Proposal

Zhu Zhengtian

04.08.20

Abstract

The correlation matrix is a fundamental statistic that used in many fields. It is used as a basic element of recommendation systems, which provide information of interest to users by referring to databases. For example, GroupLens, a collaborative filtering system, uses the correlation between users for predictive purposes. However, the estimated correlation matrix sometimes has a serious defect: although the correaltion matrix is originally positive semidefinite, the estimated one may not be positive semidefinite when not all ratings are observed. To obtain a positive semidefinite correlation matrix, the nearest correlation matrix problem has been recently studied in the fields of numerical analysis and optimization. Since the optimization problem has a prescribed rank and bound constraints on the correlations, it can be converted into a orthogonal constrained optimization problem. In this project, we use a first-order feasible method to estimate the correlation matrix.

Introduction

GroupLens Data

MovieLens data sets were collected by the GroupLens Research Project at the University of Minnesota. This data set consists of:

```
*100,000 ratings (1-5) from 943 users on 1682 movies.
*Each user has rated at least 20 movies.
*Simple demographic info for the users (age, gender, occupation, zip)
```

The data was collected through the MovieLens web site (movielens.umn.edu) during the seven-month period from September 19th, 1997 through April 22nd, 1998. This data has been cleaned up - users who had less than 20 ratings or did not have complete demographic information were removed from this data set.

The data can be downloaded at https://grouplens.org/datasets/movielens/.

```
library(dslabs)
```

```
## Warning: package 'dslabs' was built under R version 3.6.3
```

```
data(movielens)
dim(movielens)
```

```
## [1] 100004      7
```

```
head(movielens)
```

```
##   movieId                                    title year
## 1      31                     Dangerous Minds 1995
## 2    1029                                Dumbo 1941
## 3    1061                             Sleepers 1996
## 4    1129                Escape from New York 1981
## 5    1172 Cinema Paradiso (Nuovo cinema Paradiso) 1989
## 6    1263                     Deer Hunter, The 1978
```

```
##                               genres userId rating  timestamp
## 1                              Drama      1    2.5 1260759144
## 2 Animation|Children|Drama|Musical      1    3.0 1260759179
## 3                           Thriller      1    3.0 1260759182
## 4 Action|Adventure|Sci-Fi|Thriller      1    2.0 1260759185
## 5                              Drama      1    4.0 1260759205
## 6                         Drama|War      1    2.0 1260759151
```

The data includes 100004 ratings, each rating has 7 features. The 1st column is movieID which ranges from 1 to 1682. The 2nd column is the title of movie. Notice that the title may not be unique. The 3rd column is the time of first roadshow of a movie. The 4th column identifies the genre(s) of a movie. Notice that some genres may have overlapping. This kind of classification is not very explicit in this dataset. The 5th column is the userID which ranges from 1 to 943. The data is arranges by the order of userID originally. The 6th column is the rating which ranges from 0.5 to 5 and has interval 0.5. The last column is the timestamp of a user rating a movie.

Algorithm Used in GroupLens

Most collaborative filtering algorithms can be categorized into two groups: one composed of model-based algorithms, and the other of memory-based algorithms. We consider an algorithm of the second category.

The algorithm used in GroupLens is explained in the following. Let R_{ia} be the rating of item a by user i and U_{ij} be the set of items rated by both user i and user j. The correlation coefficient between user i and user j is estimated by

$$\widehat{\rho}_{ij} = \frac{\sum_{a\in U_{ij}}(R_{ia} - \bar{R}_i(U_{ij}))(R_{ja} - \bar{R}_j(U_{ij}))}{\sqrt{\sum_{a\in U_{ij}}(R_{ia} - \bar{R}_i(U_{ij}))^2}\sqrt{\sum_{a\in U_{ij}}(R_{ja} - \bar{R}_j(U_{ij}))^2}}.$$

Here, $\bar{R}_i(U_{ij}) = \sum_{a\in U_{ij}} R_{ia}/n_{ij}$ is the mean of R_{ia} in U_{ij}, and n_{ij} is the number of elements in U_{ij}.

