Claude Shyaka
ID#: 801326243
ECGR-5105

Homework 5: Gradient Descent, Backpropagation, and ANN

GitHub Repo:
https://github.com/claudeshyaka/ml/tree/main/GD_and_BackPropagation_Pytorch/Homework5

In this exercise, we used gradient descent, backpropagation, and ANN with the Pytorch neural network library to develop a model for temperature prediction and house pricing. In the first part, we developed a non-linear model for temperature prediction, then developed a one-layer and a three-layer neural network for house price prediction. The results of our analysis are presented below.

1. In this section, a non-linear model was developed for temperature prediction and compared to the linear model developed in lectures. The non-linear model was trained with 5000 epochs and a learning rate of 0.0001, and the linear model was trained with 5000 epochs and a learning rate of 0.01. In addition, both models used the stochastic gradient descent (SGD) optimizer. Figure 1 below shows a graph of predictions from the linear and non-linear models. As shown in the figure, the linear model is better at predicting the temperature that is, the linear model generated training and validation losses of 2.9692 and 3.3047 respectively, whereas the nonlinear model produced training and validation losses of 4.5239 and 0.3298 respectively, after 5000 epochs. In conclusion, Results from the non-linear model suggest that the model is not a fit for this data.

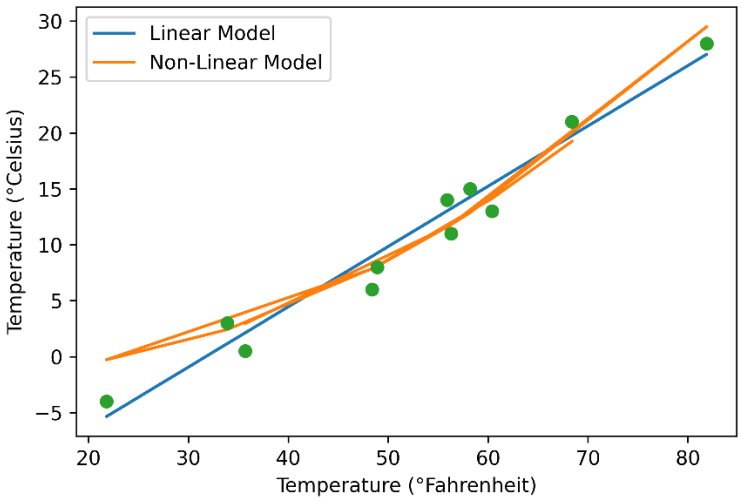

Figure 1: Linear and Non-Linear models for temperature prediction.

2. Using the housing dataset, a linear regression model was trained to predict the housing price. The features used to train the model are area, bedrooms, bathrooms, stories, and parking. In addition, the dataset was split into 80% and 20% for training and validation respectively. After training for 5000 epochs with a learning rate of 0.1, the final training and validation losses were reported as 0.0119 and 0.0099 respectively. The model would need more data to produce better estimates based on these results. Figure 2 shows the training and validation losses for 5000 epochs with a learning rate of 0.1.

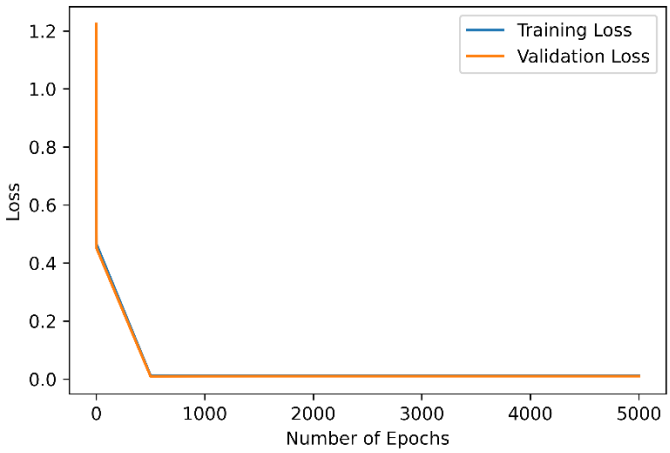

Figure 2: Training and validation loss for a linear regression model

3. In this section, a fully connected neural network (NN) with one hidden layer having 8 nodes was built for the housing dataset from part 2. In addition, the ReLU activation function was used instead of the Tanh. The NN was trained for 200 epochs with a learning rate of 0.001. The model reported training and validation losses of 0.1076 and 0.1006 respectively, and the total training time and model size were 0.08473 seconds and 0.002 MB respectively. Figure 3 shows the training and validation losses for 200 epochs with a learning rate of 0.001. These results suggest that the model should be trained longer, that is the number of epochs should be increased.

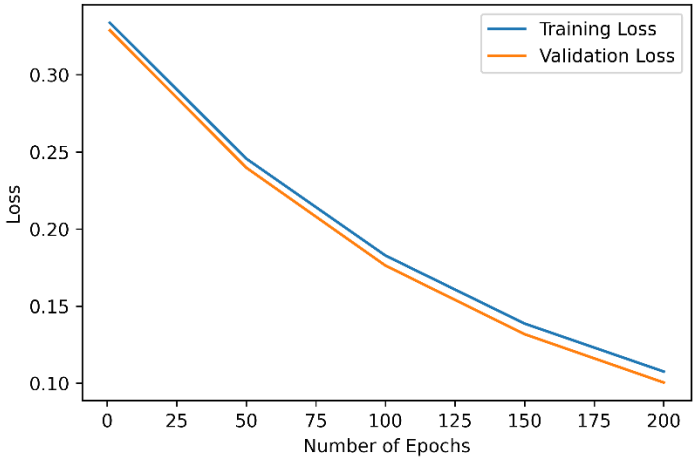

Figure 3: Training and validation losses for a 1 layer with 8 nodes neural network.

4. The network from part 3 was extended with two additional hidden layers. The first hidden layer had 8 nodes, the second has 100 nodes and the third has 8 nodes. In addition, the ReLU activation function was used instead of the Tanh. The NN was trained for 200 epochs with a learning rate of 0.001. The model reported training and validation losses of 0.0844 and 0.0797 respectively, and the total training time and model size were 0.27245 seconds and 0.0067 MB respectively. Figure 4 shows the training and validation losses for 200 epochs with a learning rate of 0.001. These results suggest that the model should be trained longer, that is the number of epochs should be increased. In addition, the model in part 4 is three

time larger in terms of MB compared to the model in part 3. Also, this model reported lower training and validation losses compared to the model in part 3.

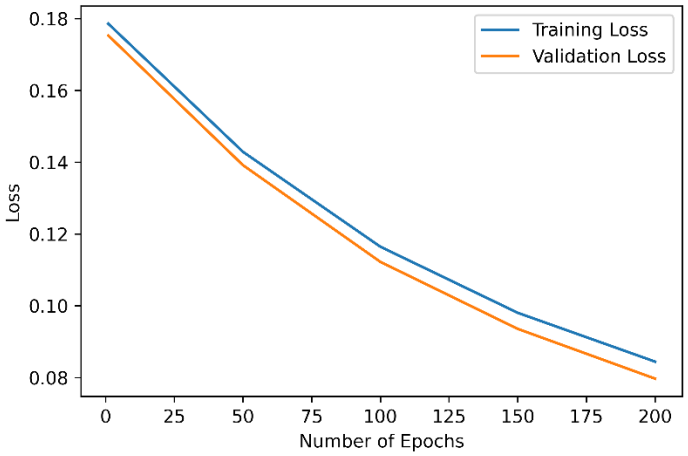

Figure 4: Training and validation losses for a 3-layer with 8, 100, and 8 nodes, respectively, neural network.