scPADGRN: A preconditioned ADMM approach for reconstructing dynamic gene regulatory network using single-cell RNA sequencing data

2024

1. Introduction

The article *scPADGRN: A preconditioned ADMM approach for reconstructing dynamic gene regulatory network using single-cell RNA sequencing data* proposes a preconditioned ADMM approach to construct dynamic gene regulatory networks. The dynamic network $\{A(1), ..., A(N-1)\}$ is defined in the form of random 0-1 matrices at N different time points. Thus, there are $N-1$ matrices, which need to be optimized. According to the method described in the article all matrices have to be updated sequentially. Each matrix $A(t), \quad 1 \leq t \leq N-1$, has to be updated while the other $N-2$ matrices are fixed.

Package *scPADGRN* contains all tools described in the article. In this report we are going to talk about some possible modifications of function *preADMM()*, which is the main function in the package. The new version of function *preADMM()*, denoted *preADMM new()* has one advantage: it is shorter and simpler. The new method allows us to simplify three subproblems, which were given in the original algorithm. In the next section we will consider the simplified version of algorithm PADMM.

2. The ADMM algorithm and the PADMM algorithm

We have the following optimization problem:

(2.1)

$$\min_{A(1),...,A(N-1)} \frac{1}{2} \sum ||[Y(t+1) - Y(t)] - A(t)Y(t)||_F^2 + \alpha \sum_{t=1}^{N-1} ||A(t)||_1 + \beta \sum_{t+1}^{N-2} ||A(t+1) - A(t)||_1$$

In this equation the second term is responsible the sparsity of the network, and the third term is responsible for the continuity. Both sparsity and continuity should be considered in biological networks. However, according to the article it is possible to drop the third term for practical purposes. In our paper, we will consider the scenario without the third term, $\beta \sum_{t+1}^{N-2} ||A(t+1) - A(t)||_1$ and analyze the effect it has on the final result. Since PADMM is a version of the alternating direction method of multipliers (ADMM), first of all we consider ADMM algorithm. Since we simplified continuity, instead of three subproblems, we can use just one algorithm.

$$\min_{A(t),B(t)} \frac{1}{2}||A(t)Y(t) - [Y(t+1) - Y(t)]||_F^2 + \alpha||B(t)||_1 \tag{2.2}$$

such that $B(t) - A(t) = 0$

Augmented Lagrangian:

$$L_\rho(A(t), B(t), U(t)) = \frac{1}{2}||A(t)Y(t) - [Y(t+1) - Y(t)]||_F^2 + \alpha||A(t)||_1 + \frac{\rho}{2}||B(t) - A(t) + U(t)||_F^2 - \frac{\rho}{2}||U(t)||_F^2 \tag{2.3}$$

Iterations:

$$\begin{cases} A(t)^{k+1} = [(Y(t+1) - Y(t)) \cdot Y(t)^T + \rho^k(B(t)^k + U(t)^k][Y(t)Y(t)^T + 2\rho^k I]^{-1} \\ B(t)^{k+1} = S_{\alpha/\rho^k}(A(t)^k - U(t)^k) \\ U(t)^{k+1} = U(t)^k + B(t)^k - A(t)^k \end{cases} \tag{2.4}$$

With some adjustments to the ADMM described above, we can derive the PADMM algorithm.

$$\frac{\partial L_\rho(A(t), B(t), U(t))}{\partial A(t)} = A(t) \cdot [Y(t)Y(t)^T + 2\rho I] - (Y(t+1) - Y(t)) \cdot Y(t)^T - \rho(B(t) + U(t) \tag{2.5}$$

is equivalent to

$$A(t) \cdot [Y(t)Y(t)^T + 2\rho I] = (Y(t+1) - Y(t)) \cdot Y(t)^T + \rho(B(t) + U(t) \tag{2.6}$$

Then

$$A(t) = [(Y(t+1) - Y(t)) \cdot Y(t)^T + \rho(B(t) + U(t)] \cdot [Y(t)Y(t)^T + 2\rho I]^{-1} \tag{2.7}$$

Finally,

$$A(t) = [(Y(t+1) - Y(t)) \cdot Y(t)^T + \rho(B(t) + U(t) - 2A(t)] \cdot [Y(t)Y(t)^T + 2\rho I]^{+}, \tag{2.8}$$

where $(M)^+$ is the general inverse of the matrix M. We obtain the PADMM iterations of $A(t)$ with $1 \leq t \leq N$:

$$A(t)^{k+1} = [(Y(t+1) - Y(t)) \cdot Y(t)^T + \rho(B(t)^k + U(t)^k - 2A(t)^k] \cdot [Y(t)Y(t)^T]^+ \tag{2.9}$$