Let us give an example to show the estimation of correlation matrix.

```
Edata <- matrix(data = c(4,3,0,0,3,5,0,2,4,0,1,3,3,2,4,2,0,5,4,0,3),byrow = TRUE, nrow = 7, ncol = 3)
Edata
```

```
##      [,1] [,2] [,3]
## [1,]    4    3    0
## [2,]    0    3    5
## [3,]    0    2    4
## [4,]    0    1    3
## [5,]    3    2    4
## [6,]    2    0    5
## [7,]    4    0    3
```

The row stands for items and the column stands for users. The number in the i-th row and j-th column is the rating of item i rated by user j. 0 means the rating is unobserved. The correlation matrix $\widehat{\rho}$ obtained from the data is

```
hrho <- matrix(0, nrow = 3, ncol = 3)
for (i in 1:3) {
  for (j in 1:3){
    find = which(Edata[,i] > 0 & Edata[,j] > 0)
    if (length(find) == 0 | length(find) == 1){
      hrho[i,j] = 0
    }
    else
    {hrho[i,j] = cor(Edata[find,i],Edata[find,j]) }
```

```
  }
}
hrho
```

```
##      [,1] [,2] [,3]
## [1,]    1    1   -1
## [2,]    1    1    1
## [3,]   -1    1    1
```

Let T_a be the set of users who rated item a and T_i' be the set of items rated by user i. The point prediction for the rating of item a by user i is given by

$$\widehat{R}_{ia} = \bar{R}_i + \frac{\sum_{j \in T_a} \widehat{\rho}_{ij}(R_{ja} - \bar{R}_j)}{\sum_{j \in T_a} |\widehat{\rho}_{ij}|}.$$

where $\bar{R}_i$ is the mean of R_{ia} in T_i'.

EDA

Movie Data

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.6.3
```

```
## -- Attaching packages ---------------------------------------------------------------------- tidyverse 1
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## Warning: package 'readr' was built under R version 3.6.3
```

```
## Warning: package 'forcats' was built under R version 3.6.3
```

```
## -- Conflicts ------------------------------------------------------------------------ tidyverse_conflic
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

At first, we load required packages. Most of the packages that were used come from the tidyverse - a collection of packages that share common philosophies of tidy data.

```
na_movielens <- movielens %>%
  filter(is.na(title) | is.na(year))
glimpse(na_movielens)
```

```
## Observations: 7
## Variables: 7
## $ movieId   <int> 108583, 162376, 108583, 108548, 151307, 108583, 143410
## $ title     <chr> NA, NA, NA, NA, NA, NA, NA
## $ year      <int> NA, NA, NA, NA, NA, NA, NA
## $ genres    <fct> Comedy, Drama, Comedy, Comedy, (no genres listed), C...
## $ userId    <int> 73, 73, 262, 270, 547, 615, 624
## $ rating    <dbl> 5.0, 4.5, 2.5, 3.0, 4.5, 5.0, 2.0
## $ timestamp <int> 1435472609, 1474255532, 1433899163, 1469278482, 1472...
```

We have a glimpse of the missing data. We can see some movies do not have debut year or title.

```
summary(movielens)
```

```
##     movieId          title                year
##  Min.   :     1   Length:100004      Min.   :1902
##  1st Qu.:  1028   Class :character   1st Qu.:1987
##  Median :  2406   Mode  :character   Median :1995
##  Mean   : 12549                      Mean   :1992
##  3rd Qu.:  5418                      3rd Qu.:2001
##  Max.   :163949                      Max.   :2016
##                                      NA's   :7
##                   genres          userId        rating
##  Drama               : 7757   Min.   :  1   Min.   :0.500
##  Comedy              : 6748   1st Qu.:182   1st Qu.:3.000
##  Comedy|Romance      : 3973   Median :367   Median :4.000
##  Drama|Romance       : 3462   Mean   :347   Mean   :3.544
##  Comedy|Drama        : 3272   3rd Qu.:520   3rd Qu.:4.000
##  Comedy|Drama|Romance: 3204   Max.   :671   Max.   :5.000
##  (Other)             :71588
##    timestamp
##  Min.   :7.897e+08
##  1st Qu.:9.658e+08
##  Median :1.110e+09
##  Mean   :1.130e+09
##  3rd Qu.:1.296e+09
##  Max.   :1.477e+09
##
```

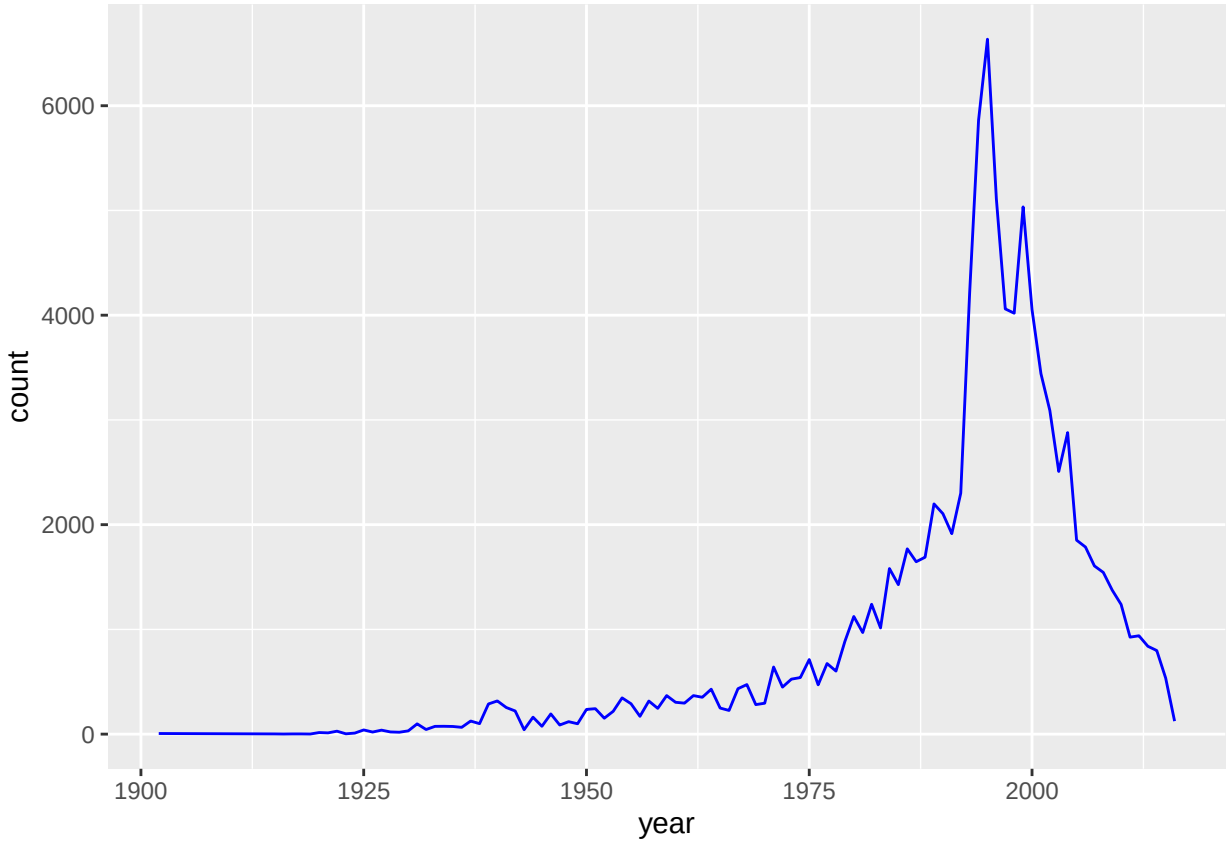
We then give a summary of our original data.

```
movie_per_year <- movielens %>%
  na.omit() %>%
  select(movieId, year) %>%
  group_by(year) %>%
  summarise(count = n()) %>%
  arrange(year)

print(movie_per_year)
```

```
## # A tibble: 103 x 2
##     year count
##    <int> <int>
##  1  1902     6
##  2  1915     2
##  3  1916     1
##  4  1917     2
##  5  1918     2
##  6  1919     1
##  7  1920    15
##  8  1921    12
##  9  1922    28
## 10  1923     3
## # ... with 93 more rows
```

```
movie_per_year %>%
  ggplot(aes(x = year, y = count)) +
  geom_line(color="blue")
```



