(CS 534) Machine Learning

Implementation Assignment 1:

Linear Regression

A-young Kang (Student ID: 933-610-350)

Part 0 (20 pts):

(a) Remove the ID feature. Why do you think it is a bad idea to use this feature in learning?

The ID feature has nothing to do with the price of houses. Therefore, it should be removed when training the data. If we use this feature in learning, it could adversely affect our model accuracy, which leads to bad predictions.

(b) Split the date feature into three separate numerical features: month, day, and year. Can you think of better ways of using this date feature?

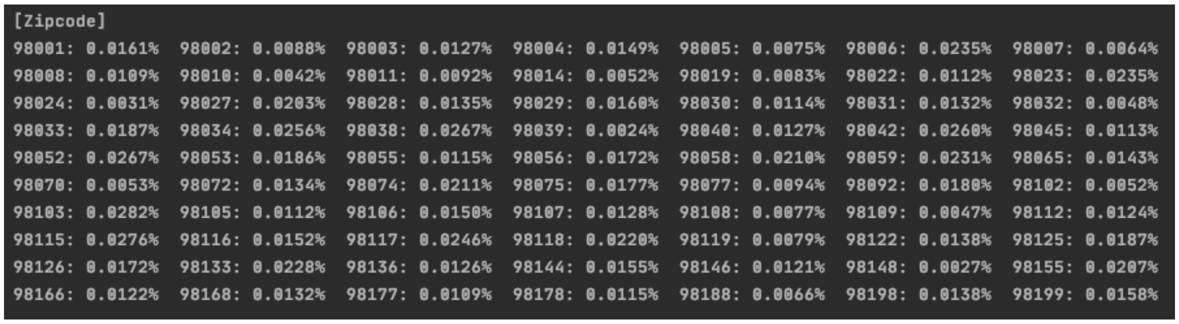
By splitting the date feature into three separate features, we would be able to capture some trends in house prices over a certain period of time (e.g. month or year) since day and month features are cyclic. We can also treat the year feature as categorical, which would let us identify specific years associated with high/low target value homes.

(c) Build a table that reports the statistics for each feature.

	Mean	Standard Deviation	Range
month	6.59	3.11	11.00
day	15.80	8.62	30.00
year	2014.32	0.47	1.00
bedrooms	3.38	0.94	32.00
bathrooms	2.12	0.77	7.25
sqft_living	2080.22	911.29	9520.00
sqft_lot	15089.20	41201.83	1650787.00
floors	1.50	0.54	2.50
view	0.23	0.76	4.00
sqft_above	1793.10	830.82	8490.00
sqft_basement	287.12	434.98	2720.00
year_built	1971.12	29.48	115.00
year_renovated	81.23	394.36	2015.00
zipcode	98078.29	53.52	198.00
lat	47.56	0.14	0.62
long	-122.21	0.14	1.19
sqft_living15	1994.33	691.87	5650.00
sqft_lot15	12746.32	28239.83	870540.00

	Category	Percentage (%)
Waterfront	0	0.993
	1	0.007
Condition	1	0.0013
	2	0.0076
	3	0.653
	4	0.2569
	5	0.0812
Grade	4	0.0011
	5	0.0105
	6	0.0933
	7	0.413
	8	0.2838
	9	0.1182
	10	0.0547
	11	0.021
	12	0.0039
	13	0.0005

Table 1 Statistics for each feature



```
[Zipcode]
98001: 0.0161%  98002: 0.0088%  98003: 0.0127%  98004: 0.0149%  98005: 0.0075%  98006: 0.0235%  98007: 0.0064%
98008: 0.0109%  98010: 0.0042%  98011: 0.0092%  98014: 0.0052%  98019: 0.0083%  98022: 0.0112%  98023: 0.0235%
98024: 0.0031%  98027: 0.0203%  98028: 0.0135%  98029: 0.0160%  98030: 0.0114%  98031: 0.0132%  98032: 0.0048%
98033: 0.0187%  98034: 0.0256%  98038: 0.0267%  98039: 0.0024%  98040: 0.0127%  98042: 0.0260%  98045: 0.0113%
98052: 0.0267%  98053: 0.0186%  98055: 0.0115%  98056: 0.0172%  98058: 0.0210%  98059: 0.0231%  98065: 0.0143%
98070: 0.0053%  98072: 0.0134%  98074: 0.0211%  98075: 0.0177%  98077: 0.0094%  98092: 0.0180%  98102: 0.0052%
98103: 0.0282%  98105: 0.0112%  98106: 0.0150%  98107: 0.0128%  98108: 0.0077%  98109: 0.0047%  98112: 0.0124%
98115: 0.0276%  98116: 0.0152%  98117: 0.0246%  98118: 0.0220%  98119: 0.0079%  98122: 0.0138%  98125: 0.0187%
98126: 0.0172%  98133: 0.0228%  98136: 0.0126%  98144: 0.0155%  98146: 0.0121%  98148: 0.0027%  98155: 0.0207%
98166: 0.0122%  98168: 0.0132%  98177: 0.0109%  98178: 0.0115%  98188: 0.0066%  98198: 0.0138%  98199: 0.0158%
```

