SBF vignette

Amal Thomas

Contents

1 Background

Joint matrix factorization facilitates the comparison of expression profiles from different species without using gene mapping. Transforming gene expression profiles into reduced eigengene space using singular value decomposition (SVD) has been shown to capture meaningful biological information (Alter, Brown, and Botstein 2000). Tamayo et al. (2007) used a non-negative matrix factorization approach to learn a low-dimensional approximation of the microarray expression datasets and used the reduced space for comparisons. Matrix factorization-based methods are commonly used for gene expression analysis (Alter, Brown, and Botstein 2000; Tamayo et al. 2007). An orthology independent matrix factorization framework based on generalized singular value decomposition [GSVD; Van Loan (1976)] was used by Alter, Brown, and Botstein (2003) to compare gene-expression profiles from two species. This framework was later extended to develop higher-order generalized singular value decomposition (HO GSVD) to analyze data from more than two species (Ponnapalli et al. 2011).

This study developed a joint diagonalization approach called approximate shared basis factorization (A-SBF) for cross-species expression comparisons. This approach extends the exact factorization approach we developed called shared basis factorization (SBF). We discuss the details of the two methods in the following sections.

2 Shared basis factorization

Consider a set of real matrices $D_i \in \mathbb{R}^{m_i \times k}$ $(i = 1, \ldots, N)$ with full column rank. We define shared basis factorization (SBF) as

$$
\begin{aligned}
D_1 &= U_1 \Delta_1 V^T, \\
D_2 &= U_2 \Delta_2 V^T, \\
&\vdots \\
D_N &= U_N \Delta_N V^T.
\end{aligned}
$$

Here each $U_i \in \mathbb{R}^{m_i \times k}$ is a dataset-specific left basis matrix, each $\Delta_i \in \mathbb{R}^{k \times k}$ is a diagonal matrix with positive values δ_{ik}, and V is a square invertible matrix.

2.1 Estimating the shared right basis matrix

Let M be the scaled sum of the $D_i^T D_i$. We define M is defined as

$$
M = \frac{\sum_{i=1}^{N} D_i^T D_i / w_i}{\alpha}.
$$

The scaling factor w_i is the total variance explained by the column vectors of D_i, and α is the inverse sum of the total variance of D_i, for $i = 1 \cdots N$. The weights w_i and α are defined as

$$
\begin{aligned}
w_i &= \sum_{j=1}^{k} {\sigma_{jj}^2}^{(i)} \text{ and} \\
\alpha &= \sum_{i=1}^{N} \frac{1}{\sum_{j=1}^{k} {\sigma_{jj}^2}^{(i)}}.
\end{aligned}
$$

Here $\sum_{j=1}^{k} {\sigma_{jj}^2}^{(i)} = \mathrm{tr}(D_i^T D_i) = \mathrm{tr}(A_i)$. Using the w_i and α, individual $D_i^T D_i$ are standardized. If all the variances are equal, M becomes the arithmetic mean of the sum of $D_i^T D_i$. The shared right basis matrix V is then determined from the eigenvalue decomposition of M, where $M = V \Theta V^T$. The shared right basis matrix V is an orthogonal matrix as M is symmetric. Given V, we compute U_i and Δ_i by solving the linear system $D_i V = U_i \Delta_i = L_i$. By normalizing the columns of L_i, we have $\delta_{ik} = \|l_{ik}\|$ and $\Delta_i = \mathrm{diag}(\delta_{i1}, \ldots, \delta_{ik})$.

3 Approximate shared basis factorization

Consider a set of matrices $D_i \in \mathbb{R}^{m_i \times k}$ $(i = 1, \ldots, N)$, each with full column rank. We define approximate shared basis factorization (A-SBF) as

$$
\begin{aligned}
D_1 &= U_1 \Delta_1 V^T + \epsilon_1, \\
D_2 &= U_2 \Delta_2 V^T + \epsilon_2, \\
&\vdots \\
D_N &= U_N \Delta_N V^T + \epsilon_N.
\end{aligned}
$$

Each $U_i \in \mathbb{R}^{m_i \times k}$ is a species-specific left basis matrix with **orthonormal** columns (eigengenes), $\Delta_i \in \mathbb{R}^{k \times k}$ is a diagonal matrix with positive values Δ_{ik} and V is a non singular square matrix. The right basis matrix V is identical in all the N matrix factorizations and defines the common space shared by all species. We learn the factorization such that the learned V is closest to that in the exact decomposition and by minimizing the total decomposition error $\sum_{i=1}^{N} \epsilon_i = \sum_{i=1}^{N} \|D_i - U_i \Delta_i V^T\|^2{}_F$.

4 Usage cases

4.1 SBF examples

```
# load SBF package
library(SBF)
```

Let's create some random matrices using `createRandomMatrices` function from SBF package. We will create four matrices, each with three columns with rows varying from 4 to 6.

```
set.seed(1231)
mymat <- createRandomMatrices(n = 4, ncols = 3, nrows = 4:6)
sapply(mymat, dim)
#>      mat1 mat2 mat3 mat4
#> [1,]    5    6    4    5
#> [2,]    3    3    3    3
```

Rank of each of this matrices

```
sapply(mymat, function(x) {qr(x)$rank})
#> mat1 mat2 mat3 mat4
#>    3    3    3    3
```

Let's compute SBF using different approaches.

- Estimate V using sum of $D_i^T D_i$
- Estimate V using sum of $D_i^T D_i$ with inverse variance weighting
- Estimate V using inter-sample correlation

```
sbf <- SBF(matrix_list = mymat, check_col_matching = FALSE, weighted = FALSE,
           approximate = FALSE, transform_matrix = FALSE)
sbf_inv <- SBF(matrix_list = mymat, check_col_matching = FALSE, weighted = TRUE,
          approximate = FALSE, transform_matrix = FALSE)
#>
#> Inverse variance weighting applied
sbf_cor <- SBF(matrix_list = mymat, check_col_matching = FALSE, weighted = FALSE,
               approximate = FALSE, transform_matrix = TRUE)
#>
#> V is computed using inter-sample correlation
```

When D_i's are transformed to compute inter-sample correlation, we do not need to scale it using inverse-variance weighting anymore. We recommend using inverse variance weights, giving a more robust estimate of V when noisy datasets are present. We usually estimate V using inter-sample correlation when dealing with gene expression data sets from different species.

Let's look into the outputs of `sbf`.

```
names(sbf)
#> [1] "v"      "lambda" "u"      "delta"  "m"
```

`sbf$u`, `sbf$v`, and `sbf$delta` correspond to the estimated left basis matrix, shared right basis matrix, and diagonal matrices.

The estimated $V \in R^{k \times k}$ has a dimension of $k \times k$, where k is the number of columns in D_i.

```
sbf$v
#>            [,1]       [,2]       [,3]
#> [1,] 0.4793022  0.8669998  0.1363110
#> [2,] 0.7027353 -0.2860780 -0.6514004
```

```
#> [3,] 0.5257684 -0.4080082  0.7463892
sbf_inv$v
#>           [,1]       [,2]        [,3]
#> [1,] 0.4798100  0.8722298 -0.09485504
#> [2,] 0.7013364 -0.4462526 -0.55586503
#> [3,] 0.5271714 -0.2001843  0.82584296
sbf_cor$v
#>           [,1]        [,2]       [,3]
#> [1,] 0.4973865 -0.69637575 -0.5173659
#> [2,] 0.4799193  0.71767234 -0.5046027
#> [3,] 0.7226923  0.00268868  0.6911647
```

The delta values for each matrix for the three cases is shown below.

```
printDelta <- function(l) {
  for (eachmat in names(l$delta)) {
  cat(eachmat, ":", l$delta[[eachmat]], "\n")
  }
}
cat("sbf\n");printDelta(sbf)
#> sbf
#> mat1 : 205.4915 29.6746 71.43295
#> mat2 : 206.5816 71.72548 55.682
#> mat3 : 189.9136 52.6758 42.36825
#> mat4 : 192.6911 80.22868 58.57913
cat("sbf_inv\n");printDelta(sbf_inv)
#> sbf_inv
#> mat1 : 205.5109 22.4888 73.95623
#> mat2 : 206.5963 77.00352 48.05638
#> mat3 : 189.8719 58.60394 33.92988
#> mat4 : 192.6942 72.41955 67.98802
cat("sbf_cor\n");printDelta(sbf_cor)
#> sbf_cor
#> mat1 : 200.4197 44.25134 77.99852
#> mat2 : 199.8621 80.34494 67.23723
#> mat3 : 176.1738 76.95449 60.64485
#> mat4 : 185.4579 67.51833 89.69184
```

The $V \in R^{k \times k}$ estimated in SBF is also orthogonal. So $V^T V = I$.

```
zapsmall(t(sbf$v) %*% sbf$v)
#>      [,1] [,2] [,3]
#> [1,]    1    0    0
#> [2,]    0    1    0
#> [3,]    0    0    1
```

The estimated V is an invertible matrix.

```
qr(sbf$v)$rank
#> [1] 3
```

The U_i matrices estimated in the SBF do not have orthonormal columns. Let's explore that.

```
sapply(sbf$u, dim)
#>      mat1 mat2 mat3 mat4
#> [1,]    5    6    4    5
```

```
#> [2,]    3    3    3    3
```

Let's take the first matrix $U_i \in R^{m_i \times k}$ to check this. For this matrix, $U_i^T U_i$ will be $k \times k$ matrix where $k = 3$.

```
t(sbf$u[[names(sbf$u)[1]]]) %*% sbf$u[[names(sbf$u)[1]]]
#>              [,1]        [,2]       [,3]
#> [1,]  1.00000000 -0.07071457  0.1405487
#> [2,] -0.07071457  1.00000000 -0.6201468
#> [3,]  0.14054867 -0.62014676  1.0000000
```

The estimated M matrix is stored `sbf$m` and `sbf$lambda` gives the eigenvalues in the eigenvalue decomposition ($M = V\Theta V^T$).

```
sbf$lambda
#> [1] 39524.940  3809.127  3357.434
sbf_inv$lambda
#> [1] 39757.875  3623.628  3398.462
sbf_cor$lambda
#> [1] 1.118512 1.011059 0.870429
```

```
names(mymat)
#> [1] "mat1" "mat2" "mat3" "mat4"
sbf$delta
#> $mat1
#> [1] 205.49149  29.67460  71.43295
#>
#> $mat2
#> [1] 206.58163  71.72548  55.68200
#>
#> $mat3
#> [1] 189.91364  52.67580  42.36825
#>
#> $mat4
#> [1] 192.69106  80.22868  58.57913
```

SBF is an exact factorization. Let compute the factorization error for the three cases.

```
calcDecompError(mymat, sbf$delta, sbf$u, sbf$v)
#> [1] 1.851693e-26
calcDecompError(mymat, sbf_inv$delta, sbf_inv$u, sbf_inv$v)
#> [1] 1.894292e-26
calcDecompError(mymat, sbf_cor$delta, sbf_cor$u, sbf_cor$v)
#> [1] 2.835482e-26
```

The errors are close to zero in all three cases.

4.1.1 Noisy dataset

```
sapply(mymat, function(x) sum(diag(cov(x))))
#>    mat1    mat2    mat3    mat4
#> 2076.80 2273.50 2375.25 2860.40
```

The total column variance of matrix 1-4 is approximately in the same range. Now, let's add a dataset with similar and high variance.

```
mat5 <- matrix(c(130, 183, 62, 97, 147, 94, 102, 192, 19), byrow = T,
                    nrow = 3, ncol = 3)
```

```
mat5_highvar <- matrix(c(406, 319, 388, 292, 473, 287, 390, 533, 452), byrow = T,
                       nrow = 3, ncol = 3)

mymat_new <- mymat
mymat_new[["mat5"]] <- mat5
sapply(mymat_new, function(x) sum(diag(cov(x))))
#>     mat1     mat2     mat3     mat4     mat5
#> 2076.800 2273.500 2375.250 2860.400 2299.667
mymat_new_noisy <- mymat
mymat_new_noisy[["mat5"]] <- mat5_highvar
sapply(mymat_new_noisy, function(x) sum(diag(cov(x))))
#>     mat1     mat2     mat3     mat4     mat5
#>  2076.80  2273.50  2375.25  2860.40 22915.00
```

Let us compute SBF with the new datasets

```
sbf_new <- SBF(matrix_list = mymat_new, check_col_matching = FALSE,
               weighted = FALSE, approximate = FALSE, transform_matrix = FALSE)
sbf_inv_new <- SBF(matrix_list = mymat_new, check_col_matching = FALSE,
                   weighted = TRUE, approximate = FALSE,
                   transform_matrix = FALSE)
#>
#> Inverse variance weighting applied


sbf_new_noisy <- SBF(matrix_list = mymat_new_noisy, check_col_matching = FALSE,
                     weighted = FALSE, approximate = FALSE,
                     transform_matrix = FALSE)
sbf_inv_new_noisy <- SBF(matrix_list = mymat_new_noisy,
                         check_col_matching = FALSE, weighted = TRUE,
                         approximate = FALSE, transform_matrix = FALSE)
#>
#> Inverse variance weighting applied
```

Lets compare the decomposition error for the two cases with and without inverse variance weightinng.

```
e1 <- calcDecompError(mymat, sbf_new$delta[1:4], sbf_new$u[1:4], sbf_new$v)
e2 <- calcDecompError(mymat, sbf_new_noisy$delta[1:4], sbf_new_noisy$u[1:4],
                      sbf_new_noisy$v)
e2 / e1
#> [1] 3.887268
```

```
e3 <- calcDecompError(mymat, sbf_inv_new$delta[1:4],
                      sbf_inv_new$u[1:4], sbf_inv_new$v)
e4 <- calcDecompError(mymat, sbf_inv_new_noisy$delta[1:4],
                      sbf_inv_new_noisy$u[1:4], sbf_inv_new_noisy$v)
e4 / e3
#> [1] 1.657059
```

Inverse variance weighting reduces the total error.

4.2 ASBF examples

Now let's compute Approximate SBF for the same datasets.

- ASBF

- ASBF with inverse variance weighting
- ASBF with inter-sample correlation

```
asbf <- SBF(matrix_list = mymat, check_col_matching = FALSE, weighted = FALSE,
            approximate = TRUE, transform_matrix = FALSE)
#>
#> A-SBF is computed
asbf_inv <- SBF(matrix_list = mymat, check_col_matching = FALSE, weighted = TRUE,
           approximate = TRUE, transform_matrix = FALSE)
#>
#> Inverse variance weighting applied
#>
#> A-SBF is computed
asbf_cor <- SBF(matrix_list = mymat, check_col_matching = FALSE, weighted = FALSE,
                approximate = TRUE, transform_matrix = TRUE)
#>
#> V is computed using inter-sample correlation
#>
#> A-SBF is computed
```

ASBF is not an exact factorization. Let compute the factorization error.

```
names(asbf)
#> [1] "v"       "lambda"  "u"       "u_ortho" "delta"   "m"       "error"
```

ASBF output has two additional values. `asbf$u_ortho` is the left basis matrix with orthonormal columns and `asbf$error` gives the decomposition error.

```
asbf$error
#> [1] 2329.73
asbf_inv$error
#> [1] 1651.901
asbf_cor$error
#> [1] 14045.99
```

The same error can also be computed using `calcDecompError` function.

```
calcDecompError(mymat, asbf$delta, asbf$u_ortho, asbf$v)
#> [1] 2329.73
calcDecompError(mymat, asbf_inv$delta, asbf_inv$u_ortho, asbf_inv$v)
#> [1] 1651.901
calcDecompError(mymat, asbf_cor$delta, asbf_cor$u_ortho, asbf_cor$v)
#> [1] 14045.99
```

In ASBF factorization, U_i has orthonormal columns and V is orthogonal.

```
zapsmall(t(asbf$u_ortho[[names(asbf$u_ortho)[1]]]) %*%
           asbf$u_ortho[[names(asbf$u_ortho)[1]]])
#>      [,1] [,2] [,3]
#> [1,]    1    0    0
#> [2,]    0    1    0
#> [3,]    0    0    1
```

```
zapsmall(t(asbf$v) %*% asbf$v)
#>      [,1] [,2] [,3]
#> [1,]    1    0    0
#> [2,]    0    1    0
#> [3,]    0    0    1
```

5 Cross-species gene expression dataset analysis

For cross-species gene expression datasets, we learn the common space V based on correlation (R_i) between column phenotypes (such as tissues, cell types, etc.) within a species. In our study, we have shown that the inter-tissue gene expression correlation is similar across species. Let $X_i \in \mathbb{R}^{m_i \times k}$ be a standardized gene expression matrix where $X_i = C_i D_i {S_i}^{-1}$. Here $C_i = I_{m_i} - {m_i}^{-1} 1_{m_i} {1_{m_i}}^T$ is a centering matrix and $S_i = \text{diag}(s_1, \ldots, s_k)$ is a diagonal scaling matrix, where s_p is the standard deviation of p-th column of D_i. The matrix X_i is a matrix with columns of D_i mean-centered and scaled by the standard deviation. The correlation between expression profiles of k tissue types in species i is given by $R_i = X_i^T X_i / m_i$. We then define an expected correlation matrix ($\mathbb{E}(R_i)$) across N species as M, where M is defined as

$$M = \frac{\sum_{i=1}^{N} R_i}{N}.$$

The shared right basis matrix V capturing the inter-tissue gene expression correlation is determined from the eigenvalue decomposition of M, where $M = V \Theta V^T$. Once the V is learned, we compute U_i and Δ_i by minimizing the total decomposition error $\sum_{i=1}^{N} \epsilon_i = \sum_{i=1}^{N} \|D_i - U_i \Delta_i V^T\|^2_F$

5.1 Usage examples

Let us load the SBF package's in-built gene expression dataset. The dataset contains average gene expression profile of five similar tissues in three species.

```
# load dataset
avg_counts <- SBF::TissueExprSpecies
# check the names of species
names(avg_counts)
#> [1] "Homo_sapiens"   "Macaca_mulatta" "Mus_musculus"
```

```
# head for first species
avg_counts[[names(avg_counts)[1]]][1:5, 1:5]
#>                 hsapiens_brain hsapiens_heart hsapiens_kidney hsapiens_liver
#> ENSG00000000003         2.3109         1.9414          5.2321         5.0554
#> ENSG00000000005         0.0254         0.2227          0.5317         0.0000
#> ENSG00000000419         5.2374         5.3901          5.5659         5.0218
#> ENSG00000000457         2.0860         1.0749          2.2234         2.1266
#> ENSG00000000460         0.6529         0.0618          0.3377         0.3870
#>                 hsapiens_testis
#> ENSG00000000003          6.1414
#> ENSG00000000005          0.1423
#> ENSG00000000419          6.8038
#> ENSG00000000457          3.2602
#> ENSG00000000460          4.0250
```

The number of genes annotated in different species is different. As a result, the number of rows (genes) in the expression data will be different for different species.

```
sapply(avg_counts, dim)
#>      Homo_sapiens Macaca_mulatta Mus_musculus
#> [1,]        58676          30807        54446
#> [2,]            5              5            5
```

Lets compute A-SBF with inter-tissue correlation

```r
# A-SBF call using correlation matrix
asbf_cor <- SBF(matrix_list = avg_counts, col_index = 2, weighted = FALSE,
                approximate = TRUE, transform_matrix = TRUE)
#>
#> V is computed using inter-sample correlation
#>
#> A-SBF is computed
# calculate decomposition error
decomperror <- calcDecompError(avg_counts, asbf_cor$delta, asbf_cor$u_ortho,
                               asbf_cor$v)
decomperror
#> [1] 65865.92
```

References

Alter, Orly, Patrick O Brown, and David Botstein. 2000. "Singular Value Decomposition for Genome-Wide Expression Data Processing and Modeling." *Proceedings of the National Academy of Sciences* 97 (18): 10101–6.

———. 2003. "Generalized singular value decomposition for comparative analysis of genome-scale expression data sets of two different organisms." *Proceedings of the National Academy of Sciences* 100 (6): 3351–56.

Ponnapalli, Sri Priya, Michael A Saunders, Charles F Van Loan, and Orly Alter. 2011. "A higher-order generalized singular value decomposition for comparison of global mRNA expression from multiple organisms." *PloS One* 6 (12): e28072.

Tamayo, Pablo, Daniel Scanfeld, Benjamin L Ebert, Michael A Gillette, Charles WM Roberts, and Jill P Mesirov. 2007. "Metagene projection for cross-platform, cross-species characterization of global transcriptional states." *Proceedings of the National Academy of Sciences* 104 (14): 5959–64.

Van Loan, Charles F. 1976. "Generalizing the Singular Value Decomposition." *SIAM Journal on Numerical Analysis* 13 (1): 76–83.