We can see an exponential growth of the movie business and a sudden drop in 1998. The latter is caused by the fact that the data is collected until April 1998. The dataset keeps updated, but the later data are not complete.

```
genres <- movielens %>%
  separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarise(number = n()) %>%
  arrange(desc(number))

print(genres)
```

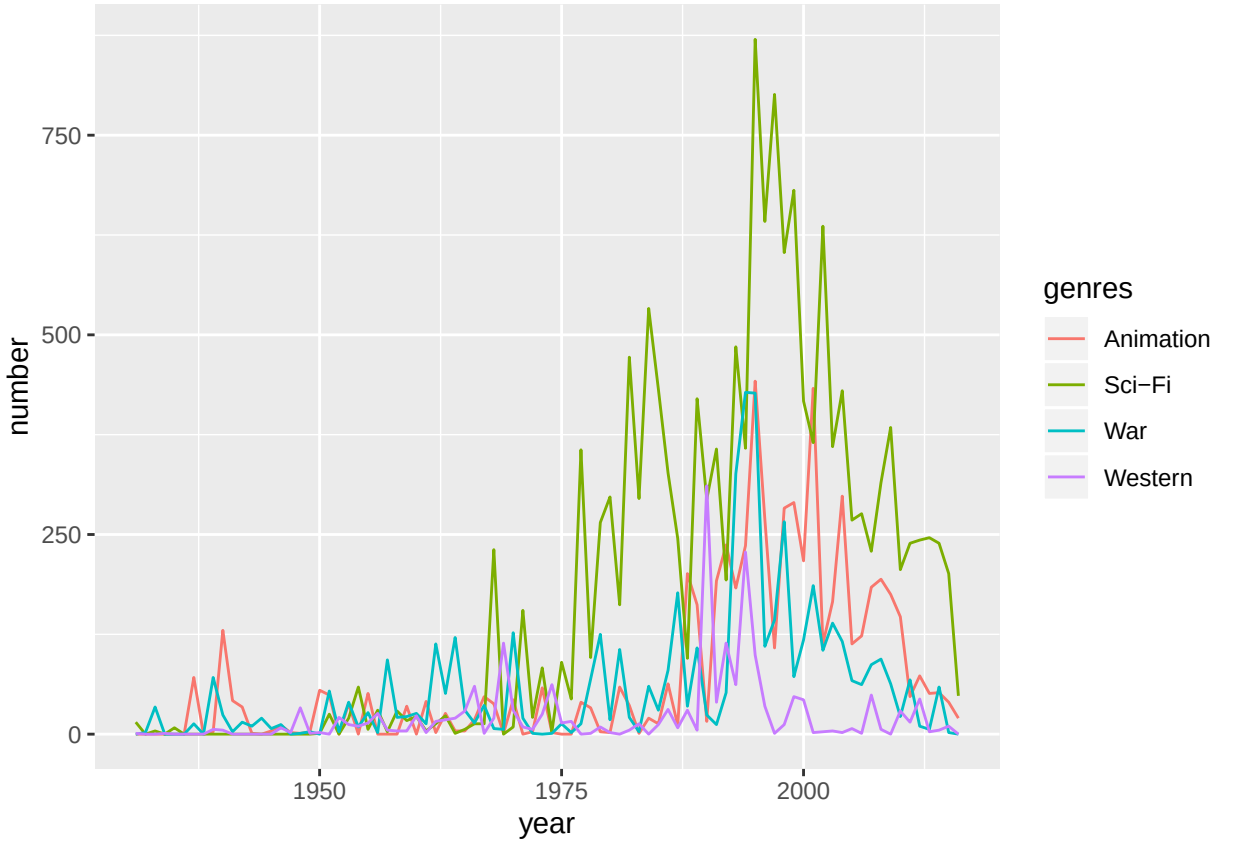
```
## # A tibble: 20 x 2
##    genres              number
##    <chr>                <int>
##  1 Drama                44752
##  2 Comedy               38026
##  3 Action               27056
##  4 Thriller             25240
##  5 Adventure            22017
##  6 Romance              19336
##  7 Crime                16266
##  8 Sci-Fi               15365
```

```
##  9 Fantasy             10657
## 10 Children             8680
## 11 Mystery              7625
## 12 Horror               6790
## 13 Animation            6170
## 14 War                  5025
## 15 Musical              4722
## 16 IMAX                 3156
## 17 Western              1912
## 18 Documentary          1564
## 19 Film-Noir            1140
## 20 (no genres listed)     18
```

