NETFLIX

Movie Recommendation System

Faichali (16)
Sanjeev (41)
Shoorvir(44)
Subhra(48)

Problem Set Details

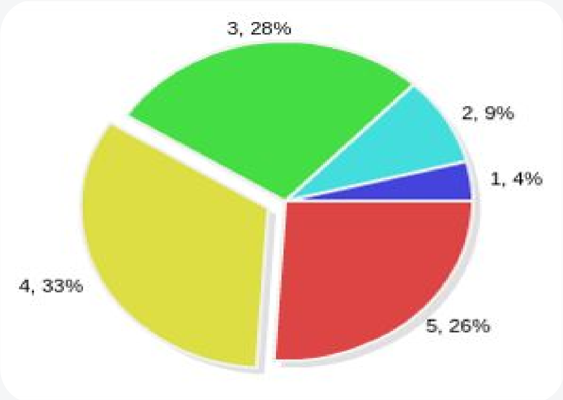
- **4,80,189** users .
- **17770** movies
- **100** million ratings. (OCT 1998 – DEC 2005)

Attributes Information

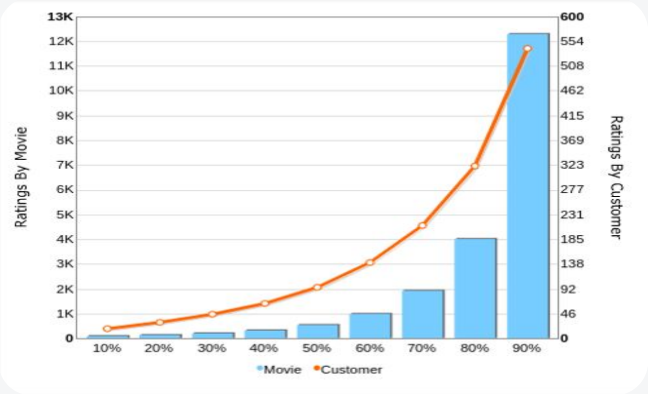
- Movie id
- Customer ID
- Rating
- Title
- Year of Release
- Date
- Netflix ID

Data Analysis

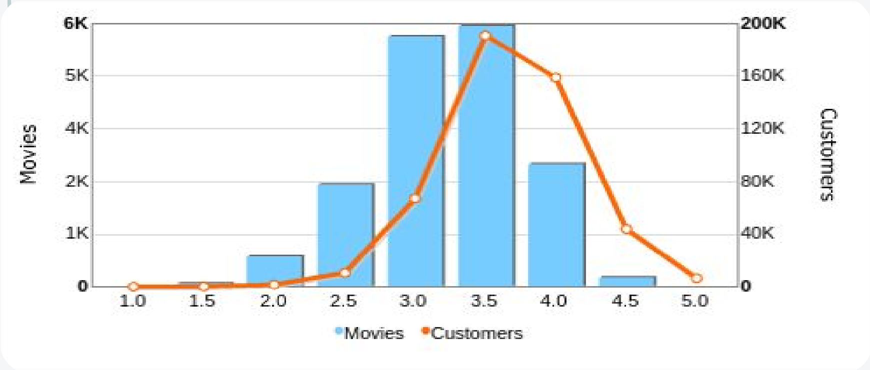
Overall rating distribution



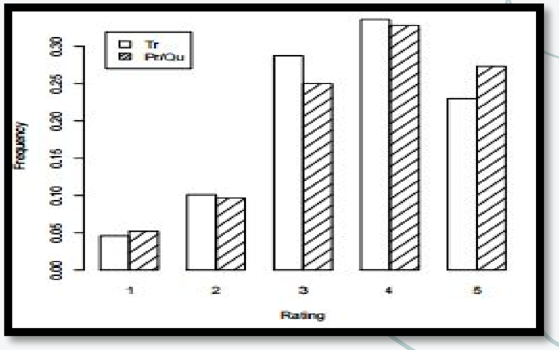
Average rating distribution



Rating count distribution



Training set vs Probe Set



Most popular movies in the dataset

Rank	Title	Average rating	Count
1	Miss Congeniality	3.359	227715
2	Independence Day	3.724	216233
3	The Patriot	3.783	200490
4	The Day After Tomorrow	3.443	194695
5	Pirates of the Caribbean: The Curse of the Black Pearl	4.153	188849
6	Pretty Woman	3.901	190320
7	Forrest Gump	4.299	180736
8	The Green Mile	4.307	180883
9	Con Air	3.454	177825
10	Twister	3.412	177212
	average of top 10	3.783	193495.8
	average overall	3.603	209.3

Most Frequently Used Methods

1. **Collaborative Filtering**

   a) User - User based Approach - find a set of users similar to users who rated the same movie, and again take the mean of their ratings

   b) Item – Item Based Approach - find a set of items similar to users who have also rated, and take the (weighted) mean of ratings on them

Calculating Similarity using Pearson Correlation, Cosine Similarity, Jaccard Index

2. **Matrix Factorization:**

   Tried to use the SVD method for the matrix factorization but even for 1000 customers got memory error.

