Nathon Tadeo
Student ID: 801265462
11/15/24
Homework 5
Github: https://github.com/nathon-tadeo/Intro-to-ML/blob/main/homework_5_intro_to_ml.ipynb

Problem 1 (30 pts):
Let's change our model to a nonlinear system in our temperature prediction example. Consider the following description for our model: w2 * t_u ** 2 + w1 * t_u + b.
1a. Modify the training loop properly to accommodate this redefinition.

Using the "ML-Lecture-12-Gradient Descent, BackPropagation in Pytorch.pdf" for the temperature prediction example and structure, I converted the linear system, t_c = w * t_u + b, into a nonlinear system by using the model given "w2 * t_u ** 2 + w1 * t_u + b". Using Lecture 12 as guidance, the training loop was modified to accommodate the new nonlinear system, and the input was normalized. The training loop computed the gradients for w2,w1, and b using PyTorch's "autograd" and calculated the loss using MSE.

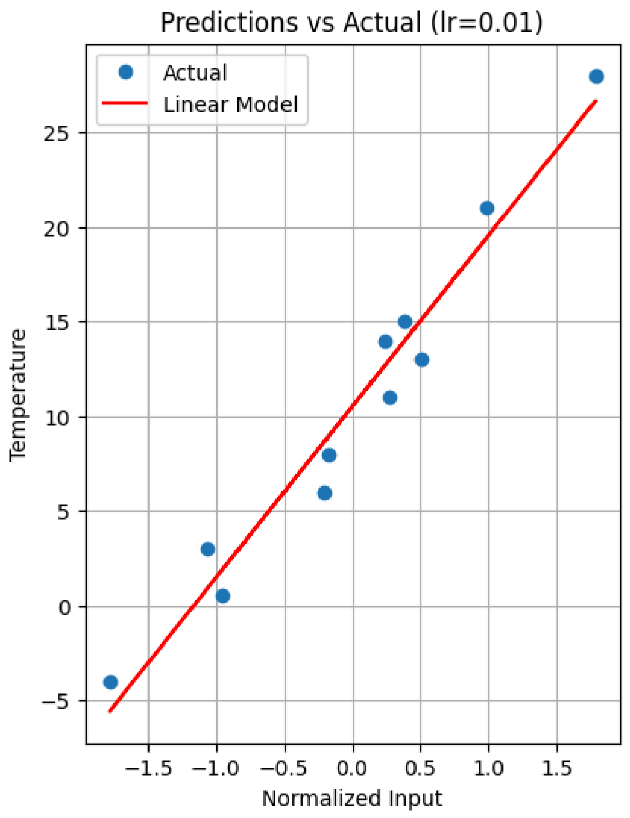
1b. Use 5000 epochs for your training. Explore different learning rates from 0.1 to 0.0001 (you need four separate trainings). Report your loss for every 500 epochs per training.

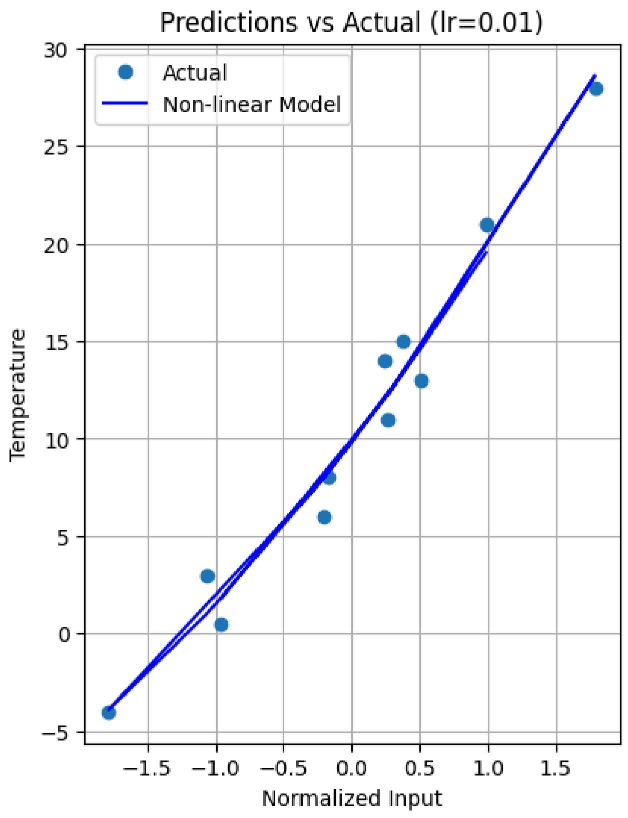
Using the preprocessing from the previous part, the model was trained for 5000 epochs with four different learning rates: 0.1, 0.01, 0.001, and 0.0001. For each learning rate, the loss was calculated and printed every 500 epochs. The model trained with a learning rate of 0.01 achieved the lowest training loss of "2.0907194" at the 5000th epoch. The model trained with a learning rate of 0.0001 had the highest training loss of "24.4275" at the 5000th epochs. The poor performance is likely due to the extremely low learning rate. The model was unable to fully optimize the parameters because the 5000 epochs are too large to accommodate the learning rate. The learning rate of 0.01 is the optimal choice, achieving both efficient and stable convergence within the 5000 epochs.

