hw4_keliu

November 26, 2018

0.1 Problem 1

```
In [23]: import numpy
         import urllib
         import scipy.optimize
         import random
         from collections import defaultdict
         import nltk
         import string
         from nltk.stem.porter import *
         from sklearn import linear_model
         from math import log10
```

```
In [4]: def parseData(fname):
          for l in urllib.request.urlopen(fname):
            yield eval(l)

        ### Just the first 5000 reviews

        print ("Reading data...")
        data = list(parseData("http://jmcauley.ucsd.edu/cse190/data/beer/beer_50000.json"))[:5
        print ("done")
```

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

```
In [188]: ### Ignore capitalization and remove punctuation

          wordCount = 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 punctuation])
            w = r.split()
              #w = stemmer.stem(w) # with stemming
            for i in range(len(w)-1):
              wordCount[w[i]+' '+w[i+1]] += 1
```

```
In [7]: frequent_bigram = [(wordCount[i],i) for i in wordCount]
        frequent_bigram.sort()
        frequent_bigram.reverse()
        frequent_bigram[:5]

Out[7]: [(4587, 'with a'),
         (2595, 'in the'),
         (2245, 'of the'),
         (2056, 'is a'),
         (2033, 'on the')]
```

0.2 Problem 2

```
In [8]: words = [x[1] for x in frequent_bigram[:1000]]
        wordId = dict(zip(words, range(len(words))))
        wordSet = set(words)

In [9]: def feature(datum):
          feat = [0]*len(words)
          r = ''.join([c for c in datum['review/text'].lower() if not c in punctuation])
          w = r.split()
          for i in range(len(w)-1):
            tmp = w[i]+' '+w[i+1]
            if tmp in words:
              feat[wordId[tmp]] += 1
          feat.append(1) #offset
          return feat

In [10]: X = [feature(d) for d in data]
         y = [d['review/overall'] for d in data]

         #No regularization
         #theta,residuals,rank,s = numpy.linalg.lstsq(X, y)

         #With regularization
         clf = linear_model.Ridge(1.0, fit_intercept=False)
         clf.fit(X, y)
         theta = clf.coef_
         predictions = clf.predict(X)

In [11]: ## mse
         err = 0
         for i in range(len(y)):
             err += (y[i]-predictions[i])**2
         err =err/5000
         err

Out[11]: 0.3431530140613628
```

0.3 Problem 3

```
In [63]: def idf(term):
             num = 0
             for d in data:
                 r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
                 if term in r.split():
                     num += 1
             return -log10(num/len(data))

In [64]: def tf(t,d):
             num = 0
             r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
             for term in r.split():
                 if term == t:
                     num += 1
             return num

In [67]: terms = ['foam', 'smell', 'banana', 'lactic', 'tart']
         for term in terms:
             print(term + ' :idf = '+str(idf(term))+'   tf-idf = '+str(tf(term,data[0])*idf(te
```

```
foam :idf = 1.1378686206869628   tf-idf = 2.2757372413739256
smell :idf = 0.5379016188648442   tf-idf = 0.5379016188648442
banana :idf = 1.6777807052660807   tf-idf = 3.3555614105321614
lactic :idf = 2.9208187539523753   tf-idf = 5.841637507904751
tart :idf = 1.8068754016455384   tf-idf = 1.8068754016455384
```

0.4 Problem 4

```
In [68]: uniCount = defaultdict(int)
         punctuation = set(string.punctuation)
         for d in data:
           r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
           for w in r.split():
             uniCount[w] += 1

In [77]: ### donnot run it again
         word_idf = defaultdict(int)
         for w in uniCount:
             word_idf[w] = idf(w)

In [80]: counts = [(uniCount[w], w) for w in uniCount]
         counts.sort()
         counts.reverse()

         words = [x[1] for x in counts]
         wordId = dict(zip(words, range(len(words))))
```

```
        wordSet = set(words)

        doc1 = data[0]
        doc2 = data[1]

        d1 = [0]*len(uniCount)
        d2 = [0]*len(uniCount)


        r = ''.join([c for c in doc1['review/text'].lower() if not c in punctuation])
        for term in r.split():
            d1[wordId[term]] = word_idf[term]*tf(term,doc1)

        r = ''.join([c for c in doc2['review/text'].lower() if not c in punctuation])
        for term in r.split():
            d2[wordId[term]] = word_idf[term]*tf(term,doc2)
```

```
In [85]: import numpy as np
         from math import sqrt
```

```
In [92]: def cosine_similarity(d1,d2):
             nominator = np.dot(d1,d2)
             denominator = sqrt(sum(i**2 for i in d1))*sqrt(sum(i**2 for i in d2))
             if denominator == 0 : return 0
             cosine = nominator/denominator
             return cosine
```

```
In [93]: cosine_similarity(d1,d2)
```

```
Out[93]: 0.0658819397474438
```