Figure 1 Percentage for the zip code feature

(d) Based on the statistics and your understanding of these features/housing prices, which set of features do you expect to be useful for this task? Why?

I think all the features provided are useful for this task but sqft_living15(Living room area) and grade are the most important because area is one of most important factors in determining housing prices and all the information about the house is reflected in the overall grade given to the house unit.

(e) Normalize all numerical features (excluding the housing prices y) to the range 0 and 1 using the training data.

I normalized all numerical features using following equation in method 'normalize'. I used ordinal features (grade and condition) and the zip code as numerical this time.

$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$ where z_i is the normalized feature for example x_i

2Part 1 (40 pts).

(a) Which learning rate or learning rates did you observe to be good for this particular dataset? What learning rates (if any) make gradient descent diverge? Report your observations together with some example curves showing the training MSE as a function of training iterations and its convergence or non-convergence behaviors.

Based on my observation, the learning rates of 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4} are good for this dataset. When using 10^{0}, the gradient descent diverges. It seems that the learning rates of 10^{-5}, 10^{-6}, 10^{-7} make gradient descent converge but they are so slow that the convergence condition ($\epsilon = 0.5$) is not satisfied within the maximum number of iterations (I set the maximum number of iterations as 700,000). Figure 2 shows the training MSE as a function of training iterations. When the learning rate is smaller than 10^{-1}, the MSE decreases as the number of iterations increases, but the smaller the learning rate is, the slower the algorithm converges.

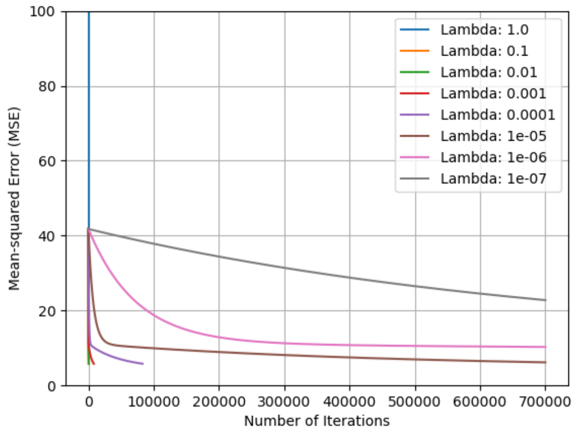


Figure 2 Mean-squared Error (MSE) vs. Number of Iterations

(b) For each learning rate that worked for you, Report the MSE on the training data and the validation data respectively and the number of iterations needed to achieve the convergence condition for training. What do you observe? Between different convergent learning rates, how should we choose one if the MSE is nearly identical?

Learning rate	10^{-1}	10^{-2}	10^{-3}	10^{-4}
MSE on the training data	5.7967	5.8079	5.8090	5.8092
Number of iterations needed to achieve the convergence condition	84 (norm: 0.4974)	830 (norm: 0.4997)	8,290 (norm: 0.4999)	82,887 (norm: 0.4999)

Table 2 Mean-squared Error on the Training Data and Number of Iterations

For the learning rates of 10^{-5}, 10^{-6} and 10^{-7}, the convergence condition is not achieved within 700,000 iterations, so the MSE's on the training data are quite large (6.1738, 10.2767, 22.7855, respectively). The MSE is almost the same for all learning rates. I think 10-1 is good for this data set because it only takes 84 iterations to achieve the convergence condition and the MSE is the smallest.

Learning rate	10^{-1}	10^{-2}	10^{-3}	10^{-4}
MSE on the validation data	6.0734	6.1035	6.1069	6.1073

Table 3 Mean-squared Error on the Validation Data

For the learning rates of 10^{-5}, 10^{-6} and 10^{-7}, the MSE's on the validation data are 6.4665, 10.3732, 22.7040, respectively.

