Problem 7

In [154]:

```
import numpy as np
import urllib
import scipy.optimize
from random import shuffle
import string
from collections import defaultdict
from nltk import bigrams
from nltk.stem.porter import *
from sklearn import linear_model
from sklearn.metrics import mean_squared_error
from sklearn.metrics.pairwise import cosine_similarity
from math import log
```

In [155]:

```
def parseData(fname):
    for l in urllib.request.urlopen(fname):
        yield eval(l)
```

In [156]:

```
print("Reading data...")
data = list(parseData("file:beer_50000.json"))
print("done")
```

```
Reading data...
done
```

In [157]:

```
shuffle(data)
```

In [158]:

```
data_train = data[:5000]
data_valid = data[5000:10000]
data_test = data[10000:15000]
```

1. unigrams, removing punctuation, tfidf

In [159]:

```
punctuation = set(string.punctuation)
reviewText = [''.join([c for c in datum['review/text'].lower() if not c in
```

In [160]:

```
unigramCount = defaultdict(int)
for d in data:
    r = ''.join([c for c in d['review/text'].lower() if not c in punctuati
    for w in r.split():
        unigramCount[w] += 1
```

In [161]:

```
countsUnigram = [(unigramCount[w], w) for w in unigramCount]
unigrams = [x[1] for x in countsUnigram[:1000]]
unigramId = dict(zip(unigrams, range(len(unigrams))))
unigramSet = set(unigrams)
```

In [162]:

```
countsUnigram.sort()
countsUnigram.reverse()
```

In [163]:

```
def feature_000(datum):
    feat = [0] * len(unigrams)
    unigramCount_rt = defaultdict(int)
    r = ''.join([c for c in datum['review/text'].lower() if not c in punct
    for w in r.split():
        unigramCount_rt[w] += 1
    for unigram in unigramId:
        if unigram in unigramCount_rt:
            feat[unigramId[unigram]] = unigramCount_rt[unigram] *  log(150
    feat.append(1) #offset
    return feat
```

In [164]:

```
X_train = [feature_000(d) for d in data_train]
y_train = [d['review/overall'] for d in data_train]
X_valid = [feature_000(d) for d in data_valid]
y_valid = [d['review/overall'] for d in data_valid]
X_test = [feature_000(d) for d in data_test]
y_test = [d['review/overall'] for d in data_test]
```

In [165]:

```
def compare_unigrams(reg):
    clf = linear_model.Ridge(reg, fit_intercept=False)
    clf.fit(X_train, y_train)
    theta = clf.coef_
    predictions = clf.predict(X_valid)
    predictions = np.matrix(predictions)
    y_validation = np.matrix(y_valid)
    diff = y_validation - predictions
    MSE = diff * diff.T / 5000
    print("reg for unigrams= ", r, "MSE = ", MSE)
```

In [166]:

```
reg = [0.01, 0.1, 1.0, 10, 100]
for r in reg:
    compare_unigrams(r)
```

```
reg for unigrams=  0.01 MSE =  [[0.40196911]]
reg for unigrams=  0.1 MSE =  [[0.39984351]]
reg for unigrams=  1.0 MSE =  [[0.39184914]]
reg for unigrams=  10 MSE =  [[0.37368687]]
reg for unigrams=  100 MSE =  [[0.38978366]]
```

In [167]:

```
clf = linear_model.Ridge(10, fit_intercept=False)
clf.fit(X_train, y_train)
theta = clf.coef_
predictions = clf.predict(X_test)
predictions = np.matrix(predictions)
y_testing = np.matrix(y_test)
diff = y_testing - predictions
MSE = diff * diff.T / 5000
print("optimal lambda = 10, testing MSE = " + str(MSE))
```

```
optimal lambda = 10, testing MSE = [[0.38394586]]
```

2. unigrams, removing punctuation, word counts

In [168]:
```
punctuation = set(string.punctuation)
reviewText = [''.join([c for c in datum['review/text'].lower() if not c in
```

In [169]:
```
unigramCount = defaultdict(int)
for d in data:
    r = ''.join([c for c in d['review/text'].lower() if not c in punctuati
    for w in r.split():
        unigramCount[w] += 1
```

In [170]:
```
countsUnigram = [(unigramCount[w], w) for w in unigramCount]
unigrams = [x[1] for x in countsUnigram[:1000]]
unigramId = dict(zip(unigrams, range(len(unigrams))))
unigramSet = set(unigrams)
```

In [171]:
```
countsUnigram.sort()
countsUnigram.reverse()
```

In [172]:
```
def feature_001(datum):
    feat = [0] * len(unigrams)
    unigramCount_rt = defaultdict(int)
    r = ''.join([c for c in datum['review/text'].lower() if not c in punct
    for w in r.split():
        if w in unigrams:
            feat[unigramId[w]] += 1
    feat.append(1) #offset
    return feat
```

In [173]:
```
X_train = [feature_001(d) for d in data_train]
y_train = [d['review/overall'] for d in data_train]
X_valid = [feature_001(d) for d in data_valid]
y_valid = [d['review/overall'] for d in data_valid]
X_test = [feature_001(d) for d in data_test]
y_test = [d['review/overall'] for d in data_test]
```

In [174]:
```
def compare_unigrams(reg):
    clf = linear_model.Ridge(reg, fit_intercept=False)
    clf.fit(X_train, y_train)
    theta = clf.coef_
    predictions = clf.predict(X_valid)
    predictions = np.matrix(predictions)
    y_validation = np.matrix(y_valid)
    diff = y_validation - predictions
    MSE = diff * diff.T / 5000
    print("reg for unigrams= ", r, "MSE = ", MSE)
```

In [175]:
```
reg = [0.01, 0.1, 1.0, 10, 100]
for r in reg:
    compare_unigrams(r)
```

```
reg for unigrams=  0.01 MSE =  [[0.40064272]]
reg for unigrams=  0.1 MSE =  [[0.3948727]]
reg for unigrams=  1.0 MSE =  [[0.38137263]]
reg for unigrams=  10 MSE =  [[0.36343046]]
reg for unigrams=  100 MSE =  [[0.39376672]]
```

In [176]:
```
clf = linear_model.Ridge(10, fit_intercept=False)
clf.fit(X_train, y_train)
theta = clf.coef_
predictions = clf.predict(X_test)
predictions = np.matrix(predictions)
y_testing = np.matrix(y_test)
diff = y_testing - predictions
MSE = diff * diff.T / 5000
print("optimal lambda = 10, testing MSE = " + str(MSE))
```

```
optimal lambda = 10, testing MSE = [[0.37596804]]
```

3. unigrams, preserving punctuation, tfidf

In [177]:
```
punctuation = set(string.punctuation)
reviewText = [''.join([c for c in datum['review/text'].lower() if not c in
```

In [178]:
```
unigramCount = defaultdict(int)
for d in data:
    unigramList = []
    for c in d['review/text'].lower():
        if c not in punctuation:
            unigramList.append(c)
        else:
            unigramList.append("  ")
            unigramList.append(c)
    r = ''.join(unigramList)
    for w in r.split():
        unigramCount[w] += 1
```

In [179]:
```
countsUnigram = [(unigramCount[w], w) for w in unigramCount]
unigrams = [x[1] for x in countsUnigram[:1000]]
unigramId = dict(zip(unigrams, range(len(unigrams))))
unigramSet = set(unigrams)
```

In [180]:
```
countsUnigram.sort()
countsUnigram.reverse()
```

In [181]:

```
def feature_010(datum):
    feat = [0] * len(unigrams)
    unigramCount_datum = defaultdict(int)

    unigramList = []
    for c in datum['review/text'].lower():
        if c not in punctuation:
            unigramList.append(c)
        else:
            unigramList.append(" ")
            unigramList.append(c)
    r = ''.join(unigramList)

    for w in r.split():
        unigramCount_datum[w] += 1

    for unigram in unigramId:
        if unigram in unigramCount_datum:
            feat[unigramId[unigram]] = unigramCount_datum[unigram] * log(1

    feat.append(1) #offset
    return feat
```

In [182]:

```
X_train = [feature_010(d) for d in data_train]
y_train = [d['review/overall'] for d in data_train]
X_valid = [feature_010(d) for d in data_valid]
y_valid = [d['review/overall'] for d in data_valid]
X_test = [feature_010(d) for d in data_test]
y_test = [d['review/overall'] for d in data_test]
```

In [183]:

```
def compare_unigrams(reg):
    clf = linear_model.Ridge(reg, fit_intercept=False)
    clf.fit(X_train, y_train)
    theta = clf.coef_
    predictions = clf.predict(X_valid)
    predictions = np.matrix(predictions)
    y_validation = np.matrix(y_valid)
    diff = y_validation - predictions
    MSE = diff * diff.T / 5000
    print("reg for unigrams= ", r, "MSE = ", MSE)
```

In [184]:

```
reg = [0.01, 0.1, 1.0, 10, 100]
for r in reg:
    compare_unigrams(r)
```

```
reg for unigrams=  0.01 MSE =  [[0.40508638]]
reg for unigrams=  0.1 MSE =  [[0.40344859]]
reg for unigrams=  1.0 MSE =  [[0.39609762]]
reg for unigrams=  10 MSE =  [[0.37728858]]
reg for unigrams=  100 MSE =  [[0.39379484]]
```

In [185]:

```
clf = linear_model.Ridge(10, fit_intercept=False)
clf.fit(X_train, y_train)
theta = clf.coef_
predictions = clf.predict(X_test)
predictions = np.matrix(predictions)
y_testing = np.matrix(y_test)
diff = y_testing - predictions
MSE = diff * diff.T / 5000
print("optimal lambda = 10, testing MSE = " + str(MSE))
```

```
optimal lambda = 10, testing MSE = [[0.38785107]]
```

4. unigrams, preserving punctuation, word counts

In [186]:

```
punctuation = set(string.punctuation)
reviewText = [''.join([c for c in datum['review/text'].lower() if not c in
```

In [187]:

```
unigramCount = defaultdict(int)
for d in data:
    unigramList = []
    for c in d['review/text'].lower():
        if c not in punctuation:
            unigramList.append(c)
        else:
            unigramList.append(" ")
            unigramList.append(c)
    r = ''.join(unigramList)
    for w in r.split():
        unigramCount[w] += 1
```

In [188]:

```
countsUnigram = [(unigramCount[w], w) for w in unigramCount]
unigrams = [x[1] for x in countsUnigram[:1000]]
unigramId = dict(zip(unigrams, range(len(unigrams))))
unigramSet = set(unigrams)
```

In [189]:

```
countsUnigram.sort()
countsUnigram.reverse()
```

In [190]:

```
def feature_011(datum):
    feat = [0] * len(unigrams)
    #unigramCount_rt = defaultdict(int)
    r = ''.join([c for c in datum['review/text'].lower() if not c in punct
    for w in r.split():
        if w in unigrams:
            feat[unigramId[w]] += 1
    feat.append(1) #offset
    return feat
```

In [191]:

```
X_train = [feature_010(d) for d in data_train]
y_train = [d['review/overall'] for d in data_train]
X_valid = [feature_010(d) for d in data_valid]
y_valid = [d['review/overall'] for d in data_valid]
X_test = [feature_010(d) for d in data_test]
y_test = [d['review/overall'] for d in data_test]
```

In [192]:

```
def compare_unigrams(reg):
    clf = linear_model.Ridge(reg, fit_intercept=False)
    clf.fit(X_train, y_train)
    theta = clf.coef_
    predictions = clf.predict(X_valid)
    predictions = np.matrix(predictions)
    y_validation = np.matrix(y_valid)
    diff = y_validation - predictions
    MSE = diff * diff.T / 5000
    print("reg for unigrams= ", r, "MSE = ", MSE)
```

In [193]:

```
reg = [0.01, 0.1, 1.0, 10, 100]
for r in reg:
    compare_unigrams(r)
```

```
reg for unigrams=  0.01 MSE =  [[0.40508638]]
reg for unigrams=  0.1 MSE =  [[0.40344859]]
reg for unigrams=  1.0 MSE =  [[0.39609762]]
reg for unigrams=  10 MSE =  [[0.37728858]]
reg for unigrams=  100 MSE =  [[0.39379484]]
```

In [194]:

```
clf = linear_model.Ridge(10, fit_intercept=False)
clf.fit(X_train, y_train)
theta = clf.coef_
predictions = clf.predict(X_test)
predictions = np.matrix(predictions)
y_testing = np.matrix(y_test)
diff = y_testing - predictions
MSE = diff * diff.T / 5000
print("optimal lambda = 10, testing MSE = " + str(MSE))
```

```
optimal lambda = 10, testing MSE = [[0.38785107]]
```

5. bigrams, removing punctuation, tfidf

In [195]:

```
bigramCount = defaultdict(int)
punctuation = set(string.punctuation)
#stemmer = PorterStemmer()
for d in data:
    r = ''.join([c for c in d['review/text'].lower() if not c in punctuati
    words = r.split()
    for i in range(0,len(words)-1):
        bigram = words[i] + " " + words[i+1]
        bigramCount[bigram] += 1
```

In [196]:

```
countsBigram = [(bigramCount[d], d) for d in bigramCount.keys()]
countsBigram.sort()
countsBigram.reverse()
```

In [197]:

```
bigrams = [c[1] for c in countsBigram[:1000]]
bigramId = dict(zip(bigrams, range(len(bigrams))))
bigramSet = set(bigrams)
```

In [198]:

```
def feature_100(datum):
    feat = [0] * len(bigrams)
    bigramCount_datum = defaultdict(int)
    r = ''.join([c for c in datum['review/text'].lower() if not c in punct
    words = r.split()
    for i in range(len(words) - 1):
        bigram = words[i] + " " + words[i + 1]
        bigramCount_datum[bigram] += 1

    for bigram in bigramId:
        if bigram in bigramCount_datum:
            feat[bigramId[bigram]] = bigramCount_datum[bigram] * log(15000

    feat.append(1) #offset
    return feat
```

In [199]:

```
X_train = [feature_100(d) for d in data_train]
y_train = [d['review/overall'] for d in data_train]
X_valid = [feature_100(d) for d in data_valid]
y_valid = [d['review/overall'] for d in data_valid]
X_test = [feature_100(d) for d in data_test]
y_test = [d['review/overall'] for d in data_test]
```

In [200]:

```
def compare_bigrams(reg):
    clf = linear_model.Ridge(reg, fit_intercept=False)
    clf.fit(X_train, y_train)
    theta = clf.coef_
    predictions = clf.predict(X_valid)
    predictions = np.matrix(predictions)
    y_validation = np.matrix(y_valid)
    diff = y_validation - predictions
    MSE = diff * diff.T / 5000
    print("reg for bigrams= ", r, "MSE = ", MSE)
```

In [201]:

```
reg = [0.01, 0.1, 1.0, 10, 100]
for r in reg:
    compare_bigrams(r)
```

```
reg for bigrams=  0.01 MSE =  [[0.44638918]]
reg for bigrams=  0.1 MSE =  [[0.44612483]]
reg for bigrams=  1.0 MSE =  [[0.44349019]]
reg for bigrams=  10 MSE =  [[0.42654711]]
reg for bigrams=  100 MSE =  [[0.4132646]]
```

In [202]:

```
clf = linear_model.Ridge(100, fit_intercept=False)
clf.fit(X_train, y_train)
theta = clf.coef_
predictions = clf.predict(X_test)
predictions = np.matrix(predictions)
y_testing = np.matrix(y_test)
diff = y_testing - predictions
MSE = diff * diff.T / 5000
print("optimal lambda = 100, testing MSE = " + str(MSE))
```

```
optimal lambda = 100, testing MSE = [[0.42533635]]
```

6. bigrams, removing punctuation, word counts

In [203]:

```
bigramCount = defaultdict(int)
punctuation = set(string.punctuation)
#stemmer = PorterStemmer()
for d in data:
    r = ''.join([c for c in d['review/text'].lower() if not c in punctuati
    words = r.split()
    for i in range(0,len(words)-1):
        bigram = words[i] + " " + words[i+1]
        bigramCount[bigram] += 1
```

In [204]:

```
countsBigram = [(bigramCount[d], d) for d in bigramCount.keys()]
countsBigram.sort()
countsBigram.reverse()
```

In [205]:

```
bigrams = [c[1] for c in countsBigram[:1000]]
bigramId = dict(zip(bigrams, range(len(bigrams))))
bigramSet = set(bigrams)
```

In [206]:

```
def feature_101(datum):
    feat = [0]*len(bigrams)
    r = ''.join([c for c in datum['review/text'].lower() if not c in punct
    words = r.split()
    for i in range(len(words)-1):
        bigram = words[i] + " " + words[i + 1]
        if bigram in bigrams:
            feat[bigramId[bigram]] += 1
    feat.append(1) #offset
    return feat
```

In [207]:

```
X_train = [feature_101(d) for d in data_train]
y_train = [d['review/overall'] for d in data_train]
X_valid = [feature_101(d) for d in data_valid]
y_valid = [d['review/overall'] for d in data_valid]
X_test = [feature_101(d) for d in data_test]
y_test = [d['review/overall'] for d in data_test]
```

In [208]:

```
def compare_bigrams(reg):
    clf = linear_model.Ridge(reg, fit_intercept=False)
    clf.fit(X_train, y_train)
    theta = clf.coef_
    predictions = clf.predict(X_valid)
    predictions = np.matrix(predictions)
    y_validation = np.matrix(y_valid)
    diff = y_validation - predictions
    MSE = diff * diff.T / 5000
    print("reg for bigrams= ", r, "MSE = ", MSE)
```

In [209]:

```
reg = [0.01, 0.1, 1.0, 10, 100]
for r in reg:
    compare_bigrams(r)
```

```
reg for bigrams=  0.01 MSE =  [[0.44597822]]
reg for bigrams=  0.1 MSE =  [[0.44569662]]
reg for bigrams=  1.0 MSE =  [[0.44299749]]
reg for bigrams=  10 MSE =  [[0.42425457]]
reg for bigrams=  100 MSE =  [[0.41755767]]
```

In [210]:

```
clf = linear_model.Ridge(100, fit_intercept=False)
clf.fit(X_train, y_train)
theta = clf.coef_
predictions = clf.predict(X_test)
predictions = np.matrix(predictions)
y_testing = np.matrix(y_test)
diff = y_testing - predictions
MSE = diff * diff.T / 5000
print("optimal lambda = 100, testing MSE = " + str(MSE))
```

```
optimal lambda = 100, testing MSE = [[0.43058102]]
```

7. bigrams, preserving punctuation, tfidf

In [211]:

```
punctuation = set(string.punctuation)
reviewText = [''.join([c for c in datum['review/text'].lower() if not c in
```

In [212]:

```
bigramCount = defaultdict(int)
for d in data:
    bigramList = []
    for c in d['review/text'].lower():
        if c not in punctuation:
            bigramList.append(c)
        else:
            bigramList.append(" ")
            bigramList.append(c)
    r = ''.join(bigramList)
    words = r.split()
    for i in range(len(words) - 1):
        bigram = words[i] + " " + words[i + 1]
        bigramCount[bigram] += 1
```

In [213]:

```
countsBigram = [(bigramCount[d], d) for d in bigramCount.keys()]
countsBigram.sort()
countsBigram.reverse()
```

In [214]:

```
bigrams = [c[1] for c in countsBigram[:1000]]
bigramId = dict(zip(bigrams, range(len(bigrams))))
bigramSet = set(bigrams)
```

In [215]:

```
def feature_110(datum):
    feat = [0] * len(bigrams)
    bigramCount_datum = defaultdict(int)

    bigramList = []
    for c in datum['review/text'].lower():
        if c not in punctuation:
            bigramList.append(c)
        else:
            bigramList.append(" ")
            bigramList.append(c)
    r = ''.join(bigramList)
    words = r.split()
    for i in range(len(words) - 1):
        bigram = words[i] + " " + words[i + 1]
        bigramCount_datum[bigram] += 1

    for bigram in bigramId:
        if bigram in bigramCount_datum:
            feat[bigramId[bigram]] = bigramCount_datum[bigram] * log(15000

    feat.append(1) #offset
    return feat
```

In [216]:

```
X_train = [feature_110(d) for d in data_train]
y_train = [d['review/overall'] for d in data_train]
X_valid = [feature_110(d) for d in data_valid]
y_valid = [d['review/overall'] for d in data_valid]
X_test = [feature_110(d) for d in data_test]
y_test = [d['review/overall'] for d in data_test]
```

In [217]:

```
def compare_bigrams(reg):
    clf = linear_model.Ridge(reg, fit_intercept=False)
    clf.fit(X_train, y_train)
    theta = clf.coef_
    predictions = clf.predict(X_valid)
    predictions = np.matrix(predictions)
    y_validation = np.matrix(y_valid)
    diff = y_validation - predictions
    MSE = diff * diff.T / 5000
    print("reg for bigrams= ", r, "MSE = ", MSE)
```

In [218]:

```
reg = [0.01, 0.1, 1.0, 10, 100]
for r in reg:
    compare_bigrams(r)
```

```
reg for bigrams=  0.01 MSE =  [[0.44922545]]
reg for bigrams=  0.1 MSE =  [[0.44880617]]
reg for bigrams=  1.0 MSE =  [[0.44514748]]
reg for bigrams=  10 MSE =  [[0.42573408]]
reg for bigrams=  100 MSE =  [[0.41411603]]
```

In [219]:

```
clf = linear_model.Ridge(100, fit_intercept=False)
clf.fit(X_train, y_train)
theta = clf.coef_
predictions = clf.predict(X_test)
predictions = np.matrix(predictions)
y_testing = np.matrix(y_test)
diff = y_testing - predictions
MSE = diff * diff.T / 5000
print("optimal lambda = 100, testing MSE = " + str(MSE))
```

```
optimal lambda = 100, testing MSE = [[0.4299674]]
```

8. bigrams, preserving punctuation, word counts

In [220]:

```
punctuation = set(string.punctuation)
reviewText = [''.join([c for c in datum['review/text'].lower() if not c in
```

In [221]:

```
bigramCount = defaultdict(int)
for d in data:
    bigramList = []
    for c in d['review/text'].lower():
        if c not in punctuation:
            bigramList.append(c)
        else:
            bigramList.append(" ")
            bigramList.append(c)
    r = ''.join(bigramList)
    words = r.split()
    for i in range(len(words) - 1):
        bigram = words[i] + " " + words[i + 1]
        bigramCount[bigram] += 1
```

In [222]:

```
countsBigram = [(bigramCount[d], d) for d in bigramCount.keys()]
countsBigram.sort()
countsBigram.reverse()
```

In [223]:

```
bigrams = [c[1] for c in countsBigram[:1000]]
bigramId = dict(zip(bigrams, range(len(bigrams))))
bigramSet = set(bigrams)
```

In [224]:

```
def feature_111(datum):
    feat = [0] * len(bigrams)
    # bigramCount_datum = defaultdict(int)

    bigramList = []
    for c in datum['review/text'].lower():
        if c not in punctuation:
            bigramList.append(c)
        else:
            bigramList.append(" ")
            bigramList.append(c)
    r = ''.join(bigramList)
    words = r.split()
    for i in range(len(words) - 1):
        bigram = words[i] + " " + words[i + 1]
        if bigram in bigrams:
            feat[bigramId[bigram]] += 1

    feat.append(1) #offset
    return feat
```

In [225]:

```
X_train = [feature_111(d) for d in data_train]
y_train = [d['review/overall'] for d in data_train]
X_valid = [feature_111(d) for d in data_valid]
y_valid = [d['review/overall'] for d in data_valid]
X_test = [feature_111(d) for d in data_test]
y_test = [d['review/overall'] for d in data_test]
```

In [226]:

```
def compare_bigrams(reg):
    clf = linear_model.Ridge(reg, fit_intercept=False)
    clf.fit(X_train, y_train)
    theta = clf.coef_
    predictions = clf.predict(X_valid)
    predictions = np.matrix(predictions)
    y_validation = np.matrix(y_valid)
    diff = y_validation - predictions
    MSE = diff * diff.T / 5000
    print("reg for bigrams= ", r, "MSE = ", MSE)
```

In [227]:

```
reg = [0.01, 0.1, 1.0, 10, 100]
for r in reg:
    compare_bigrams(r)
```

```
reg for bigrams=  0.01 MSE =  [[0.44906436]]
reg for bigrams=  0.1 MSE =  [[0.4487985]]
reg for bigrams=  1.0 MSE =  [[0.44627841]]
reg for bigrams=  10 MSE =  [[0.42936252]]
reg for bigrams=  100 MSE =  [[0.42557995]]
```

In [228]:

```
clf = linear_model.Ridge(100, fit_intercept=False)
clf.fit(X_train, y_train)
theta = clf.coef_
predictions = clf.predict(X_test)
predictions = np.matrix(predictions)
y_testing = np.matrix(y_test)
diff = y_testing - predictions
MSE = diff * diff.T / 5000
print("optimal lambda = 100, testing MSE = " + str(MSE))
```

```
optimal lambda = 100, testing MSE = [[0.44222335]]
```