Chunyuan Shen

ID:801322013

Homework 1

https://github.com/cryyin/ECGR5105/tree/main/homework1

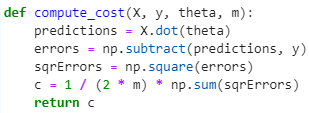
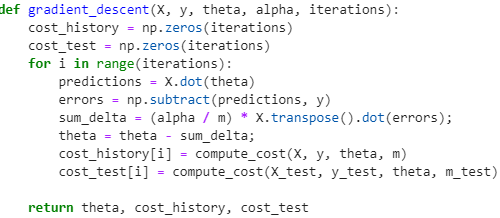
1.a) Develop a gradient decent training and evaluation code that predicts housing price based on the following input variables:

area, bedrooms, bathrooms, stories, parking

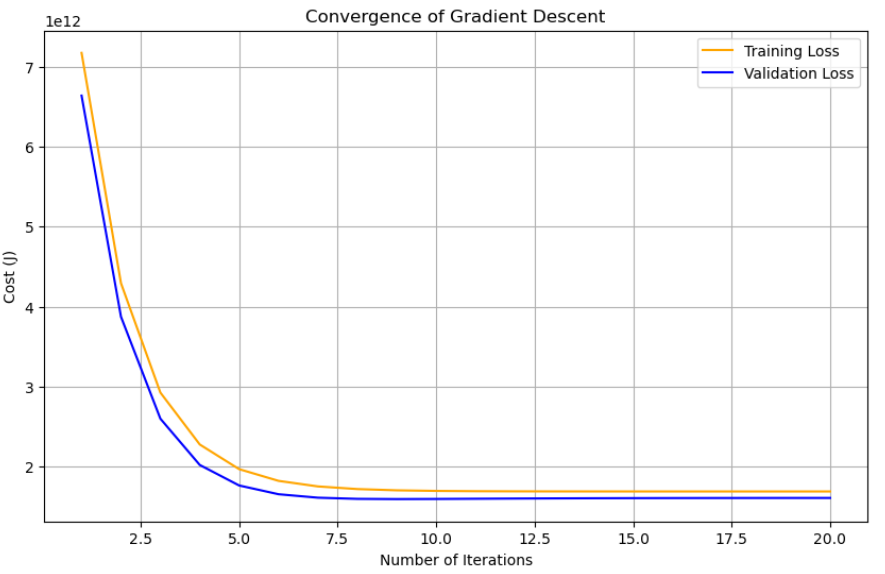
Identify the best parameters for your linear regression model, based on the above input variables.

Plot the training and validation losses (in a single graph, but two different lines). For the learning rate, explore different values between 0.1 and 0.01 (your choice). Initialize your parameters (thetas to zero). For the training iteration, choose what you believe fits the best.

We can suppose that y=θ+θ1\*x1+θ2\*x2…. , so we can substitute the value into the function. So wo got error from prediction and real Y, than we can compute the loss by using J = 1 / (2 \* m) \* sum(sqrErrors), and save the cost of each iteration.

I choose 0.00000001 as learning rate because this data is too easy to train and if learning rate is 0.01 cost will going to nan. And here is the picture of training and validation loss:

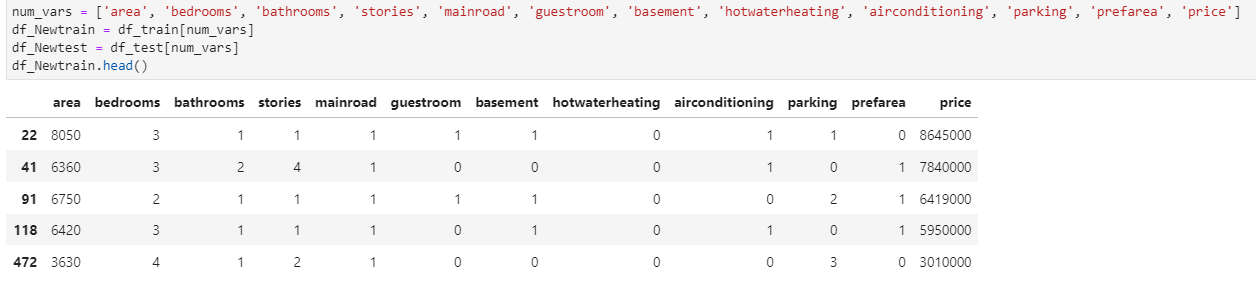


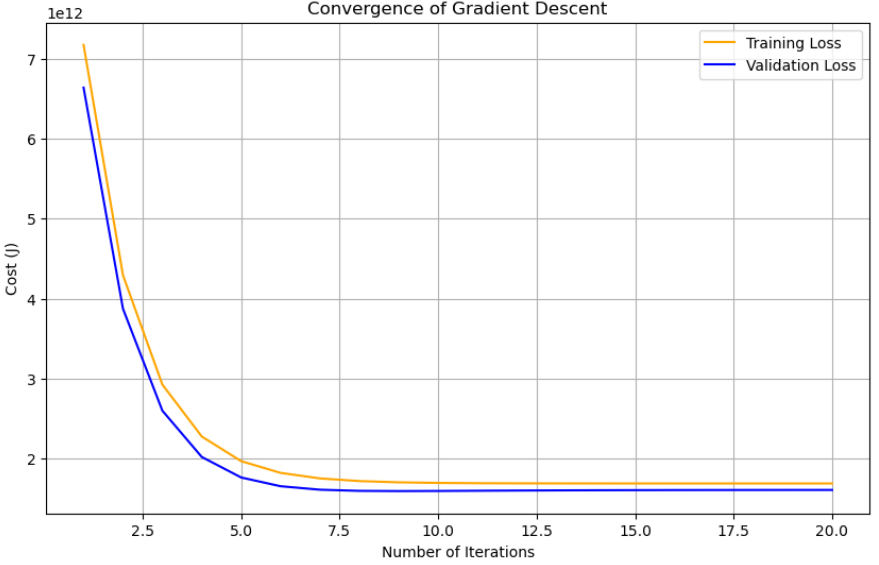
1.b) Develop a gradient decent training and evaluation code that predicts housing price based on the following input variables:

Area, bedrooms, bathrooms, stories, mainroad, guestroom, basement, hotwaterheating, airconditioning, parking, prefarea

Identify the best parameters for your linear regression model, based on the above input variables.

Plot the training and validation losses (in a single graph, but two different lines) over your training iteration. Compare your linear regression model against problem 1 a. For the learning rate, explore different values between 0.1 and 0.01 (your choice). Initialize your parameters (thetas to zero). For the training iteration, choose what you believe fits the best.

This is similar with 1a, we just need to add other 6 variables and 6 thetas.

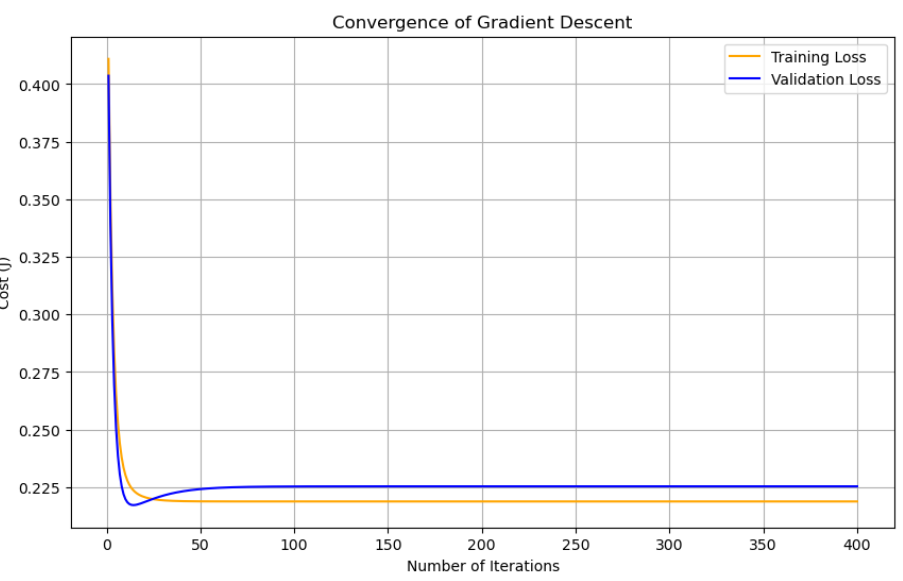
And here is the picture of training and validation loss:

**Problem 2 (30 points)**

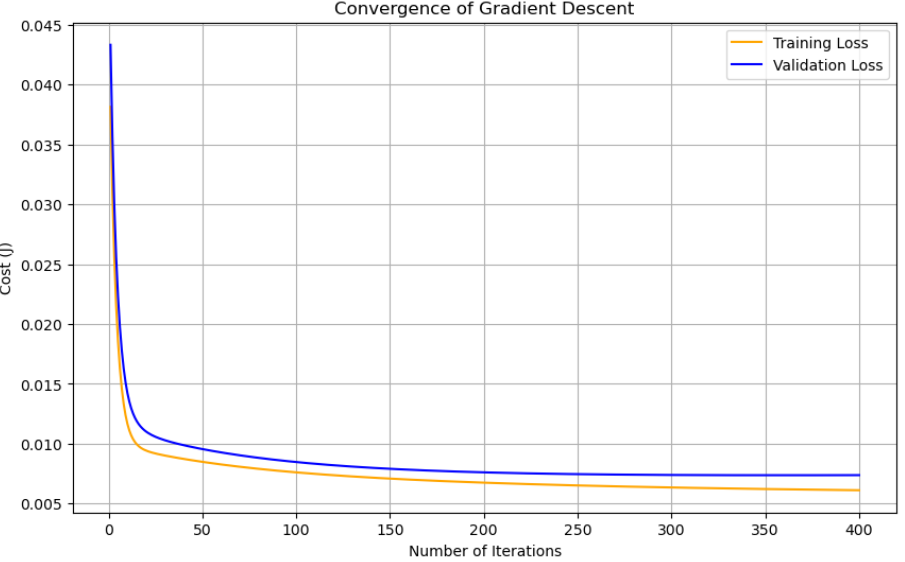
2.a) Repeat problem 1 a, this time with input normalization and input standardization as part of your pre-processing logic. You need to perform two separate trainings for standardization and normalization.

Plot the training and validation losses for both training and validation set based on input standardization and input normalization. Compare your training accuracy between both scaling approaches as well as the baseline training in problem 1 a. Which input scaling achieves the best training? Explain your results.

Firstly we should look at the plot of standardization and normalization:

standardization :

normalization:



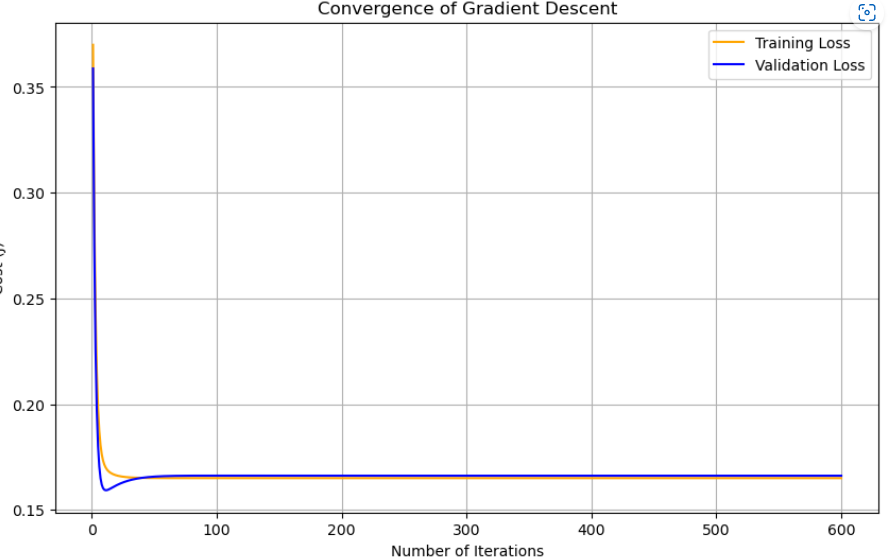
as we can see, standardization’s loss get down really fast, but finally loss is about 0.225 and validation loss go down but get up, that means there is overfitting in the model.

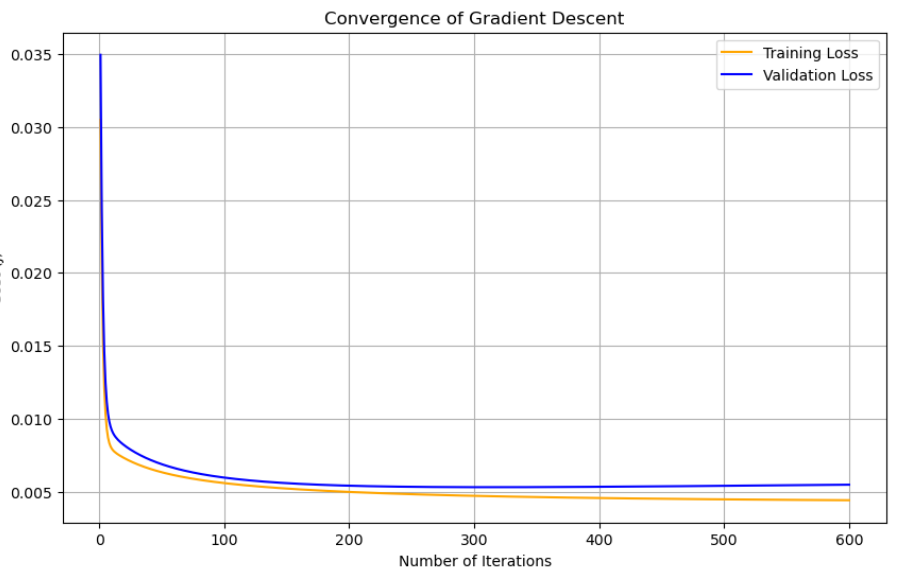
For normalization, finally loss is about 0.005, much lower than 0.225 and no overfitting, so I choose normalization in question2 and 3.

