Part 0.

(a) Original train dataset. [Cropped the table because of the size of the picture.]

	dummy	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade
0	1	3066410850	7/9/2014	4	2.50	2720	10006	2.0	0	0	3	9
1	1	9345400350	7/18/2014	2	2.50	2600	5000	1.0	0	0	5	8
2	1	7128300060	7/7/2014	5	1.75	1650	3000	1.5	0	0	3	8
3	1	2155500030	4/28/2015	4	1.75	1720	9600	1.0	0	0	4	8
4	1	3999300080	9/4/2014	6	2.25	3830	11180	1.0	0	2	5	9
5	1	1222069133	2/24/2015	4	2.50	2210	213008	1.0	0	0	4	7
6	1	6329000185	3/29/2015	3	2.50	2600	23361	1.5	1	4	3	8
7	1	3336000296	11/13/2014	4	1.50	1220	4900	1.0	0	0	3	6

sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	long	sqft_living15	sqft_lot15	price
2720	0	1989	0	98074	47.6295	-122.042	2720	10759	5.9495
1300	1300	1926	0	98126	47.5806	-122.379	2260	5000	6.6500
1650	0	1902	0	98144	47.5955	-122.306	1740	4000	4.4300
1720	0	1969	0	98059	47.4764	-122.155	1660	10720	3.8000
2440	1390	1962	0	98008	47.5849	-122.113	2500	10400	8.8700
1210	1000	1975	0	98038	47.4039	-121.980	2270	52707	4.1500
2150	450	1912	0	98146	47.4997	-122.379	1700	14700	5.4000
1220	0	1942	0	98118	47.5292	-122.269	1410	3000	2.5000

Removed the ID columns.

	dummy	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade
0	1	7/9/2014	4	2.50	2720	10006	2.0	0	0	3	9
1	1	7/18/2014	2	2.50	2600	5000	1.0	0	0	5	8
2	1	7/7/2014	5	1.75	1650	3000	1.5	0	0	3	8
3	1	4/28/2015	4	1.75	1720	9600	1.0	0	0	4	8
4	1	9/4/2014	6	2.25	3830	11180	1.0	0	2	5	9
5	1	2/24/2015	4	2.50	2210	213008	1.0	0	0	4	7
6	1	3/29/2015	3	2.50	2600	23361	1.5	1	4	3	8
7	1	11/13/2014	4	1.50	1220	4900	1.0	0	0	3	6

sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	long	sqft_living15	sqft_lot15	price
2720	0	1989	0	98074	47.6295	-122.042	2720	10759	5.9495
1300	1300	1926	0	98126	47.5806	-122.379	2260	5000	6.6500
1650	0	1902	0	98144	47.5955	-122.306	1740	4000	4.4300
1720	0	1969	0	98059	47.4764	-122.155	1660	10720	3.8000
2440	1390	1962	0	98008	47.5849	-122.113	2500	10400	8.8700
1210	1000	1975	0	98038	47.4039	-121.980	2270	52707	4.1500
2150	450	1912	0	98146	47.4997	-122.379	1700	14700	5.4000
1220	0	1942	0	98118	47.5292	-122.269	1410	3000	2.5000

ID is random data which has no numerical, categorial and ordinal value. It will show no relevancy with the target data, which means the change in value ID will have no effect in target data. So, using this feature will have no effect when calculating the function.