0.5 Problem 5

```
In [94]: max_cos = 0
         beer_id = ''
         profile_name = ''
         for d in data[1:]:
             d_cur = [0]*len(uniCount)
             r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
             for term in r.split():
                d_cur[wordId[term]] = word_idf[term]*tf(term,d)
             tmp = cosine_similarity(d1,d_cur)
             if tmp > max_cos :
                 max_cos = tmp
                 beer_id = d['beer/beerId']
                 profile_name = d['user/profileName']
         print(beer_id)
         print(profile_name)
```

```
72146
spicelab
```

0.6 Problem 6

In [194]:
```
### Just take the most popular words...

wordCount = defaultdict(int)
punctuation = set(string.punctuation)
for d in data:
  r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
  for w in r.split():
    wordCount[w] += 1

counts = [(wordCount[w], w) for w in wordCount]
counts.sort()
counts.reverse()

words = [x[1] for x in counts[:1000]]
wordId = dict(zip(words, range(len(words))))
wordSet = set(words)
```

In [195]:
```
def feature(datum):
  feat = [0]*len(words)
  r = ''.join([c for c in datum['review/text'].lower() if not c in punctuation])
  w = r.split()
  for i in w:
    tmp = word_idf[i]*tf(i,datum)
    if i in wordSet:
       feat[wordId[i]] = tmp
  feat.append(1) #offset
  return feat
```

In [196]:
```
X = [feature(d) for d in data]
y = [d['review/overall'] for d in data]

#No regularization
#theta,residuals,rank,s = numpy.linalg.lstsq(X, y)

#With regularization
clf = linear_model.Ridge(1.0, fit_intercept=False)
clf.fit(X, y)
theta = clf.coef_
predictions = clf.predict(X)
```

In [199]:
```
## mse
err = 0
for i in range(len(y)):
```

```
            err += (y[i]-predictions[i])**2
        err =err/5000
        print("mse with tf-idf is: "+str(err))
```

```
mse with tf-idf is: 0.2787595600777218
```

0.7 Problem 7

```
In [227]: print ("Reading data...")
          data_all = list(parseData("http://jmcauley.ucsd.edu/cse190/data/beer/beer_50000.json"
          print ("done")
```

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

```
In [228]: import random
          random.shuffle(data_all)
```

```
In [229]: train = data_all[:5000]
          validation = data_all[5000:10000]
          test = data_all[10000:15000]
```

0.7.1 model 1 : unigram, preserve punctuation, word count

```
In [230]: ### unigram
          wordCount = defaultdict(int)
          for d in train:
              r = ''.join(c for c in d['review/text'].lower())
              for w in r.split():
                    wordCount[w] += 1
```

```
In [231]: counts = [(wordCount[w], w) for w in wordCount]
          counts.sort()
          counts.reverse()

          words = [x[1] for x in counts[:1000]]
          wordId = dict(zip(words, range(len(words))))
          wordSet = set(words)
```

```
In [232]: def feature(datum):
            feat = [0]*len(words)
            r = ''.join(c for c in datum['review/text'].lower())
            for w in r.split():
              if w in words:
                feat[wordId[w]] += 1
            feat.append(1) #offset
            return feat
```

```
In [233]: X_train = [feature(d) for d in train]
          y_train = [d['review/overall'] for d in train]


          X_validation = [feature(d) for d in validation]
          y_validation = [d['review/overall'] for d in validation]

          X_test = [feature(d) for d in test]
          y_test = [d['review/overall'] for d in test]

In [234]: #No regularization
          #theta,residuals,rank,s = numpy.linalg.lstsq(X, y)

          #With regularization

          def mse(lamda):
             clf = linear_model.Ridge(lamda, fit_intercept=False)
             clf.fit(X_train, y_train)
             theta = clf.coef_
             predictions = clf.predict(X_validation)

             ## mse
             err = 0
             for i in range(len(y_validation)):
                err += (y_validation[i]-predictions[i])**2
             err =err/5000
             return err

In [235]: print("lambda = 0.01  mse of validation: " +str(mse(0.01)))
          print("lambda = 0.1  mse of validation: " +str(mse(0.1)))
          print("lambda = 1  mse of validation: " +str(mse(1)))
          print("lambda = 10  mse of validation: " +str(mse(10)))
          print("lambda = 100  mse of validation: " +str(mse(100)))
```

```
lambda = 0.01  mse of validation: 0.39608492544268364
lambda = 0.1  mse of validation: 0.3959317131915895
lambda = 1  mse of validation: 0.39446116829442684
lambda = 10  mse of validation: 0.3842667979968166
lambda = 100  mse of validation: 0.40611767673772076
```