Data Extraction

1. Implemented Using the python
2. If the dataset is represented in the form of matrix(#users x # movies), then the matrix is highly sparse.

   i.e % Sparsity = 1- (Total No. of Ratings)/(Total no possible entries in the matrix)

   = 1 - 100480507/(17770*480189) = 0.98823 ~ 98.82 %
3. Pseudo Code:

   1. Create list for the movieratings, movieids, userids, dates
   2. For each training Set file

   update the movie rating, movieids, userids

   convert the dates to epoch time for particular index.

   Mapped the userid's to continuous ids from 0 to 480189
4. Storing of data using list data-structure of all the training data makes csv of more than 2GB.
5. Instead of Matrix method , this structure makes the store training data scalable.
6. Using list data structure we can reproduce the matrix if desired.

Data Extraction

```
data = matrix(data=0,nrow=5000,ncol = 17770)
for (i in 1:17770){
  if(i <10){
    fname=paste("mv_000000",i,".txt",sep="")
  }
  else if(i <100){
    fname=paste("mv_00000",i,".txt",sep="")
  }
  else if(i <1000){
    fname=paste("mv_0000",i,".txt",sep="")
  }
  else if(i <10000){
    fname=paste("mv_000",i,".txt",sep="")
  }
  else{
    fname=paste("mv_00",i,".txt",sep="")
  }

  fl = file(fname,open="r")
  linn = readLines(fl)
  for(j in 1:length(linn)){
    if(j == 1){
      mv_id = as.numeric(gsub(':','',linn[1]))
    }
    else{
      list = strsplit(linn[j],",",fixed = TRUE)
      rat_vec = c(do.call("cbind",list))
      uid = as.numeric(rat_vec[1])
      if(uid > 600000 & uid <= 605000){
        temp_uid = uid - 600000
        rating = as.numeric(rat_vec[2])
        data[temp_uid,mv_id] = rating
      }
    }
  }
}
```

Basic Ideology For Recommendation

Three Basic Steps :

1. Calculating the Similarity Matrix or Related Correlation Matrix

2. Finding out the Weighted Matrix using the Similarity Matrix

3. Movie Recommendation

Pearson Correlation

Pearson Correlation is the correlation between sets of data giving the measure of how well they are related.

The formula for Pearson correlation coefficient is:

$$\frac{\sum (r_{ui} - \mu_i)(r_{uj} - \mu_j)}{\sqrt{\sum (r_{ui} - \mu_i)^2}\sqrt{\sum (r_{uj} - \mu_j)^2}}$$

```
> x2 = cor(t(data11))
> x2
             [,1]        [,2]         [,3]         [,4]        [,5]        [,6]        [,7]
 [1,]  1.00000000 -0.22438960  0.609077280 -0.502832105 -0.09171709 -0.26004016 -0.09512255
 [2,] -0.22438960  1.00000000 -0.226445957  0.408600583 -0.13287057 -0.07452088  0.11558417
 [3,]  0.60907728 -0.22644596  1.000000000  0.006812027 -0.28099374 -0.17875053 -0.37572314
 [4,] -0.50283210  0.40860058  0.006812027  1.000000000  0.14069649  0.48327701 -0.43870359
 [5,] -0.09171709 -0.13287057 -0.280993741  0.140696485  1.00000000 -0.08330399  0.26531584
 [6,] -0.26004016 -0.07452088 -0.178750526  0.483277006 -0.08330399  1.00000000 -0.44950040
 [7,] -0.09512255  0.11558417 -0.375723140 -0.438703589  0.26531584 -0.44950040  1.00000000
 [8,]  0.19364533  0.27772870 -0.209320343 -0.583312236 -0.16320735 -0.49356340  0.71028470
 [9,] -0.35639569  0.16712157 -0.292685296 -0.031390657  0.04893342  0.09111119  0.09168322
[10,] -0.32738810 -0.12230216 -0.162758031  0.235464743 -0.53812580  0.50595757 -0.49866315
```

```
> final = which(rp == tmp4)
> final
[1] 6
> movie = tmp2[1,final]
> movie
movie11
```

Cosine Similarity

Cosine similarity is a measure of **similarity** between two vectors of an inner product space that measures the **cosine** of the angle between them.

