Recommendation Engine

Venkateswarlu A

28 Dec 2017

This `Recommendation Engine` uses `User-Based collaborative filtering` to create similarity matrix between users and recommends movies to customers. This recommendation engine creates `Centered Cosine similarity` matrix of users. Based on the high similarity, it recommends movies to similar user.

Packages required for this project

```
library(dplyr)
library(tidyr)
library(ggplot2)
library(arules)
library(caret)
library(animation)
library(lsa) #Latent semantic analysis (for cosine function)
```

```
## 'data.frame':    100004 obs. of  4 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : int  31 1029 1061 1129 1172 1263 1287 1293 1339 1343 ...
##  $ rating   : num  2.5 3 3 2 4 2 2 2 3.5 2 ...
##  $ timestamp: int  1260759144 1260759179 1260759182 1260759185 1260759205 1260759151 1260759187 1260
```

`User-Based` *Collaborative Filtering*

In this collaborative filtering, users are arranged in rows and movies are arranged in columns and cells contains the ratings given by user to a particular movie. After transforming the dataset into required format, row means are calculated for every row. This row mean is nothing but an average rating given by user. Next, every value in the row is substracted by row mean to normalize the ratings and transformed the dataframe as matrix. Used `cosine` function to get similarity matrix.

```
ubcfm <- spread(ds, key = movieId, value = rating) #Arranging rows as users, columns as movies and rati
str(ubcfm)
rownames(ubcfm) <- ubcfm[,1] #Changing Row names with user id's
ubcfm <- ubcfm[-1] #We have transformed row names so the first column is no longer needed

rmeans <- rowMeans(ubcfm,na.rm = TRUE) #To calculate centered cosine, calculate row means
uccs <- sweep(ubcfm, 1,rmeans,"-") #Created centered cosine by substracting every row value with respec
uccs[is.na(uccs)] <- 0 #Transform NA's as 0

muccs <- t(as.matrix(uccs)) #Transforming data frame as matrix, cosine will accept vectors or matrix

sim <- cosine(muccs) #Calculating similarity matrix
sim <- round(sim,3) #Similarity values are having longer decimal numbers, so rouding to 3 decimal point
```

Once created the similarity matrix, this is the time for predict the recommends. To predict the recommends, identified all the users, who are similar to a customer, to whom we are going to recommend movies. Then again filtered the users, who have given the rating to the movie. Multiplied their rating with the weight, which is nothing but cosine similarity value divided by summation of all the cosine similarities. So, to predict recommendation, I have created a custome funciton, which will take user Id , to which we want to predict as an argument and it will return all the movies which will be liked by the user.

Let's say recommend movies for user Id `15`. The function will return rating along with movie id. Whichever movies' rating would be high those movies will be recommended to a user.

```r
recommend_movies(15)
```

```
##     movie    rating
## 1      49 0.3233797
## 2      53 0.3143035
## 3      55 0.3087920
## 4      54 0.3082328
## 5      59 0.3054951
## 6      57 0.3049106
## 7      60 0.2985963
## 8       3 0.2970352
## 9      58 0.2965219
## 10     61 0.2919994
## 11     63 0.2848537
## 12     66 0.2822154
## 13     64 0.2795960
## 14     65 0.2782788
## 15     68 0.2772801
## 16     69 0.2725936
## 17     71 0.2679872
## 18     46 0.2625780
## 19     72 0.2624421
## 20     73 0.2587493
## 21    305 0.2570957
## 22    285 0.2559362
## 23    309 0.2557972
## 24    303 0.2555146
## 25    299 0.2551967
## 26    289 0.2551414
## 27    287 0.2547878
## 28    312 0.2547514
## 29    304 0.2546979
## 30     74 0.2545711
## 31     48 0.2544712
## 32    301 0.2544094
## 33    295 0.2543448
## 34    290 0.2539254
## 35    313 0.2537363
## 36    302 0.2531000
## 37     77 0.2527886
## 38    291 0.2526354
## 39     76 0.2521997
## 40    294 0.2516253
## 41    103 0.2511541
## 42     78 0.2482173
## 43    102 0.2481250
## 44    105 0.2477053
## 45    144 0.2450682
## 46    108 0.2448429
```

```
## 47     269 0.2441992
## 48     141 0.2436515
## 49      79 0.2434005
## 50     114 0.2432589
## 51     277 0.2431545
## 52     283 0.2431058
## 53     270 0.2427711
## 54     146 0.2427694
## 55     131 0.2426055
## 56     276 0.2422927
## 57     278 0.2419393
## 58     271 0.2417730
## 59     113 0.2415277
## 60      80 0.2415023
## 61     272 0.2413633
## 62     130 0.2411437
## 63     279 0.2409007
## 64     275 0.2408420
## 65     148 0.2404651
## 66     156 0.2403439
## 67     147 0.2403164
## 68     132 0.2402037
## 69     267 0.2401233
## 70     273 0.2400798
## 71     116 0.2400652
## 72     268 0.2396608
## 73     280 0.2396299
## 74     152 0.2393743
## 75     274 0.2391871
## 76     155 0.2388495
## 77     281 0.2388306
## 78     282 0.2386700
## 79     151 0.2385659
## 80     158 0.2384307
## 81      81 0.2383264
## 82     135 0.2381329
## 83     140 0.2380096
## 84     154 0.2370699
## 85     137 0.2369587
## 86     117 0.2369515
## 87      83 0.2369216
## 88     159 0.2363476
## 89      88 0.2362679
## 90      84 0.2345779
## 91     166 0.2342573
## 92     118 0.2339225
## 93     167 0.2329076
## 94      89 0.2323543
## 95      99 0.2323014
## 96      98 0.2322355
## 97      85 0.2320059
## 98      97 0.2316532
## 99     119 0.2316202
## 100    168 0.2307775
```

```
## 101    87 0.2304204
## 102   174 0.2301601
## 103   171 0.2294075
## 104   169 0.2293467
## 105    92 0.2291650
## 106   100 0.2290357
## 107    86 0.2288065
## 108   121 0.2286940
## 109   173 0.2281630
## 110   177 0.2281366
## 111    93 0.2273940
## 112   124 0.2269489
## 113   122 0.2267963
## 114   178 0.2260436
## 115   129 0.2250026
## 116   264 0.2247213
## 117   179 0.2244058
## 118   126 0.2243601
## 119    96 0.2243372
## 120   181 0.2238666
## 121   220 0.2236049
## 122   266 0.2235292
## 123   222 0.2234335
## 124   209 0.2233690
## 125   206 0.2231288
## 126   183 0.2228103
## 127   184 0.2222846
## 128   205 0.2222606
## 129   239 0.2221453
## 130   207 0.2219793
## 131   224 0.2218509
## 132   211 0.2217801
## 133   219 0.2215763
## 134   240 0.2211782
## 135   204 0.2210719
## 136   186 0.2207200
## 137   227 0.2204506
## 138   213 0.2204158
## 139   238 0.2201891
## 140   241 0.2198521
## 141   234 0.2196688
## 142   228 0.2190834
## 143   191 0.2190668
## 144   187 0.2190526
## 145   229 0.2190311
## 146   217 0.2188464
## 147   242 0.2186400
## 148   190 0.2182348
## 149   236 0.2181860
## 150   188 0.2178801
## 151   243 0.2173932
## 152   194 0.2173922
## 153   218 0.2172776
## 154   245 0.2172646
```

```
## 155   189 0.2169629
## 156   263 0.2169390
## 157   199 0.2164366
## 158   244 0.2161598
## 159   248 0.2160130
## 160   195 0.2156697
## 161   200 0.2153890
## 162   249 0.2152239
## 163   259 0.2149187
## 164   250 0.2147702
## 165   261 0.2147044
## 166   201 0.2136885
## 167   251 0.2136753
## 168   203 0.2136649
## 169   255 0.2136526
## 170   262 0.2135114
## 171   254 0.2133724
## 172   202 0.2131239
## 173   256 0.2123903
## 174   257 0.2110767
## 175   258 0.2100390
## 176    38 0.2047753
## 177    40 0.2031172
## 178     7 0.2000598
## 179    43 0.1993190
## 180    41 0.1971873
## 181    45 0.1941931
## 182    42 0.1929371
## 183     4 0.1782083
## 184    35 0.1753719
## 185    37 0.1676564
## 186     8 0.1532168
## 187    31 0.1371943
## 188     9 0.1371839
## 189    29 0.1314241
## 190    12 0.1302511
## 191    30 0.1289831
## 192    24 0.1244464
## 193    13 0.1187659
## 194    18 0.1183783
## 195    26 0.1177211
## 196    23 0.1160112
## 197    28 0.1137318
## 198    20 0.1134655
## 199    27 0.1117200
## 200    15 0.1109557
```