kaggle_Loss_Function

August 10, 2016

In [1]:
```
#This code run gradient descent on the kaggle RMSLE loss function with L2 Regularization
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import pandas as pd
from copy import copy
from sklearn.ensemble import RandomForestRegressor
import csv
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder


dateparse=lambda x:pd.datetime.strptime(x,'%Y-%m-%d %H:%M:%S')
train=pd.read_csv('train.csv',parse_dates=['datetime'],date_parser=dateparse)
test=pd.read_csv('test.csv',parse_dates=['datetime'],date_parser=dateparse)

#This made very little difference for the kaggle loss function using gradient decent

#test['windspeed']=np.log(test['windspeed']+1)
#train['windspeed']=np.log(train['windspeed']+1)
print 'train.shape is ',train.shape,' and test.shape is ',test.shape


def extractFeaturesTrain(data):
    #print 'data is ',data
    data['Hour']=data.datetime.dt.hour
    labels=data['count']
    train_years=data.datetime.dt.year
    train_months=data.datetime.dt.month
    data=data.drop(['datetime','count','casual','registered'], axis = 1)

    return np.array(data),np.array(labels),np.array(train_years),np.array(train_months),(data.c

def extractFeaturesTest(data):
    #print 'data is \n',data
    data['Hour']=data.datetime.dt.hour
    test_years=data.datetime.dt.year
    test_months=data.datetime.dt.month
    data=data.drop(['datetime'], axis = 1)
    return np.array(data),np.array(test_years),np.array(test_months)

train2=copy(train)
test2=copy(test)
```

```
        test=np.array(test)
        #print 'train2 is ',train2
        traind,labelsTrain,train_years,train_months,headers=extractFeaturesTrain(train2)
        testd,test_years,test_months=extractFeaturesTest(test2)
        print 'traind.shape is ',traind.shape,' and testd.shape is ',testd.shape
```

```
train.shape is  (10886, 12)  and test.shape is  (6493, 9)
traind.shape is  (10886, 9)  and testd.shape is  (6493, 9)
```

In [2]:
```
        enc=OneHotEncoder(categorical_features=[0,1,2,3,8],sparse=False)
        traind2=enc.fit_transform(traind)
        print traind2.shape
        testd2=enc.fit_transform(testd)
        print testd2.shape
        ones1=np.ones((traind.shape[0],1))
        ones2=np.ones((testd.shape[0],1))
        traind2=copy(np.hstack((traind2,ones1)))
        testd2=copy(np.hstack((testd2,ones2)))
        print traind2.shape
        print testd2.shape
```

```
(10886, 40)
(6493, 40)
(10886, 41)
(6493, 41)
```

In [3]:
```
        train=np.array(train)

        def getSplits(years,months):
            locsTrain=[]
            locsTest=[]
            print 'in getSplits ,train is \n',train
            for i in range(0,train.shape[0]):
                    if (train[i,0].year==years[0] or train[i,0].year==years[1]) and (train[i,0].month in
                        locsTest.append(i)
                    else:
                        locsTrain.append(i)

            return locsTrain,locsTest

        def getCustomLocsTest(year,month,data):
            locs=[]
            for i in range(0,data.shape[0]):
                if data[i][0].year==year and data[i][0].month==month:
                    locs.append(i)
            return locs

        def replaceNegaticeValuesWithZeroAndCountThem(ypred):
            count=0
            for i in range(ypred.shape[0]):
                if(ypred[i]<0):
                    ypred[i]=0
                    count+=1
            #print 'Number of Negative values predicted are ',count
            return ypred,count
```

```
def calculateGradientAndLoss(weights,x,y,year,month):
    y2=y.reshape(-1,1)
    weights=weights.reshape(1,-1)
    print 'weights are \n',weights.shape
    #print 'y2 is ',y2.shape
    losses=[]
    lambdas=np.arange(0.0,2,0.1)
    #lambdas=[0.0]
    minLoss=9999999
    learning_rate=0.4
    fig=plt.figure()
