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 ()
```