Homework 5 Report

Gilberto Feliu
Student ID: 801257813
https://github.com/gfeliu-rgb/Intro-to-ML-ECGR-4105

Problem 1A

In this problem, I modified the temperature prediction example from the lecture by turning the linear model into a nonlinear model. Originally, the model predicted output using the equation:

$$y = w_1 t_u + b$$

I updated this to include a quadratic term:

$$y = w_2 t_u^2 + w_1 t_u + b$$

The training loop was adjusted to accommodate this change, while the rest of the code remained the same. After adding the quadratic term, the model's behavior changed—it became more flexible but also more sensitive to learning rate. At higher learning rates, convergence became unstable.

Problem 1B

The modified quadratic model was trained for 5000 epochs across multiple learning rates: 0.1, 0.01, 0.001, and 0.0001. The dataset and normalization method were identical to those used in the lecture. During training, the loss was recorded every 500 epochs.

The results showed that learning rates above 0.0001 caused numerical instability (NaN losses). Only the smallest rate, 0.0001, produced stable convergence.

Key Results

- Best learning rate: 0.0001
- Final loss: 3.8615

- Final parameters: `tensor([-0.8881, 0.5570, -0.8753])`
- All larger learning rates produced NaN results

Problem 1C

For this part, I used the best-performing learning rate (0.0001) from Part 1B and compared the nonlinear model's results with the original linear model from the lecture.

Both models were plotted on the same graph with T_U (Unnormalized Temperature) on the X-axis and T_C (Celsius Temperature) on the Y-axis. The linear model fit the data points more accurately. Numerically, the linear model achieved a minimum loss of 2.928, while the nonlinear model's loss was 3.8615.

This confirmed that the original linear model performed better for this dataset.

Problem 2A

For this section, I used the housing dataset and selected five features: **area**, **bedrooms**, **bathrooms**, **stories**, and **parking**. All inputs were normalized, and the dataset was split into 80% training and 20% validation.

The model used the equation:

$$U = W_5X_5 + W_4X_4 + W_3X_3 + W_2X_2 + W_1X_1 + B$$

After training, the weight values were analyzed to identify the most influential feature. The **area** variable had the largest weight (2,933,135.8), making it the strongest predictor of housing price.

Problem 2B

Next, I trained the same model for 5000 epochs using different learning rates (0.1, 0.01, 0.001, and 0.0001). Each version was evaluated on both training and validation losses.

The learning rate of 0.1 converged the fastest and reached the lowest loss (2.29×10^{12}) around epoch 2000. Smaller learning rates converged more slowly and produced higher losses, showing that a more aggressive learning rate was beneficial for this dataset.

Summary

- Best learning rate: 0.1
- Lowest validation loss: 2.29×10^{12}

- Most important input feature: Area

Problem 3A

Here, I built a fully connected neural network (NN) to predict housing prices using the same normalized dataset as before. The network had one hidden layer with 8 neurons and ReLU activation. It was trained for 200 epochs using a learning rate of 0.01 with an 80/20 training-validation split.

Results

- Training time: 0.333 s
- Final training loss: 3.98×10^{12}
- Evaluation accuracy (R^2): -0.3001

The negative R^2 indicates that the model did not learn meaningful relationships in the data, performing worse than random guessing.

Problem 3B

For this problem, I expanded the neural network from Problem 3A by adding two additional hidden layers, resulting in a structure of: Input $\rightarrow$ 8 $\rightarrow$ 8 $\rightarrow$ 8 $\rightarrow$ Output, all with ReLU activations.

When trained with the same learning rate (0.01), the model's loss became NaN, meaning it diverged. Using a much smaller rate (1e-9) allowed training to complete, but performance worsened overall.

Results

- Learning rate 0.01: Loss = NaN, R^2 = NaN
- Learning rate 1e-9: Loss = 2.52×10^{13}, R^2 = -4.96
- Training time: 0.28 s

Although the deeper model trained quickly, it failed to generalize or capture meaningful relationships. The high loss and negative accuracy confirm that the training was ineffective. This was not a case of overfitting—the model simply failed to learn the data patterns.

Conclusion

Across all experiments, simpler and linear models performed better than nonlinear or deeper neural networks. Larger learning rates caused divergence, while smaller ones improved stability but slowed convergence.

Overall, the linear regression model from Problem 1 and the linear housing model from Problem 2 produced the most stable and interpretable results.