Non-negative matrix factorization

Statistical Machine Learning

Contents

1 Non-negative matrix factorization

Non-negative matrix factorization (NMF) is an unsupervised learning technique that has found applications in several fields, including for example signal and image processing, text mining, and bioinformatics. Developments in NMF theory and applications have resulted in a variety of algorithms and methods.

The R package `NMF` includes functionalities and tools for implementing NMF and visualization of data via low-dimensional matrices. The package is accompanied by a detailed vignette and an article, available here:

- Vignette
- Article BMC Bioinformatics

2 A simple recommendation system

Non-negative matrix factorization can be used as a simple recommendation system, where the entries of the approximated matrix are taken as an estimated of the rating a user would give for a given item. Through NMF, these ratings are defined in terms of linear combinations of latent basis and coefficients.

We consider an application to rating data from Movielens (https://movielens.org/), a large platform where users can rate movies and obtain recommendations on the basis of their taste. The data are non-negative, as each user can give a rating to a movie from 0 to 5, in increments of 0.5. The dataset `movie_ratings` in the file `data_movie_lens.RData` include ratings from 100 users for more than 2500 movies.

```r
library(NMF)

load("data_movie_lens.RData")
head(movie_ratings)

# how many movies each user had rated?
sort( apply( movie_ratings, 2, function(v) sum( !is.na(v) ) ) )

# plot, a NA entry is indicated with a white cell
#  NOTE : Only a subset of movies is shown, as the whole matrix will not fit in the screen
heatmap(movie_ratings, Rowv = NA, Colv = NA, symm = FALSE,
        scale = "none", keep.dendro = FALSE, revC = FALSE,
        col = hcl.colors(20, "Inferno", rev = TRUE))
```

Not all the users have watched and rated all the movies in the data. Those instances are recorded with an `NA` entry. The goal of building a recommendation system is then filling those entries with an estimated rating for each user, which can then be employed to rank and suggest unseen movies to a user. With NMF the assumption is that ratings are composed of latent basis that are shared across movies. Hence, for a user, the rating of an unseen movie will be derived as a linear combination of ratings for other movies from other users.

The main function for implementing NMF is `nmf`, see `?nmf`. The function allows to specify the algorithm for factorization via the argument `method`. We implement the algorithm `"ls-nmf"`, based on the Frobenius norm and similar to that one presented in class. The algorithm also allows to give weights to the entries of the input data matrix (see below). We implement NMF with a rank of 10, setting `rank = 10`.

Because NMF is usually initialized randomly, the argument `nrun` allows to run the optimization multiple times from multiple starting points, keeping only the one leading to the best (local) optimum overall (similar to the function `kmeans`). The argument `seed` is used to set the random number generator in the initialization and make the results reproducible.

The function does not allow direct implementation to input data with `NA` entries. To do so, we replace the `NA` entries with a dummy value (`99999` in this case). Subsequently, we need to specify an indicator matrix that tells the algorithm to not consider those entries with the missing value. This indicator matrix is then inputted in the function through the argument `weight`.

```
# see also https://renozao.github.io/NMF/devel/demo-lsNMF-nmf.html
x <- movie_ratings   # to create matrix with dummy entries

# replace NA with fixed dummy value
x[ is.na(movie_ratings) ] <- 99999

# create NA indicator matrix
na_ind <- matrix(1, nrow(x), ncol(x))
na_ind[ is.na(movie_ratings) ] <- 0

# NMF
res <- nmf(x, rank = 10, method = "ls-nmf",
           weight = na_ind,
           nrun = 10,
           seed = 19197)
# NOTE: this will take some time to run!
```

Function `fitted` can be called to extract the approximated matrix. Note that the ratings of the approximated matrix are not necessarily upper-bounded by 5, but they can be still used for ranking purposes. Functions `basis` and `coef` can be called to extract the basis matrix $\mathbf{W}$ and the coefficient matrix $\mathbf{H}$, respectively.

We can look at the top `r` movies in each basis to interpret the latent dimension identified by a given basis vector. The highest ranking movies will be likely related by genre and/or other latent characteristic, and, given their large weight, they can be used to aid interpretation of the latent dimension.

```
# reconstructed rating matrix
x_hat <- fitted(res)
range(x_hat)

# basis and coefficient matrices
w <- basis(res)
h <- coef(res)

# top r movies in a basis
r <- 20
tail( sort(w[,3]), 20 )
tail( sort(w[,7]), 20 )
```

The estimated ratings in `x_hat` can be employed to recommend movies to a specific user. We can inspect the top `s` unseen movies by rating for a given user `u`. This list can be used as a proxy of what movies will likely receive a high rate from that user, and hence recommended.

```r
# select user - change it
u <- 25

# plot observed ratings and reconstructed ratings
plot( movie_ratings[,u], pch = 19 )
points(x_hat[,u], pch = 19, col = adjustcolor("darkorange2", 0.2))

# create list of recommendations of top s unseen movies
s <- 10
notseen <- which( is.na(movie_ratings[,u]) )
tail( sort(x_hat[notseen, u]), s )
```

2.1 Task

- We used a rank of 10 in this example, which may be too small. Re-implement NMF with a larger rank.
- Function `predict` in application to an object can be used to extract the "allocation" to the dominant basis, see `?`predict,NMF-method". Explore its use in this example.