## 源代码

import csv

import math

import datetime

#user\_habit\_dict:每个用户的乘车记录:起点,终点,距离

user\_habit\_dict={}

#start\_end\_dict:每条记录的起点,终点对

start\_end\_dict={}

#end\_start\_dict:每条记录的起点,终点对

end\_start\_dict={}

#user\_habit\_dict\_test:test中每个用户的记录

user\_habit\_dict\_test={}

#bike\_dict:bike中的记录

bike\_dict={}

#弧度转换

def rad(tude):

    return (math.pi/180.0)\*tude

#geohash模块提取的

\_\_base32 = '0123456789bcdefghjkmnpqrstuvwxyz'

\_\_decodemap = { }

for i in range(len(\_\_base32)):

    \_\_decodemap[\_\_base32[i]] = i

del i

#返回 精确的经纬度和误差

def decode\_exactly(geohash):

    lat\_interval, lon\_interval = (-90.0, 90.0), (-180.0, 180.0)

    lat\_err, lon\_err = 90.0, 180.0

    is\_even = True

    for c in geohash:

        cd = \_\_decodemap[c]

        for mask in [16, 8, 4, 2, 1]:

            if is\_even: # adds longitude info

                lon\_err /= 2

                if cd & mask:

                    lon\_interval = ((lon\_interval[0]+lon\_interval[1])/2, lon\_interval[1])

                else:

                    lon\_interval = (lon\_interval[0], (lon\_interval[0]+lon\_interval[1])/2)

            else:      # adds latitude info

                lat\_err /= 2

                if cd & mask:

                    lat\_interval = ((lat\_interval[0]+lat\_interval[1])/2, lat\_interval[1])

                else:

                    lat\_interval = (lat\_interval[0], (lat\_interval[0]+lat\_interval[1])/2)

            is\_even = not is\_even

    lat = (lat\_interval[0] + lat\_interval[1]) / 2

    lon = (lon\_interval[0] + lon\_interval[1]) / 2

    return lat, lon, lat\_err, lon\_err

#返回 欧式距离  （其实还可以返回南北方向距离,东西方向距离,曼哈顿距离,方向(-0.5:0.5)，但是删了，没啥吊用）

def produceLocationInfo(latitude1, longitude1,latitude2, longitude2):

    radLat1 = rad(latitude1)

    radLat2 = rad(latitude2)

    a = radLat1-radLat2

    b = rad(longitude1)-rad(longitude2)

    R = 6378137

    d = R\*2\*math.asin(math.sqrt(math.pow(math.sin(a/2),2)+math.cos(radLat1)\*math.cos(radLat2)\*math.pow(math.sin(b/2),2)))

    detallat = abs(a)\*R

    detalLon = math.sqrt(d\*\*2-detallat\*\*2)

    if b==0:

        direction = 1/2 if a\*b>0 else -1/2

    else:

        direction = math.atan(detallat/detalLon\*(1 if a\*b>0 else -1))/math.pi

    return round(d)

#返回 欧式距离

def loc\_2\_dis(hotStartLocation,hotEndLocation):

    StartLocation = decode\_exactly(hotStartLocation[:7])

    EndLocation = decode\_exactly(hotEndLocation[:7])

    latitude1 = StartLocation[0]

    longitude1 = StartLocation[1]

    latitude2 = EndLocation[0]

    longitude2 = EndLocation[1]

    return produceLocationInfo(latitude1, longitude1, latitude2, longitude2)

#返回 是否放假,距0点的分钟数,距5月1的天数

def produceTimeInfo(TimeData):

    TimeData = TimeData.split(' ')

    baseData = datetime.datetime(2017, 5, 1, 0, 0, 1)

    mydata = TimeData[0].split('-')

    mytime = TimeData[1].split(':')

    mydata[0] = int(mydata[0])

    mydata[1] = int(mydata[1])

    mydata[2] = int(mydata[2])

    mytime[0] = int(mytime[0])

    mytime[1] = int(mytime[1])

    mytime[2] = int(mytime[2].split('.')[0])

    dt = datetime.datetime(mydata[0], mydata[1], mydata[2], mytime[0], mytime[1], mytime[2])

    minute = mytime[1]+mytime[0]\*60

    # return int((dt-baseData).\_\_str\_\_().split(' ')[0]),miao,dt.weekday(),round(miao/900)

    isHoliday = 0

    if dt.weekday()in [5,6] or int((dt-baseData).\_\_str\_\_().split(' ')[0]) in [29,28]:

        isHoliday=1

    return isHoliday,minute,int((dt-baseData).\_\_str\_\_().split(' ')[0])