According to the mse we obtained, we should choose lambda = 10

```
In [236]: ## choose lambda = 10
          clf = linear_model.Ridge(10, fit_intercept=False)
          clf.fit(X_train, y_train)
          theta = clf.coef_
          predictions = clf.predict(X_test)
```

```
## mse
err = 0
for i in range(len(y_test)):
      err += (y_test[i]-predictions[i])**2
err =err/5000
print("mse on the test set is: "+str(err))
```

```
mse on the test set is: 0.4032378768323837
```

0.7.2 model 2: bigram, preserve punctuation, word counts

In [237]:
```
wordCount = defaultdict(int)
punctuation = set(string.punctuation)
stemmer = PorterStemmer()
for d in train:
  r = ''.join(c for c in d['review/text'].lower())
  w = r.split()
    #w = stemmer.stem(w) # with stemming
  for i in range(len(w)-1):
    wordCount[w[i]+' '+w[i+1]] += 1
```

In [238]:
```
frequent_bigram = [(wordCount[i],i) for i in wordCount]
frequent_bigram.sort()
frequent_bigram.reverse()

words = [x[1] for x in frequent_bigram[:1000]]
wordId = dict(zip(words, range(len(words))))
wordSet = set(words)
```

In [239]:
```
def feature(datum):
  feat = [0]*len(words)
  r = ''.join(c for c in datum['review/text'].lower())
  w = r.split()
  for i in range(len(w)-1):
    tmp = w[i]+' '+w[i+1]
    if tmp in words:
      feat[wordId[tmp]] += 1
  feat.append(1) #offset
  return feat
```

In [240]:
```
X_train = [feature(d) for d in train]
y_train = [d['review/overall'] for d in train]


X_validation = [feature(d) for d in validation]
y_validation = [d['review/overall'] for d in validation]

X_test = [feature(d) for d in test]
y_test = [d['review/overall'] for d in test]
```

```
In [241]: #No regularization
          #theta,residuals,rank,s = numpy.linalg.lstsq(X, y)

          #With regularization

          def mse(lamda):
             clf = linear_model.Ridge(lamda, fit_intercept=False)
             clf.fit(X_train, y_train)
             theta = clf.coef_
             predictions = clf.predict(X_validation)

             ## mse
             err = 0
             for i in range(len(y_validation)):
                err += (y_validation[i]-predictions[i])**2
             err =err/5000
             return err

In [242]: print("lambda = 0.01  mse of validation: " +str(mse(0.01)))
          print("lambda = 0.1  mse of validation: " +str(mse(0.1)))
          print("lambda = 1  mse of validation: " +str(mse(1)))
          print("lambda = 10  mse of validation: " +str(mse(10)))
          print("lambda = 100  mse of validation: " +str(mse(100)))

lambda = 0.01  mse of validation: 0.4587312780018877
lambda = 0.1  mse of validation: 0.4583889598679832
lambda = 1  mse of validation: 0.4552387407523705
lambda = 10  mse of validation: 0.4359396997199695
lambda = 100  mse of validation: 0.4362181043469898
```