We have a look at the genres.

```
genres_popularity <- movielens %>%
  na.omit() %>%
  select(movieId, year, genres) %>%
  separate_rows(genres, sep = "\\|") %>%
  mutate(genres = as.factor(genres)) %>%
  group_by(year, genres) %>%
  summarise(number = n()) %>%
  complete(year = full_seq(year, 1), genres, fill = list(number = 0))

genres_popularity %>%
  filter(year > 1930) %>%
  filter(genres %in% c("War", "Sci-Fi", "Animation", "Western")) %>%
  ggplot(aes(x = year, y = number)) +
  geom_line(aes(color=genres)) +
  scale_fill_brewer(palette = "Paired")
```



Here we have some interesting observations. First we can notice a rapid growth of sci-fi movies shortly after 1969, the year of the first Moon landing. Secondly, we notice high number of westerns in 1950s and 1960s that was the time when westerns popularity was peaking. Next, we can see the rise of popularity of animated movies, the most probable reason might be the computer animation technology advancement which made the production much easier. War movies were popular around the time when big military conflicts occured - World War II, Vietnam War and most recently War in Afghanistan and Iraq. It's interesting to see how the world of movie reflected the state of the real world.

Correlation Matrix

```
movielens[,-which(names(movielens)%in%c("title","year","genres","timestamp"))]
rating <- subset(movielens,select=-c(title,year,genres,timestamp))
```

To construct the correlation matrix, we only consider about movieID, userID and rating.

```
omatrix <- matrix(0, nrow = 1682, ncol = 943)
for (i in 1:1682){
  for (j in 1:943){
    find = which(rating[,1] == i & rating[,2] == j)
    if (length(find) == 0){
      omatrix[i,j] = 0
    }
    else
    omatrix[i,j] = rating[find,3]
  }
}
```

We rearrange the data as matrix whose row is movieID and column is userID. The element in i-th row and j-th column is the rating of movie$i$ by user$j$. 0 means the rating is unobserved. The code above runs slowly, we read from the file which is saved earlier.

```
omatrix <- read.csv("omatrix.csv", header = FALSE)
dim(omatrix)
```

```
## [1] 1683  944
```

According to the algorithm by GroupLens, we have the following 943×943 correlation matrix:

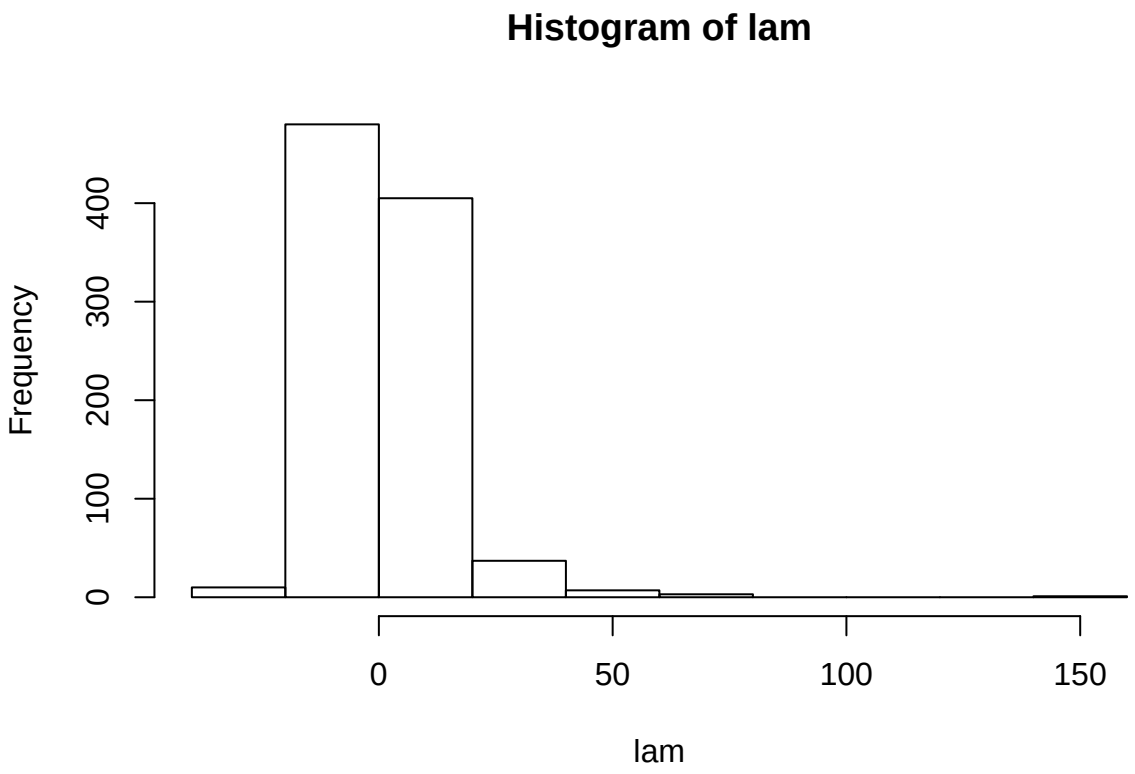
```
a <- matrix(0, nrow = 943, ncol = 943)
for (i in 1:943) {
  for (j in 1:943){
    find = which(omatrix[,i] > 0 & omatrix[,j] > 0)
    if (length(find) == 0 | length(find) == 1){
      a[i,j] = 0
    }
    else
    {a[i,j] = cor(omatrix[find,i],omatrix[find,j]) }
  }
}
a[is.na(a)] = 0
```

The Pearson correlation efficient is not defined when the variable has 0 variance. The coefficient is set to 0 in this situation.

```
a <- read.table("a.txt")
```

We also read the file directly for saving time.

```
lam <- eigen(a)$values
hist(lam)
```



We check the histogram of the 943 eigenvalues of $\hat{\rho}$. There are a lot of negative eigenvalues; hence, $\hat{\rho}$ is highly indefinite. The positive semidefiniteness of the correlation matrix is not required in the above algorithm, but the correlation matrices are originally positive semidefinite. Thus, the existence of large negative eigenvalues suggests that $\hat{\rho}$ has a lot of noise.

Model

Our goal is to find a low-rank positive semidefinite matrix of a given symmetric matrix. A low-rank matrix optimization problem is to be solved. This kind of problem belongs to semidefinite programming (SDP) which is a useful tool for modeling many applications.