#模型之间的融合，粗暴的取了最值，这个可以再提升

def add2result(result1,result2):

    for each in result2:

        if each in result1:

            result1[each] = min(result1[each] ,result2[each] )

        else:

            result1[each] = result2[each]

    return result1

# 其实就是knn算法，结合了leak。一般的knn+leak应该是0.26分。这里主要有两点创新。一是给算出来的距离值除以频度的1.1次方，

# 这个加了很多分，二是对于新用户又使用了一个新的knn，其他算法在处理新用户的时候也可以参考下。

# knn算法产生的特征可以融合进xgb再训练，已实现，但内存不够弃赛

def training(trainfile = 'train.csv',testfile = 'test.csv',subfile = 'submission.csv' ,

             leak1 = 0.01 ,leak2 = 4 ,leak3 = 20,              #leak

             qidianquan = 10,shijianquan = 10,jiejiaquan = 2,bikequan = 0.5,  #都是拼音，字面意思，越大则这个特征比重越大

             zhishu = 1.1  #对结果影响很大

             ):

    tr = csv.DictReader(open(trainfile))

    #利用train.csv建立user\_habit\_dict和start\_end\_dict

    for rec in tr:

        user = rec['userid']

        start = rec['geohashed\_start\_loc']

        end = rec['geohashed\_end\_loc']

        rec['isHoliday'] , rec['minute'] , rec['data'] = produceTimeInfo(rec['starttime'])

        if user in user\_habit\_dict:

            user\_habit\_dict[user].append(rec)

        else:

            user\_habit\_dict[user] = [rec]

        if start in start\_end\_dict:

            start\_end\_dict[start].append(rec)

        else:

            start\_end\_dict[start] = [rec]

        if end in end\_start\_dict:

            end\_start\_dict[end].append(rec)

        else:

            end\_start\_dict[end] = [rec]

    print('train done!')

    # te是测试文件

    te = csv.DictReader(open(testfile))

    for rec in te:

        user = rec['userid']

        bike = rec['bikeid']

        rec['isHoliday'], rec['minute'], rec['data'] = produceTimeInfo(rec['starttime'])

        if user in user\_habit\_dict\_test:

            user\_habit\_dict\_test[user].append(rec)

        else:

            user\_habit\_dict\_test[user] = [rec]

        if bike in bike\_dict:

            bike\_dict[bike].append(rec)

        else:

            bike\_dict[bike] = [rec]

    print("test done!")

    #sub是提交文件

    sub = open(subfile, 'w')

    iter1 = 0

    # AllhotLocSort = sorted(end\_start\_dict.items(), key=lambda d: len(d[1]), reverse=True)

    te1 = csv.DictReader(open(testfile))

    for rec in te1:

        iter1 += 1

        if iter1  % 10000== 0:

            print(iter1/20000,'%',sep='')

        # testTime = timeSlipt(rec['minute'])

        rec['isHoliday'], rec['minute'], rec['data'] = produceTimeInfo(rec['starttime'])

        user1 = rec['userid']

        bikeid1 = rec['bikeid']

        order1 = rec['orderid']

        start1 = rec['geohashed\_start\_loc']

        hour1 = rec['minute']/60

        minute1 = rec['minute']

        isHoliday1 = rec['isHoliday']

        biketype1 = rec['biketype']

        data1 = rec['data']

        result = {}

        hotLoc = {}

        #knn

        if user1 in user\_habit\_dict:

            for eachAct in user\_habit\_dict[user1]:

                start2 = eachAct['geohashed\_start\_loc']

                end2 = eachAct['geohashed\_end\_loc']

                hour2 = eachAct['minute']/60

                isHoliday2 = eachAct['isHoliday']

                biketype2 = eachAct['biketype']

                data2 = rec['data']

                dis = loc\_2\_dis(start1, start2)

                dis = min(dis, 1000)    #1000

                qidian= qidianquan \* (dis / 100) \*\* 2

                detalaTime = abs(hour2 - hour1) if abs(hour2 - hour1) < 12 else 24 - abs(hour2 - hour1)

                shijian= shijianquan \* (detalaTime / 12 \* 10) \*\* 2

                dayType = isHoliday2 - isHoliday1

                jiejia= jiejiaquan \* (dayType \* 10) \*\* 2         #?

