# COMP9313 Final

By Chen Wu

Id: z5244467

Q1. HDFS

1. According to Erasure Coding: (6,3)-Reed-Solomon, a matrix should contain 6 raw data and 3 parity data. And the lost data can be recovered by any 6 rows. Thereby, the files to be stored are all divided into cells and stored in the block in the order from left to right and top to bottom. For every 6 cells stored, 3 cells are stored as parties in order to calculate the lost data in the future. Therefore, they belong to the same striped block group.
2. 2. When x => 3 and y = 3, the achieved tolerance is the same as (6,3)-Reed-Solomon. Under this condition, the lost data cannot exceed 3 copies, and the data will not be restored.

Q2. Spark and MapReduce

1. from pyspark import SparkConf, SparkContext

conf = SparkConf().setMaster("local").setAppName("practice\_RDD")

sc = SparkContext(conf = conf)

record = [('z3212321',66),('z3212321',77),('z3212321',77),

('z5672322',74),('z4212331',98),('z4212331',87),

('z4212331',57),('z4212331',62),('z3212431',78),('z3212431',70)]

student\_rdd = sc.parallelize(record)

tup =()

def createCombiner(value):

return (value)

def mergeValue(acc,value):

tup = (max(acc,value))

return tup

def mergeCombiners(acc1,acc2):

return (acc1[0]+acc2[0],acc1[1]+acc2[1])

result = student\_rdd.combineByKey(createCombiner,mergeValue,mergeCombiners)

print(result.collect())

2. Obviously, the position of the grace offset condition in the code is wrong. The offset should be operated after judging the size of cand\_num and beta\_n.

Modify as follows:

def collision\_count(a, b, offset):  
 counter = 0  
 for i in range(len(a)):  
 if abs(a[i]-b[i]) <= offset:  
 counter += 1  
 return counter  
def c2lsh(data\_hashes, query\_hashes, alpha\_m, beta\_n):  
 offset = 0  
 cand\_num = 0  
 while cand\_num < beta\_n :  
 offset += 1  
 candidates = data\_hashes.flatMap(lambda x :  
 [x[0]] if collision\_count(x[1], query\_hashes, offset)>=alpha\_m else [])  
 cand\_num = candidates.count()  
 return candidates

Q3. LSH

Q4. Spark SQL

In this question, I suppose a dataset, which is tup including [(3, "9321",69), (1, "9004",85), (1, "9012",75), (2, "9313",70), (1, "9900",90), (3, "9023",50),(4,"213",71),(4,"321",89)].

import pandas as pd

from pyspark.sql import \*

from pyspark.sql import SQLContext

from pyspark import SparkContext,SparkConf

import pyspark.sql.functions as F

conf = SparkConf().setAppName("abc")

sc = SparkContext(conf=conf)

sqlContext=SQLContext(sc)

tup = [(3, "9321",69), (1, "9004",85), (1, "9012",75), (2, "9313",70), (1, "9900",90), (3, "9023",50),(4,"213",71),(4,"321",89)]

record = sqlContext.createDataFrame(tup, ["Id", "Course","Score"])

record.show(5)

maxmin=record.orderBy('Id').groupBy('Id').agg(F.max('Score').alias('max'),F.min('Score').alias('min'))

maxmin.show(3)

Q5. Stacking

1. According to the question, there are 3 base classifiers, and 1 meta classifier. Thus, we suppose clf1,clf2,clf3 and mcl. Then using stackingCVClassifier as follow:

sclf=StackingCVClassifier(classifiers=[clf1,clf2,clf3],meta\_classifier=mcl,random\_state=RANDOM\_SEED)

Then we do 5-fold cross validation

for clf in zip([clf1, clf2, clf3, sclf]):

scores = model\_selection.cross\_val\_score(clf, X, y, cv=5, scoring='accuracy')

print("Accuracy: %0.2f (+/- %0.2f) [%s]"

% (scores.mean(), scores.std(), label))

Q6 Mining Data Streams

1. S = “hello” h = 7, e = 4, l = 11, o = 14

Thus, hello = 7+4+11+11+14 = 47

map = 12+0+15 = 27

reduce = 17+4+3+20+2+4 = 50

The 8-bit array is initialized as below

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Insert hello with h1(“hello”,”map”,”reduce”) = {7,3,2}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

Insert hello with h2(“hello”) = {5,3,6}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |

1. If s = “spark”, spark = 18+15+0+17+10 = 60

Query “spark” with h1(“spark”) = {4}

Query “spark” with h2(“spark”) = {5}

Thus H(“spark”) = {4,5}.

However, in the table of S, 5 corresponds to 0, so “spark” is not included in S

1. False positive probability =

K= 2, m = 3, n = 8

Thus, = = 0.2784

Q7. Recommender System

1. r1 = [3,5,0,0,2], m1 = (3+5+2)/3 = row1: [,0,0,]

r2 = [0,4,0,1,0], m2 = (4+1)/2 = , row 2: [0,]

r3 = [4,0,5,2,0], m3 = (4+5+2)/3 = , row 3: [,0]

S1,3 = = -0.025717

S2,1 = = 0.583898

S3,2 = = 0.583898

According to bxi = µ + bx +bi

µ = 3+5+2+4+1+5+4+2 = 26, bx = (avg. rating of user x) - µ = 26/8 – 26 =

1. b1 = (avg. rating of movie i) - µ = -26 =

b3 = (avg. rating of movie i) - µ = -26 =

bx3 = 26+（）+（）=, bx1 = 26+（）+（）=,

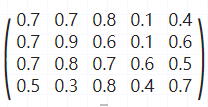
= + bx1= 5-（）+()= = 4.667

2） b2 = (avg. rating of movie i) - µ = -26 =

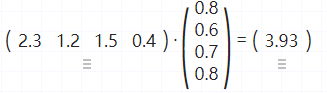
bx2 = 26+（）+（）=,

= + bx2= 2 -（）+()= = = 1.667

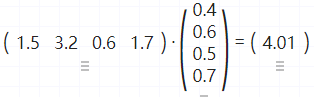
3) = = 5

2. PT = 

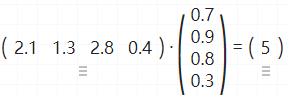
1）



2）



3)



Therefore, using matrix factorization we can get 3.93, 4.01 ,5 respectively.

3. According to RMSE,

using baseline estimator, we can get RMSE1 = sqrt((4.6667-3)2 + (1.667-4)2 + (5.15-5)2 )=2.8713

using matrix factorization， we can get RMSE2 = sqrt((3.93-3)2 + (4.01-4)2 + (5-5)2 )=0.93005

RMSE1 > RMSE2

Therefore, in this question using matrix factorization is better.