Assignment 1
CSE 258
Winter 2017

Xiaowen Mao
PID: A53220159
x9mao@eng.ucsd.edu

Helpfulness Prediction

1. Linear Model

$$\begin{aligned}\frac{nHelpful}{outOf} &= \alpha \\ &+ \theta_1 \times sigmoid(outOf \times 0.8) \\ &+ \theta_2 \times rating \\ &+ \theta_3 \times (if\ CategoryID = 0) \\ &+ \theta_4 \times (if\ CategoryID = 1) \\ &+ \theta_5 \times (if\ CategoryID = 2) \\ &+ \theta_6 \times (if\ CategoryID = 3) \\ &+ \theta_7 \times (if\ CategoryID = 4)\end{aligned}$$

2. Optimization

I use a python library named *sklearn.ensemble.GradientBoostingRegressor*, referenced from scikit-learn.org.
The parameters for the regressor I choose are as following:

$$learning\_rate = 0.01$$
$$max\_depth = 5$$
$$loss = `huber`$$

The loss parameter works for loss function to be optimized. 'huber' is a combination of least squares regression and least absolute deviation.

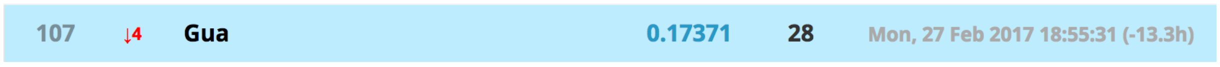
3. Training

I trained the model based on the whole 200,000 dataset.
The validation MAE on 100,000 dataset is

$$MAE = 0.16928$$

5. Team Name on kaggle

My team name on kaggle is **Gua**.



107	↓4	Gua	0.17371	28	Mon, 27 Feb 2017 18:55:31 (-13.3h)

Figure 1: Kaggle Name

6. Source Code

Attached.

Rating Prediction

1. Linear Model

$$rating(u,i) = \alpha + \beta_u + \beta_i$$

2. Optimization

$$argmin_{\alpha,\beta} = \sum_{u,i}(\alpha + \beta_u + \beta_i - R_{u,i})^2 + \lambda_u \times \sum_u (\beta_u)^2 + \lambda_i \times \sum_i (\beta_i)^2$$

3. Iteration

$$\alpha^{(t+1)} = \frac{\sum_{u,i \in train}(R_{u,i} - (\beta_u^{(t)} + \beta_i^{(t)}))}{N_{train}}$$

$$\beta_u^{(t+1)} = \frac{\sum_{i \in I_u}(R_{u,i} - (\alpha^{(t+1)} + \beta_i^{(t)}))}{\lambda_u + |I_u|}$$

$$\beta_i^{(t+1)} = \frac{\sum_{u \in U_i}(R_{u,i} - (\alpha^{(t+1)} + \beta_u^{(t+1)}))}{\lambda_i + |U_i|}$$

4. Training

I trained my model using the whole 200,000 training dataset. After testing on Kaggle, I choose my parameters as following:

$$\lambda_u = 4.5$$

$$\lambda_i = 9.3$$

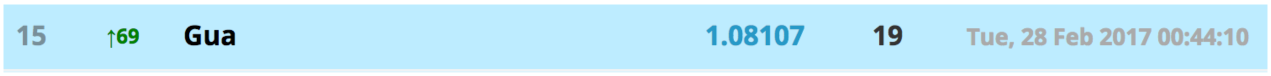
$$iterations = 30$$

And the validation MSE on 100,000 valid dataset is

$$MSE = 0.806253035$$

5. Team Name on kaggle

My team name on kaggle is **Gua**.



15	↑69	Gua	1.08107	19	Tue, 28 Feb 2017 00:44:10

Figure 2: Kaggle Name

6. Source Code

Attached.

