CSE258 - Homework1

October 13, 2019

1 CSE 258, Fall 2019: Homework 1

Student ID: A53308934 Student Name: Deng Zhang

1.1 Tasks - Regression

[2]:
```
import csv
import gzip
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from random import shuffle
```

[3]:
```
# Read the tsv
c = csv.reader(gzip.open('./Data/hw1/amazon_reviews_us_Gift_Card_v1_00.tsv.gz',␣
 ↪'rt', encoding="utf8"), delimiter = '\t')
dataset = []
```

[4]:
```
first = True
for line in c:
    # The first line is the header
    if first:
        header = line
        first = False
    else:
        d = dict(zip(header, line))
        # Convert strings to integers for some fields:
        d['star_rating'] = int(d['star_rating'])
        d['helpful_votes'] = int(d['helpful_votes'])
        d['total_votes'] = int(d['total_votes'])
        dataset.append(d)

dataset[0]
```

[4]:
```
{'marketplace': 'US',
 'customer_id': '24371595',
 'review_id': 'R27ZP1F1CD0C3Y',
 'product_id': 'B004LLIL5A',
```

```
 'product_parent': '346014806',
 'product_title': 'Amazon eGift Card - Celebrate',
 'product_category': 'Gift Card',
 'star_rating': 5,
 'helpful_votes': 0,
 'total_votes': 0,
 'vine': 'N',
 'verified_purchase': 'Y',
 'review_headline': 'Five Stars',
 'review_body': 'Great birthday gift for a young adult.',
 'review_date': '2015-08-31'}
```

1.1.1 Question 1:

[5]:
```python
rating = {}
for data in dataset:
    # use dictionary to count the review of each start_rating
    rating[data['star_rating']] = rating.get(data['star_rating'], 0) + 1

print (rating)
```

```
{5: 129029, 1: 4766, 4: 9808, 2: 1560, 3: 3147}
```

1.1.2 Question 3:

[11]:
```python
# define the feature that we are looking for
def feature(data):
    return [1, data['verified_purchase'] == 'Y', len(data['review_body'])]
```

[12]:
```python
# Cluster the feature
star_predict = [int(data['star_rating']) for data in dataset]
X = [feature(d) for d in dataset] # X is the record which satisified feature(d)
```

[13]:
```python
# Return the least-squares solution to linear matrix equation
theta, residuals, ranks, s = np.linalg.lstsq(X, star_predict, rcond = None)
```

[14]:
```python
theta
```

[14]:
```
array([ 4.84503504e+00,  4.98580589e-02, -1.24545526e-03])
```

Explaination

[15]:
```python
# MY EXPLAINATION FOR QUESTION 3
# 0 equals to 4.84503504e+00, 1 equals to 4.98580589e-02,
# 3 equals to -1.24545526e-03

# 0 represents bias, 1 and 2 shows the influence of verified-review and review
# length to the result of star_rating
```

```
# 1 shows that verified-review is more likely to give a higher rating
# 2 shows that the longer the length is, the lower the star_rating would be
```

1.1.3 Question 4:

[16]:
```
# define new feature according to the requirement
def feature(data):
    return [1, float(data['verified_purchase'] == 'Y')]
```

[17]:
```
star_predict = [int(data['star_rating']) for data in dataset]
X = [feature(d) for d in dataset]
```

[18]:
```
theta, residuals, ranks, s = np.linalg.lstsq(X, star_predict, rcond = None)
```

[19]:
```
theta
```

[19]:
```
array([4.578143  , 0.16793392])
```

Explaination

[20]:
```
# MY EXPLAINATION FOR QUESTION 4
# 0 equals to 4.578143, 1 equals to 0.16793392
# Thetas here are larger than the one from Question 3.

# This probably because the average value of `verified_purchase` is less
# than 1, but the average value of `review_body` is much bigger than that.

# As a result, review_body would play a more important role in finding out
# the result in Question 3. But in Question 4, we only have the feature
# `verified_purchase`, so it is much more important than before.
```

1.1.4 Qustion 5:

[21]:
```
N = len(dataset)
print (N)
```

```
148310
```

[22]:
```
# distribute training and testing part
training_data = dataset[0:int(N*0.9)]
testing_data = dataset[int(N*0.9):]
```