3. R code

The original code:

```
# A[[1]]
j <- 1
sum <- B[[j]]+U[[j]]+D[[j]]+W[[j]]
A[[j]] <- (rho*sum+(X[[j+1]]-X[[j]])%*%t(X[[j]])-
    2*rho*A[[j]])%*% K[[j]]
B[[j]] <- sto(A[[j]] - U[[j]],alpha, rho)
U[[j]] <- U[[j]]-A[[j]]+B[[j]]
D[[j]] <- sto(A[[j]]-W[[j]]-A[[j+1]],beta,rho)+A[[j+1]]
W[[j]] <- W[[j]]-A[[j]]+D[[j]]

# A[[2:N-2]]
for(j in c(2:(N-2))){
    sum <- B[[j]]+U[[j]]+C[[j]]+V[[j]]+D[[j]]+W[[j]]
    A[[j]] <- (rho*sum+(X[[j+1]]-X[[j]])%*%t(X[[j]])-
        3*rho*A[[j]])%*% K[[j]]
    B[[j]] <- sto(A[[j]] - U[[j]],alpha, rho)
    U[[j]] <- U[[j]]-A[[j]]+B[[j]]
    C[[j]] <- sto(A[[j]]-V[[j]]-A[[j-1]],beta,rho)+A[[j-1]]
    V[[j]] <- V[[j]]-A[[j]]+C[[j]]
    D[[j]] <- sto(A[[j]]-W[[j]]-A[[j+1]],beta,rho)+A[[j+1]]
    W[[j]] <- W[[j]]-A[[j]]+D[[j]]
}
```

```
        # A[[N-1]]
        j <- N-1
        sum <- B[[j]]+U[[j]]+C[[j]]+V[[j]]
        A[[j]] <- (rho*sum+(X[[j+1]]-X[[j]])%*%t(X[[j]])-
            2*rho*A[[j]])%*% K[[j]]
        B[[j]] <- sto(A[[j]] - U[[j]],alpha, rho)
        U[[j]] <- U[[j]]-A[[j]]+B[[j]]
        C[[j]] <- sto(A[[j]]-V[[j]]-A[[j-1]],beta,rho)+A[[j-1]]
        V[[j]] <- V[[j]]-A[[j]]+C[[j]]
```

The modified code without adjustment for continuity:

```
    for(j in c(1:(N-1))){
            sum <- B[[j]]+U[[j]]
            A[[j]] <- (rho*sum+(X[[j+1]]-X[[j]])%*%t(X[[j]])-
                2*rho*A[[j]])%*% K[[j]]
            B[[j]] <- sto(A[[j]] - U[[j]],alpha, rho)
            U[[j]] <- U[[j]]-A[[j]]+B[[j]]
        }
```

We need to notice that according to the code provided by author, the output matrices, which we get from function *preADMM()* should be converted to matrices containing only 0 or 1 as their entries. The author uses the following algorithm:

(1) we set all diagonal entries to 0.
(2) For each matrix we take the maximum of all positive values. Then we divide all positive values by this number.
(3) For each matrix we take the maximum of absolute values of all negative values. Then we divide all negative values by this number and take the absolute value of the result.
(4) Thus, for all entries we get values between 0 and 1. All numbers which are greater or equal to 0.2268 become 1; all numbers which are smaller than 0.2268 become 0.

If the absolute value of the correlation between two genes is lower than 0.2268, then we can conclude that these genes are not connected. If the absolute value of the correlation between two genes is greater than 0.2268, then we can conclude that these genes are connected.

4. Results

In order to compare the results from the original method and the modified method, we take the matrices calculated by each method and find their difference. If the resulting matrices are close to zero matrices, then we can assume that both methods give the same results. *Figure 1* illustrates the algorithm.

$$\underbrace{\begin{bmatrix}\cdots & \cdots & \cdots\\ \vdots & \ddots & \vdots\\ \cdots & \cdots & \cdots\end{bmatrix}}_{\text{matrix calculated with original method}} - \underbrace{\begin{bmatrix}\cdots & \cdots & \cdots\\ \vdots & \ddots & \vdots\\ \cdots & \cdots & \cdots\end{bmatrix}}_{\text{matrix calculated with modified method}} = \underbrace{\begin{bmatrix}\cdots & \cdots & \cdots\\ \vdots & \ddots & \vdots\\ \cdots & \cdots & \cdots\end{bmatrix}}_{\text{result}}$$

Figure 1

We repeated the procedure 100 times and got the following results.

Dataset 1

For the first matrix we have 7 different entries on average; for the second matrix we have 4 different entries on average; for the third matrix we have 3 different entries on average; for the fourth matrix we have 5 different entries on average.

Dataset 2

For the first matrix we have 71 different entries on average; for the second matrix we have 98 different entries on average; for the third matrix we have 22 different entries on average.

Dataset 3

For the first matrix we do not have any different entries on average; for the second matrix we have 5 different entries on average; for the third matrix we have just 1 different entry on average; for the fourth matrix we have 3 different entries on average; for the fifth matrix we have just 1 different entry on average.

It seems that for the second dataset the result, which we got using modified function is different from the original result. However, given that on average each matrix has around 2000 non-zero entries, we can conclude that the change which we made does not change the result significantly.

link for github