Based on the performance on validation set, we choose lambda = 10

```
In [243]: ## choose lambda = 10
          clf = linear_model.Ridge(10, fit_intercept=False)
          clf.fit(X_train, y_train)
          theta = clf.coef_
          predictions = clf.predict(X_test)

          ## mse
          err = 0
          for i in range(len(y_test)):
                err += (y_test[i]-predictions[i])**2
          err =err/5000
          print("mse on the test set is: "+str(err))

mse on the test set is: 0.44809417453688355
```

0.7.3 model 3 : unigram, remove punctuation, word counts

In [244]:
```
### Ignore capitalization and remove punctuation

wordCount = defaultdict(int)
punctuation = set(string.punctuation)
stemmer = PorterStemmer()
for d in train:
  r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
  for w in r.split():
    #w = stemmer.stem(w) # with stemming
    wordCount[w] += 1
```

In [246]:
```
counts = [(wordCount[w], w) for w in wordCount]
counts.sort()
counts.reverse()

words = [x[1] for x in counts[:1000]]
wordId = dict(zip(words, range(len(words))))
wordSet = set(words)
```

In [247]:
```
def feature(datum):
  feat = [0]*len(words)
  r = ''.join([c for c in datum['review/text'].lower() if not c in punctuation])
  for w in r.split():
    if w in words:
      feat[wordId[w]] += 1
  feat.append(1) #offset
  return feat
```

In [248]:
```
X_train = [feature(d) for d in train]
y_train = [d['review/overall'] for d in train]


X_validation = [feature(d) for d in validation]
y_validation = [d['review/overall'] for d in validation]

X_test = [feature(d) for d in test]
y_test = [d['review/overall'] for d in test]
```

In [249]:
```
#No regularization
#theta,residuals,rank,s = numpy.linalg.lstsq(X, y)

#With regularization

def mse(lamda):
   clf = linear_model.Ridge(lamda, fit_intercept=False)
   clf.fit(X_train, y_train)
   theta = clf.coef_
```

```
            predictions = clf.predict(X_validation)

            ## mse
            err = 0
            for i in range(len(y_validation)):
               err += (y_validation[i]-predictions[i])**2
            err =err/5000
            return err
```

```
In [250]: print("lambda = 0.01  mse of validation: " +str(mse(0.01)))
          print("lambda = 0.1  mse of validation: " +str(mse(0.1)))
          print("lambda = 1  mse of validation: " +str(mse(1)))
          print("lambda = 10  mse of validation: " +str(mse(10)))
          print("lambda = 100  mse of validation: " +str(mse(100)))
```

```
lambda = 0.01  mse of validation: 0.379454667980617
lambda = 0.1  mse of validation: 0.37930442186825364
lambda = 1  mse of validation: 0.3778577799756602
lambda = 10  mse of validation: 0.3677593169062544
lambda = 100  mse of validation: 0.39213104419367617
```

choose lambda = 10

```
In [251]: ## choose lambda = 10
          clf = linear_model.Ridge(10, fit_intercept=False)
          clf.fit(X_train, y_train)
          theta = clf.coef_
          predictions = clf.predict(X_test)

          ## mse
          err = 0
          for i in range(len(y_test)):
                err += (y_test[i]-predictions[i])**2
          err =err/5000
          print("mse on the test set is: "+str(err))
```

```
mse on the test set is: 0.38943060307690436
```

0.7.4 model 4: bigram, remove punctuation, word counts

```
In [252]: ### Ignore capitalization and remove punctuation

          wordCount = defaultdict(int)
          punctuation = set(string.punctuation)
          stemmer = PorterStemmer()
          for d in train:
            r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
```

```
        w = r.split()
          #w = stemmer.stem(w) # with stemming
        for i in range(len(w)-1):
          wordCount[w[i]+' '+w[i+1]] += 1
```

In [253]:
```
frequent_bigram = [(wordCount[i],i) for i in wordCount]
frequent_bigram.sort()
frequent_bigram.reverse()

words = [x[1] for x in frequent_bigram[:1000]]
wordId = dict(zip(words, range(len(words))))
wordSet = set(words)
```

In [254]:
```
def feature(datum):
  feat = [0]*len(words)
  r = ''.join([c for c in datum['review/text'].lower() if not c in punctuation])
  w = r.split()
  for i in range(len(w)-1):
    tmp = w[i]+' '+w[i+1]
    if tmp in words:
      feat[wordId[tmp]] += 1
  feat.append(1) #offset
  return feat
```

In [255]:
```
X_train = [feature(d) for d in train]
y_train = [d['review/overall'] for d in train]


X_validation = [feature(d) for d in validation]
y_validation = [d['review/overall'] for d in validation]

X_test = [feature(d) for d in test]
y_test = [d['review/overall'] for d in test]
```

In [256]:
```
#No regularization
#theta,residuals,rank,s = numpy.linalg.lstsq(X, y)

#With regularization

def mse(lamda):
   clf = linear_model.Ridge(lamda, fit_intercept=False)
   clf.fit(X_train, y_train)
   theta = clf.coef_
   predictions = clf.predict(X_validation)

   ## mse
   err = 0
   for i in range(len(y_validation)):
      err += (y_validation[i]-predictions[i])**2
```

```
            err =err/5000
            return err
```

```
In [257]: print("lambda = 0.01  mse of validation: " +str(mse(0.01)))
          print("lambda = 0.1  mse of validation: " +str(mse(0.1)))
          print("lambda = 1  mse of validation: " +str(mse(1)))
          print("lambda = 10  mse of validation: " +str(mse(10)))
          print("lambda = 100  mse of validation: " +str(mse(100)))
```

```
lambda = 0.01  mse of validation: 0.4426702835166532
lambda = 0.1  mse of validation: 0.44243655374939134
lambda = 1  mse of validation: 0.44023705689289544
lambda = 10  mse of validation: 0.424838839599379
lambda = 100  mse of validation: 0.4273780468358466
```

Choose lambda = 10

```
In [258]: ## choose lambda = 10
          clf = linear_model.Ridge(10, fit_intercept=False)
          clf.fit(X_train, y_train)
          theta = clf.coef_
          predictions = clf.predict(X_test)

          ## mse
          err = 0
          for i in range(len(y_test)):
                err += (y_test[i]-predictions[i])**2
          err =err/5000
          print("mse on the test set is: "+str(err))
```

```
mse on the test set is: 0.4483852646172084
```

0.7.5 model 5: unigram, remove punctuation, tf-idf

```
In [259]: def unigram_idf(term):
              num = 0
              for d in train:
                  r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
                  if term in r.split():
                      num += 1
              return -log10(num/len(train))
```

```
In [260]: ### Just take the most popular words...

          wordCount = defaultdict(int)
          punctuation = set(string.punctuation)
          for d in train:
```

```
    r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
    for w in r.split():
      wordCount[w] += 1

  counts = [(wordCount[w], w) for w in wordCount]
  counts.sort()
  counts.reverse()

  words = [x[1] for x in counts[:1000]]
  wordId = dict(zip(words, range(len(words))))
  wordSet = set(words)
```

In [261]:
```
uni_idf = defaultdict(int)
for i in words:
    uni_idf[i] = unigram_idf(i)
```

In [263]:
```
def feature(datum):
  feat = [0]*len(words)
  r = ''.join([c for c in datum['review/text'].lower() if not c in punctuation])
  w = r.split()
  for i in w:
    if i in wordSet:
       tmp = uni_idf[i]*tf(i,datum)
       feat[wordId[i]] = tmp
  feat.append(1) #offset
  return feat
```

In [264]:
```
X_train = [feature(d) for d in train]
y_train = [d['review/overall'] for d in train]


X_validation = [feature(d) for d in validation]
y_validation = [d['review/overall'] for d in validation]

X_test = [feature(d) for d in test]
y_test = [d['review/overall'] for d in test]
```