1c. Pick the best non-linear model and compare your final best loss against the linear model that we did during the lecture. For this, visualize the non-linear model against the linear model over the input dataset, as we did during the lecture. Is the actual result better or worse than our baseline linear model?

The non-linear model outperformed the linear model in terms of training loss. The linear model had a final loss of 2.9276 while the non-linear had a final loss of 2.0907 both at epoch 5000. The nonlinear model had an improvement of approximately 0.8369. The models were plotted against the actual data for visual comparison. The nonlinear model showed an advantage by gravitating more closely toward the actual temperatures between -0.5 and 0.0, while the linear model could not capture the curvature of the data due to its inherent linearity. A non-linear model should outperform a linear model due to its flexible nature. The visualization is shown in the figure below.

Linear: Epoch 5000, Loss 2.927645
Nonlinear: Epoch 5000, Loss: 2.09071946144104





Problem 2 (40 pts):

2a. Develop preprocessing and a training loop to train a linear regression model that predicts housing price based on the following input variables: area, bedrooms, bathrooms, stories, parking For this, you need to use the housing dataset. For training and validation use 80% (training) and 20% (validation) split. Identify the best parameters for your linear regression model, based on the above input variables. In this case, you will have six parameters:

The training split and preprocessing for this question simply followed the same format and structure as homework 2. The processing differed when using PyTorch with the usage of torch tensors. This included reshaping the target variable y to match the expected dimensions for the training loop.

2b. Use 5000 epochs for your training. Explore different learning rates from 0.1 to 0.0001 (you need four separate trainings). Report your loss and validation accuracy for every 500 epochs per training. Pick the best linear model.

The linear regression model was implemented using Pytorch's "nn.Module" This model uses a single linear layer to map the input features to the target variable. The training loop utilized the SGD as the optimizer and the MSE as the loss function. Similar to problem 1, the model was run for 5000 epochs. The validation and training loss was calculated every 500 epochs for the learning rates: 0.1, 0.01, 0.001, and 0.0001.

The results are as follows:

- **LR=0.1**: Best Validation Loss = 1,173,790,261,248.0
- **LR=0.01**: Best Validation Loss = 1,173,788,950,528.0
- **LR=0.001**: Best Validation Loss = **1,173,262,434,304.0** (lowest loss)
- **LR=0.0001**: Best Validation Loss = 4,150,343,499,776.0

The training loss and validation losses were the lowest for the learning rate of 0.001. The 0.0001 demonstrated a much slower convergence and a higher loss, and 0.1 and 0.01 failed to outperform the learning rate of 0.001.

2c. Compare your results against the linear regression you did in homework 1. Do you see meaningful differences?

When comparing the results of convergence with the model using PyTorch and the model without using PyTorch the key differences lie in convergence speed and optimization methods. The model not using PyTorch seems to have a steeper rate of convergence over the 1000 iterations with standardization and regularization. The Pytorch model had a more gradual convergence within the 5000 Epochs. This gradual convergence could be due to the usage of Stochastic Gradient Descent (SDG). SGD updates weights in smaller, noisier steps and requires more epochs to stabilize.