(c) Use the validation data to pick the best converged solution, and report the learned weights for each feature. Which features are the most important in deciding the house prices according to the learned weights? Compare them to your pre-analysis results (Part 0 (d)).

Table 4 shows the learned weights for each feature. The top 3 features with the highest |weight| are grade, view and sqft_living15. Grade and sqft_living are consistent with our pre-analysis results. I found out that how many times the house has been viewed is an important factor in determining housing price from this observation, which is kind of surprise.

Feature	Weight (learning rate = 10^{-1})	Feature	Weight (learning rate = 10^{-1})
grade	3.233111806	condition	0.674361914
view	2.619704327	bedrooms	0.304792542
sqft_living15	2.568067833	dummy	0.186705042
lat	2.48868746	year	0.101384584
sqft_living	2.432419393	sqft_lot15	0.096048473
sqft_above	2.174340302	sqft_lot	0.087123778
bathrooms	1.837291964	month	-0.076837021
sqft_basement	1.726648332	long	-0.129008721
floors	1.424070421	day	-0.336364342
waterfront	0.809085257	zipcode	-0.347815912
yr_renovated	0.724651394	yr_built	-0.537106712

Table 4 Learned Weights of Each feature (Learning Rate = 10^{-1})

Part 2 (20 pts). Training with non-normalized data Use the preprocessed data but skip the normalization. Plot the training MSE and validation MSE respectively as a function of the number of iterations. What do you observe? Specify the learning rate value (if any) that prevents the gradient descent from exploding? Compare between using the normalized and the non-normalized versions of the data. Which one is easier to train and why?

Figure 3 and Figure 4 show the training MSE and the validation MSE, respectively. For all values of learning rate that works (e.g., from 100 to 10^{-4}), we can see that the training is diverging. I found out that when the learning rate is less than 10^{-11}, the gradient descent starts to converge (Figure 5), which means the non-normalized version of the data requires much smaller learning rate in order to converge and thus, the normalized data is much easier to train. This is because the difference in ranges of features will cause different step sizes for each feature; therefore, the gradient descent algorithm would oscillate a lot back and forth without normalization. By scaling the data, the steps for gradient descent are updated at the same rate for all the features and the gradient descent moves smoothly towards the local minimum.