[23]:
```
X_training = [feature(d) for d in training_data]
X_testing = [feature(d) for d in testing_data]

Y_training = [int(d['star_rating']) for d in training_data]
Y_testing = [int(d['star_rating']) for d in testing_data]
```

[24]:
```
theta, residuals, ranks, s = np.linalg.lstsq(X_training, Y_training, rcond =␣
↪None)
```

[25]:
```
# find out the mse
mse_train = np.mean((np.dot(X_training, theta) - Y_training)**2)
mse_test = np.mean((np.dot(X_testing, theta) - Y_testing)**2)
```

Result

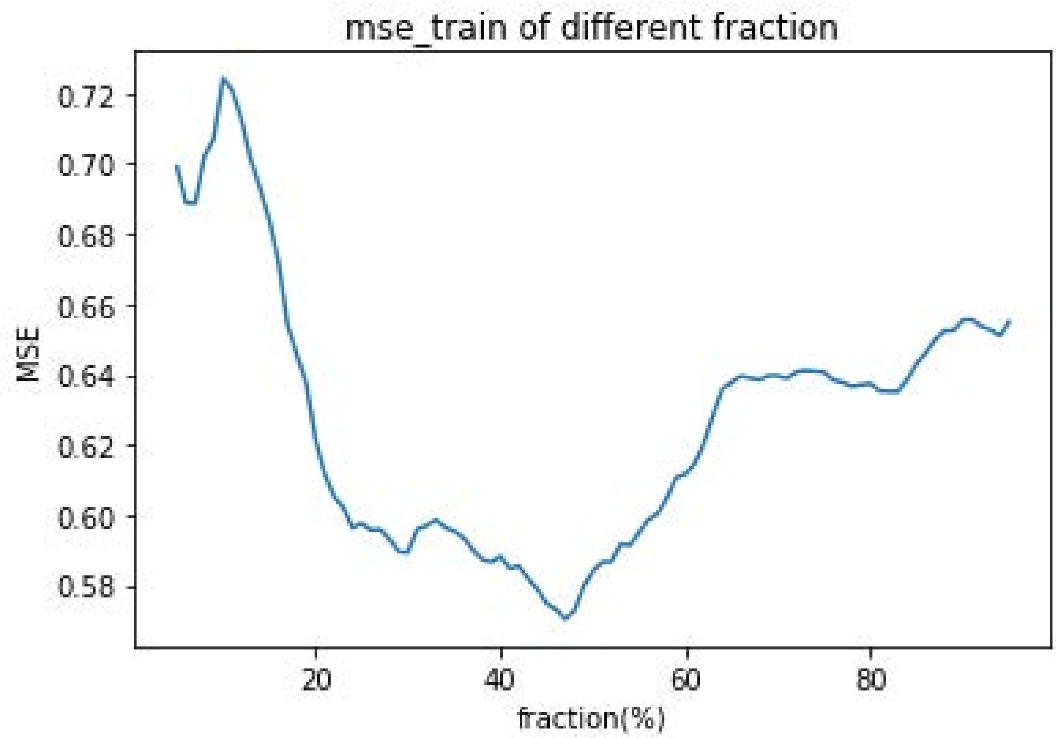
[26]:
```
print('Train MSE: %f \n Test MSE: %f' % (mse_train, mse_test))
```

```
Train MSE: 0.655484
 Test MSE: 0.972385
```

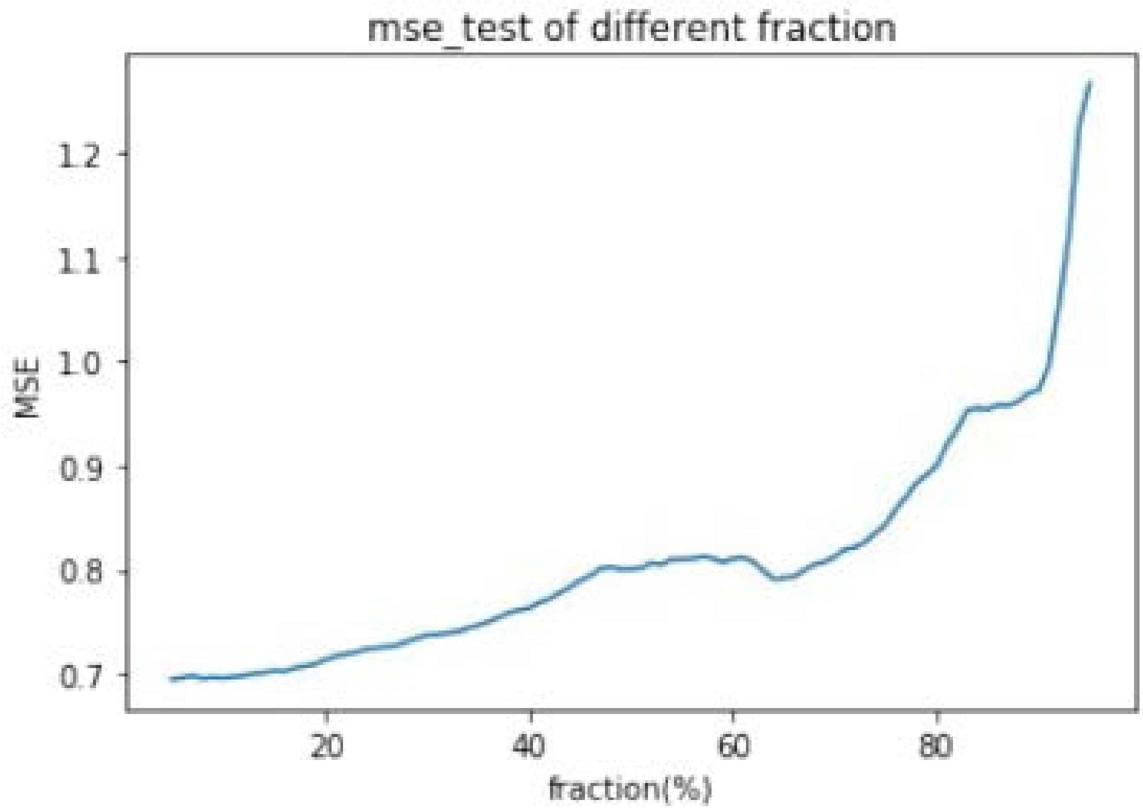
1.1.5 Question 7:

[27]:
```
mse_train_list = []
mse_test_list = []

# find out the mse at different distribution
for fraction in range(5, 96, 1):
    training_data = dataset[0:int(N*fraction/100)]
    testing_data = dataset[int(N*fraction/100):]

    X_training = [feature(d) for d in training_data]
    X_testing = [feature(d) for d in testing_data]
    Y_training = [int(d['star_rating']) for d in training_data]
    Y_testing = [int(d['star_rating']) for d in testing_data]

    theta, residuals, ranks, s = np.linalg.lstsq(X_training, Y_training, rcond␣
↪= None)

    mse_train = np.mean((np.dot(X_training, theta) - Y_training)**2)
    mse_test = np.mean((np.dot(X_testing, theta) - Y_testing)**2)

    mse_train_list.append(mse_train)
    mse_test_list.append(mse_test)
```

[28]:
```
# draw a picture for the trend of mse_train
plt.plot(range(5, 96, 1), mse_train_list)
plt.title("mse_train of different fraction")
plt.xlabel("fraction(%)")
plt.ylabel("MSE")
plt.show()
```



[29]:
```
# draw a picture for the trend of mse_test
plt.plot(range(5, 96, 1), mse_test_list)
plt.title("mse_test of different fraction")
plt.xlabel("fraction(%)")
plt.ylabel("MSE")
plt.show()
```