In [265]:
```
#No regularization
#theta,residuals,rank,s = numpy.linalg.lstsq(X, y)

#With regularization

def mse(lamda):
   clf = linear_model.Ridge(lamda, fit_intercept=False)
   clf.fit(X_train, y_train)
   theta = clf.coef_
   predictions = clf.predict(X_validation)

   ## mse
```

```
            err = 0
            for i in range(len(y_validation)):
               err += (y_validation[i]-predictions[i])**2
            err =err/5000
            return err
```

```
In [266]: print("lambda = 0.01  mse of validation: " +str(mse(0.01)))
          print("lambda = 0.1  mse of validation: " +str(mse(0.1)))
          print("lambda = 1  mse of validation: " +str(mse(1)))
          print("lambda = 10  mse of validation: " +str(mse(10)))
          print("lambda = 100  mse of validation: " +str(mse(100)))
```

```
lambda = 0.01  mse of validation: 0.3794683914844995
lambda = 0.1  mse of validation: 0.37943754775378147
lambda = 1  mse of validation: 0.37900731516842195
lambda = 10  mse of validation: 0.3755721347449421
lambda = 100  mse of validation: 0.40353962517928066
```

Choose lambda = 10

```
In [267]: ## choose lambda = 10
          clf = linear_model.Ridge(10, fit_intercept=False)
          clf.fit(X_train, y_train)
          theta = clf.coef_
          predictions = clf.predict(X_test)

          ## mse
          err = 0
          for i in range(len(y_test)):
                err += (y_test[i]-predictions[i])**2
          err =err/5000
          print("mse on the test set is: "+str(err))
```

```
mse on the test set is: 0.398328745046825
```

0.7.6 model 6: unigram, preserve punctuation, tf-idf

```
In [268]: def unigram_idf2(term):
              num = 0
              for d in train:
                  r = ''.join(c for c in d['review/text'].lower())
                  if term in r.split():
                      num += 1
              return -log10(num/len(train))
```

```
In [306]: def tf_preserve(t,d):
              num = 0
```

```
    r = ''.join(c for c in d['review/text'].lower())
    for term in r.split():
        if term == t:
            num += 1
    return num
```

In [307]:
```
### Just take the most popular words...

wordCount = defaultdict(int)
punctuation = set(string.punctuation)
for d in train:
  r = ''.join(c for c in d['review/text'].lower())
  for w in r.split():
    wordCount[w] += 1

counts = [(wordCount[w], w) for w in wordCount]
counts.sort()
counts.reverse()

words = [x[1] for x in counts[:1000]]
wordId = dict(zip(words, range(len(words))))
wordSet = set(words)
```

In [308]:
```
uni_idf2 = defaultdict(int)
for i in words:
    uni_idf2[i] = unigram_idf2(i)
```

In [309]:
```
def feature(datum):
  feat = [0]*len(words)
  r = ''.join(c for c in datum['review/text'].lower())
  w = r.split()
  for i in w:
    if i in wordSet:
       tmp = uni_idf2[i]*tf_preserve(i,datum)
       feat[wordId[i]] = tmp
  feat.append(1) #offset
  return feat
```

In [310]:
```
X_train = [feature(d) for d in train]
y_train = [d['review/overall'] for d in train]


X_validation = [feature(d) for d in validation]
y_validation = [d['review/overall'] for d in validation]

X_test = [feature(d) for d in test]
y_test = [d['review/overall'] for d in test]
```

In [311]:
```
#No regularization
#theta,residuals,rank,s = numpy.linalg.lstsq(X, y)
```

```
#With regularization

def mse(lamda):
    clf = linear_model.Ridge(lamda, fit_intercept=False)
    clf.fit(X_train, y_train)
    theta = clf.coef_
    predictions = clf.predict(X_validation)

    ## mse
    err = 0
    for i in range(len(y_validation)):
        err += (y_validation[i]-predictions[i])**2
    err =err/5000
    return err

print("lambda = 0.01  mse of validation: " +str(mse(0.01)))
print("lambda = 0.1  mse of validation: " +str(mse(0.1)))
print("lambda = 1  mse of validation: " +str(mse(1)))
print("lambda = 10  mse of validation: " +str(mse(10)))
print("lambda = 100  mse of validation: " +str(mse(100)))
```

```
lambda = 0.01  mse of validation: 0.3960970367051074
lambda = 0.1  mse of validation: 0.3960494003577027
lambda = 1  mse of validation: 0.3954696222066391
lambda = 10  mse of validation: 0.39089098243413944
lambda = 100  mse of validation: 0.41681817393572
```

