A53271697

Bosi Cheng

1.

businessCount\_train,businessCount\_valid = defaultdict(int),defaultdict(int)

negative\_valid=defaultdict(list)

train,positive\_valid=defaultdict(list),defaultdict(list)

user\_set,item\_set=set(),set()

totalPurchases = 0

for l in readGz("train.json.gz"):

if totalPurchases<100000:

user,business = l['reviewerID'],l['itemID']

user\_set.add(user)

item\_set.add(business)

train[user].append(business)

businessCount\_train[business] += 1

totalPurchases += 1

else:

user,business = l['reviewerID'],l['itemID']

user\_set.add(user)

item\_set.add(business)

positive\_valid[user].append(business)

businessCount\_valid[business] += 1

totalPurchases += 1

valid\_if\_purchase=0

while valid\_if\_purchase<100000:

random\_user=user\_set.pop()

user\_set.add(random\_user)

random\_item=item\_set.pop()

item\_set.add(random\_item)

if random\_item not in train[random\_user] :

if random\_item not in negative\_valid[random\_user]:

negative\_valid[random\_user].append(random\_item)

valid\_if\_purchase+=1

mostPopular = [(businessCount\_train[x], x) for x in businessCount\_train]

mostPopular.sort()

mostPopular.reverse()

return1 = set()

count = 0

for ic, i in mostPopular:

count += ic

return1.add(i)

if count > totalPurchases/2: break

correction=0

for user in positive\_valid.keys():

for item in positive\_valid[user]:

if item in return1:

correction+=1

for user in negative\_valid.keys():

for item in negative\_valid[user]:

if item not in return1:

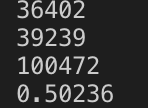
correction+=1

print(len(positive\_valid))

print(len(negative\_valid))

print(correction)

print(float(correction/200000))



2.

No, it is not the best choice. This model has nothing to do with the training set. We should include the non-purchase pair only in the validation sets.

businessCount\_train,businessCount\_valid = defaultdict(int),defaultdict(int)

negative\_valid=defaultdict(list)

train,positive\_valid=defaultdict(list),defaultdict(list)

user\_set,item\_set=set(),set()

train\_user\_set=set()

totalPurchases = 0

valid\_user\_set,valid\_item\_set=set(),set()

for l in readGz("train.json.gz"):

if totalPurchases<100000:

user,business = l['reviewerID'],l['itemID']

if user not in user\_set:

user\_set.add(user)

if business not in item\_set:

item\_set.add(business)

train[user].append(business)

businessCount\_train[business] += 1

totalPurchases += 1

else:

user,business = l['reviewerID'],l['itemID']

if user not in valid\_user\_set:

valid\_user\_set.add(user)

if business not in valid\_item\_set:

valid\_item\_set.add(business)

positive\_valid[user].append(business)

businessCount\_valid[business] += 1

totalPurchases += 1

mostPopular = [(businessCount\_train[x], x) for x in businessCount\_train]

mostPopular.sort()

mostPopular.reverse()

return1 = set()

count = 0

for ic, i in mostPopular:

count += ic

return1.add(i)

if count > totalPurchases/2: break

valid\_if\_purchase=0

while valid\_if\_purchase<100000:

random\_user=valid\_user\_set.pop()

valid\_user\_set.add(random\_user)

random\_item=valid\_item\_set.pop()

valid\_item\_set.add(random\_item)

if random\_item not in train[random\_user] :

if random\_item not in negative\_valid[random\_user]:

negative\_valid[random\_user].append(random\_item)

valid\_if\_purchase+=1

correction=0

for user in positive\_valid.keys():

for item in positive\_valid[user]:

if item in return1:

correction+=1

print(correction)

for user in negative\_valid.keys():

for item in negative\_valid[user]:

if item not in return1:

correction+=1

print(len(positive\_valid))

print(len(negative\_valid))

print(correction)

print(float(correction/200000))



3.

while loading the json, create a dict that matches every item and its categories item\_categories[item].

repeat\_purchase=defaultdict()

for user in user\_set:

temp=set()

for item in purchase[user]:

for category in item\_categories[item]:

if category in temp:

repeat\_purchase[user]=True

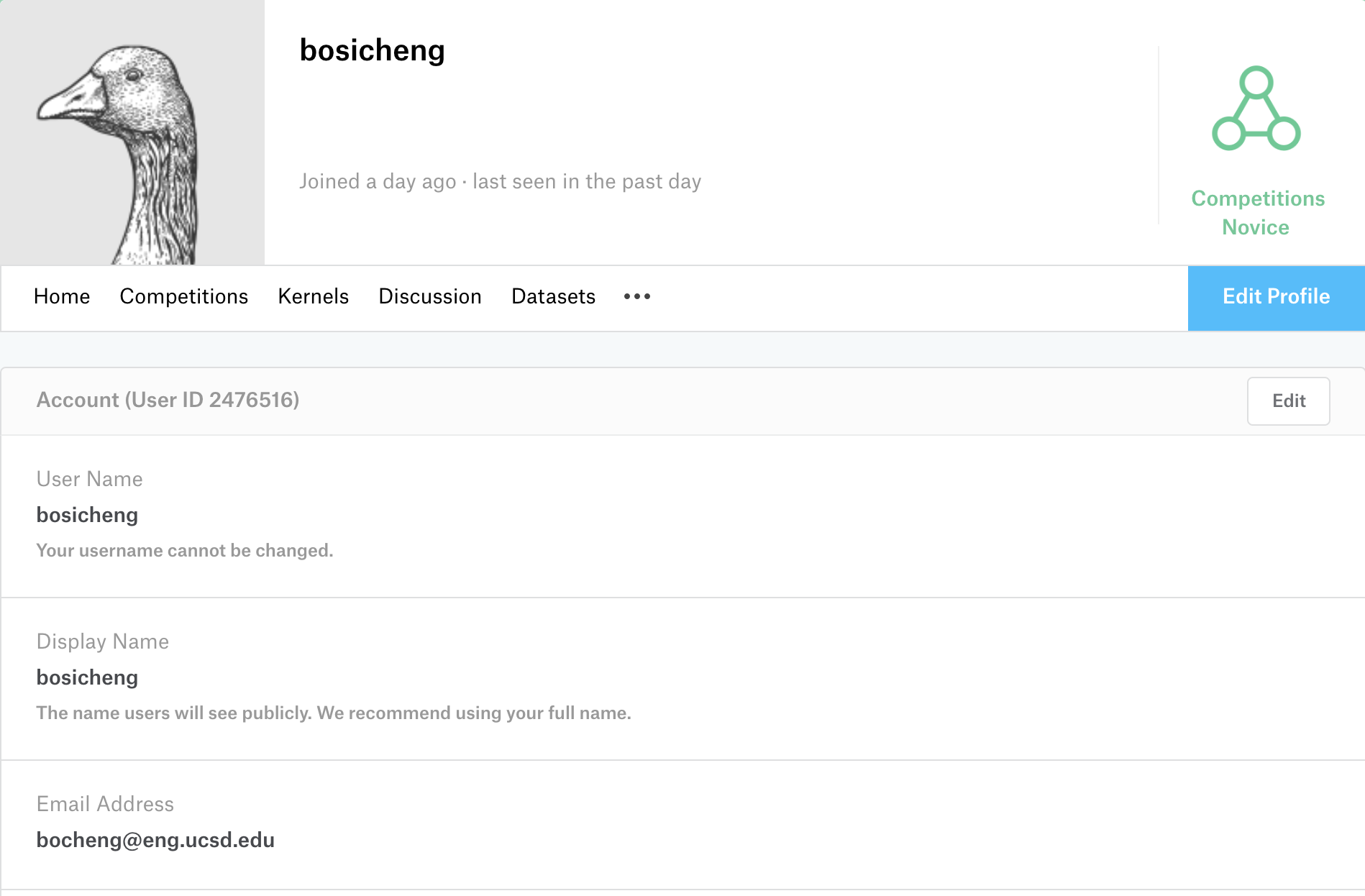
temp.add(category)

if user not in repeat\_purchase.keys():

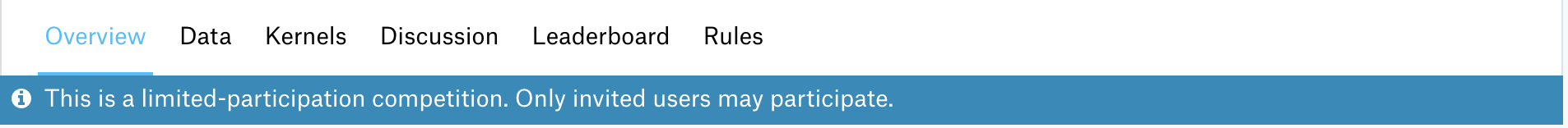
repeat\_purchase[user]=False

4.

My username is bosicheng. But I can't participate because I'm not invited. (why?) I swear it's my UCSD email.



Here's the issue:



5.

allRatings\_train = []

userRatings = defaultdict(list)

totalPurchase=0

sum\_mse=0

for l in readGz("train.json.gz"):

user,business = l['reviewerID'],l['itemID']

if totalPurchase<100000:

allRatings\_train.append(l['rating'])

userRatings[user].append(l['rating'])

totalPurchase+=1

else:

if totalPurchase==100000:

globalAverage = sum(allRatings\_train) / len(allRatings\_train)

sum\_mse+=abs(l['rating']-globalAverage)\*abs(l['rating']-globalAverage)

totalPurchase+=1

globalAverage = sum(allRatings\_train) / len(allRatings\_train)

userAverage = {}

for u in userRatings:

userAverage[u] = sum(userRatings[u]) / len(userRatings[u])

print(globalAverage)

print(sum\_mse/100000)



6.