    fig.set_size_inches(18.5, 10.5)
    fig.suptitle('Gradient Descent on Cross Validation for year  '+str(year)+' with test month a
    fig.subplots_adjust(hspace=0.5)
    fig.text(0.5,0.04,'-------------------------------------------------------------------------Itera
    fig.text(0.08,0.5,'-----------------------------------------------------Kaggle Loss function
    counter=1
    for lambda1 in lambdas:
        weights=np.random.rand(1,x.shape[1])
        print 'For regularization factor ',lambda1
        losses=[]
        for i in range(1000):
            h=np.dot(x,weights.T)
            #print 'h is \n',h
            h,count=replaceNegaticeValuesWithZeroAndCountThem(h)
            err=np.log(y2+1)-np.log(h+1)
            #print 'err is ',err.shape
            loss=np.sum(err**2)
            loss=np.sqrt(loss/x.shape[0])
            grad1=(err)/(h+1)
            #print 'grad1 is ',grad1.shape
            grad=( 2*np.dot(grad1.T,x)/x.shape[0] ) - lambda1*weights
            #print 'Loss at iteration ',i,' is ',loss
            losses.append(loss)
            #print 'grad is ',grad.shape
            weights=weights+learning_rate*grad
        if loss<minLoss:
            minLoss=loss
            weightsRes=weights


        print 'final training loss is ',loss
        #plt.figure().set_size_inches(18.5, 10.5)
        #plt.plot(losses)

        title= ' Regularization of '+ str(lambda1).split('.')[0]+'_'+str(lambda1).split('.')[1]

        ax=fig.add_subplot(4,5,counter)
        counter=counter+1
        ax.plot(losses)
        #print 'title is ',title
        ax.set_title(title)
        #plt.xlabel('Iterations')
```

```
        #plt.ylabel('Kaggle  Loss functions')
    plt.savefig('Gradient Descent on Cross Validation for year  '+str(year)+' with test month as
    plt.show()
    return weightsRes

def TrainFucntion(x,y,year,month):
    weights=np.random.rand(1,x.shape[1])
    for i in range(x.shape[1]):
        max1=max(x[:,i])
        if max1!=0:
            x[:,i]=x[:,i]/max1

    weights=calculateGradientAndLoss(weights,x,y,year,month)
    return weights


def Predict(weights,test):

    return np.dot(test,weights.T)

def findLoss(gold,predicted):
    loss=0

    #print 'predicted is ',predicted
    for i in range(gold.shape[0]):
        loss+=(np.log(predicted[i]+1) -np.log(gold[i]+1))**2

    loss=loss/gold.shape[0]
    return np.sqrt(loss)

def crossValidate():
        months=[12]
        locsTrain,locsTest=getSplits([2011,2012],months)

        testSubset=traind2[locsTest]
        testSubset2=train[locsTest]
        testLabels=labelsTrain[locsTest]
        #rf3=RandomForestRegressor(20)

        trainSubset=traind2[locsTrain]
        trainSubset2=train[locsTrain]
        trainLabels=labelsTrain[locsTrain]

        for i in [2011,2012]:
            for j in months:
                testLocs=getCustomLocsTest(i,j,testSubset2)
                testSubset3=testSubset2[testLocs]
                testSubset4=testSubset[testLocs]
                testLabels4=testLabels[testLocs]

                trainLocs2=np.where(trainSubset2[:,0]<=min(testSubset3[:,0]))

                trainSubset3=trainSubset[trainLocs2]
                trainLabels3=trainLabels[trainLocs2]
```

```
                        x1=trainSubset2[trainLocs2]
                        x2=testSubset2[testLocs]

                        print 'trainSubset min is  ', min(x1[:,0]),' and max is ',max(x1[:,0])
                        print 'testSubset  min is  ', min(x2[:,0]),' and max is ',max(x2[:,0])

                        #rf3.fit(trainSubset3,trainLabels3)change here to program new function to train
                        weights=TrainFucntion(trainSubset3,trainLabels3,i,j)

                        ypred=Predict(weights,testSubset4)
                        ypred,count=replaceNegaticeValuesWithZeroAndCountThem(ypred)
                        print 'Number of Negative values predicted are ',count
                        print 'loss with year =',i,' and month = ',j,' is ',findLoss(testLabels4,ypred)

        crossValidate()
```

```
in getSplits ,train is
[[Timestamp('2011-01-01 00:00:00') 1 0 ..., 13 16 0]
 [Timestamp('2011-01-01 01:00:00') 1 0 ..., 32 40 1]
 [Timestamp('2011-01-01 02:00:00') 1 0 ..., 27 32 2]
 ...,
 [Timestamp('2012-12-19 21:00:00') 4 0 ..., 164 168 21]
 [Timestamp('2012-12-19 22:00:00') 4 0 ..., 117 129 22]
 [Timestamp('2012-12-19 23:00:00') 4 0 ..., 84 88 23]]
trainSubset min is   2011-01-01 00:00:00  and max is  2011-11-19 23:00:00
testSubset  min is   2011-12-01 00:00:00  and max is  2011-12-19 23:00:00
weights are
(1, 41)
For regularization factor  0.0
final training loss is  1.36496093134
For regularization factor  0.1
final training loss is  2.24331121767
For regularization factor  0.2
final training loss is  2.45930489673
For regularization factor  0.3
final training loss is  2.59113775189
For regularization factor  0.4
final training loss is  2.68654447052
For regularization factor  0.5
final training loss is  2.76139717248
For regularization factor  0.6
final training loss is  2.8229991281
For regularization factor  0.7
final training loss is  2.87533170724
For regularization factor  0.8
final training loss is  2.92080960883
For regularization factor  0.9
final training loss is  2.96100970462
For regularization factor  1.0
final training loss is  2.99701939817
For regularization factor  1.1
final training loss is  3.02962065707
For regularization factor  1.2
final training loss is  3.05939484218
```

```
For regularization factor  1.3
final training loss is  3.08678604799
For regularization factor  1.4
final training loss is  3.1121412365
For regularization factor  1.5
final training loss is  3.13573668248
For regularization factor  1.6
final training loss is  3.15779598073
For regularization factor  1.7
final training loss is  3.17850265585
For regularization factor  1.8
final training loss is  3.19800920779
For regularization factor  1.9
final training loss is  3.2164437387
Number of Negative values predicted are  0
loss with year = 2011  and month =  12  is  [ 2.73863954]
trainSubset min is   2011-01-01 00:00:00  and max is  2012-11-19 23:00:00
testSubset  min is   2012-12-01 00:00:00  and max is  2012-12-19 23:00:00
weights are
(1, 41)
For regularization factor  0.0
final training loss is  1.44264136432
For regularization factor  0.1
final training loss is  2.43784771087
For regularization factor  0.2
final training loss is  2.66352617924
For regularization factor  0.3
final training loss is  2.80019313493
For regularization factor  0.4
final training loss is  2.89871218921
For regularization factor  0.5
final training loss is  2.97581685078
For regularization factor  0.6
final training loss is  3.03916257802
For regularization factor  0.7
final training loss is  3.09290681143
For regularization factor  0.8
final training loss is  3.13956399085
For regularization factor  0.9
final training loss is  3.18077256284
For regularization factor  1.0
final training loss is  3.21766041795
For regularization factor  1.1
final training loss is  3.25103746293
For regularization factor  1.2
final training loss is  3.28150509082
For regularization factor  1.3
final training loss is  3.30952221225
For regularization factor  1.4
final training loss is  3.3354470334
For regularization factor  1.5
final training loss is  3.35956454944
For regularization factor  1.6
final training loss is  3.38210524481
```

```
For regularization factor  1.7
final training loss is  3.40325817513
For regularization factor  1.8
final training loss is  3.4231803432
For regularization factor  1.9
final training loss is  3.44200356238
Number of Negative values predicted are  0
loss with year = 2012  and month =  12  is  [ 2.48524151]
```