 training samples.

w^T = [[8.40850036e-01], [-3.33546017e+01], [5.64125183e+02], [-5.36273723e+03], [3.48082234e+04], [-1.68791402e+05], [5.73302384e+05], [-1.27160483e+06], [1.77393403e+06][-1.49381540e+06], [6.93028249e+05], [-1.36028110e+05]]

To limit the negative effects of overfitting, and lower the validation error, I used regularization. The lambda value was selected using trial and error. I found only very small values have a positive effect on the validation error. Below is a graph of 9 different lambda values used after retraining the standardized data. The values range from 0 to 4e-5, having a step of 0.0001.

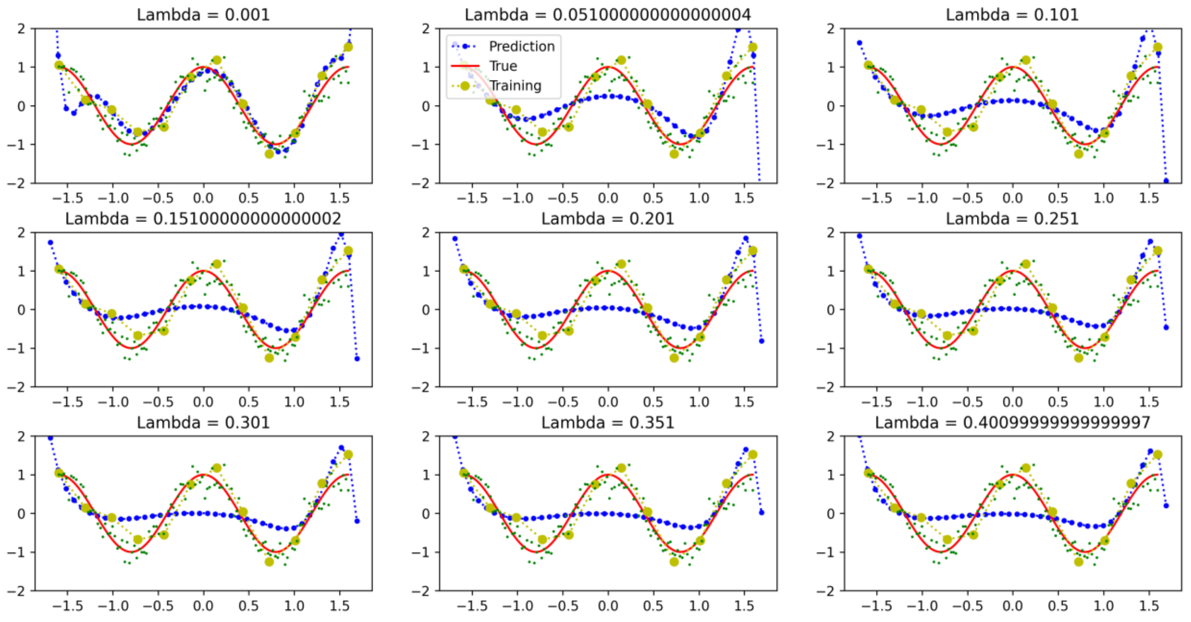


See the errors calculated from the different lambdas below.

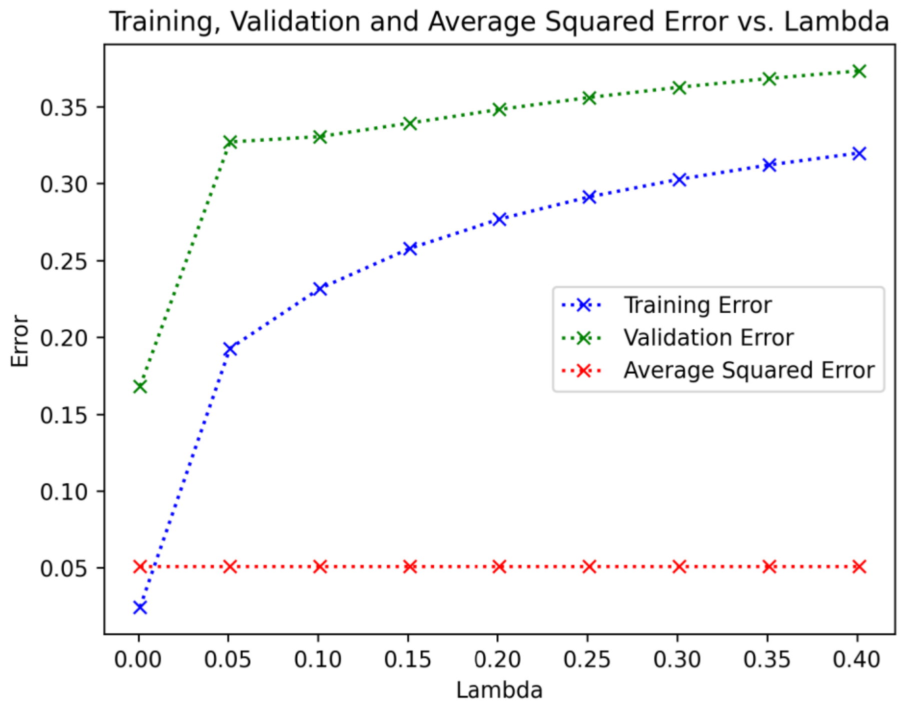


I experimented with many, many different values of lambda from the order of 10e-6 to 10. I found that within the range shown above I could see the validation error improve. At lambda$_1$ = 1.5e5, the validation error is 0.14699248. This is 5% lower than without regularization. The parameter vector for lambda$_1$ is w^T = [[1.11744718], [1.67157514], [-9.13551665], [-6.04498116], [12.56044425], [2.88013891], [-5.77784796], [3.80011814], [0.71882734], [-3.31158016], [0.06723346], [0.66988301]]. These parameters have a much smaller/reasonable magnitude. In the original w vector, the magnitude of the weights was close to 1 million, while now, they are all within –10 to 10. The large values in the original parameters were due to each training point becoming perfectly tuned to a sample in the training set. The lower magnitude weights show how the improved prediction function was able to learn an overall trend and ignore noise.

Below is a graph of lambda values which creates an inaccurate model of the validation data, also called underfitting.



I chose a range of 0.001 to 0.04001 to illustrate the smoothening effect of a large lambda. The obvious areas of overfitting which were highlighted with pink arrows during regular training begin to smoothen out. However, it also smooths out the whole function and brings the prediction much further away from the validation data. Below is the graph of errors associated with the above lambdas.



The validation and training error both increase as lambda becomes large. I will choose lambda$_2$ = 0.05 as an arbitrary choice for an underfitted model. The parameter vector for lambda$_2$ is equal to [[0.24819728], [0.04618372], [-1.2604492], [-0.38253536], [-0.13763802], [-0.1650261], [0.35485664], [0.39787138], [0.39787138], [0.26349504], [-0.16315142], [-0.09992606]]. The values are very small, which makes logical sense. The small values mean no one input is heavily weighted which would result in a smooth, low-profile curve.