task1

February 28, 2017

```
In [ ]: # last submit: Validation MAE = 0.16928, gdr(huber), outOf * 0.8
```

```
In [3]: import gzip
        import numpy
        from collections import defaultdict

        def readGz(f):
            for l in gzip.open(f):
                yield eval(l)
```

```
In [4]: data = []
        for l in readGz("assignment1/train.json.gz"):
            data.append(l)
```

```
In [5]: data_train = data[:100000]
        data_valid = data[100000:]
```

```
In [6]: import math
        import numpy

        def inner(x,y):
            return sum([x[i]*y[i] for i in range(len(x))])

        def sigmoid(x):
            return 1.0 / (1 + numpy.exp(-x))
```

```
In [7]: def feature(datum):
            feat= [1]
            feat.append(sigmoid(datum['helpful']['outOf'] * 0.8))
            feat.append(datum['rating'])

        #   categoryID
            if (datum['categoryID'] == 0):
                for i in [1,0,0,0,0]:
                    feat.append(i)
            elif (datum['categoryID'] == 1):
                for i in [0,1,0,0,0]:
                    feat.append(i)
            elif (datum['categoryID'] == 2):
                for i in [0,0,1,0,0]:
                    feat.append(i)
            elif (datum['categoryID'] == 3):
                for i in [0,0,0,1,0]:
                    feat.append(i)
            elif (datum['categoryID'] == 4):
                for i in [0,0,0,0,1]:
                    feat.append(i)

            return feat
```

```
In [8]: X = [feature(d) for d in data if d['helpful']['outOf'] > 0]
        y = [d['helpful']['nHelpful'] * 1.0 / d['helpful']['outOf'] for d in data if d['helpful'
```

```
In [15]: from sklearn.ensemble import GradientBoostingRegressor as gdr
         regressor = gdr(learning_rate=0.01, max_depth=5, loss='huber')
         # regressor = gdr(loss='ls')
         regressor.fit(X,y)
```

```
Out[15]: GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
                      learning_rate=0.01, loss='huber', max_depth=5,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_split=1e-07, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, presort='auto', random_state=None,
                      subsample=1.0, verbose=0, warm_start=False)
```

```
In [16]: # validation MAE
         X_v = [feature(d) for d in data_valid if d['helpful']['outOf'] > 0]
         o_v = [d['helpful']['outOf'] for d in data_valid if d['helpful']['outOf'] > 0]
         y_v = [d['helpful']['nHelpful'] for d in data_valid if d['helpful']['outOf'] > 0]
         predict = regressor.predict(X_v)
```

```
In [17]: MAE = 0
         for i in range(len(predict)):
             res = round(predict[i] * o_v[i])
             MAE += math.fabs(res - y_v[i])

         print("Validation MAE = " + str(MAE / len(data_valid)))
```

```
Validation MAE = 0.16928
```

```
In [18]: # predict
         data_test = []
         for l in readGz("assignment1/test_Helpful.json.gz"):
             if l['helpful']['outOf'] > 0:
                 data_test.append(l)
         X_test = [feature(d) for d in data_test]
         o_test = [d['helpful']['outOf'] for d in data_test]
         predict = regressor.predict(X_test)

In [19]: reviews = {}
         for i in range(len(data_test)):
             user,item, outOf = data_test[i]['reviewerID'], data_test[i]['itemID'], data_test[i]
             key = user + ' ' + item + ' ' + str(outOf)
             reviews[key] = predict[i]

In [20]: # kaggle
         predictions = open("assignment1/predictions_Helpful.txt", 'w')

         for l in open("assignment1/pairs_Helpful.txt"):
             if l.startswith("userID"):
                 # first line
                 predictions.write(l)
                 continue

             u, i, outOf = l.strip().split('-')
             key = u + ' ' + i + ' ' + outOf
             outOf = int(outOf)
             if outOf > 0:
                 res = round(outOf * reviews[key])
             else:
                 res = 0.0
             predictions.write(u + '-' + i + '-' + str(outOf) + ',' + str(res)+ '\n')

         predictions.close ()
```

task2

February 28, 2017

```
In [ ]: # last submit: 4.5, 9.3
```

```
In [2]: import gzip
        from collections import defaultdict
        import math
        import scipy.optimize
        from sklearn import svm
        import numpy
        import string

        def readGz(f):
            for l in gzip.open(f):
                yield eval(l)
```

```
In [3]: data = []
        for l in readGz("assignment1/train.json.gz"):
            data.append(l)

        data_train = data[:100000]
        data_valid = data[100000:]
        UserRating = defaultdict(list)
        ItemRating = defaultdict(list)
        for r in data:
            UserRating[r['reviewerID']].append(r)
            ItemRating[r['itemID']].append(r)
```

```
In [4]: trainRatings = [r['rating'] for r in data]
        globalAverage = sum(trainRatings) * 1.0 / len(trainRatings)

        betaU = {}
        betaI = {}
        for u in UserRating:
            betaU[u] = 0

        for i in ItemRating:
            betaI[i] = 0

        alpha = globalAverage
```

```
In [5]: def iterate(lamU, lamI):
            # update alpha
            newAlpha = 0
            for r in data:
                newAlpha += r['rating'] - (betaU[r['reviewerID']] + betaI[r['itemID']])
            alpha = newAlpha / len(data)

            # update betaU
            for u in UserRating:
                newBetaU = 0
                for r in UserRating[u]:
                    newBetaU += r['rating'] - (alpha + betaI[r['itemID']])
                betaU[u] = newBetaU / (lamU + len(UserRating[u]))

            # update betaI
            for i in ItemRating:
                newBetaI = 0
                for r in ItemRating[i]:
                    newBetaI += r['rating'] - (alpha + betaU[r['reviewerID']])
                betaI[i] = newBetaI / (lamI + len(ItemRating[i]))

            # cal mse
            mse = 0
            for r in data:
                predict = alpha + betaU[r['reviewerID']] + betaI[r['itemID']]
                mse += (r['rating'] - predict)**2

            # add regularizer
            regU = 0
            regI = 0
            for u in betaU:
                regU += betaU[u]**2
            for i in betaI:
                regI += betaI[i]**2

            mse /= len(data)
            return mse, mse + lamU*regU + lamI*regI

In [6]: # lamU = 4.5
        # lamI = 9.3
        # MSE = 0.806253035
        # iteration = 30

        mse,objective = iterate(1,1)
        newMSE,newObjective = iterate(1,1)

        n = 1
        while n < 30 or objective - newObjective > 0.0001:
            mse, objective = newMSE, newObjective
            newMSE, newObjective = iterate(4.5, 9.3)
            n += 1

        validMSE = 0
        for r in data_valid:
            bu = 0
            bi = 0
            if r['reviewerID'] in betaU:
                bu = betaU[r['reviewerID']]
            if r['itemID'] in betaI:
                bi = betaI[r['itemID']]
            prediction = alpha + bu + bi
            validMSE += (r['rating'] - prediction)**2

        validMSE /= len(data_valid)
        print("MSE = " + str(validMSE))
```

```
MSE = 0.806253035
```

```
In [9]: predictions = open("assignment1/predictions_Rating.txt", 'w')
        for l in open("assignment1/pairs_Rating.txt"):
            if l.startswith("userID"):
                #header
                predictions.write(l)
                continue
            u,i = l.strip().split('-')

            x = alpha
            if u in betaU:
                x += betaU[u]
            if i in betaI:
                x += betaI[i]
            predictions.write(u + '-' + i + ',' + str(x) + '\n')

        predictions.close()
```