(b) Table with day, month, year and dropped the date column.

	dummy	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade
0	1	4	2.50	2720	10006	2.0	0	0	3	9
1	1	2	2.50	2600	5000	1.0	0	0	5	8
2	1	5	1.75	1650	3000	1.5	0	0	3	8
3	1	4	1.75	1720	9600	1.0	0	0	4	8
4	1	6	2.25	3830	11180	1.0	0	2	5	9
5	1	4	2.50	2210	213008	1.0	0	0	4	7
6	1	3	2.50	2600	23361	1.5	1	4	3	8
7	1	4	1.50	1220	4900	1.0	0	0	3	6

yr_renovated	zipcode	lat	long	sqft_living15	sqft_lot15	price	month	day	year
0	98074	47.6295	-122.042	2720	10759	5.9495	7	9	2014
0	98126	47.5806	-122.379	2260	5000	6.6500	7	18	2014
0	98144	47.5955	-122.306	1740	4000	4.4300	7	7	2014
0	98059	47.4764	-122.155	1660	10720	3.8000	4	28	2015
0	98008	47.5849	-122.113	2500	10400	8.8700	9	4	2014
0	98038	47.4039	-121.980	2270	52707	4.1500	2	24	2015
0	98146	47.4997	-122.379	1700	14700	5.4000	3	29	2015
0	98118	47.5292	-122.269	1410	3000	2.5000	11	13	2014

We can use this year as categorial data since we have two values. There could be a possibility that certain year has high rate of house trading than other year.

(c) For numerical features.

1) table of means.

	means
dummy	1.000000
bedrooms	3.375200
bathrooms	2.118875
sqft_living	2080.223200
sqft_lot	15089.201400
floors	1.503700
view	0.229400
sqft_above	1793.099300
sqft_basement	287.123900
yr_built	1971.124900
yr_renovated	81.226700
lat	47.559814
long	-122.213287
sqft_living15	1994.326100
sqft_lot15	12746.323400
year	2014.318500
month	6.592400
day	15.802100

2) table of standard deviation.

	standard deviation
dummy	0.000000
bedrooms	0.943246
bathrooms	0.765128
sqft_living	911.334358
sqft_lot	41203.894918
floors	0.542647
view	0.755932
sqft_above	830.865434
sqft_basement	435.005264
yr_built	29.480594
yr_renovated	394.379804
lat	0.138651
long	0.141405
sqft_living15	691.900301
sqft_lot15	28241.243043
year	0.465918
month	3.111435
day	8.621761

3) table of range

	range
dummy	0.000000e+00
bedrooms	3.200000e+01
bathrooms	7.250000e+00
sqft_living	9.520000e+03
sqft_lot	1.650787e+06
floors	2.500000e+00
view	4.000000e+00
sqft_above	8.490000e+03
sqft_basement	2.720000e+03
yr_built	1.150000e+02
yr_renovated	2.015000e+03
lat	6.217000e-01
long	1.195000e+00
sqft_living15	5.650000e+03
sqft_lot15	8.705400e+05
year	1.000000e+00
month	1.100000e+01
day	3.000000e+01

For categorical features

: waterfront, grade, condition, zip code

1) percentage of waterfront

	waterfront
0	99.3
1	0.7

2) percentage of grade

	grade
7	41.30
8	28.38
9	11.82
6	9.33
10	5.47
11	2.10
5	1.05
12	0.39
4	0.11
13	0.05

3) percentage of condition

	condition
3	65.30
4	25.69
5	8.12
2	0.76
1	0.13

4) zip code

	zipcode
98103	2.82
98115	2.76
98038	2.67
98052	2.67
98042	2.60
98034	2.56
98117	2.46
98023	2.35
98006	2.35
98059	2.31
98133	2.28
98118	2.20
98074	2.11
98058	2.10
98155	2.07
98027	2.03
98125	1.87
98033	1.87
98053	1.86
98092	1.80
98075	1.77
98126	1.72
98056	1.72
98001	1.61
98029	1.60
98199	1.58
98144	1.55
98116	1.52
98106	1.50
98004	1.49
98065	1.43
98198	1.38
98122	1.38
98028	1.35
98072	1.34
98168	1.32
98031	1.32
98107	1.28
98003	1.27
98040	1.27
98136	1.26
98112	1.24
98166	1.22
98146	1.21
98055	1.15
98178	1.15
98030	1.14
98045	1.13
98022	1.12
98105	1.12
98177	1.09
98008	1.09
98077	0.94
98011	0.92
98002	0.88
98019	0.83
98119	0.79
98108	0.77
98005	0.75
98188	0.66
98007	0.64
98070	0.53
98014	0.52
98102	0.52
98032	0.48
98109	0.47
98010	0.42
98024	0.31
98148	0.27
98039	0.24