Let S^n and S^n_+ be the space of $n \times n$ symmetric matrices and the cone of positive semidefinite matrices in S^n, respectively. Denote the Frobenius norm induced by the standard trace inner product $< \cdot, \cdot >$ in S^n by $\| \cdot \|$. Let C be a given matrix in S^n and $H \in S^n$ a given weight matrix whose entries are nonnegative. Then the rank constrained nearest correlation matrix problem (rank-NCM) can be formulated as follows:

$$\begin{aligned} & \min \quad \frac{1}{2}\|H \circ (X - C)\|^2 \\ \textbf{s.t.} \quad & X_{ii} = 1, \quad i = 1, \ldots, n, \\ & X \in S^n_+, \\ & rank(X) \leq r, \end{aligned}$$

where "$\circ$" denotes the Hadamard product, i.e., $(A \circ B)_{ij} = A_{ij}B_{ij}$,, $i, j = 1, \ldots, n$ and $r \in \{1, \ldots, n\}$ is a given integer. The weight matrix H is introduced by adding larger weights to correlations that are better estimated or are of higher confidence in their correctness. Zero weights are usually assigned to those correlations that are missing or not estimated.

The weight matrix H is either the identity or the one provided by T.Fushiki at Institute of Statistical Mathematics, Japan.

According to T.Fushiki, if $\{(X_1, Y_1), \ldots, (X_N, Y_N)\}$ is assumed to be independent and identically distributed (i.i.d.), an asymptotic variance of the correlation coefficient between X and Y is obtained with the delta method, which uses estimates of higher-order moments. If $\{(X_1, Y_1), \ldots, (X_N, Y_N)\}$ is, moreover, assumed to have a gaussian distribution, an asymptotic variance of $\widehat{\rho}_{X,Y}$ is obtained as $(1 - \widehat{\rho}^2_{X,Y})^2/N$ by using simply $\widehat{\rho}_{X,Y}$. Let $\widehat{V}_{ij}$ be an estimate of the variance of $\widehat{\rho}_{ij}$ and $\widehat{H}$ be the elementwise inverse of $\widehat{V}$:

$$\widehat{H}_{ij} = \begin{cases} 1/\widehat{V}_{ij} & \textbf{if } \widehat{V}_{ij} \neq 0 \\ 0 & \textbf{otherwise}. \end{cases}$$

If $\widehat{H}$ is used, the relaxed problem can be written as

$$\begin{aligned} \min \quad & \frac{1}{2}\mathbf{tr}[(X - \widehat{\rho})\{\widehat{H} \circ (X - \widehat{\rho})\}] \\ \textbf{s.t.} \quad & diag(X) = \mathbf{1}, \quad X \succeq 0. \end{aligned}$$

We express the rank constraints $rank(X) \leq r$ explicitly as $X = V^T V$ with $V = [V_1, \ldots, V_n] \in \mathbb{R}^{r \times n}$. Hence the original optimization problem is reduced to the minimization of a quadratic polynomial over spheres as follows:

$$\min_{V \in \mathbb{R}^{r \times n}} \quad \frac{1}{2}\|\widehat{H} \circ (V^T V - C)\|^2, \quad \textbf{s.t.} \quad \|V_i\|_2 = 1, i = 1, \ldots, n.$$

Future Plan

On one hand, we can cast the original problem into QSDP (quadratic semidefinite programming) form by dropping the constant term:

$$\begin{aligned} \min \quad & \frac{1}{2}\mathbf{tr}[X(\widehat{H} \circ X)] - \mathbf{tr}[X(\widehat{H} \circ \widehat{\rho})] \\ \textbf{s.t.} \quad & diag(X) = \mathbf{1}. \quad X \succeq 0. \end{aligned}$$

The loss function and the gradient are easy to compute. We can implement feasible manifold optimization algorithm to achieve V or X.

On the other hand, the dual problem is written as

$$\begin{aligned} \max \quad & -\frac{1}{2}\mathbf{tr}[X(\widehat{H} \circ X)] + \mathbf{1}^T \mathbf{y} \\ \textbf{s.t.} \quad & Diag(\mathbf{y}) - \widehat{H} \circ X + S = -\widehat{H} \circ \widehat{\rho}, \quad S \succeq 0, \end{aligned}$$

where $Diag : \mathbb{R}^n \to S^n$ is a linear map that makes a diagonal matrix with the given diagonal elements. We can implement parallelizable manifold optimization algorithm to achieve V or X. We then compare the accuracy and efficiency between two approaches.