Zeppelin

Movie Ratings Data Normalization

FINISHED

Movie Ratings Data Normalization

- User-Movie Rating Matrix
- User-User Correlation Matrix
- Item-Item Correlation Matrix

Took 1 seconds (outdated)

```
%dep
z.reset() // clean up previously added artifact and repository

// add maven repository
z.addRepo("nexus-releases").url("http://oss.sonatype.org/service/local/staging/deploy/mav
z.addRepo("sonatype-nexus-snapshots").url("https://oss.sonatype.org/content/repositories/

z.load("org.nd4j:nd4j-x86:0.4-rc3.8")
z.load("org.nd4j:nd4s_2.11:0.4-rc3.8")
z.load("org.deeplearning4j:deeplearning4j-core:0.4-rc3.8")
z.load("org.deeplearning4j:deeplearning4j-ui:0.4-rc3.8")
```

```
Must be used before SparkInterpreter (%spark) initialized
Hint: put this paragraph before any Spark code and restart Zeppelin/Interpreter
```

```
import org.nd4j.linalg.factory.Nd4j;
import org.nd4s.Implicits._;
import org.nd4j.linalg.api.ndarray.INDArray;
import org.nd4j.linalg.dataset.DataSet;
import org.nd4j.linalg.dataset.api.DataSetPreProcessor;
import org.nd4j.linalg.factory.Nd4j;
import org.nd4j.linalg.lossfunctions.LossFunctions;
```

```
import org.nd4j.linalg.factory.Nd4j
import org.nd4s.Implicits._
import org.nd4j.linalg.api.ndarray.INDArray
import org.nd4j.linalg.dataset.DataSet
import org.nd4j.linalg.dataset.api.DataSetPreProcessor
import org.nd4j.linalg.factory.Nd4j
import org.nd4j.linalg.lossfunctions.LossFunctions
```

```
import java.io.File

import scala.io.Source

import org.apache.log4j.Logger
import org.apache.log4j.Level

import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
```

```
import java.io.File
import scala.io.Source
import org.apache.log4j.Logger
import org.apache.log4j.Level
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.spark.rdd._
```

Load Sample Data

```
val sampleFile  = "/Volumes/EXTRADRIVE/data/netflix_sample.csv"

val ratings = sc.textFile(new File(sampleFile).toString)
    .mapPartitionsWithIndex { (idx, iter) => if (idx == 0) iter.drop(1) else iter }
    .map { line =>
  val fields = line.split(",")
  // format: (Index,UID, Rating, MID)
  // Rating(userId, MID, Rating)
  Array(fields(1).toInt, fields(3).toInt,fields(2).toInt).toNDArray
}
```

```
sampleFile: String = /Volumes/EXTRADRIVE/data/netflix_sample.csv
ratings: org.apache.spark.rdd.RDD[org.nd4j.linalg.api.ndarray.INDArray] = MapPartitionsRD
D[5958] at map at <console>:138
```

Load Data into Spark SQL

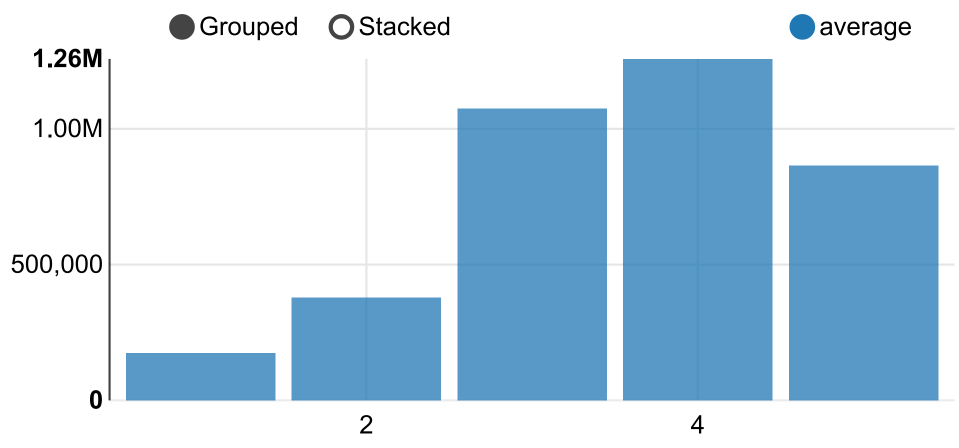
```
case class dataTable(userId:Integer, movieID: Integer, rating: Double)

// split each line, filter out header (starts with "age"), and map it into Bank case clas
val dTable = ratings.map(
    s=>dataTable(s(0).toInt,
                 s(1).toInt,
                 s(2).toInt)
)

// convert to DataFrame and create temporal table
dTable.toDF().registerTempTable("dTable")
```

```
defined class dataTable
dTable: org.apache.spark.rdd.RDD[dataTable] = MapPartitionsRDD[5959] at map at <consol
e>:141
```

Data Exploration

```
%sql
select rating, count(rating) average
from dTable
group by rating
order by rating
```

settings ▾

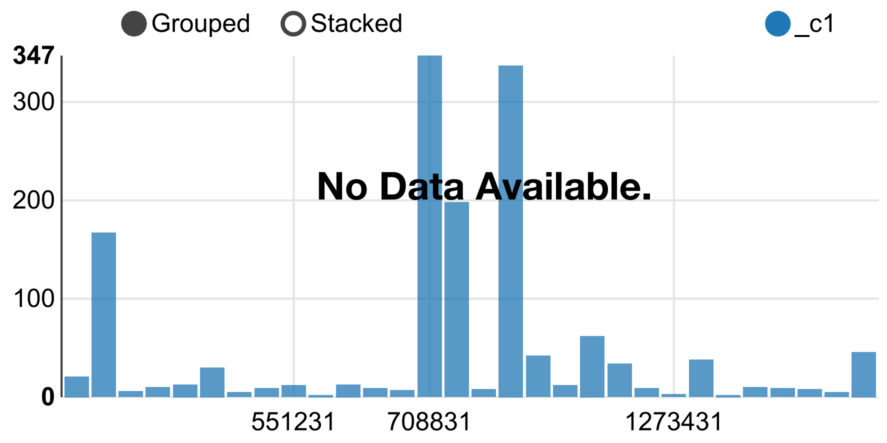


```
%sql
select userID, count(rating)
from dTable
where rating = ${rating =3}
group by userID
limit 30
```

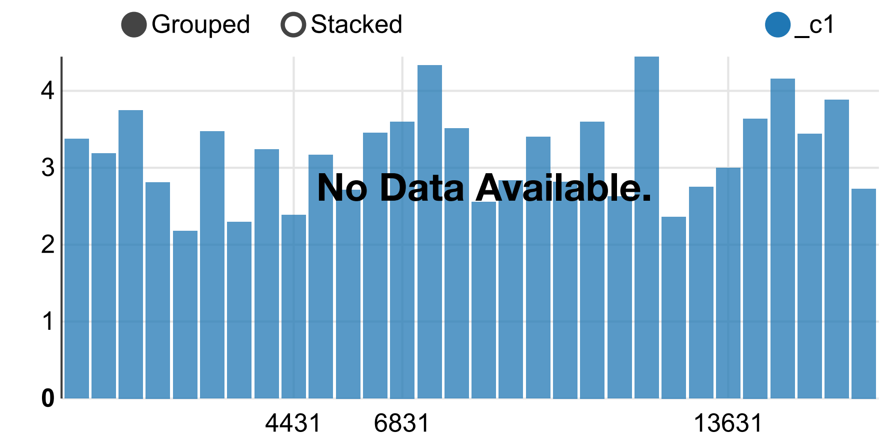
rating

2

settings ▾



```
%sql
select movieID, avg(rating)
from dTable
group by movieID
limit 30
```



Subsample Data due to Memory Constraints

```
ratings.count
// Subsample 0.1% of orignal dataset
```

```
val ratings_short = ratings.sample(false, 0.001, 1)
ratings_short.count
```

```
res66: Long = 3749377
ratings_short: org.apache.spark.rdd.RDD[org.nd4j.linalg.api.ndarray.INDArray] = Partition
wiseSampledRDD[20] at sample at <console>:82
res68: Long = 3860
```

Assign a row number for each user and a column number for each movie

```
val users = ratings_short.map(s => s(0)).distinct.zipWithIndex.toArray.toMap
val movies = ratings_short.map(s => s(1)).distinct.zipWithIndex.toArray.toMap
```

```
warning: there were 1 deprecation warning(s); re-run with -deprecation for details
users: scala.collection.immutable.Map[Double,Long] = Map(1570833.0 -> 1580, 2310565.0 ->
720, 91071.0 -> 326, 589088.0 -> 1065, 699566.0 -> 1184, 2024298.0 -> 1912, 1288299.0 ->
1575, 83720.0 -> 2092, 2274152.0 -> 942, 739277.0 -> 1400, 1468524.0 -> 1706, 1081134.0 -
> 1214, 255283.0 -> 651, 2396004.0 -> 2040, 1339689.0 -> 1795, 759729.0 -> 350, 974375.0
-> 1072, 2250666.0 -> 2064, 565666.0 -> 2323, 2405965.0 -> 602, 1555276.0 -> 2437, 204153
3.0 -> 1863, 85562.0 -> 2713, 763194.0 -> 687, 1089277.0 -> 2779, 2308676.0 -> 792, 20298
35.0 -> 599, 193148.0 -> 2830, 1944228.0 -> 1433, 1036542.0 -> 614, 1812758.0 -> 704, 261
3757.0 -> 845, 1498129.0 -> 2551, 2635599.0 -> 1534, 390751.0 -> 797, 1404157.0 -> 2229,
1486072.0 -> 2507, 1693339.0 -> 2210, 2564120.0 -> 344, 2422528.0 -> 1719, 2108034.0 ...w
arning: there were 1 deprecation warning(s); re-run with -deprecation for details
movies: scala.collection.immutable.Map[Double,Long] = Map(15874.0 -> 1814, 3021.0 -> 108
8, 5686.0 -> 243, 14852.0 -> 1143, 9131.0 -> 1056, 13052.0 -> 886, 6085.0 -> 307, 2452.0
-> 312, 809.0 -> 478, 3962.0 -> 1990, 7427.0 -> 1125, 629.0 -> 230, 13955.0 -> 1119, 261
2.0 -> 1615, 4450.0 -> 74, 14367.0 -> 245, 9886.0 -> 1627, 10785.0 -> 494, 13402.0 -> 193
5, 15009.0 -> 550, 13854.0 -> 89, 7691.0 -> 1001, 11192.0 -> 970, 2072.0 -> 1642, 4262.0
-> 740, 3798.0 -> 731, 12526.0 -> 1870, 10577.0 -> 1229, 12570.0 -> 563, 15440.0 -> 765,
12975.0 -> 619, 14574.0 -> 607, 11922.0 -> 475, 16604.0 -> 129, 14890.0 -> 1751, 7922.0 -
> 1141, 15472.0 -> 1684, 5320.0 -> 303, 11313.0 -> 582, 13058.0 -> 920, 10730.0 -> 1198,
11164.0 -> 2048, 8507.0 -> 906, 9685.0 -> 1428, 12440.0 -> 1273, 11752.0 -> 1886, 259
5....
```

```
movies.get(3021.0)
```

```
res76: Option[Long] = Some(1088)
```

Count Numbers of Ratings, Users and Movies

```
val numRatings = ratings_short.count.toInt
val numUsers = ratings_short.map(s => s(0)).distinct.count.toInt
val numMovies = ratings_short.map(s => s(1)).distinct.count.toInt
```

```
println("Got " + numRatings + " ratings from "
  + numUsers + " users on " + numMovies + " movies.")
```

```
numRatings: Int = 3860
numUsers: Int = 2949
numMovies: Int = 2065
Got 3860 ratings from 2949 users on 2065 movies.
```

Split Table in 3 Arrays; Ratings, MovieIds and UserIds

```
val ratingsarr = ratings_short.map(s => s(2)).toArray.toNDArray
val moviesids = ratings_short.map(s => s(1)).toArray
val userids = ratings_short.map(s => s(0)).toArray
```

```
warning: there were 1 deprecation warning(s); re-run with -deprecation for details
ratingsarr: org.nd4j.linalg.api.ndarray.INDArray = [ 4.00, 2.00, 4.00, 3.00, 3.00, 5.00,
4.00, 3.00, 5.00, 4.00, 5.00, 3.00, 3.00, 3.00, 3.00, 3.00, 3.00, 3.00, 5.00, 3.00, 4.00,
4.00, 2.00, 4.00, 4.00, 3.00, 3.00, 3.00, 3.00, 4.00, 5.00, 3.00, 1.00, 4.00, 3.00, 5.00,
3.00, 3.00, 1.00, 4.00, 4.00, 4.00, 2.00, 3.00, 4.00, 5.00, 4.00, 2.00, 3.00, 3.00, 1.00,
5.00, 4.00, 3.00, 4.00, 2.00, 3.00, 1.00, 3.00, 5.00, 1.00, 3.00, 3.00, 5.00, 4.00, 3.00,
2.00, 3.00, 4.00, 3.00, 4.00, 5.00, 3.00, 2.00, 3.00, 1.00, 5.00, 4.00, 4.00, 4.00, 3.00,
3.00, 3.00, 5.00, 3.00, 4.00, 5.00, 3.00, 5.00, 4.00, 5.00, 3.00, 2.00, 2.00, 2.00, 3.00,
3.00, 3.00, 3.00, 5.00, 2.00, 5.00, 1.00, 4.00, 5.00, 4.00, 5.00, 4.00, 1.00, 2.00, 5.00,
4.00, 2.00, 4.00, 1.00, 4.00, 5.00, 2.00, 4.00, 4.00, 5.00, 5.00, 2.00, 5.00, ...warning:
there were 1 deprecation warning(s); re-run with -deprecation for details
moviesids: Array[Double] = Array(3593.0, 7879.0, 7879.0, 7879.0, 3551.0, 12904.0, 1290
4.0, 17324.0, 17324.0, 17324.0, 17324.0, 8524.0, 11867.0, 3256.0, 3256.0, 3256.0, 3256.0,
3256.0, 3256.0, 6702.0, 4640.0, 4640.0, 4640.0, 4640.0, 4640.0, 4640.0, 8744.0, 8744.0, 8
744.0, 15375.0, 5071.0, 5071.0, 4141.0, 4141.0, 5345.0, 5345.0, 5345.0, 5345.0, 3009.0, 4
216.0, 4216.0, 17730.0, 689.0, 8851.0, 8851.0, 269.0, 16580.0, 1012.0, 13378.0, 3796.0, 8
384.0, 3905.0, 13216.0, 13216.0, 7826.0, 7826.0, 12739.0, 4130.0, 2342.0, 2342.0, 1680
5.0, 6068.0, 6068.0, 16175.0, 16359.0, 5897.0, 5897.0, 4683.0, 12694.0, 12694.0, 15529.0,
10372.0, 10372.0, 5360.0, 5360.0, 2340.0, 14618.0, 7331.0, 7331.0, 7331.0, 7331.0, 241.0,
13493.0, 14312.0, 14312.0, 14312.0, 1409.0, 14725.0, 14725.0, 14725.0, 14725.0, 1472
5.0,...warning: there were 1 deprecation warning(s); re-run with -deprecation for details
userids: Array[Double] = Array(784019.0, 111615.0, 1333442.0, 2616365.0, 2386328.0, 22741
52.0, 552590.0, 2548244.0, 303028.0, 1765266.0, 2382987.0, 1766574.0, 1563935.0, 173560
5.0, 1601196.0, 1516418.0, 337915.0, 1716946.0, 392113.0, 950677.0, 1570833.0, 2239261.0,
110493.0, 1286681.0, 744433.0, 1924750.0, 1847565.0, 2410143.0, 2090242.0, 467092.0, 4659
7.0, 1692798.0, 424883.0, 919670.0, 1334430.0, 626328.0, 2270680.0, 755506.0, 230112.0, 1
83908.0, 1718160.0, 143011.0, 974375.0, 403769.0, 354138.0, 593862.0, 1995832.0, 68269
7.0, 1946049.0, 1246351.0, 305580.0, 40845.0, 1692512.0, 2142230.0, 588771.0, 852514.0, 1
630361.0, 1563935.0, 934472.0, 1178182.0, 2208411.0, 1582107.0, 362038.0, 472831.0, 29113
5.0, 581309.0, 2211057.0, 2512816.0, 1819271.0, 2193332.0, 1439261.0, 716649.0, 73877
0.0,...
```

Map each IDs into Corresponding Row and Column Number

```
val moviemaparr = moviesids.map(s => movies.getOrElse(s,0))
val usermaparr = userids.map(s => users.getOrElse(s,0))
```

```
moviemaparr: Array[AnyVal] = Array(1452, 483, 483, 483, 1744, 1934, 1934, 518, 518, 518,
518, 1610, 1122, 589, 589, 589, 589, 589, 589, 1572, 1037, 1037, 1037, 1037, 1037, 1037,
905, 905, 905, 358, 132, 132, 533, 533, 801, 801, 801, 801, 730, 203, 203, 982, 11, 1365,
1365, 634, 406, 1183, 1551, 606, 702, 1716, 2057, 2057, 1087, 1087, 1441, 991, 1413, 141
3, 126, 1032, 1032, 1184, 235, 1344, 1344, 2019, 1616, 1616, 1950, 2041, 2041, 1225, 122
5, 1182, 1346, 1216, 1216, 1216, 1216, 1908, 1260, 711, 711, 711, 1972, 1992, 1992, 1992,
1992, 1992, 717, 243, 243, 1750, 966, 966, 1745, 1745, 1745, 1745, 1745, 434, 617, 328, 1
188, 1188, 1188, 1188, 1991, 878, 1468, 781, 1308, 827, 628, 661, 661, 661, 661, 166, 41
9, 731, 2018, 2018, 345, 1398, 1533, 103, 953, 953, 953, 489, 658, 770, 1445, 804, 19
0...usermaparr: Array[AnyVal] = Array(1144, 782, 41, 2849, 289, 942, 1443, 226, 109, 175
3, 1448, 1873, 483, 1737, 1367, 221, 548, 2115, 35, 2095, 1580, 464, 715, 849, 2136, 233
7, 1485, 1023, 2074, 575, 1030, 1022, 1640, 1526, 1111, 730, 1920, 2403, 1757, 2101, 196
2, 980, 1072, 1589, 2237, 363, 2900, 1135, 1562, 2874, 2554, 2510, 904, 932, 184, 2580, 4
20, 483, 1983, 2755, 1460, 1625, 287, 596, 2745, 1050, 1926, 1510, 520, 601, 971, 359, 12
81, 697, 1434, 460, 2625, 2541, 141, 2631, 1208, 1362, 2023, 506, 1070, 345, 850, 372, 67
6, 1725, 2877, 1857, 291, 2347, 1649, 1022, 919, 207, 2399, 2469, 2108, 2322, 307, 739, 1
774, 721, 308, 833, 2275, 811, 2689, 2652, 2354, 441, 719, 906, 2826, 1922, 1437, 1757, 1
981, 2024, 2603, 2327, 1912, 364, 1478, 122, 1893, 525, 1863, 2615, 1080, 580, 2489, 152
8, 1...
```

Compute Global Rating Mean and Standard Deviation

```
val meanrating = org.nd4j.linalg.factory.Nd4j.mean(ratingsarr)(0)
val stdrating = org.nd4j.linalg.factory.Nd4j.std(ratingsarr)(0)
```

```
meanrating: Double = 3.591709852218628
stdrating: Double = 1.0880539417266846
```

Normalize Ratings

```
val adjratingarr = (ratingsarr - meanrating) / stdrating
```

```
adjratingarr: org.nd4j.linalg.api.ndarray.INDArray = [ 0.38, -1.46, 0.38, -0.54, -0.54,
1.29, 0.38, -0.54, 1.29, 0.38, 1.29, -0.54, -0.54, -0.54, -0.54, -0.54, -0.54, -0.54, 1.2
9, -0.54, 0.38, 0.38, -1.46, 0.38, 0.38, -0.54, -0.54, -0.54, -0.54, 0.38, 1.29, -0.54,
-2.38, 0.38, -0.54, 1.29, -0.54, -0.54, -2.38, 0.38, 0.38, 0.38, -1.46, -0.54, 0.38, 1.2
9, 0.38, -1.46, -0.54, -0.54, -2.38, 1.29, 0.38, -0.54, 0.38, -1.46, -0.54, -2.38, -0.54,
1.29, -2.38, -0.54, -0.54, 1.29, 0.38, -0.54, -1.46, -0.54, 0.38, -0.54, 0.38, 1.29, -0.5
4, -1.46, -0.54, -2.38, 1.29, 0.38, 0.38, 0.38, -0.54, -0.54, -0.54, 1.29, -0.54, 0.38,
1.29, -0.54, 1.29, 0.38, 1.29, -0.54, -1.46, -1.46, -1.46, -0.54, -0.54, -0.54, -0.54,
1.29, -1.46, 1.29, -2.38, 0.38, 1.29, 0.38, 1.29, 0.38, -2.38, -1.46, 1.29, 0.38, -1.46,
0...
```

Initialize Empty User-Movie Matrix

```
val arr = Array(0).toNDArray
val V = arr.reshape(numUsers,numMovies)
V.getClass
```

```
arr: org.nd4j.linalg.api.ndarray.INDArray = 0.00
V: org.nd4j.linalg.api.ndarray.INDArray =
[[0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0....res19: Clas
s[_ <: org.nd4j.linalg.api.ndarray.INDArray] = class org.nd4j.linalg.cpu.NDArray
```

Insert Ratings into Matrix

```
for(i <- 0 until  numRatings){
    val row = usermaparr(i).asInstanceOf[Number].intValue()
    val column = moviemaparr(i).asInstanceOf[Number].intValue()
    //V.putScalar(row,column,adjratingarr(i))
    V(row,column) = adjratingarr(i)
}
```

```
val row = usermaparr(1).asInstanceOf[Number].intValue()
val column = moviemaparr(1).asInstanceOf[Number].intValue()
V(row,column)
```

```
row: Int = 782
column: Int = 483
res28: Double = -1.4628961086273193
```

Compute User-User Correlation Matrix

```
val UU = V dot V.transpose
```

```
UU: org.nd4j.linalg.api.ndarray.INDArray =
[[0.30,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0...
```

Compute Item-Item Correlation Matrix

```
val MM = V.transpose dot V
```

```
MM: org.nd4j.linalg.api.ndarray.INDArray =
[[0.14,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0...
```

Shape of Correlatoin Matrices

```
UU.shape
MM.shape
```

```
res34: Array[Int] = Array(2949, 2949)
res35: Array[Int] = Array(2065, 2065)
```