(d) Based on the statistics that I have calculated; I think features such as square feet living, and lots, including square feet data for year 15 will be useful to get the meaningful result. This is because those features have higher standing deviation, which means each dataset varies a lot. Then we can easily show the drastic difference between each value.

(e) Normalized features.

	dummy	bedrooms	bathrooms	sqft_living	sqft_lot	floors	view	sqft_above	sqft_basement	yr_built
0	NaN	0.09375	0.275862	0.246849	0.005715	0.4	0.0	0.276796	0.000000	0.773913
1	NaN	0.03125	0.275862	0.234244	0.002682	0.0	0.0	0.109541	0.477941	0.226087
2	NaN	0.12500	0.172414	0.134454	0.001471	0.2	0.0	0.150766	0.000000	0.017391
3	NaN	0.09375	0.172414	0.141807	0.005469	0.0	0.0	0.159011	0.000000	0.600000
4	NaN	0.15625	0.241379	0.363445	0.006426	0.0	0.5	0.243816	0.511029	0.539130
5	NaN	0.09375	0.275862	0.193277	0.128688	0.0	0.0	0.098940	0.367647	0.652174
6	NaN	0.06250	0.275862	0.234244	0.013805	0.2	1.0	0.209658	0.165441	0.104348
7	NaN	0.09375	0.137931	0.089286	0.002622	0.0	0.0	0.100118	0.000000	0.365217
8	NaN	0.06250	0.310345	0.280462	0.014308	0.4	0.0	0.314488	0.000000	0.756522
9	NaN	0.03125	0.068966	0.067227	0.002930	0.0	0.0	0.075383	0.000000	0.226087

Part 1.

(a) When calculate training data with each learning rate, when the learning rate was 1, MSE diverges to infinity. With the high value of running rate, it changes the weight value dramatically which leads to a possibility to pass optimal calculated value instead of converges to it and keeps going back and forth of the optimal predicted target value. However, when we set the learning rate other than 1, with lower learning rate, since it helps the weight to move small amount, the MSE value converges to certain value, for me it was about 4.78.