Explaination

[30]:
```
# MY EXPLAINATION FOR QUESTION 7
# 1.1 the trend of first graph
# From the first graph, we got that with the increase of the size of training
# fraction, the Mean Squared Error of training set goes down at the beginning,
# and then goes up at about 50%.

# 1.2 possible reason 1
# When the training propotion is extremely low, it is easy to understand that
# the MSE would be very high. Since we do not get enough statistics to train
# the model at a low training propotion, some exceptions # would cause huge
# influence on the final result. The Fault tolerance is rather small.

# 1.3 possible reason 2
# However, the MSE of training_data goes up at about 50%. I guess this may
# because the data is not evenly distributed. Lots of exceptions are clustered
# in the second half part of the statistics.

# 2.1 the trend of second graph and its reason
# From the second graph, we got that with the increase of the size of testing
# fraction, the Mean Squared Error of testing set goes up continuously. This is
# because that with the increase of testing propotion, the propotion of
# training set goes down. The low propotion of training set would cause MSE
```

```
# goes up according to 1.2.

# 3 another special thing
# There is another issue: why this two graphs not corresponding to each other?
# I guess that is because the porpotion of two sets are changing all the time.
# The testing set is not fixed. As I have stated in 1.3, the data might not be
# evenly distributed. So the changes of testing set would cause cause the above
# results. If we fixed the testing set, the trend of two graphs might
# complement each other.
```

1.2 Tasks - Classification

1.2.1 Question 8:

[31]:
```
model = linear_model.LogisticRegression()
```

[32]:
```
def feature(data):
    return [1, float(data['star_rating']), len(data['review_body'])]
```

[33]:
```
def calculatePropotion(dataset):
    training_data = dataset[:int(len(dataset)*0.9)]
    testing_data = dataset[int(len(dataset)*0.9):]

    X_training = [feature(d) for d in training_data]
    X_testing = [feature(d) for d in testing_data]

    Y_training = [float(d['verified_purchase'] == 'Y') for d in training_data]
    Y_testing = [float(d['verified_purchase'] == 'Y') for d in testing_data]

    return X_training, Y_training, X_testing, Y_testing
```

[34]:
```
X_training, Y_training, X_testing, Y_testing = calculatePropotion(dataset)
model.fit(X_training, Y_training)
```

```
E:\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
```

[34]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

[35]:
```
def predict(X_training, Y_training, X_testing, Y_testing):
    predictions = model.predict(X_testing)
    correctPredictions = predictions == Y_testing
    correct_rate = sum(correctPredictions) / len(correctPredictions)
```

```
    print('Correct Rate in Testing: %f' % (correct_rate))

    train_Y_positive = sum(i == 1 for i in Y_training) / len(Y_training)
    print('Positive Label in Y_training: %f' % train_Y_positive)
    test_Y_positive = sum(i == 1 for i in Y_testing) / len(Y_testing)
    print('Positive Label in Y_testing: %f' % test_Y_positive)

    predictions_positive = sum(i == 1 for i in predictions) / len(Y_testing)
    print('Positive Label in predictions: %f' % predictions_positive)

predict(X_training, Y_training, X_testing, Y_testing)
```

```
Correct Rate in Testing: 0.559773
Positive Label in Y_training: 0.951386
Positive Label in Y_testing: 0.559571
Positive Label in predictions: 0.998989
```

Result

[36]:
```
# The Correct Rate in Testing is 0.559773
# The Positive Label in Y_training is 0.951386
# The Positive Label in Y_testing is 0.559571
# The Positive Label in predictions is 0.998989
```

1.2.2 Question 9:

[37]:
```
# shuffle the dataset and do the same as above
shuffle(dataset)
X_training, Y_training, X_testing, Y_testing = calculatePropotion(dataset)
model.fit(X_training, Y_training)
predict(X_training, Y_training, X_testing, Y_testing)
```

```
E:\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
```

```
Correct Rate in Testing: 0.910997
Positive Label in Y_training: 0.912106
Positive Label in Y_testing: 0.913087
Positive Label in predictions: 0.996966
```

Explaination

[38]:
```
# MY EXPLAINATION FOR QUESTION 8
# From Question 7, we got that the data in the dataset in not evenly
# distributed, the positive label mainly distributed in the training
```

```
# set, but rare in testing set.

# To improve the accuracy of our predictor, we could shuffle the dataset.
# Shuffling the dataset could make the data evenly distributed, which
# would solve the problem stated before.

# After shuffling the dataset, the accuracy has an obvious enhancement
# The Correct Rate in Testing is 0.912143
# The Positive Label in Y_training is 0.912024
# The Positive Label in Y_testing is 0.913829
# The Positive Label in predictions is 0.996831
```