The formula for calculating cosine similarity is :

$$\frac{\vec{r_i} \cdot \vec{r_j}}{\|\vec{r_i}\|_2 \|\vec{r_j}\|_2} = \frac{\sum r_{ui} r_{uj}}{\sqrt{\sum r_{ui}^2} \sqrt{\sum r_{uj}^2}}$$

--- LSA Package
--- Cosine Similarity Function
--- cosine(data)

```
> x=cosine(as.matrix(t(data11)))
> x
           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9]
 [1,] 1.0000000 0.4922832 0.8216435 0.3223795 0.4593025 0.3709539 0.4626917 0.6019275 0.3735054
 [2,] 0.4922832 1.0000000 0.5005315 0.7608306 0.4911048 0.5116912 0.6044431 0.6726035 0.6541698
 [3,] 0.8216435 0.5005315 1.0000000 0.5592213 0.3746504 0.4193010 0.3348987 0.4120817 0.4120686
 [4,] 0.3223795 0.7608306 0.5592213 1.0000000 0.5835253 0.7454097 0.3118379 0.2391644 0.5361627
 [5,] 0.4593025 0.4911048 0.3746504 0.5835253 1.0000000 0.4260103 0.6169153 0.3907637 0.5298114
 [6,] 0.3709539 0.5116912 0.4193010 0.7454097 0.4260103 1.0000000 0.2383544 0.2117021 0.5465951
 [7,] 0.4626917 0.6044431 0.3348987 0.3118379 0.6169153 0.2383544 1.0000000 0.8495299 0.5551551
 [8,] 0.6019275 0.6726035 0.4120817 0.2391644 0.3907637 0.2117021 0.8495299 1.0000000 0.6491696
 [9,] 0.3735054 0.6541698 0.4120686 0.5361627 0.5298114 0.5465951 0.5551551 0.6491696 1.0000000
[10,] 0.3699831 0.5203718 0.4561641 0.6458852 0.2216563 0.7471994 0.2483035 0.3596079 0.4860194
```

Cosine Similarity Cont……

Finding weightage using similarity matrix

```
> c = x[,6]
> c
 [1] 0.3709539 0.5116912 0.4193010 0.7454097 0.4260103 1.0000000 0.2383544 0.2117021 0.5465951
[10] 0.7471994
```

Finding the rating points and Recommendation

```
+ rp[i] = sim[,i]/sum
+ }
> rp
[1] 6.075132 6.483917 7.437675 7.232124 6.713229 5.457993
> tmp4 = max(rp)
> final = which(rp == tmp4)
> final
[1] 3
> movie = tmp2[1,final]
> movie
movie6
```

Adjusted Cosine Similarity

Here we normalize the data prior to applying the Cosine Similarity

The formula for adjusted cosine similarity

$$\frac{\vec{\hat{r}}_i \cdot \vec{\hat{r}}_j}{\|\vec{\hat{r}}_i\|_2 \|\vec{\hat{r}}_j\|_2} = \frac{\sum (r_{ui} - \mu_u)(r_{uj} - \mu_u)}{\sqrt{\sum (r_{ui} - \mu_u)^2} \sqrt{\sum (r_{uj} - \mu_u)^2}}$$

Normalized Values Using mean and duality similarity

```
> mean2
 [1] 5.500000 6.000000 6.166667 6.833333 6.333333 7.200000 6.333333 7.000000 6.666667 7.000000
[11] 5.000000 6.833333
> mean3
 [1] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
[10] 0.8571429
>
```

```
> data13
      [,1] [,2]       [,3]       [,4]       [,5] [,6]       [,7] [,8]       [,9] [,10] [,11]
 [1,]  0.0   -2 -0.1666667  0.1666667  0.0000000 -2.2  2.6666667    1  0.0000000     0    -1
 [2,]  0.0    3  1.8333333  0.1666667 -2.3333333 -1.2  0.0000000    0  1.3333333    -1     0
 [3,]  0.0    1 -1.1666667  0.0000000 -0.3333333  0.8  0.6666667    2  0.0000000     0    -1
 [4,]  0.5   -1 -0.1666667  0.0000000  1.6666667  1.8  0.0000000    0 -0.6666667     0     0
 [5,]  3.5    0  0.0000000  0.1666667  0.0000000  0.8  0.0000000   -1  0.0000000    -3     0
 [6,]  1.5    0  1.8333333  0.0000000 -0.3333333  0.0 -1.3333333    0 -3.6666667     0     0
 [7,]  0.5   -1  0.0000000  1.1666667  0.0000000  0.0  0.0000000    1  0.3333333     2     0
 [8,]  0.0    0  0.0000000  1.1666667  0.0000000  0.0 -0.3333333   -3  0.3333333     1     0
 [9,] -2.5    0  0.0000000 -2.8333333 -0.3333333  0.0 -1.3333333    0  0.0000000     1     0
[10,] -3.5    0 -2.1666667  0.0000000  1.6666667  0.0 -0.3333333    0  2.3333333     0     2
```