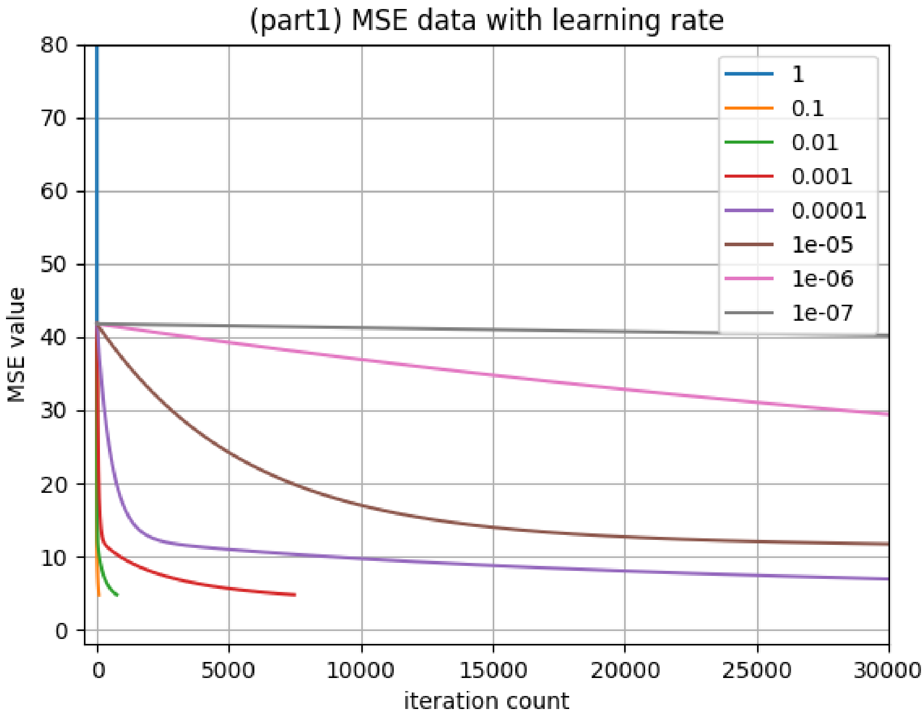

Figure 1. Convergence with learning rate

In figure 1, we can observe that all the MSE value with learning rate except 1 converges. Since learning rate lower that 0.0001 took need too many iterations, I cut the X-axis until 30000, so the figure does not show the converged value, but as these values does not diverge like learning rate 1, we can assume it also converges even just looking at this figure.

(b)

```
MSE of train data with learning rate 1 :  0.0  with iteration:  337
MSE of validation data with learning rate 1 :  0.0  with iteration:  337
iteration score:  337
time took :  0.11842155456542969
MSE of train data with learning rate 0.1 :  4.722104004420385  with iteration:  76
MSE of validation data with learning rate 0.1 :  4.93271366040207  with iteration:  76
iteration score:  76
time took :  0.024348735809326172
MSE of train data with learning rate 0.01 :  4.760923756658479  with iteration:  748
MSE of validation data with learning rate 0.01 :  4.971123606382196  with iteration:  748
iteration score:  748
time took :  0.22682619094848633
MSE of train data with learning rate 0.001 :  4.7631225403489035  with iteration:  7475
MSE of validation data with learning rate 0.001 :  4.973321107666561  with iteration:  7475
iteration score:  7475
time took :  2.2261900901794434
MSE of train data with learning rate 0.0001 :  4.7633426004894055  with iteration:  74745
MSE of validation data with learning rate 0.0001 :  4.973541029158608  with iteration:  74745
iteration score:  74745
time took :  22.248289585113525
MSE of train data with learning rate 1e-05 :  4.763367108311752  with iteration:  747444
MSE of validation data with learning rate 1e-05 :  4.973565462267589  with iteration:  747444
iteration score:  747444
time took :  227.39355373382568
MSE of train data with learning rate 1e-06 :  4.763370059117203  with iteration:  7474432
MSE of validation data with learning rate 1e-06 :  4.97356839345157  with iteration:  7474432
iteration score:  7474432
time took :  2284.240499019623
MSE of train data with learning rate 1e-07 :  4.763370454198004  with iteration:  74744308
MSE of validation data with learning rate 1e-07 :  4.97356878414037  with iteration:  74744308
hw1.py:203: UserWarning: Creating legend with loc="best" can be slow with large amounts of data.
  plt.savefig(str(learning_rate[i])+"part1_.png", format='png')
iteration score:  74744308
time took :  23129.749594449997
```

Figure 2 MSE and Iteration with Learning Rate

Learning Rate	10^{0}	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
MSE of Training Data	Inf	4.72210	4.76092	4.76312	4.76334	4.76336	4.76337	4.76337
MSE of Validation Data	Inf	4.93271	4.97112	4.97332	4.97354	4.97356	4.97356	4.97356
Iteration	337	76	748	7475	74745	747444	7474432	74744308

Table 1 MSE and Iteration with Learning Rate

Figure 2 and Table 1 shows the MSE result and iteration value of both training data and validation data. As we can see after learning rate 10^-2, MSE results are almost same. Even though the result of MSE is almost same in those learning rate, the number of iterations to achieve that learning rate varies a lot. In that case, choosing learning rate 10^-2 can be enough to get the converged MSE value where the iteration value is the lowest, because the time to get those MSE value is also important. As the iterations get higher the time the get the convergent value goes up.