                biType = int(biketype2) - int(biketype1)

                bike= bikequan \* (biType \* 10) \*\* 2  #0.5

                #利用终点预测

                # return 欧式距离,南北方向距离,东西方向距离,曼哈顿距离,方向(-0.5:0.5)

                # test2train\_dis = loc\_2\_dis(start1,end2)

                # train2train\_dis = loc\_2\_dis(start2,end2)

                # dis\_detal = min(abs(test2train\_dis[3]-train2train\_dis[3]),1000)  #1000

                # direction\_detal = abs(test2train\_dis[4]-train2train\_dis[4])

                # direction\_detal = direction\_detal if direction\_detal<0.5 else 1-direction\_detal

                # jvli = 4 \* (dis\_detal/100)\*\*2

                # fangxiang = 1 \* (direction\_detal/0.5\*10)\*\*2

                score = qidian+shijian+jiejia+bike              #jvli+fangxiang

                # print(qidian,shijian,jiejia,bike,jvli,fangxiang)

                if end2 in hotLoc:

                    hotLoc[end2] += 1

                else:

                    hotLoc[end2] = 1

                if end2 in result:

                    if result[end2] > score:

                        result[end2] = score

                else:

                    result[end2] = score

            for each in hotLoc:

                result[each] = result[each] / (hotLoc[each]\*\*zhishu)  #0

            for each in result:

                result[each] = math.sqrt(result[each])

        #利用test中的用户历史记录

        if user1 in user\_habit\_dict\_test:

            resulttest = {}

            user\_habit\_dict\_test[user1].sort(key = lambda x:x['data']\*60\*24+x['minute'])

            xuhao = 0

            for i in range(len(user\_habit\_dict\_test[user1])-1):

                if user\_habit\_dict\_test[user1][i]['orderid'] == order1:

                    xuhao = i

                    resulttest[user\_habit\_dict\_test[user1][i+1]['geohashed\_start\_loc']] = 21

            for i in range(len(user\_habit\_dict\_test[user1])):

                if i not in [xuhao,xuhao+1]:

                    resulttest[user\_habit\_dict\_test[user1][i]['geohashed\_start\_loc']] = 21+abs(i-xuhao)

                result = add2result(result, resulttest)

        # leak

        if bikeid1 in bike\_dict:

            resultleak = {}

            bike\_dict[bikeid1].sort(key = lambda x:x['data']\*60\*24+x['minute'])

            for i in range(len(bike\_dict[bikeid1])-1):

                if bike\_dict[bikeid1][i]['orderid'] == order1:

                    zhong = bike\_dict[bikeid1][i+1]['data']\*60\*24+bike\_dict[bikeid1][i+1]['minute']

                    qi = bike\_dict[bikeid1][i]['data']\*60\*24+bike\_dict[bikeid1][i]['minute']

                    detal = zhong-qi

                    if detal<30:

                        resultleak[bike\_dict[bikeid1][i + 1]['geohashed\_start\_loc']] = leak1

                    elif detal<2\*60:

                        resultleak[bike\_dict[bikeid1][i + 1]['geohashed\_start\_loc']] = leak2  #4

                    else:

                        resultleak[bike\_dict[bikeid1][i + 1]['geohashed\_start\_loc']] = leak3   #20

            result = add2result(result,resultleak)

        #起点终点对的knn

        if start1 in start\_end\_dict:

            endDict = {}

            resultqizhong={}

            for eachAct in start\_end\_dict[start1]:

                score = 0

                score += (24-abs(hour1-eachAct['minute']/60))/24

                score += (1-abs(isHoliday1-eachAct['isHoliday']))\*0.4

                if eachAct['geohashed\_end\_loc'] in endDict:

                    endDict[eachAct['geohashed\_end\_loc']] += score

                else:

                    endDict[eachAct['geohashed\_end\_loc']] = score

            hotLoc = sorted(endDict.items(),key = lambda x:x[1],reverse=True)

            if len(hotLoc)>=1:

                resultqizhong[hotLoc[0][0]] = 1000

            if len(hotLoc) >= 2:

                resultqizhong[hotLoc[1][0]] = 1001

            if len(hotLoc) >= 3:

                resultqizhong[hotLoc[2][0]] = 1002

            result = add2result(result, resultqizhong)

        #剔除不合理结果

        for each in result:

            distance = loc\_2\_dis(each,start1)

            if distance > 2500:

                result[each] = 1999

        if start1 in result:

            result[start1] = min(2000, result[start1])

        else:

            result[start1]=2000

        result['fuck2'] = 2001

        result['fuck3'] = 2002

        bestResult = sorted(result.items(), key=lambda d: d[1])

        string = rec['orderid']

        num = 0

        for item in bestResult:

            string += ',' + item[0]

            # string += ':' + str(item[1]) + '\t'

            num += 1

            if num == 3:

                break

        sub.write(string + '\n')

    sub.close()

    print('ok')

if \_\_name\_\_ =="\_\_main\_\_":

    training('train.csv', 'test.csv', 'submission.csv' )