Hackathon1 - Intro to TensorFlow

Beichen Zhang

In this assignment, to investigate the impacts of various learning rates and the number of iterations on gradient descent, I applied a third degree polynomial function to fit the natural logarithmic function (Fig 1).

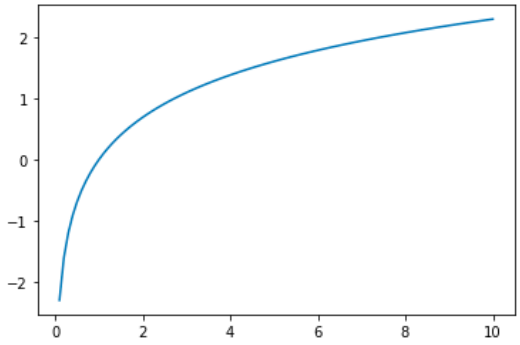

Figure 1: ln(x) with a range from (0,10].

1. Data

The original x data set ([50,1]) is an ordered series of numbers from 0.1 to 10 with an interval of 0.2. Then, I used the PolynomialFeatures function from Sklearn to generate a new feature matrix of all polynomial combinations with a degree of three. Therefore, the true X data set is a matrix with 50 samples and 4 features (1, x, x^2, x^3).

Based on the example in Hackathon 1, the objective function is XA - b = 0, and b = ln(x). The goal of the gradient decent algorithm is to find the optimal A ([4*1]) that would make the squared error of the objective function minimum.

2. Hyperparameter tuning

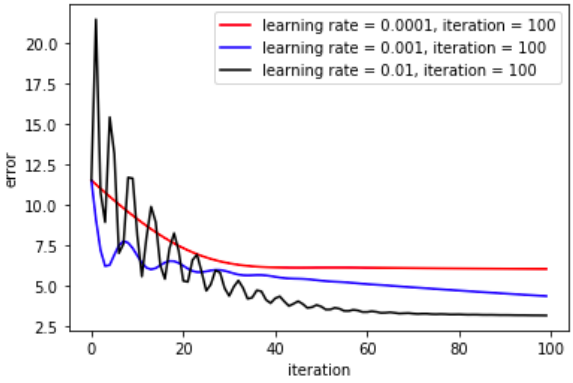

Figure 2: Comparison of the impacts of learning rates on gradient descent.

To test the effects of different learning rates and the number of iterations, I built a simple gridded search to find the best and worst combination. I used a list of learning rate: [0.0001, 0.001, 0.01] and the number of iterations: [100, 50, 10]. It turned out when the learning rate was 0.01 and the number of iterations was 100, the squared

error reached the minimum (3.16), where as the learning rate and iterations were 0.01 and 10, the squared error was the maximum (11.61).

Figure 2 shows the impacts of learning rates on gradient descent. Generally, a higher learning rate could lead to quickly converge whereas a slower learning rate could be slower to get a converging result. However, a higher learning rate would result in a larger fluctuation in the converging progression, and the lower learning rate could provide a relatively smoother and more consistent descending steps. These phenomena could be explained by the mathematical mechanism of the gradient descend. Because the learning rate is a coefficient that multiples to the differential results of features, it controls the length of the update. If the length is too large, it might cross optimal value and lead to a higher error. Also, if the length is too small, it could lead to a very slow converging progression. When we fix a learning rate (0.01), we can find that the squared error become stable after about 80 iterations, which means solely increasing the number of iterations won't be helpful if the model is already converged. Therefore, we need to select a suitable learning rate and iterations. In this case, the best learning rate and iterations are 0.01 and 100. Figure 3 shows the prediction of the optimal model.

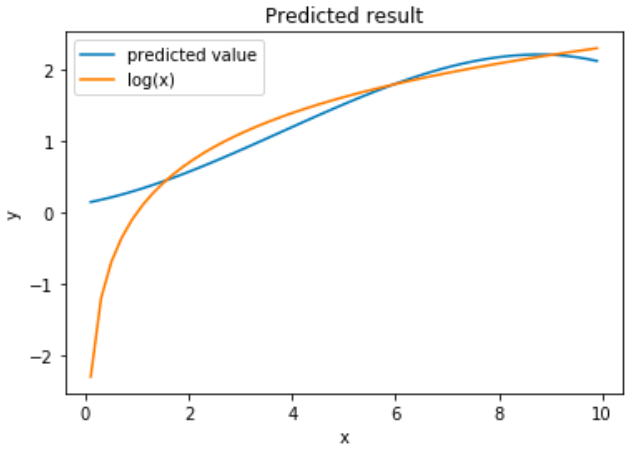

Figure 3: Comparison of the predicted and true values (log(x)).