2.b) Repeat problem 1 b, this time with input normalization and input standardization as part of your pre-processing logic. You need to perform two separate trainings for standardization and normalization.

Plot the training and validation losses for both training and validation set based on input standardization and input normalization. Compare your training accuracy between both scaling approaches as well as the baseline training in problem 1 b. Which input scaling achieves the best training? Explain your results.

Similar with 1b and 2a, we just need to add other 6 variables and 6 thetas and try different functions:

standardization:

normalization 

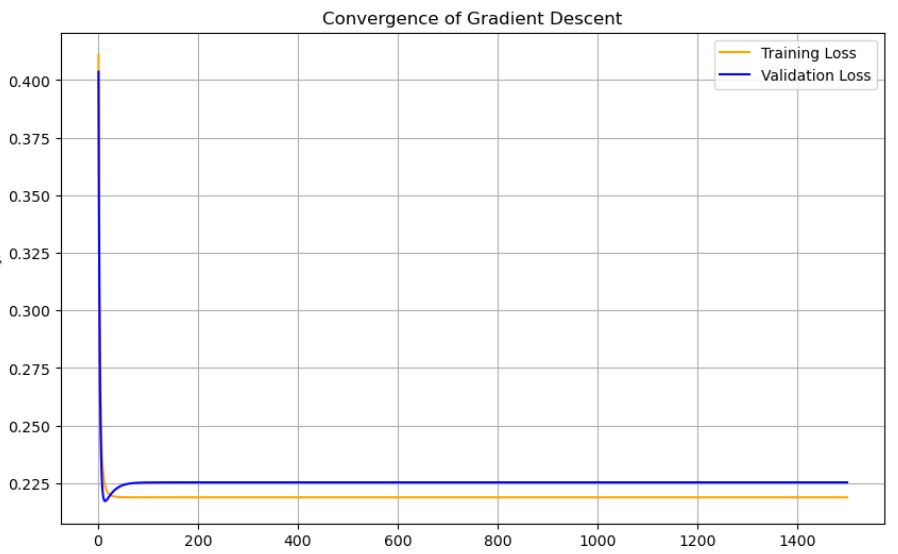
the final loss of normalization is still lower than standardization.

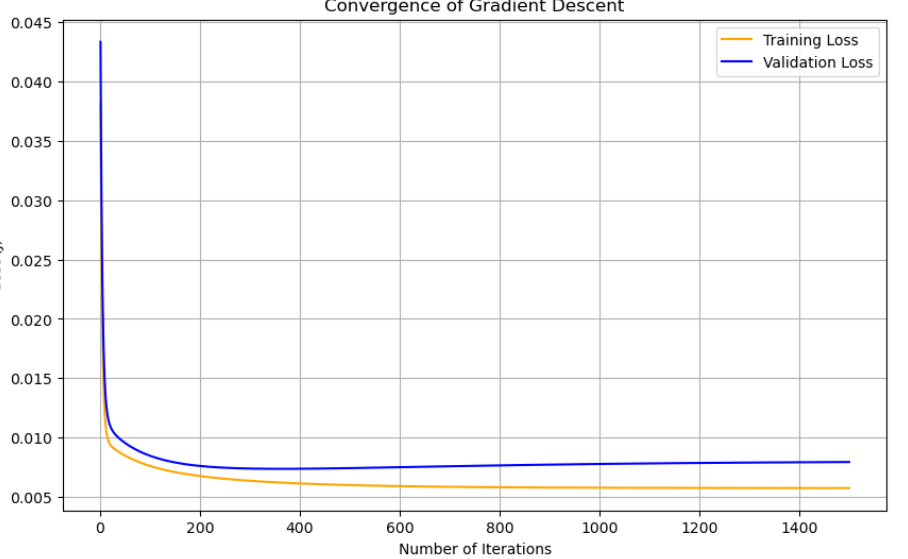
**Problem 3 (40 points)**

3.a) Repeat problem 2 a, this time by adding parameters penalty to your loss function. Note that in this case, you need to modify the gradient decent logic for your training set, but you don’t need to change your loss for the evaluation set.