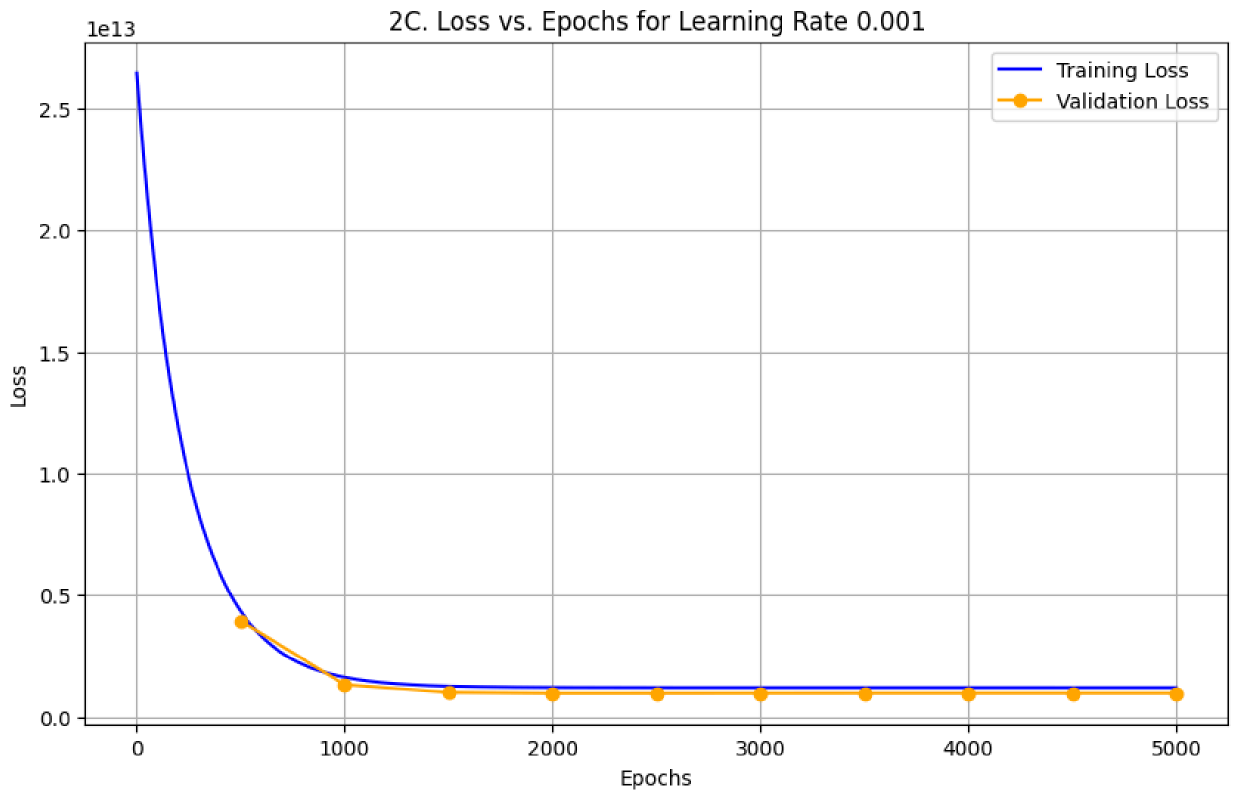


Figure 2.1 Training and Validation Loss with PyTorch

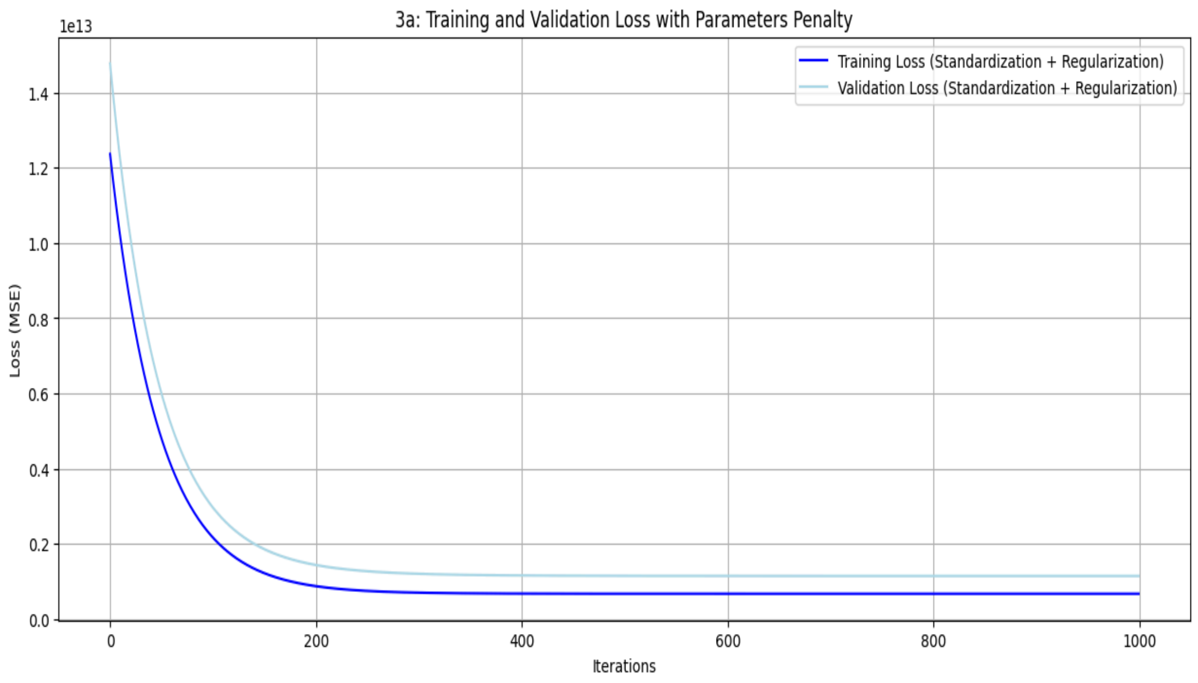


Figure 2.2 Training and Validation Loss without PyTorch

Problem 3 (30 pts):
Repeat all sections of problem 2, this time use all the input features from the housing price dataset.

The processing was relatively the same for this problem, except with the inclusion of all the input features from the housing price dataset. The features added for training were: area, bedrooms, bathrooms, stories, mainroad, guestroom, basement, hotwaterheating, airconditioning, parking, prefarea. With all these features, the data was split normalized, and converted to tensors. The validation and training loss was also calculated every 500 epochs over 5000 epochs for the learning rates: 0.1, 0.01, 0.001, and 0.0001.

The results are as follows:

- **LR=0.1**: Best Validation Loss = 968,194,916,352.0
- **LR=0.01**: Best Validation Loss = 968,194,195,456.0
- **LR=0.001**: Best Validation Loss = **962,407,956,480.0** (lowest loss)
- **LR=0.0001**: Best Validation Loss = 3,939,853,664,256.0

The 0.001 learning rate achieved the lowest validation loss, while 0.0001 showed slower convergence and higher loss. Learning rates of 0.1 and 0.01 also underperformed compared to 0.001.

When comparing the results with the model from Homework 2 using the whole housing price dataset, the primary differences were the same as the 2C. The non-PyTorch model converged faster while the Pytorch model converged more gradually due to using SGD. For both problems 2 and 3, the Pytorch model exhibited a smoother and slower convergence, while the non-PyTorch model converged more quickly.

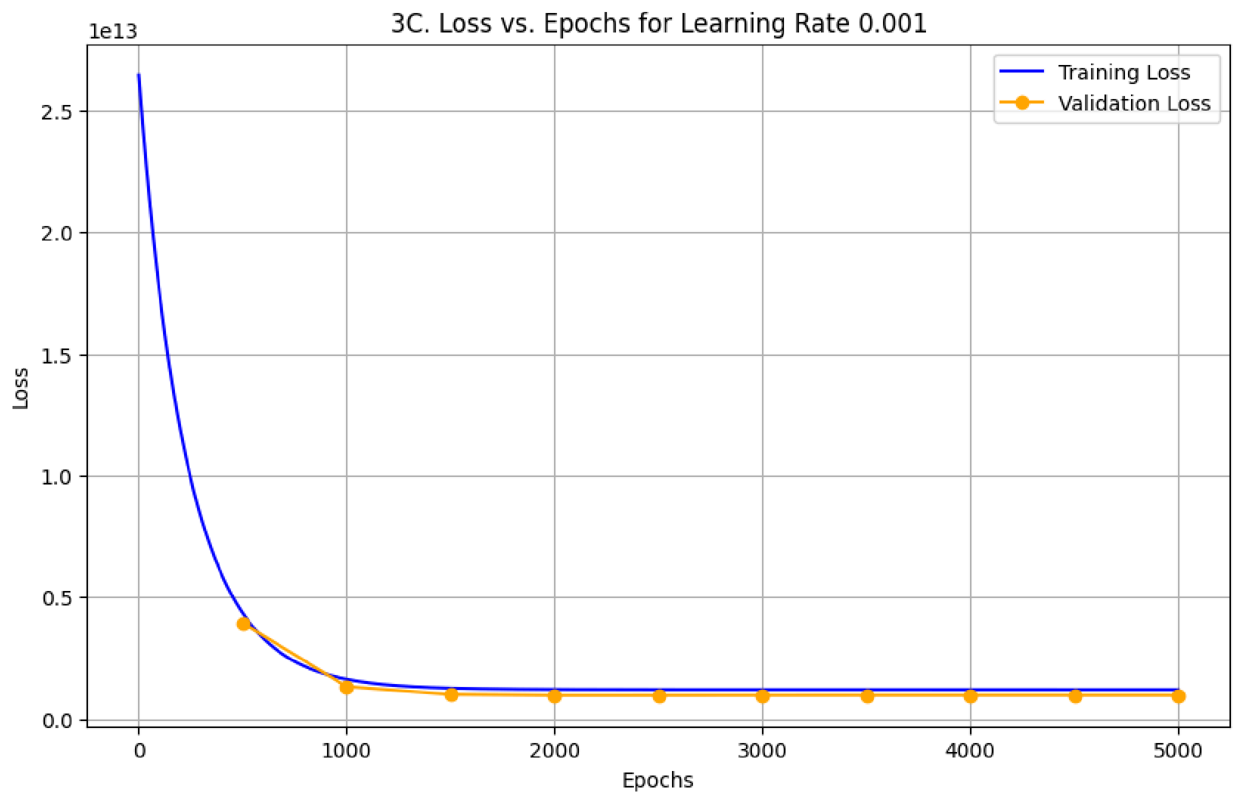


Figure 3.1 Training and Validation Loss with PyTorch

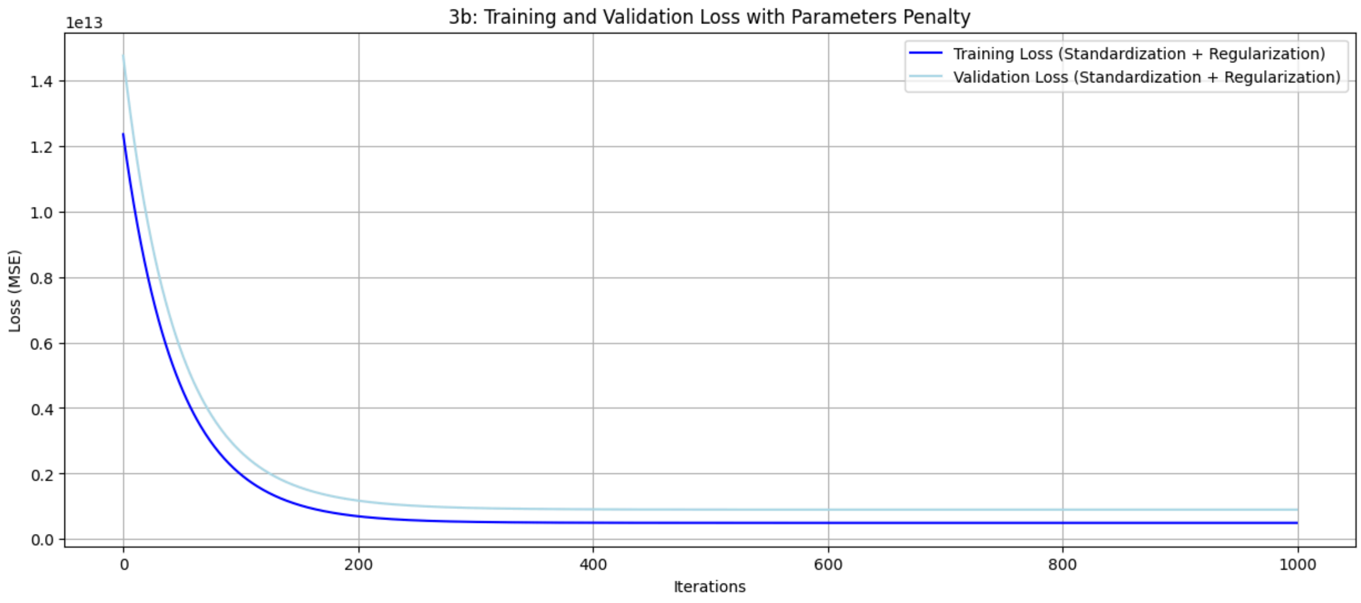


Figure 3.2 Training and Validation Loss without PyTorch