Polynomial Regression, Regularization and Learning Curves

Throughout this report, a general and brief summary of machine learning diagnostics will be discussed. There are three important elements this report will touch on, namely multi-features polynomial regression, regularized regression models and learning curves. But this report generally aims to explore the algorithms and methods studied and implemented for machine learning diagnostics and the instantiated models that were used to reach the conclusions.

Linear regression is a very common machine learning algorithm that is used to compute estimates for given inputs against target variables’ scores from a loaded dataset. It is commonly implemented due to its simplistic nature and thus, makes it good for experimenting with machine learning models and their uses. However, linear regression lacks generalization when predicting output target values as opposed to other methods that will be discussed throughout this report. As suggested from the method’s name, it has a linear hypothesis function which is used to predict the output from only one presented feature. Thus, we can work with more complex models to provide better error results.

Such models include polynomial regression which uses hypothesis functions of higher orders to predict the target values. In other words, polynomial regression allows for the use of multiple features of a dataset, instead of one, to predict outcomes. Further approaches include regularized regression, in which we optimize our hypothesis function by adding an additional control variable (lambda) to it such that by changing this lambda, we are able to change the model’s behavior towards its predictions.

The last method to be used in this project is the plotting learning curves. Learning curves illustrate the correlation between the training and testing errors for a built model while varying the size of the sets used for this model. The shape of the output graph depicts whether the model suffers from either high bias or high variance. High bias is a result of a poor hypothesis function that lacks generalization. This causes under fitting of the data where the model will not perform well with foreign testing values. As opposed to high bias, high variance has the effect of over-fitting the data. In this case, the model performs too well on its learned dataset, however, it will behave poorly for other testing sets as the model is trained specifically to predict values for its original training set. As per plotting the learning curves of our model, it can be seen that the model suffers from a high variance. This only seems logical as our optimizing hypothesis function was of the 4th order. And thus, it seemed to be slightly overfitting the data.The following sections explains the details of implementation, the encountered challenges in each of them and the reason behind their selection for this project.

To start off, we have split our dataset (HousePrices) into three subsets that were used to build our models. These subsets correspond to training, validating, and testing sets that consist of data extracted from the loaded dataset for building the model. The training set accounts for 60% of the data, which is considerably a good amount as to ensure our model will be provided with adequate amounts of inputs. Each of the validation and test sets accounts for 20% of the data and are used to test the model using values unknown to the training model. Using this approach, dividing the dataset into three subsets rather than the tradition (train/test) models allows us to cross validate the data before performing the final test. The validation set is treated by the model the same way as the test set and it is used in the model selection phase to optimize the performance of the model chosen for the final test on the testing set. In model selection, different orders of hypothesis functions are used to compute the prediction errors for the validation set. The n-th order hypothesis function that outputs the least error in performance is chosen to be test the model’s performance finally.

After splitting the dataset, the features of the training set are raised to the powers of their corresponding position in the features list. This is to construct a hypothesis function in the manner of a polynomial function. The data are then normalized using the mean normalization method in which each feature’s value would be centered of its respective column mean. This ensures that variations in the data values would not incur a penalty on the model when predicting outcomes. This also enables the model to be robust against computational overflows where the values calculated may increase beyond the threshold for storing and manipulating them.

Moving on further to polynomial regression, necessary operations were performed such as implementing the cost function and the gradient descent function. The cost function computes the mean square error (MSE) between the predicted and the actual target scores. It uses the training sample and the set of optimizing parameters (thetas) to compute the MSE for each value of theta used. Thetas’ values are updated regularly as per the gradient descent function which attempts to compute the optimal values of thetas that produce the minimum MSE. The gradient descent function runs for a fixed number of iterations such that with each succeeding iteration, the model’s learning performance increases and thus, its prediction error percentage decreases.

The second approach taken in this project is attempting to regularize the regression models and compare the resulting performance with the previous section’s results. In this method, the same operations are performed such that the gradient descent function is optimizing theta values by computing the costs of each theta and iterating over an interval to improve the model’s performance. However, the only difference is that we add an extra regularization term in terms of lambda, which is used to control the values of theta to be within standard ranges as set by the lambda parameter. By varying this parameter lambda, we can change alter the behavior of the model. Typically, large lambda values accounts for small thetas and thus, the features fit for the model will not be accounted as much when predicting the target score. Small lambda values however, makes the model highlight the importance of the represented features while partially neglecting the regularization effect. We can achieve the optimal regularization factor by iterating over the regression model and choosing the one with the optimal error results.

The final technique used in this project to further understand how our regression model performs is plotting the learning curves for the model.