Plot your results (both training and evaluation losses) for the best input scaling approach (standardization or normalization). Explain your results and compare them against problem 2 a.

For question 3 I add lamda into cost function:

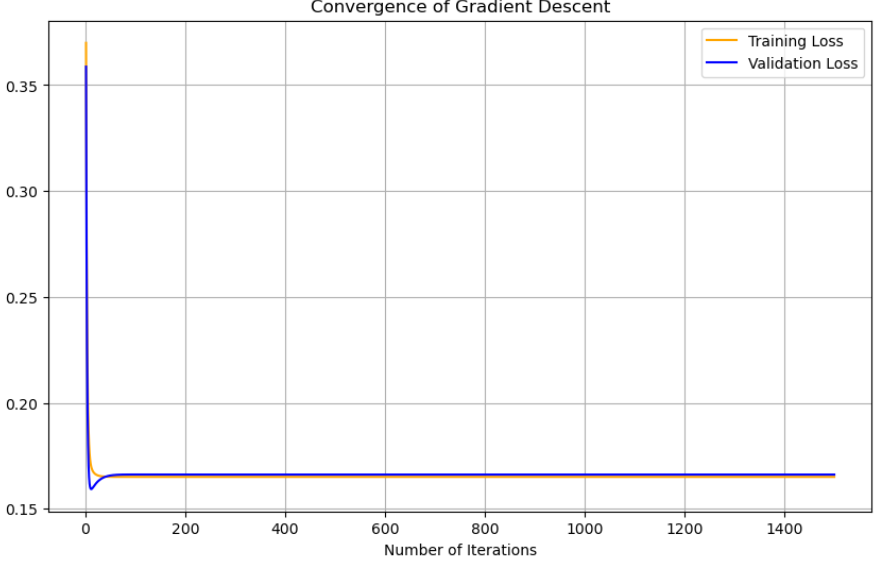
J = 1 / (2 \* m) \* (np.sum(sqrErrors) + Lambda \* np.sum(sqrTheta\_new))



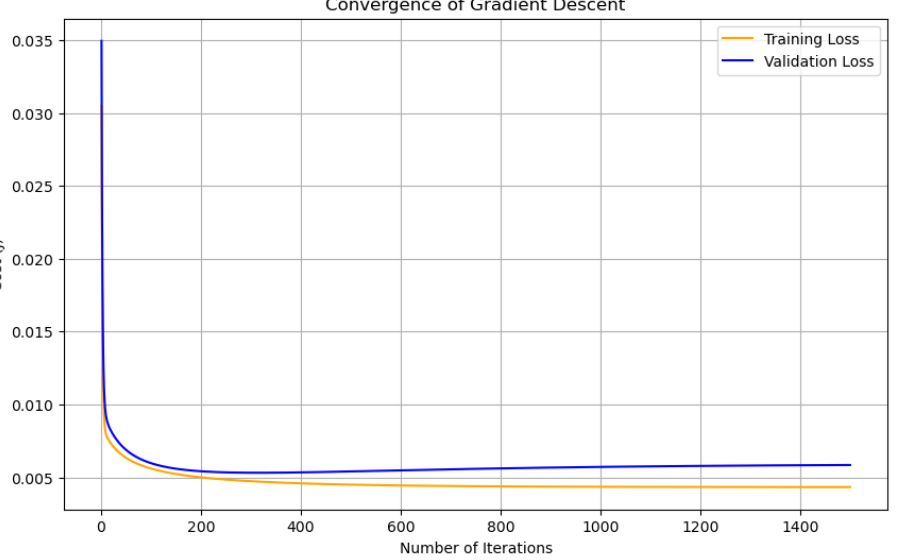
Standardization still similar with question 2a and 2b, standardization still higher than normalization.

3.b) Repeat problem 2 b, this time by adding parameters penalty to your loss function. Note that in this case, you need to modify the gradient decent logic for your training set, but you don’t need to change your loss for the evaluation set.

Plot your results (both training and evaluation losses) for the best input scaling approach (standardization or normalization). Explain your results and compare them against problem 2 b.

standardization:

normalization:



the plot still supports the result of question 2a and 2b: normalization is better than standardization, it has less overfitting. But I noticed that when we have 12 variables, the loss of standardization goes down, maybe if we have more variables it can perform better than normalization.