(c) If we see the table1, the lowest MSE value of both training dataset and validation dataset is when the learning rate it 10^-2. So, I chose this learning rate the get the weights of validation data.

dummy	1.345114
bedrooms	1.258679
bathrooms	1.688265
sqft_living	1.594858
sqft_lot	0.118525
floors	0.877523
view	2.160107
sqft_above	1.697179
sqft_basement	1.175069
yr_built	-0.11196
yr_renovated	1.053314
lat	2.734218
long	0.041831
sqft_living15	2.158409
sqft_lot15	0.154297
year	0.41641
month	0.312558
day	0.152306
waterfront_0	0.329934
waterfront_1	1.01518
grade_3	-0.00273
grade_4	-0.05516
grade_5	-0.36046
grade_6	-1.40334
grade_7	-1.52323
grade_8	-0.74244
grade_9	0.977327
grade_10	2.02682
grade_11	1.34491
grade_12	0.875393
grade_13	0.20802
condition_1	-0.00046
condition_2	-0.10241
condition_3	0.067665
condition_4	0.488427
condition_5	0.891898
zipcode_98001	-0.30773
zipcode_98002	-0.19005
zipcode_98003	-0.1982
zipcode_98004	1.34477
zipcode_98005	0.229624
zipcode_98006	0.572552
zipcode_98007	0.05547
zipcode_98008	0.226671
zipcode_98010	-0.02335
zipcode_98011	-0.15174
zipcode_98014	-0.09288
zipcode_98019	-0.22689
zipcode_98022	-0.12148
zipcode_98023	-0.43285
zipcode_98024	-0.02895
zipcode_98027	0.157974
zipcode_98028	-0.22303
zipcode_98029	0.009463
zipcode_98030	-0.26287
zipcode_98031	-0.27435
zipcode_98032	-0.10994
zipcode_98033	0.650866
zipcode_98034	-0.08535
zipcode_98038	-0.38466
zipcode_98039	0.475414
zipcode_98040	0.817131
zipcode_98042	-0.40587
zipcode_98045	-0.11522
zipcode_98052	0.137638
zipcode_98053	0.154381
zipcode_98055	-0.26664
zipcode_98056	-0.24336
zipcode_98058	-0.33915
zipcode_98059	-0.12681
zipcode_98065	-0.1256
zipcode_98070	0.023438
zipcode_98072	-0.09012
zipcode_98074	0.119244
zipcode_98075	0.124413
zipcode_98077	-0.02787
zipcode_98092	-0.2781
zipcode_98102	0.129148
zipcode_98125	-0.12056
zipcode_98126	-0.03729
zipcode_98133	-0.34979
zipcode_98136	0.066481
zipcode_98144	0.070114
zipcode_98146	-0.19237
zipcode_98148	-0.04538
zipcode_98155	-0.31573
zipcode_98166	-0.0643
zipcode_98168	-0.2721
zipcode_98177	0.090889
zipcode_98178	-0.18919
zipcode_98188	-0.14469
zipcode_98198	-0.19476
zipcode_98199	0.406825
zipcode_98103	0.21486
zipcode_98105	0.408886
zipcode_98106	-0.28702
zipcode_98107	0.059148
zipcode_98108	-0.11333
zipcode_98109	0.30036
zipcode_98112	0.808895
zipcode_98115	0.432119
zipcode_98116	0.235807
zipcode_98117	0.229447
zipcode_98118	-0.11575
zipcode_98119	0.217273
zipcode_98122	0.151095

[I split the table since it was too long]

Above table is the weight values of validation with learning rate 10^-2. Based on the table, the important features that I have found are bedrooms, bathrooms, sqft_living, sqft_above, sqft_basement, view, sqft_living15, latitude and yr_renovated. Compared to the estimation that I made on part0 (d), where I said living and lots will have big weight, I made some correct estimation but features such as bedrooms and bathrooms were also important. Those features are real important value when we are looking at new house, but in statistical data those features were very small, so I missed this point in part 0.

Part2.

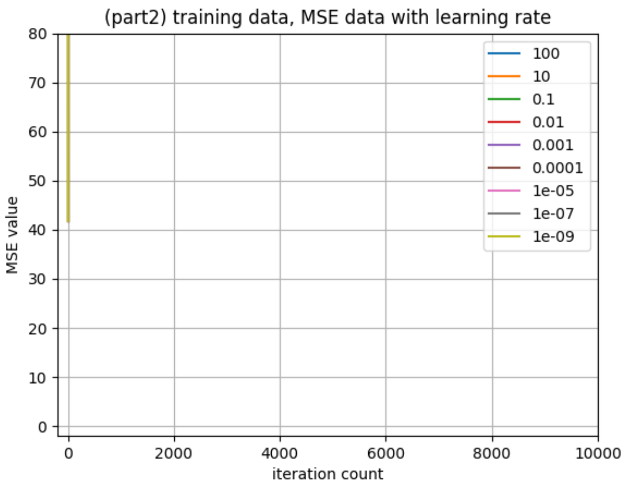

Figure 4 le-09 Training Dataset

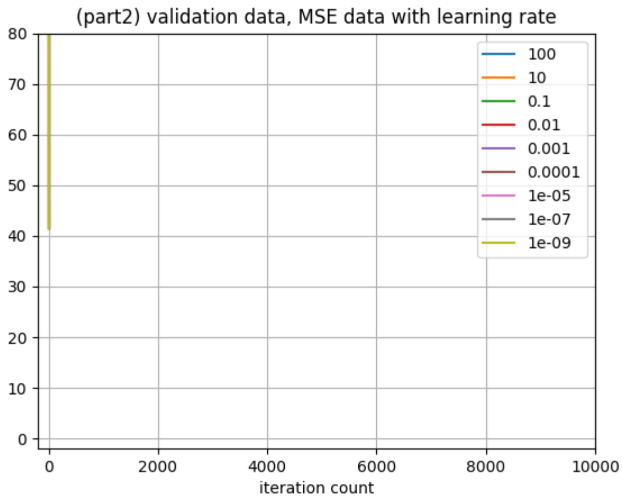

Figure 3 le-09 Validation Dataset

Figure 5 le-10 Training Dataset

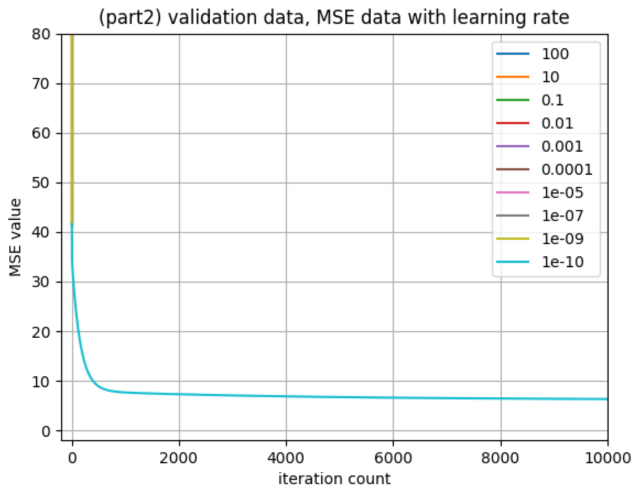

Figure 6 le-10 Validation Dataset

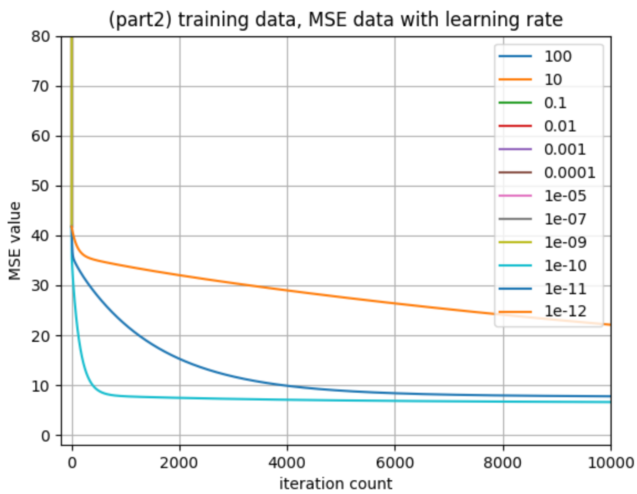

Figure 7 le-12 Training Dataset

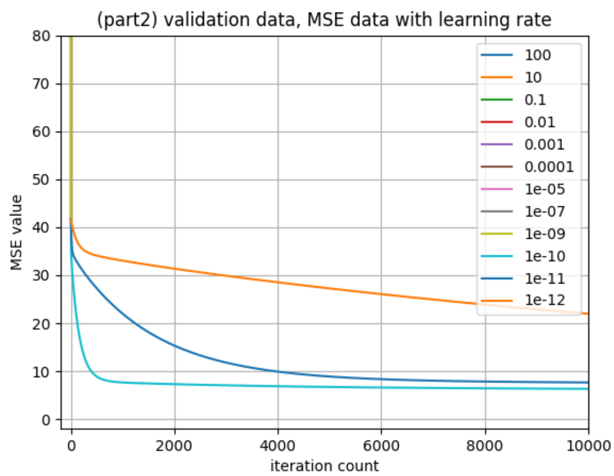

Figure 8 le-12 Validation Dataset

```
MSE of train data with learning rate 10 :  0.0  with iteration:  30
MSE of train data with learning rate 0.1 :  0.0  with iteration:  37
MSE of train data with learning rate 0.01 :  0.0  with iteration:  41
MSE of train data with learning rate 0.001 :  0.0  with iteration:  47
MSE of train data with learning rate 0.0001 :  0.0  with iteration:  55
MSE of train data with learning rate 1e-05 :  0.0  with iteration:  66
MSE of train data with learning rate 1e-07 :  0.0  with iteration:  112
MSE of train data with learning rate 1e-09 :  0.0  with iteration:  477
MSE of train data with learning rate 1e-10 :  6.608384736412362  with iteration:  10000
MSE of train data with learning rate 1e-11 :  7.761259256047734  with iteration:  10000
MSE of train data with learning rate 1e-12 :  22.102656601676216  with iteration:  10000
MSE of validation data with learning rate 100 :  0.0  with iteration:  10000
MSE of validation data with learning rate 10 :  0.0  with iteration:  10000
MSE of validation data with learning rate 0.1 :  0.0  with iteration:  10000
MSE of validation data with learning rate 0.01 :  0.0  with iteration:  10000
MSE of validation data with learning rate 0.001 :  0.0  with iteration:  10000
MSE of validation data with learning rate 0.0001 :  0.0  with iteration:  10000
MSE of validation data with learning rate 1e-05 :  0.0  with iteration:  10000
MSE of validation data with learning rate 1e-07 :  0.0  with iteration:  10000
MSE of validation data with learning rate 1e-09 :  0.0  with iteration:  10000
MSE of validation data with learning rate 1e-10 :  6.32221145872936  with iteration:  10000
MSE of validation data with learning rate 1e-11 :  7.649587815003083  with iteration:  10000
MSE of validation data with learning rate 1e-12 :  21.969163077208485  with iteration:  10000
```

1. Observation

Above figures are the MSE result of each learning rate of non-normalized training and validation dataset. I used the learning rate from 100 to 10^{-12}. Unlike normalized dataset it diverges with most of the learning rate. In figure 3 and 4 are plotted learning result of 9 different learning rate. Since in normalized dataset, the result diverges with the learning rate of 1, divergence in learning rate 100, 10 was expected. However, until the learning rate smaller to 10^{-9} the MSE result still diverges.

2. Convergence

I assume this result because of non-normalized dataset. Since all the features are non-normalized, features such as square feet data are huge so the gradient vector with those values will also be huge. Then, weight values will vary dramatically with every iteration, so it will diverge rather than converge. Since, non-normalized data is very large, running rate at least 10^{-10} will start to converge, as we can see in figure 5 and 6. I tried until 10^{-12} and from 10^{-10}, MSE results of both training and validation data converges.

3. Comparison

In the case of normalized dataset, from learning rate 0.1, training dataset converges, however in non-normalized dataset it takes much more smaller learning rate for training dataset to converges. As I mentioned above, this is because of non-normalized data has larger value which generate larger gradient vector. Also, when computing with learning rate 10^{-7}, it took very long time to get the result. In this case, we set the maximum iteration to 10000, but if we do the same iteration as normalized data, it will take much long hours to get the converged MSE result. Therefore, training with normalized data is easier.

Part 3.

1) The first change I did was to use zip code as numerical value. In my original training data, I used zip code as categorical data and applied one-hot encoding to this data. So, this time I will just consider this zip code value as normal numerical data as it mentioned in the homework description.

```
iteration considering -zipcode- as numerical data
MSE of train data with learning rate 0.001 :  5.475792777505445  with iteration:  7001
iteration score:  7001
time took :  0.9069037437438965
```

The result seems not good since it shows larger MSE value. I can interpret this result as zip code is not considered as numerical value.

2) The second change that I did was to think of 'year' value as categorical data. I did this because in dataset, the value of year is only 2014 and 2015. Also, I changed 'month' to a categorical data. There is a possibility that in certain months such as spring or fall, when there is lots of changes such as going to a university or getting a new job, people tend to move a lot to a new places. If there is lots of trading in market the price of the house can be affected.

```
iteration with considering value -year- also as categorical data
MSE of train data with learning rate 0.001 :  4.75906901539716  with iteration:  7482
iteration score:  7482
time took :  2.339142322540283
```

```
iteration with considering value -month- also as categorical data
MSE of train data with learning rate 0.001 :  4.754595986879889  with iteration:  7469
iteration score:  7469
time took :  3.0779671669006348
```

```
iteration with considering value -year, month- also as categorical data
MSE of train data with learning rate 0.001 :  4.751584922493888  with iteration:  7481
iteration score:  7481
time took :  3.1943421363830566
```

The result for this feature engineering, when I considered both year and month as categorical data, the value of MSE was the lowest with 4.75158. However, if I compare to this my original training MSE, which was 4.76312, the value got lower but not a meaningful result.

3) The third change that I did was to lower the value of epsilon. The purpose of mean squared error is to find error between true data and predicted data. We are using epsilon 0.5 as condition value. However, if we consider derivates value as 0.5 it is not that flatten value compared to 0. So, I decided to lower this epsilon value to 0.01 which is way closer to the value 0.

```
iteration with lower epsilon 0.01
MSE of train data with learning rate 0.01 :  2.0915444371053318  with iteration:  64658
iteration score:  64658
time took :  20.660179138183594
```

```
iteration with lower epsilon 0.01
MSE of train data with learning rate 0.001 :  2.0915460129400594  with iteration:  646568
iteration score:  646568
time took :  209.9405210018158
```

The result of changing the epsilon value definitely reduced the value of MSE. However, the iteration got way larger than usual case. So, setting lower epsilon value has ups and downs. However, based on the learning rate analysis result in part 1, learning rate 0.01 is small enough to get converged MSE value. I tried both 0.01 and 0.001 and the results are almost same. Therefore, setting up lower epsilon shows the most changes in getting small MSE convergence value.