Adjusted Cosine Similarity Cont…

Final Adjusted Cosine Similarity Matrix

```
> x1 = cosine(t(data13))
> x1
            [,1]         [,2]         [,3]         [,4]        [,5]         [,6]        [,7]
 [1,]  1.00000000 -0.179750964  0.097659225 -0.150936127 -0.13067273 -0.195604192  0.27089908
 [2,] -0.17975096  1.000000000  0.048662915 -0.779516493  0.04661994 -0.002504158 -0.32906200
 [3,]  0.09765922  0.048662915  1.000000000  0.008939564 -0.09435658 -0.214346804  0.12302023
 [4,] -0.15093613 -0.779516493  0.008939564  1.000000000  0.30616520  0.096990432  0.12259729
 [5,] -0.13067273  0.046619942 -0.094356581  0.306165201  1.00000000  0.186632098 -0.36921840
 [6,] -0.19560419 -0.002504158 -0.214346804  0.096990432  0.18663210  1.000000000 -0.03653068
 [7,]  0.27089908 -0.329061997  0.123020233  0.122597289 -0.36921840 -0.036530683  1.00000000
 [8,] -0.25580893 -0.022247223 -0.624994842 -0.021643301  0.01159483 -0.049127117  0.04992965
 [9,] -0.22890473 -0.042138241 -0.064122309 -0.128790613 -0.58867680 -0.103685600 -0.22177884
[10,] -0.10880848 -0.155325446 -0.015610910 -0.068956415 -0.49015177 -0.675166985 -0.06390501
```

Output

```
> tmp4 = max(rp)
> final = which(rp == tmp4)
> final
[1] 6
> movie = tmp2[1,final]
> movie
movie11
```

Weighted Mean And Adjusted Cosine Similarity

- Assigning Weights to Users Prior to applying Adjusted Cosine similarity matrix

```
> sum1
 [1]  9.200000 11.700000  6.966667  6.966667  9.633333  9.500000  6.000000  5.833333  8.166667
[10] 12.833333
> avger
 [1] 1.3142857 1.4625000 0.9952381 0.9952381 1.6055556 1.5833333 1.0000000 1.1666667 1.3611111
[10] 1.8333333
> 
```

```
> data15
      [,1] [,2]      [,3]      [,4]      [,5] [,6]      [,7] [,8]      [,9] [,10] [,11]     [,12]
 [1,]  0.0    2 0.1666667 0.1666667 0.0000000  2.2 2.6666667    1 0.0000000     0     1 0.0000000
 [2,]  0.0    3 1.8333333 0.1666667 2.3333333  1.2 0.0000000    0 1.3333333     1     0 0.8333333
 [3,]  0.0    1 1.1666667 0.0000000 0.3333333  0.8 0.6666667    2 0.0000000     0     1 0.0000000
 [4,]  0.5    1 0.1666667 0.0000000 1.6666667  1.8 0.0000000    0 0.6666667     0     0 1.1666667
 [5,]  3.5    0 0.0000000 0.1666667 0.0000000  0.8 0.0000000    1 0.0000000     3     0 1.1666667
 [6,]  1.5    0 1.8333333 0.0000000 0.3333333  0.0 1.3333333    0 3.6666667     0     0 0.8333333
 [7,]  0.5    1 0.0000000 1.1666667 0.0000000  0.0 0.0000000    1 0.3333333     2     0 0.0000000
 [8,]  0.0    0 0.0000000 1.1666667 0.0000000  0.0 0.3333333    3 0.3333333     1     0 0.0000000
 [9,]  2.5    0 0.0000000 2.8333333 0.3333333  0.0 1.3333333    0 0.0000000     1     0 0.1666667
[10,]  3.5    0 2.1666667 0.0000000 1.6666667  0.0 0.3333333    0 2.3333333     0     2 0.8333333
> rmse = sqrt(sum(data15^2)/sum(data15!=0))
> rmse
[1] 1.621016
> weightage = rmse/avger
> weightage
 [1] 1.2333816 1.1083869 1.6287719 1.6287719 1.0096292 1.0237995 1.6210158 1.3894421 1.1909504
[10] 0.8841905
> 
```

Weighted Mean And Adjusted Cosine Similarity

- Applying Adjusted Cosine similarity matrix on the weighted mean

```
> final = which(rp == tmp4)
> final
[1] 5
> movie = tmp2[1,final]
> movie
movie10
```

Future Scope

1. Matrix Factorization
2. Item-Item Collaborative Filtering
3. Linear Regression Based Analysis

Thanks!

Questions?