# Collaborative_Filtering_and_KNN

October 29, 2019

## 1 Collaborative Filtering and Recommender Systems

Businesses like Amazon, Netflix, and Hulu want to keep you engaged, and they accomplish this by recommending products you'll likely find interesting. The more movies and television shows Netflix and Hulu have you watch, the more advertisements you'll watch. The more products Amazon can find for you, the more likely you are to buy them.

The principle behind how businesses like this are able to recommend products you will like is called **collaborative filtering**. Collaborative filtering assumes people who consume similar content are likely to have similar preferences. If this principle is true, it can be exploited.

We will see how K nearest neighbors classification can recommend movies based on a user's past ratings. The data we will use is from Netflix. This dataset contains 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users.

### 1.1 Movie Tags

[50]:
```
movies2ids = read.csv('movies.csv')
print(movies2ids[1:5,1:2])
print(dim(movies2ids))
```

```
  movieId                              title
1       1                   Toy Story (1995)
2       2                     Jumanji (1995)
3       3            Grumpier Old Men (1995)
4       4           Waiting to Exhale (1995)
5       5 Father of the Bride Part II (1995)
[1] 9742    3
```

### 1.2 Ratings

[72]:
```
ratings = read.csv("ratings.csv")
print(ratings[1:5,])
```

```
  userId movieId rating timestamp
1      1       1      4 964982703
2      1       3      4 964981247
3      1       6      4 964982224
4      1      47      5 964983815
5      1      50      5 964982931
```

## 1.3 Data manipulation

[74]:
```
library(reshape2)
d = dcast(ratings, userId~movieId, value.var="rating")
d[is.na(d)] = 0
print(dim(d))

rated = ifelse(d>0,1,0)

numRated = apply(rated,1,sum)
d = d[numRated>100,]

numMoviesRated = apply(rated,2,sum)
d = d[,numMoviesRated>100,]

print(dim(d))
```

```
[1]  610 9725
[1] 248 135
```

## 1.4 KNN model for 1 movie

[75]:
```
library(class)

N = ncol(d)
d = d[,2:N]
N = ncol(d)

print(dim(d))

knnModel <- knn(train = d[,2:N]
                ,test = d[,2:N]
                ,cl = d[,1]
                ,k=2)

comparisons = data.frame('prediction'=knnModel,'truth'=d[,1])
print(mean(comparisons$prediction==comparisons$truth))
```

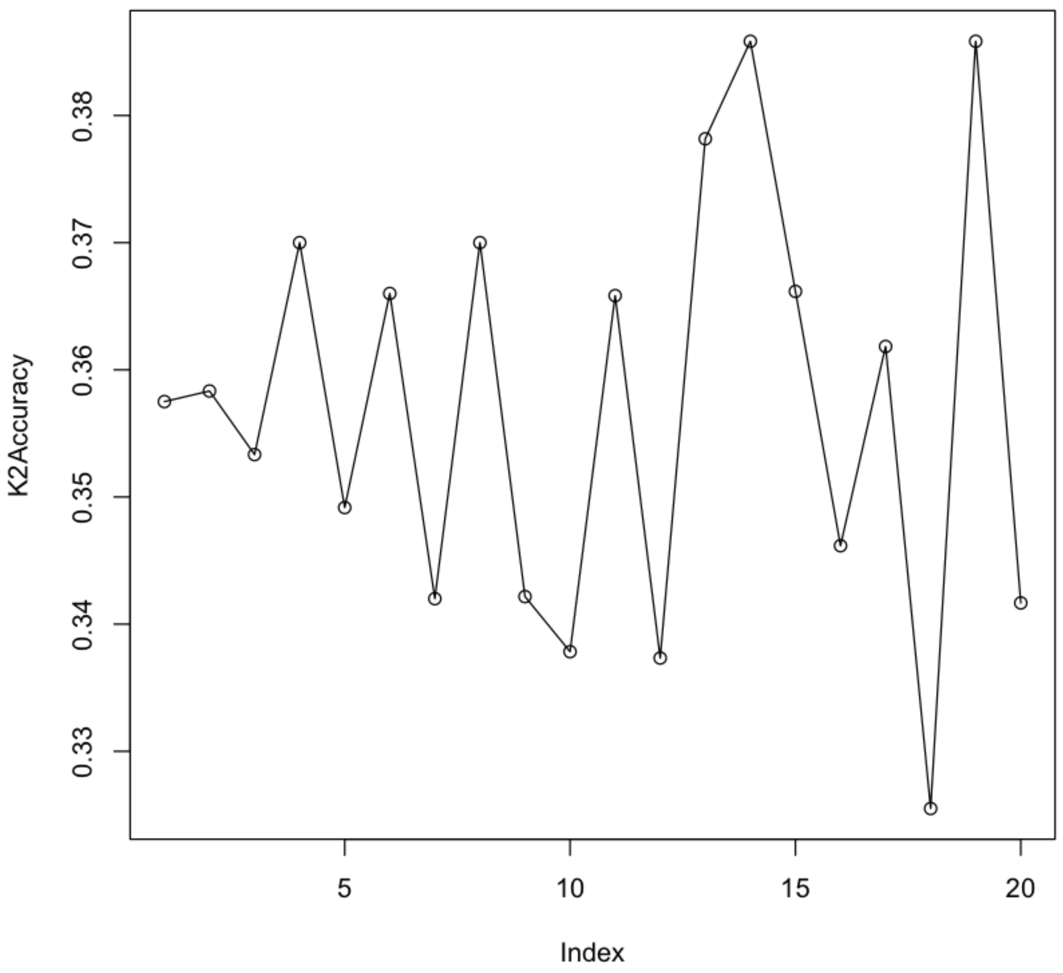
```
[1] 248 134
[1] 0.6935484
```

## 1.5 Cross-validation

[76]:
```
numObservations = nrow(d)
foldIds = split(sample(1:numObservations,numObservations),1:10)

j=1
K2Accuracy = rep(20,0)
for (K in 1:20){
    Accuracy = rep(0,10)
    for (k in 1:length(foldIds)){
        test = d[foldIds[[k]],]
        training = d[-foldIds[[k]],]

        knnModel <- knn(train = training[,setdiff(1:N,j)]
                    ,test = test[,setdiff(1:N,j)]
                    ,cl = training[,j]
                    ,k=2)

        comparisons = data.frame('prediction'=knnModel,'truth'=test[,j])
        Accuracy[k] = mean(comparisons$prediction==comparisons$truth)
    }
    K2Accuracy[K] = mean(Accuracy)
}
plot(K2Accuracy)
plot(lines(K2Accuracy))
```

```
Warning message in split.default(sample(1:numObservations, numObservations),
1:10):
âĂIJdata length is not a multiple of split variableâĂİ
Warning message in min(x):
âĂIJno non-missing arguments to min; returning InfâĂİ
Warning message in max(x):
âĂIJno non-missing arguments to max; returning -InfâĂİ
Warning message in min(x):
âĂIJno non-missing arguments to min; returning InfâĂİ
Warning message in max(x):
âĂIJno non-missing arguments to max; returning -InfâĂİ


        Error in plot.window(...): need finite 'xlim' values
    Traceback:


        1. plot(lines(K2Accuracy))
```

```
2. plot.default(lines(K2Accuracy))

3. localWindow(xlim, ylim, log, asp, ...)

4. plot.window(...)
```



### 1.6 Recommending movies to a user

[156]:
```
moviesViewed = ifelse(d>0,1,0)

distances <- as.matrix(dist(moviesViewed, method="euclidean"))

user=10
neighbors = 10
ordered.neighbors <- order(distances[user, ])[2:(neighbors+1)]
ordered.neighbors
```

```
movieProbs   = apply(moviesViewed[ordered.neighbors,],2,mean)
movieProbs

sortedMovies = movieProbs[order(movieProbs,decreasing=TRUE)]
top5Movies   = data.frame(sortedMovies[1:5])

movies2ids[movies2ids$movieId %in% as.integer(row.names(top5Movies)),]
```

1. 77 2. 64 3. 222 4. 126 5. 88 6. 226 7. 5 8. 158 9. 189 10. 129

**1** 0.1 **2** 0.1 **6** 0 **10** 0 **32** 0.1 **34** 0.3 **39** 0.1 **47** 0 **50** 0.1 **110** 0 **111** 0 **150** 0.1 **153** 0 **161** 0 **165** 0 **185** 0 **208** 0 **223** 0 **231** 0 **253** 0.1 **260** 0.2 **292** 0 **293** 0 **296** 0.2 **316** 0 **318** 0.2 **329** 0 **344** 0.1 **349** 0 **356** 0.3 **357** 0 **364** 0.1 **367** 0 **377** 0 **380** 0 **434** 0 **454** 0 **457** 0 **480** 0 **500** 0 **527** 0.1 **539** 0 **541** 0 **586** 0 **587** 0 **588** 0.3 **589** 0 **590** 0 **592** 0 **593** 0 **595** 0.2 **597** 0.2 **608** 0 **648** 0.1 **733** 0 **736** 0 **778** 0 **780** 0 **858** 0 **924** 0.1 **1036** 0 **1073** 0.1 **1089** 0.1 **1097** 0 **1136** 0.1 **1193** 0 **1196** 0.2 **1197** 0.2 **1198** 0.3 **1200** 0.1 **1206** 0 **1208** 0.1 **1210** 0 **1213** 0.1 **1214** 0.2 **1221** 0 **1222** 0 **1240** 0 **1258** 0.1 **1265** 0.2 **1270** 0.2 **1291** 0 **1527** 0 **1580** 0 **1682** 0.1 **1704** 0 **1721** 0 **1732** 0 **1923** 0.1 **1968** 0.2 **2028** 0.1 **2115** 0 **2329** 0 **2571** 0.3 **2628** 0 **2683** 0.1 **2706** 0 **2716** 0.2 **2762** 0.3 **2858** 0.3 **2918** 0.2 **2959** 0.2 **3147** 0 **3578** 0.4 **3793** 0.4 **3996** 0.3 **4226** 0.1 **4306** 0.4 **4878** 0 **4886** 0.3 **4896** 0.2 **4963** 0.2 **4973** 0 **4993** 0.4 **4995** 0.1 **5349** 0.2 **5418** 0.2 **5445** 0.1 **5816** 0.3 **5952** 0.4 **5989** 0.1 **6377** 0.3 **6539** 0.2 **6874** 0.2 **7153** 0.4 **7361** 0 **7438** 0.2 **8961** 0.4 **33794** 0.4 **48516** 0 **58559** 0.4 **60069** 0.2 **68954** 0.5 **79132** 0.3

A data.frame: 5 ÃŮ 3

| | movieId | title | genres |
|---|---|---|---|
| | &lt;int&gt; | &lt;fct&gt; | &lt;fct&gt; |
| 2675 | 3578 | Gladiator (2000) | Action \| Adv |
| 2837 | 3793 | X-Men (2000) | Action \| Adv |
| 3195 | 4306 | Shrek (2001) | Adventure \| |
| 3639 | 4993 | Lord of the Rings: The Fellowship of the Ring, The (2001) | Adventure \| |
| 7040 | 68954 | Up (2009) | Adventure \| |