Choose lambda = 10

```
In [312]: ## choose lambda = 10
clf = linear_model.Ridge(10, fit_intercept=False)
clf.fit(X_train, y_train)
theta = clf.coef_
predictions = clf.predict(X_test)

## mse
err = 0
for i in range(len(y_test)):
      err += (y_test[i]-predictions[i])**2
err =err/5000
print("mse on the test set is: "+str(err))
```

```
mse on the test set is: 0.4106750195508106
```

0.7.7 model 7: bigram, remove punctuation, tf-idf

```
In [282]: def bigram_idf(term):
              num = 0
              for d in train:
                  r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
                  w = r.split()
                  for i in range(len(w)-1):
                      if (w[i]+' '+w[i+1]) == term:
                          num += 1
              return -log10(num/len(train))

In [283]: def bigram_tf(t,d):
              num = 0
              r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
              w= r.split()
              for i in range(len(w)-1):
                  if (w[i]+' '+w[i+1]) == t:
                      num += 1
              return num

In [284]: ### Ignore capitalization and remove punctuation

          wordCount = defaultdict(int)
          punctuation = set(string.punctuation)
          stemmer = PorterStemmer()
          for d in train:
            r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
            w = r.split()
            #w = stemmer.stem(w) # with stemming
            for i in range(len(w)-1):
              wordCount[w[i]+' '+w[i+1]] += 1

In [285]: frequent_bigram = [(wordCount[i],i) for i in wordCount]
          frequent_bigram.sort()
          frequent_bigram.reverse()

          words = [x[1] for x in frequent_bigram[:1000]]
          wordId = dict(zip(words, range(len(words))))
          wordSet = set(words)

In [287]: bi_idf = defaultdict(int)
          for tmp in words:
              bi_idf[tmp] = bigram_idf(tmp)

In [288]: def feature(datum):
            feat = [0]*len(words)
            r = ''.join([c for c in datum['review/text'].lower() if not c in punctuation])
            w = r.split()
```

```
            for i in range(len(w)-1):
              tmp = w[i]+' '+w[i+1]
              if tmp in words:
                feat[wordId[tmp]] = bi_idf[tmp]*bigram_tf(tmp,datum)
            feat.append(1) #offset
            return feat
```

In [289]:
```
X_train = [feature(d) for d in train]
y_train = [d['review/overall'] for d in train]


X_validation = [feature(d) for d in validation]
y_validation = [d['review/overall'] for d in validation]

X_test = [feature(d) for d in test]
y_test = [d['review/overall'] for d in test]
```

In [290]:
```
#No regularization
#theta,residuals,rank,s = numpy.linalg.lstsq(X, y)

#With regularization

def mse(lamda):
   clf = linear_model.Ridge(lamda, fit_intercept=False)
   clf.fit(X_train, y_train)
   theta = clf.coef_
   predictions = clf.predict(X_validation)

   ## mse
   err = 0
   for i in range(len(y_validation)):
      err += (y_validation[i]-predictions[i])**2
   err =err/5000
   return err

print("lambda = 0.01  mse of validation: " +str(mse(0.01)))
print("lambda = 0.1  mse of validation: " +str(mse(0.1)))
print("lambda = 1  mse of validation: " +str(mse(1)))
print("lambda = 10  mse of validation: " +str(mse(10)))
print("lambda = 100  mse of validation: " +str(mse(100)))
```

```
lambda = 0.01  mse of validation: 0.44268492861956155
lambda = 0.1  mse of validation: 0.44258261376942376
lambda = 1  mse of validation: 0.44166302414275777
lambda = 10  mse of validation: 0.4349377766043576
lambda = 100  mse of validation: 0.441876929275772
```

choose lambda = 10

```
In [291]: ## choose lambda = 10
          clf = linear_model.Ridge(10, fit_intercept=False)
          clf.fit(X_train, y_train)
          theta = clf.coef_
          predictions = clf.predict(X_test)

          ## mse
          err = 0
          for i in range(len(y_test)):
                err += (y_test[i]-predictions[i])**2
          err =err/5000
          print("mse on the test set is: "+str(err))
```

```
mse on the test set is: 0.46019509380338913
```

0.7.8 model 8 : bigram, preserve punctuation, tf-idf

```
In [292]: def bigram_idf2(term):
              num = 0
              for d in train:
                  r = ''.join(c for c in d['review/text'].lower())
                  w = r.split()
                  for i in range(len(w)-1):
                      if (w[i]+' '+w[i+1]) == term:
                          num += 1
              return -log10(num/len(train))
```

```
In [293]: def bigram_tf2(t,d):
              num = 0
              r = ''.join(c for c in d['review/text'].lower())
              w= r.split()
              for i in range(len(w)-1):
                  if (w[i]+' '+w[i+1]) == t:
                      num += 1
              return num
```

```
In [294]: wordCount = defaultdict(int)
          punctuation = set(string.punctuation)
          stemmer = PorterStemmer()
          for d in train:
            r = ''.join(c for c in d['review/text'].lower())
            w = r.split()
            #w = stemmer.stem(w) # with stemming
            for i in range(len(w)-1):
              wordCount[w[i]+' '+w[i+1]] += 1
```

```
In [295]: frequent_bigram = [(wordCount[i],i) for i in wordCount]
          frequent_bigram.sort()
```

```
frequent_bigram.reverse()

words = [x[1] for x in frequent_bigram[:1000]]
wordId = dict(zip(words, range(len(words))))
wordSet = set(words)
```

In [296]:
```
bi_idf2 = defaultdict(int)
for tmp in words:
    bi_idf2[tmp] = bigram_idf2(tmp)
```

In [297]:
```
def feature(datum):
  feat = [0]*len(words)
  r = ''.join(c for c in datum['review/text'].lower())
  w = r.split()
  for i in range(len(w)-1):
    tmp = w[i]+' '+w[i+1]
    if tmp in words:
      feat[wordId[tmp]] = bi_idf2[tmp]*bigram_tf2(tmp,datum)
  feat.append(1) #offset
  return feat
```

In [298]:
```
X_train = [feature(d) for d in train]
y_train = [d['review/overall'] for d in train]


X_validation = [feature(d) for d in validation]
y_validation = [d['review/overall'] for d in validation]

X_test = [feature(d) for d in test]
y_test = [d['review/overall'] for d in test]
```

In [299]:
```
#No regularization
#theta,residuals,rank,s = numpy.linalg.lstsq(X, y)

#With regularization

def mse(lamda):
   clf = linear_model.Ridge(lamda, fit_intercept=False)
   clf.fit(X_train, y_train)
   theta = clf.coef_
   predictions = clf.predict(X_validation)

   ## mse
   err = 0
   for i in range(len(y_validation)):
      err += (y_validation[i]-predictions[i])**2
   err =err/5000
   return err
```

```
print("lambda = 0.01  mse of validation: " +str(mse(0.01)))
print("lambda = 0.1  mse of validation: " +str(mse(0.1)))
print("lambda = 1  mse of validation: " +str(mse(1)))
print("lambda = 10  mse of validation: " +str(mse(10)))
print("lambda = 100  mse of validation: " +str(mse(100)))
```

```
lambda = 0.01  mse of validation: 0.4587524316647613
lambda = 0.1  mse of validation: 0.45860018942688857
lambda = 1  mse of validation: 0.45725953885052295
lambda = 10  mse of validation: 0.4482363334483097
lambda = 100  mse of validation: 0.4495139426053207
```

Choose lambda = 10

```
In [300]: ## choose lambda = 10
          clf = linear_model.Ridge(10, fit_intercept=False)
          clf.fit(X_train, y_train)
          theta = clf.coef_
          predictions = clf.predict(X_test)

          ## mse
          err = 0
          for i in range(len(y_test)):
                err += (y_test[i]-predictions[i])**2
          err =err/5000
          print("mse on the test set is: "+str(err))
```

```
mse on the test set is: 0.4600827802914737
```

0.8 summary of 8 models

Unigram/bigram	Remove/preserve punctuation	tf-idf/word counts	mse on test set
Unigram	has punctuation	word counts	0.4032
bigram	has	word counts	0.4481
Unigram	no	word counts	0.3894
bigram	no	word counts	0.4484
Unigram	no	tf-idf	0.3983
Unigram	has	tf-idf	0.4107
bigram	no	tf-idf	0.4602
bigram	has	tf-idf	0.4601

According to the comparison, model with unigram, tf-idf and no punctuation has the best performance.

In [ ]: