CSE 258 - HW 1

Jin Dai / A92408103

Regression (week 1)

Question. 2

55

```
import numpy as np
```

56

```
def parse_data(fname):
    for line in open(fname):
        yield eval(line)
```

57

```
book_reviews = list(parse_data("fantasy_10000.json"))
```

58

```
book_reviews[0]
```

58

```
{'user_id': '8842281e1d1347389f2ab93d60773d4d',
 'book_id': '18245960',
 'review_id': 'dfdbb7b0eb5a7e4c26d59a937e2e5feb',
 'rating': 5,
 'review_text': 'This is a special book. It started slow for about the first third, then in the middle th
 'date_added': 'Sun Jul 30 07:44:10 -0700 2017',
 'date_updated': 'Wed Aug 30 00:00:26 -0700 2017',
 'read_at': 'Sat Aug 26 12:05:52 -0700 2017',
 'started_at': 'Tue Aug 15 13:23:18 -0700 2017',
 'n_votes': 28,
 'n_comments': 1}
```

59

```
len(book_reviews[0]['review_text'])
```

59

```
2086
```

60

```
data = [d for d in book_reviews if 'review_text' in d]
```

61

```
data[0]['review_text']
```

61

```
'This is a special book. It started slow for about the first third, then in the middle third it started t
```

62

```
def featurize(data, featurizer):
    return [featurizer(d) for d in data]
```

63

```
X = featurize(data, lambda d : [1, len(d['review_text'])])
```

64

```
X[0:10]
```

64

```
[[1, 2086],
 [1, 1521],
 [1, 1519],
 [1, 1791],
 [1, 1762],
 [1, 470],
 [1, 823],
 [1, 532],
 [1, 616],
 [1, 548]]
```

65

```
def extract_labels(data):
    return [d['rating'] for d in data]
```

66

```
y = extract_labels(data)
```

67

```
y[0:10]
```

67

```
[5, 5, 5, 4, 3, 5, 5, 5, 4, 5]
```

68

```
def mean_sq_error(theta, features, labels):
    return np.square(np.matrix(labels).T - np.matmul(np.matrix(features), np.matrix(theta).T))[:,0].mean(
```

69

```
def fit(features, labels):
    theta = np.linalg.lstsq(features, labels)[0]
    mse = mean_sq_error(theta, features, labels)
    return theta, mse
```

70

```
theta, mse = fit(X, y)
```

```
C:\Users\Fabul\AppData\Local\Temp/ipykernel_19652/1547723004.py:2: FutureWarning: `rcond` parameter will
To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old,
  theta = np.linalg.lstsq(features, labels)[0]
```

71

```
theta
```

71

```
array([3.68568136e+00, 6.87371675e-05])
```

72

```
print("Theta0 = %.4f, Theta1 = %.8f" % (theta[0], theta[1]))
```

```
Theta0 = 3.6857, Theta1 = 0.00006874
```

73

```
print("MSE = %.4f" % mse)
```

```
MSE = 1.5522
```

Question. 3

74

```
import dateutil.parser
```

```
75  data = [d for d in book_reviews if 'review_text' in d and 'date_added' in d]
```

```
76  data[0]['review_text']
```

```
76  'This is a special book. It started slow for about the first third, then in the middle third it started t
```

```
77  data[0]['date_added']
```

```
77  'Sun Jul 30 07:44:10 -0700 2017'
```

```
78  t = dateutil.parser.parse(data[0]['date_added'])
```

```
79  t.weekday()
```

```
79  6
```

```
80  t.year
```

```
80  2017
```

```
81  times = [dateutil.parser.parse(d['date_added']) for d in data]
```

```
82  weekdays = list(set([t.weekday() for t in times]))
    weekdays.sort()
    years = list(set([t.year for t in times]))
    years.sort()
```

```
83  weekdays
```

```
83  [0, 1, 2, 3, 4, 5, 6]
```

```
84  years
```

```
84  [2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017]
```

```
85  # the featurizer that uses one-hot encoding for weekdays & years
    def featurizer_t_ohe(datum):
        f = [1, len(datum['review_text'])]
        t = dateutil.parser.parse(datum['date_added'])
        f_wkd = [1 if t.weekday() == d else 0 for d in weekdays]
        f_yr = [1 if t.year == y else 0 for y in years]
        return f + f_wkd + f_yr
```

```
86  X = featurize(data, featurizer_t_ohe)
```

```
87  X[0:2]
```

```
87  [[1, 2086, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
     [1, 1521, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]]
```

```
88  print("For example 1: ")
    data[0]
```

```
For example 1:
```

```
88 {'user_id': '8842281e1d1347389f2ab93d60773d4d',
    'book_id': '18245960',
    'review_id': 'dfdbb7b0eb5a7e4c26d59a937e2e5feb',
    'rating': 5,
    'review_text': 'This is a special book. It started slow for about the first third, then in the middle th
    'date_added': 'Sun Jul 30 07:44:10 -0700 2017',
    'date_updated': 'Wed Aug 30 00:00:26 -0700 2017',
    'read_at': 'Sat Aug 26 12:05:52 -0700 2017',
    'started_at': 'Tue Aug 15 13:23:18 -0700 2017',
    'n_votes': 28,
    'n_comments': 1}
```

```
89 d0_l = len(data[0]['review_text'])
   d0_t = data[0]['date_added']
   d0_t_parsed = dateutil.parser.parse(d0_t)

   print("where its review length is %d, review date is \"%s\", weekday is %d, and year is %d" % (d0_l, d0_t
   print("The feature is " + str(X[0]))
```

```
where its review length is 2086, review date is "Sun Jul 30 07:44:10 -0700 2017", weekday is 6, and year
The feature is [1, 2086, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
```

```
90 print("For example 2: ")
   data[1]
```

```
For example 2:
```

```
90 {'user_id': '8842281e1d1347389f2ab93d60773d4d',
    'book_id': '5577844',
    'review_id': '52c8ac49496c153e4a97161e36b2db55',
    'rating': 5,
    'review_text': 'A beautiful story. Neil Gaiman is truly a unique storyteller. I did a combo of reading a
    'date_added': 'Wed Sep 24 09:29:29 -0700 2014',
    'date_updated': 'Wed Oct 01 00:31:56 -0700 2014',
    'read_at': 'Tue Sep 30 00:00:00 -0700 2014',
    'started_at': 'Sun Sep 21 00:00:00 -0700 2014',
    'n_votes': 5,
    'n_comments': 1}
```

```
91 d1_l = len(data[1]['review_text'])
   d1_t = data[1]['date_added']
   d1_t_parsed = dateutil.parser.parse(d1_t)

   print("where its review length is %d, review date is \"%s\", weekday is %d, and year is %d" % (d1_l, d1_t
   print("The feature is " + str(X[1]))
```

```
where its review length is 1521, review date is "Wed Sep 24 09:29:29 -0700 2014", weekday is 2, and year
The feature is [1, 1521, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
```

Question. 4

```
92 # the featurizer that uses the direct value weekdays & years
   def featurizer_t_dv(datum):
       f = [1, len(datum['review_text'])]
       t = dateutil.parser.parse(datum['date_added'])
       f.append(t.weekday())
```

```
    f.append(t.year)
    return f
```

```
93  X = featurize(data, featurizer_t_dv)
```

```
94  X[0:2]
```

```
94  [[1, 2086, 6, 2017], [1, 1521, 2, 2014]]
```

```
95  theta, mse = fit(X, y)
```

```
C:\Users\Fabul\AppData\Local\Temp/ipykernel_19652/1547723004.py:2: FutureWarning: `rcond` parameter will
To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old,
  theta = np.linalg.lstsq(features, labels)[0]
```

```
96  print("The model's MSE by using time values directly is: %.4f" % mse)
```

```
The model's MSE by using time values directly is: 1.5368
```

```
97  X = featurize(data, featurizer_t_ohe)
```

```
98  X[0:2]
```

```
98  [[1, 2086, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
     [1, 1521, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]]
```

```
99  theta, mse = fit(X, y)
```

```
C:\Users\Fabul\AppData\Local\Temp/ipykernel_19652/1547723004.py:2: FutureWarning: `rcond` parameter will
To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old,
  theta = np.linalg.lstsq(features, labels)[0]
```

```
100 print("The model's MSE by using one-hot encoding is: %.4f" % mse)
```

```
The model's MSE by using one-hot encoding is: 1.5124
```

Question. 5

```
101 import random
```

```
102 shuffled_data = data.copy()
    random.shuffle(shuffled_data)
```

```
103 split_index = int(len(shuffled_data)/2)
    split_index
```

```
103 5000
```

```
104 training_set = shuffled_data[:split_index]
    test_set = shuffled_data[split_index:]
```

105

```
theta_training, mse_training = fit(featurize(training_set, featurizer_t_dv), extract_labels(training_set)
mse_test = mean_sq_error(theta_training, featurize(test_set, featurizer_t_dv), extract_labels(test_set))
```

```
C:\Users\Fabul\AppData\Local\Temp/ipykernel_19652/1547723004.py:2: FutureWarning: `rcond` parameter will
To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old,
  theta = np.linalg.lstsq(features, labels)[0]
```

106

```
print("The model's MSE by using time values directly is: %.4f on training set, %.4f on test set" % (mse_t
```

```
The model's MSE by using time values directly is: 1.5335 on training set, 1.5403 on test set
```

107

```
theta_training, mse_training = fit(featurize(training_set, featurizer_t_ohe), extract_labels(training_set
mse_test = mean_sq_error(theta_training, featurize(test_set, featurizer_t_ohe), extract_labels(test_set))
```

```
C:\Users\Fabul\AppData\Local\Temp/ipykernel_19652/1547723004.py:2: FutureWarning: `rcond` parameter will
To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old,
  theta = np.linalg.lstsq(features, labels)[0]
```

108

```
print("The model's MSE by using using one-hot encoding is: %.4f on training set, %.4f on test set" % (mse
```

```
The model's MSE by using using one-hot encoding is: 1.5013 on training set, 1.5294 on test set
```

Question. 6

$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \theta X_i| = \frac{1}{n}\sum_{i=1}^{n}|y_i - \theta_0|$ for the given predictor $y = \theta_0$.

We can further derive:

$MAE = \frac{1}{n}\sum_{y_i>\theta_0}(y_i - \theta_0) + \frac{1}{n}\sum_{y_i<\theta_0}(\theta_0 - y_i) + \frac{1}{n}\sum_{y_i=\theta_0}(0)$

$\frac{\partial MAE}{\partial \theta_0} = \frac{1}{n}\sum_{y_i>\theta_0}(-1) + \frac{1}{n}\sum_{y_i<\theta_0}(1)$

Let $p = \sum_{j=1}^{n}[y_j > \theta_0]$ and $q = \sum_{j=1}^{n}[y_j < \theta_0]$:

$\frac{\partial MAE}{\partial \theta_0} = \frac{1}{n}\cdot p \cdot (-1) + \frac{1}{n}\cdot q \cdot 1$

$\frac{\partial MAE}{\partial \theta_0} = \frac{q-p}{n}$

When $p = q$, we have $MAE' = 0$ indicating the minimum is reached at this point.

It is only when $\theta_0 = \tilde{y}$ that we can satisfy the condition:

$p = q, i.e. \sum_{j=1}^{n}[y_j > \theta_0] = \sum_{j=1}^{n}[y_j < \theta_0]$.

Therefore, for $y = \theta_0$, the best possible value of θ_0 in terms of the Mean Absolute Error is the median of the label y.

108