**IA1**

Yongsung Cho

Computer Science, Oregon State University

AI 534: Machine Learning

Prof. Fern

Due Oct 15, 2021

**Part 0 Data preprocessing**

(a) The unused ID column is deleted by using a pd.drop function, and the dummy feature is added to the training dataset to calculate the Bias term w0. Weight vectors (The number of x features +1) were created to apply the weight and linear model.

(b) using year, yr\_built, and yr\_renovated features, I added columns with the name age\_since\_renovated. After calculating, yr\_renovated column was deleted.

(c) Each feature was normalized through mean and standard deviation, respectively. Mean and standard deviation of training data were also stored for normalizing the validation data, and normalization was performed for the same method for validation data. When normalizing Validation data, I thought I should normalize with the mean and standard deviation of validation data, but I was surprised that I had to do it with the same value as the training set.

**Part 1 Implement batch gradient descent and explore different learning rates**

1. Learning rate 101 learning rate 100

Chart

Description automatically generated with medium confidence Chart

Description automatically generated

When the learning rate is 101, 100, the learning rate is too large, so the MSE does not converge and shows an increasing graph.

Learning rate 10-1 learning rate 10-2

Chart

Description automatically generated Chart

Description automatically generated

When learning rate was 10-1, it started at low MSE and converged to the lowest MSE value.

Learning rate 10-3 learning rate 10-4

Chart

Description automatically generated Chart, line chart

Description automatically generated

The smaller the Learing rate, the slower the MSE value decreases because the gradient update is not enough.

Learning rate 10-5 learning rate 10-6

Chart

Description automatically generated Chart

Description automatically generated

Similarly, when the learning rate is 10-5 and 10-6, the learning rate is too small to update, and the MSE value is almost not reduced.

**Question: Which learning rate or learning rates did you observe to be good for this particular dataset? What learning rates make gradient descent diverge?**

10-1 or 10-2 looks good for learning rate.

101 and 100 learning rates are too large, and gradient is increased without decreasing. So, the value of MSE is not converged.

1. MSE value of validation dataset for learning rate 10^-1

4.526155653005665

MSE value of validation dataset for learning rate 10^-2

4.655614497032713

MSE value of validation dataset for learning rate 10^-3

5.297228766802436

MSE value of validation dataset for learning rate 10^-4

20.50656055727645

MSE value of validation dataset for learning rate 10^-5

41.06516367311085

MSE value of validation dataset for learning rate 10^-6

46.1767401210956

When the learning rate is 10-1, 10-2, and 10-3, the converged MSE value is small and it is nearly similar. However, the smaller the learning rate, the less the gradient update, so the MSE value is almost unchanged.

**Question: Which learning rate leads to the best validation MSE? Between different convergent learning rates, how should we choose one if the validation MSE is nearly identical?**

The best MSE result is shown when the learning rate is 10-1

If MSE is almost the same, it seems good to pick a learning rate that updates quickly in low iteration. For example, the value of MSE with learning rate 10-2 is shown to be lower than the value of MSE with learning rate 10-3 in low iteration.

1. The updated weight vector with learning rate 10-1

Table

Description automatically generated

**Question: learned feature weights are often used to understand the importance of the features. What features are the most important in deciding the house prices according to the learned weights?**

12th, 13th, 15th, 17th and 21st features are highly impacted which are Grade, sqft\_above, yr\_bulit, latitude and waterfront. The value of these features is higher than other features.

**Part 2 Training with non-normalized data**

1. Learning rate 10-9 Learning rate 10-10

Chart

Description automatically generated Chart

Description automatically generated

Unlike training normalized data, the learning rate does not converge from 101 to 10-10 and the MSE value diverges.

Learning rate 10-11 Learning rate 10-12

**Chart

Description automatically generated** **Chart

Description automatically generated**

Learning rate 10-13 Learning rate 10-14

Chart

Description automatically generated with medium confidence Chart, line chart

Description automatically generated

When the learning rate is 10-11, the more training is repeated, the less MSE value is shown, but the final value is very large. The Normailed data has been down to almost 3, but the non-normalized data is very large, approximately 1e10.

**Question: What learning rates work for the un-normalized data? Compare between using the normalized and the non-normalized versions of the data. Which one is easier to train and why?**

When the learning rate is smaller than 10-11, the MSE value of non-normalized data converges. Compared to using the normalized and non-normalized versions of the data, normalized data is easier to train. The number of data varies, and the value of the zip code is about 98000, but it is not a significant number, and it has a great influence on making the calculated result big.

1. Compute and report the MSE of the validation data. Pick the best converged solution that minimizes the validation MSE and report the learned weights.

MSE value of validation dataset for learning rate 10^-11

42.07517877387992

MSE value of validation dataset for learning rate 10^-12

42.8385201578033

MSE value of validation dataset for learning rate 10^-13

45.13389819515084

MSE value of validation dataset for learning rate 10^-14

46.40623910134708

Weight vector for learning rate 10^-11

Table

Description automatically generated

The MSE result value is similar to the initial MSE value which is around 50.

**Question: Compare the weights to those of 1(c), what difference do you observe? What is the explanation for such difference? How does this impact the interpretation of the weights as the measure of feature importance.**

Unlike 1c, the weight values ​​converge to 1 or 0. I think it is because the value of the first weight vector was 1, and the learning rate was minimal, so it was almost not updated.

Weight vectors obtained from un-normalized data do not know which features have a significant impact. In general, the elements with large numbers have negative values, and the other features have a little update.

**Part 3 Redundancy in features**

MSE value of validation dataset for learning rate 10^-1

4.553334757583371

**Table

Description automatically generated**

**Question: how does this new model compare to the one in part 1 (c)? What do you observe when comparing the weight for sqrt living in both versions? Consider the situation in general when two features x1 and x2 are redundant, what do you expect to happen to the weights (w1 and w2) when learning with both features, in comparison with w1 which is learned with just x1? Why?**

Compared to 1c, the MSE results of the new model were expected to be lower but slightly higher than the results of 1c. However, when I analyzed the process of updating the MSE, the starting point of the MSE value was lower than 1c. This means that the MSE result value with only one value(x1) is lower in low iteration. Although X1 and x2 features are redundant, there is a slight difference between x1 and x2 values, so I think that the more training is repeated, the better performance is shown. However, because the number of weights increases and the calculation becomes complicated, the redundant features combine into one if the dataset is more significant. I think it is a more efficient learning method.

**Part 4 Kaggle**

**For this portion of your report, you must explain and motivate (this is important!) the ideas that you have tried for this part and their corresponding performances in testing MSE reported on Kaggle (please provide your team name so that it can be cross-verified). Discuss the successes and failures of your attempts and the lessons you learned from the exploration.**

**Default features:** ['month','day','year','age\_since\_renovated','bedrooms','bathrooms','sqft\_living','sqft\_lot','floors','view','condition','grade','sqft\_above','sqft\_basement','yr\_built','zipcode','lat','long', 'waterfront']

**-feature difference**

**<TRY1>**

**Insert yr\_old / drop yr\_bulit**

( yr\_old = 2021 – yr\_bulit )

**Insert cat\_zipcode / drop zipcode**

cat\_zipcode (int)

98000 < zipcode < 98100 ~ 1

98100 < zipcode < 98200 ~ 2

98200 < zipcode < 98300 ~ 3

98300 < zipcode < 98400 ~ 4

98400 < zipcode < 98500 ~ 5

98500 < zipcode < 98600 ~ 6

98600 < zipcode < 98700 ~ 7

98700 < zipcode < 98800 ~ 8

98800 < zipcode < 98900 ~ 9

98900 < zipcode < 99000 ~ 10

**Insert latXlong**

latXlong = abs(lat-47)\*10 \* abs(long+122)\*10

Kaggle score: 3.60597

=====================================================================

**<TRY2>**

**Insert yr\_old / drop yr\_bulit**

( yr\_old = 2021 – yr\_bulit )

**Insert cat\_zipcode / drop zipcode**

cat\_zipcode (int)

98000 < zipcode < 98100 ~ 1

98100 < zipcode < 98200 ~ 2

98200 < zipcode < 98300 ~ 3

98300 < zipcode < 98400 ~ 4

98400 < zipcode < 98500 ~ 5

98500 < zipcode < 98600 ~ 6

98600 < zipcode < 98700 ~ 7

98700 < zipcode < 98800 ~ 8

98800 < zipcode < 98900 ~ 9

98900 < zipcode < 99000 ~ 10

Kaggle score: 3.63002

=====================================================================

**<TRY3>**

**Insert yr\_old / drop yr\_bulit**

( yr\_old = 2021 – yr\_bulit )

**Insert cat\_zipcode / drop zipcode**

cat\_zipcode (int)

98000 < zipcode < 98100 ~ 1

98100 < zipcode < 98200 ~ 2

98200 < zipcode < 98300 ~ 3

98300 < zipcode < 98400 ~ 4

98400 < zipcode < 98500 ~ 5

98500 < zipcode < 98600 ~ 6

98600 < zipcode < 98700 ~ 7

98700 < zipcode < 98800 ~ 8

98800 < zipcode < 98900 ~ 9

98900 < zipcode < 99000 ~ 10

**Insert latXlong**

latXlong = abs(lat-47)\*10 \* abs(long+122)\*10

**Insert sq\_above\_percent**

sq\_above\_percent = (sq\_living – sq\_basement)/ sq\_living \*100

Kaggle score: 3.36387

=====================================================================

**<TRY4>**

**Insert sq\_above\_percent**

sq\_above\_percent = (sq\_living – sq\_basement)/ sq\_living \*100

Kaggle score: 3.45277

=====================================================================

Description for the new feature

**yr\_old**

The given data has a year built, a yr\_sold, and a year\_renovateㅇ. However, these features have difficulty in training because the value indicates a year. Therefore, I have created a new yr\_old feature (2 021 - the year built) and deleted the yr\_bulit column.

**cat\_zipcode**

The Zipcode features are all between 98000 and 99000, where the number 98000 is not significant, and the last three digits are important. The last three digits represent the city's location; in addition, if the last three digits are similar, they will usually be located nearby. Therefore, the number was allocated to each area in 100 and preprocessed with accessible data for training.

98000 < zipcode < 98100 ~ 1

98100 < zipcode < 98200 ~ 2

98200 < zipcode < 98300 ~ 3

98300 < zipcode < 98400 ~ 4

98400 < zipcode < 98500 ~ 5

98500 < zipcode < 98600 ~ 6

98600 < zipcode < 98700 ~ 7

98700 < zipcode < 98800 ~ 8

98800 < zipcode < 98900 ~ 9

98900 < zipcode < 99000 ~ 10

**latXlong**

These features, which represent latitude and longitude, have similar numbers, and do not make much difference. So, we changed the two features into real numbers with one digit and then made columns multiplying the two features. The product of these two numbers appears as a linear graph on the map and has the same value at the point above the virtual graph.

**Sq\_above\_percent**

Sq\_above = sq\_living – sq\_basement

Using this, I calculated the ratio of sq\_above area by sq\_living and made it a new column.

Let's compare TRY1 and TRY2 first. The difference between the two is that TRY1 includes latXlong, and TRY2 does not have latXlong, so the number of features in TRY2 is one less than TRY1. The better score of TRY1 suggests that latXlong affects score; however, it is also known that the effect size is not significant because the score decreases slightly.

Let's compare TRY1 and TRY3. TRY3 has the sq\_above\_percent feature compared to the features of TRY1. At this time, the Kaggle score was significantly lower in try3. As a result, the sq\_above\_percent feature was affected the total score of Kaggle data considerably.

In Try3, I tried TRY4 with only sq\_above\_percent without modifying yr\_old, cat\_zipcode, and latXlong features. As a result, Kaggle's score was increased compared to TRY3. However, since the results are better than TRY1 and TRY2, the sq\_above\_percent part significantly influences the score.