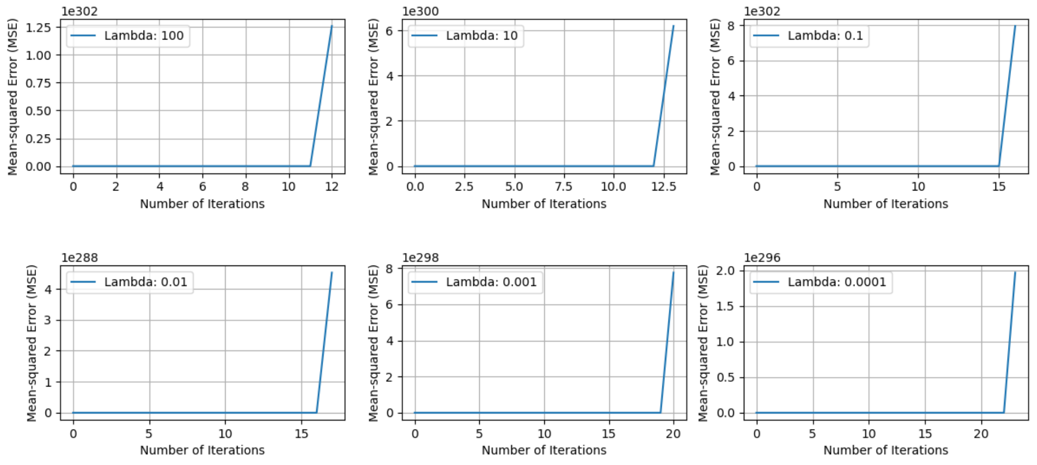


Figure 3 Training Mean-squared Error (MSE) without Normalization

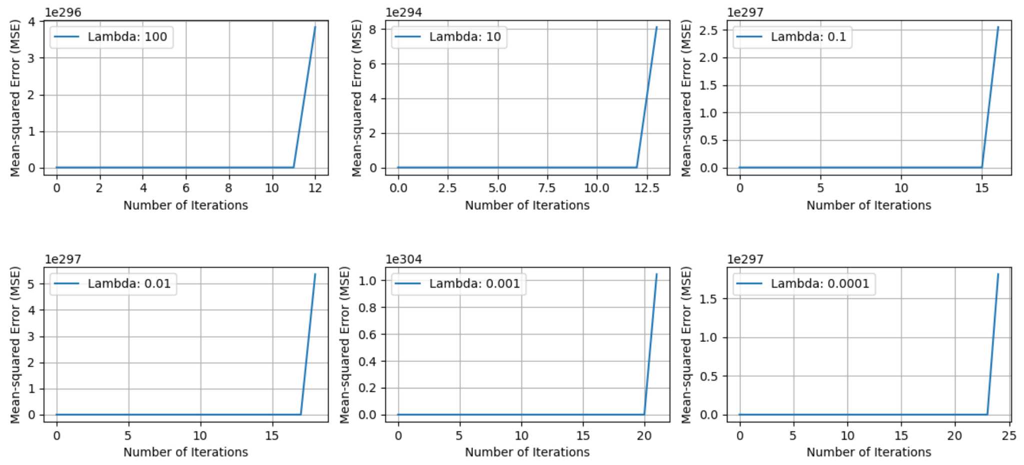


Figure 4 Validation Mean-squared Error (MSE) without Normalization

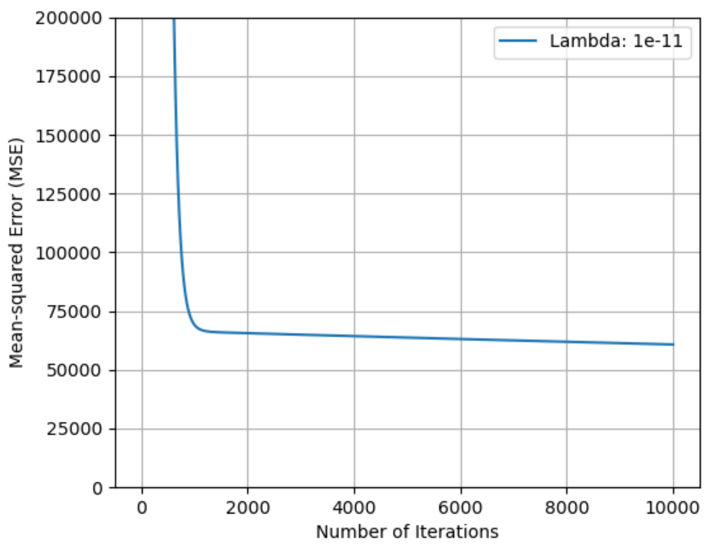


Figure 5 Training Mean-squared Error (MSE) without Normalization (learning rate = 10^{-11})

Part 3 (10 pts). Feature engineering (exploration) Similar to Part 0(b), list any modifications to the features provided that you think may be useful for making better predictions. Implement 3 (or more) ideas you have, and report the results. Please report what you have tried and what (if any) effect it has on the predictions.

I encoded 'year' as categorical. When using year as a numerical feature, it would capture information like the housing price goes up each year. However, for example, during the COVID-19 pandemic, the price may not go up, so treating year as categorical would capture information about how the COVID-19 year will affect the housing prices. I think during the peak moving season, housing prices would rise, so I added 'quarter' feature to see quarterly price trends. I also added a new feature that is a product of grade and sqft_living15 because I thought two features were the most import ones and thus a product of two features gave us a better prediction of housing price. When applying these three ideas, the MSE's on both the training and validation data slightly decrease (Table 5).

Learning rate	10^{-1}	10^{-2}	10^{-3}	10^{-4}
MSE on the training data	5.6130	5.6174	5.6188	5.6189
MSE on the validation data	5.8214	5.8482	5.8520	5.8523

Table 5 Training and Validation MSE when adding features

I tried standardization when scaling features instead of normalization. ($z_i = \frac{x_i - \mu}{\sigma}$) Standardization is good when there are outliers in our data because they are not affected by standardization. When using standardization, the MSE's on both the training and validation data much smaller than when using normalization (Table 6).

Learning rate	10^{-1}	10^{-2}	10^{-3}	10^{-4}
MSE on the training data	3.9679	3.9648	3.9662	3.9662
MSE on the validation data	3.8515	3.8667	3.8702	3.8704

Table 6 Training and Validation MSE when using standardization