import gzip

import numpy as np

from collections import defaultdict

#get the data

def readGz(f):

for l in gzip.open(f):

yield eval(l)

allRatings = []

data=[]

userRatings = defaultdict(list)

for l in readGz("train.json.gz"):

user,business = l['reviewerID'],l['itemID']

allRatings.append(l['rating'])

userRatings[user].append(l['rating'])

data.append(l)

#shuffle the data

np.random.shuffle(data)

def getPrediction2(alpha,uB,iB,i,j,y\_u,y\_i,uMap,iMap):

rating = alpha + (uB[i] if i in uB else 0) + (iB[j] if j in iB else 0)

if i in uMap and j in iMap:

rating +=np.inner(y\_u[uMap[i]],y\_i[iMap[j]])

return rating

#Method to Train The Latent Factor Model. This method doesn't use any Machine Learning library.

def trainLFModel(lam,tData,vData,trials):

uTrainDict = defaultdict(lambda: defaultdict(int))

iTrainDict = defaultdict(lambda: defaultdict(int))

uValidDict = defaultdict(lambda: defaultdict(int))

# iValidDict = defaultdict(lambda: defaultdict(int))

uB = defaultdict(float)

iB = defaultdict(float)

uMap = defaultdict(int)

uCount=0

iMap = defaultdict(int)

iCount=0

for i in tData:

user, item, rating = i['reviewerID'], i['itemID'], i['rating']

uTrainDict[user][item] = rating

iTrainDict[item][user] = rating

if user not in uMap:

uMap[user]=uCount

uCount+=1

if item not in iMap:

iMap[item]=iCount

iCount+=1

for i in vData:

user, item, rating = i['reviewerID'], i['itemID'], i['rating']

uValidDict[user][item] = rating

y\_u=np.random.normal(scale=1,size=(len(uTrainDict),1))

y\_i=np.random.normal(scale=1,size=(len(iTrainDict),1))

alpha = 0

totalTrials=trials

for \_ in range(totalTrials):

alpha=0

for i in uTrainDict:

for j in uTrainDict[i]:

alpha += uTrainDict[i][j] - uB[i] -iB[j] - np.inner(y\_u[uMap[i]],y\_i[iMap[j]])

alpha /= len(tData)

print(alpha)

for i in uTrainDict:

uB[i] = 0

for j in uTrainDict[i]:

uB[i] += uTrainDict[i][j] - alpha - iB[j] - np.inner(y\_u[uMap[i]],y\_i[iMap[j]])

uB[i] /= (lam + len(uTrainDict[i]))

for j in iTrainDict:

iB[j] = 0

for i in iTrainDict[j]:

iB[j] += iTrainDict[j][i] -alpha - uB[i] - np.inner(y\_u[uMap[i]],y\_i[iMap[j]])

iB[j] /= (lam + len(iTrainDict[j]))

for i in uTrainDict:

for lf in range(1):

y\_u[uMap[i]][lf] = 0

for j in uTrainDict[i]:

y\_u[uMap[i]][lf] += y\_i[iMap[j]][lf]\*(uTrainDict[i][j] - alpha - iB[j] +y\_i[iMap[j]][lf]\*y\_i[iMap[j]][lf]-np.inner(y\_u[uMap[i]],y\_i[iMap[j]]) )

y\_u[uMap[i]][lf] /= (lam + y\_i[iMap[j]][lf]\*y\_i[iMap[j]][lf])

for j in iTrainDict:

for lf in range(1):

y\_i[iMap[j]][lf] = 0

for i in iTrainDict[j]:

y\_i[iMap[j]][lf] += y\_u[uMap[i]][lf]\*(uTrainDict[i][j] - alpha - uB[i] - np.inner(y\_u[uMap[i]],y\_i[iMap[j]]) +y\_u[uMap[i]][lf]\*y\_u[uMap[i]][lf] )

y\_i[iMap[j]][lf] /= (lam + y\_u[uMap[i]][lf]\*y\_u[uMap[i]][lf])

vMSE = 0

for i in uValidDict:

for j in uValidDict[i]:

# vMSE += ((alpha + (uB[i] if i in uB else 0) + (iB[j] if j in iB else 0) - uValidDict[i][j]) \*\*2)

vMSE += ((getPrediction2(alpha,uB,iB,i,j,y\_u,y\_i,uMap,iMap) - uValidDict[i][j]) \*\*2)

vMSE /= len(vData)

print (vMSE)

return vMSE,alpha,uB,iB,uMap,iMap

print("done")

tData=data[:100000]

vData=data[100000:]

lamdas=[1,2,3,5]

trials=[2]

for i in lamdas:

tempvMSE=1

for t in trials:

tempvMSE,alpha,uB,iB,uMap,iMap=trainLFModel(i,tData,vData,t)

print ("MSE:"+str(tempvMSE))



7.biggest: 'U536579649', 'U910050838', 'I471768594'

smallest: 'I011994385', 'U926666668'

8.

when λ=5:

