Numerical comparisons between algorithms

Chanwoo Lee, August 22, 2020

1 Symmetric trick comparison

We have discussed whether the linear SMM with symmetric trick gives us the same output $\{\tilde{\boldsymbol{X}}_1, \ldots, \tilde{\boldsymbol{X}}_n\}$ where $\tilde{\boldsymbol{X}}_i = \begin{pmatrix} 0 & \boldsymbol{X}_i^T \\ \boldsymbol{X}_i & 0 \end{pmatrix}$ for $i = 1, \ldots, n$. To be specific, let $\boldsymbol{X}_i \in \mathbb{R}^{d_1 \times d_2}$ and assume that SMM function with input data $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n\}$ and rank r gives us coefficient $\boldsymbol{B} \in \mathbb{R}^{d_1 \times d_2}$ and $\tilde{\boldsymbol{B}}$ with input data $\{\tilde{\boldsymbol{X}}_1, \ldots, \tilde{\boldsymbol{X}}_n\}$ and rank $2r$. Theoretically, the best coefficient $\tilde{\boldsymbol{B}}$ should have the form of

$$\tilde{\boldsymbol{B}} = \begin{pmatrix} \boldsymbol{B}_1 & \boldsymbol{B}_2 \\ \boldsymbol{B}_3 & \boldsymbol{B}_4 \end{pmatrix} \text{ where } \boldsymbol{B}_2^T = \boldsymbol{B}_3 = \boldsymbol{B} \text{ and } \boldsymbol{B}_1 = \boldsymbol{B}_4 = 0. \tag{1}$$

To tell the conclusion first, I verified that (1) is optimal but needs to have strict conditions to achieve from the SMM algorithm from simulations. Simulation setting is as follows.

1. Generate feature matrix $\boldsymbol{X}_i$ where $d_1 = d_2 = 2$ and $n = 200$. Each entry is from i.i.d. normal distribution.
2. Assign labels y_i for $i = 1, \ldots, n$ such that $\boldsymbol{X}|y = 1 \sim N\left(\begin{pmatrix} 1 & -1 \\ -1 & 1, \end{pmatrix}, \frac{1}{2} I_2\right)$ otherwise mean 0.

   U: 4-by-2; (d1+d2)-by-r
   U = [0, u1; u2,0]
3. Obtain estimation of $\boldsymbol{B}$ with $\{\boldsymbol{X}_i\}_{i=1}^n$ and rank r.
4. Obtain estimation of $\tilde{\boldsymbol{B}}$ with $\{\tilde{\boldsymbol{X}}_i\}_{i=1}^n$ and rank $2r$.

Numerical outputs and loss function values at the outputs are compared. We have quite stable $\boldsymbol{B}$ verified by many repetition and $\boldsymbol{B}$ and objective value are

$$\boldsymbol{B} = \begin{pmatrix} 1.893295 & -1.894880 \\ 1.971372 & -1.973023 \end{pmatrix}, \quad L(\boldsymbol{B}) = 63.76447. \tag{2}$$

However, $\tilde{\boldsymbol{B}}$ is unstable. We needs more than 50 multiple initializations and error threshold less than 10^{-4} (default was 10^{-3}) to have the similar form of (1). When we have rep = 1, we get

$$\tilde{\boldsymbol{B}} = \begin{pmatrix} -0.2812529 & -0.2148449 & 1.0178422 & 0.9978474 \\ 0.2347047 & 0.2156805 & -0.9160169 & -0.8984503 \\ 0.8591895 & -0.8892121 & -0.2797090 & -0.2560450 \\ 0.9790218 & -1.2932482 & 0.1939514 & 0.2141366 \end{pmatrix}, \quad L(\tilde{\boldsymbol{B}}) = 60.53156$$

When we have rep = 50 and 10^{-4} threshold, we have

$$\tilde{\boldsymbol{B}} = \begin{pmatrix} 0.00005 & -0.00003 & 0.53664 & 0.87369 \\ -0.00001 & 0.000001 & -0.898825 & -1.463362 \\ 1.352981 & -0.904122 & -0.00002 & -0.00003 \\ 1.081815 & -0.722917 & 0.00004 & 0.00008 \end{pmatrix}, \quad L(\tilde{\boldsymbol{B}}) = 60.10062. \tag{3}$$

we can check $\boldsymbol{B}_1 = \boldsymbol{B}_4$ converges to 0 but $\boldsymbol{B}_2^T \neq \boldsymbol{B}_3$ yet. One thing to notice is that when we define $\boldsymbol{B}' = \boldsymbol{B}_2^T + \boldsymbol{B}_3$, we obtain similar value of $\boldsymbol{B}$ and objective value in (2).

$$\boldsymbol{B}' = \boldsymbol{B}_2^T + \boldsymbol{B}_3 = \begin{pmatrix} 1.889621 & -1.802948 \\ 1.955510 & -2.186280 \end{pmatrix}, \quad \text{Ł}(\boldsymbol{B}') = 63.57433.$$

In addition if we define $\tilde{\boldsymbol{B}}' = \begin{pmatrix} 0 & (\boldsymbol{B}')^T \\ \boldsymbol{B}' & 0 \end{pmatrix}$, we can improve the result (3) much better with the loss $L(\tilde{\boldsymbol{B}}') = 59.71805$ from the loss $L(\tilde{\boldsymbol{B}}) = 60,10062$. Therefore, we can conclude that theoretically, the form (1) is optimal but algorithmically, it is a little bit hard to obtain. One possible reason for not having the form (1) is that we did not use the information $\boldsymbol{B}_1 = \boldsymbol{B}_4 = 0$ is not reflected in the algorithm.

2 Concatenated mapping and SMM method

One good thing of using concatenated mapping is that we can find directly $\boldsymbol{B}_2$ and $\boldsymbol{B}_3$ and we have $\boldsymbol{B}_1 = \boldsymbol{B}_4 = 0$ from the begining. For this reason, we do not need to have strict convergence threshold and repretition to have $\boldsymbol{B}_1 = \boldsymbol{B}_4 = 0$ like SMM with symmetric. When we use concatenated mapping with identity feature map, the decision function has the for of

$$f(\boldsymbol{X}) = \langle \boldsymbol{B}_2, \boldsymbol{X} \rangle + \langle \boldsymbol{B}_3, \boldsymbol{X} \rangle.$$

Numerically, I verified $\boldsymbol{B}_2 + \boldsymbol{B}_3 \approx \boldsymbol{B}$ where $\boldsymbol{B}$ is an optimal coefficient of SMM method. With new algorithm, I obtain $\boldsymbol{P}_{\text{row}}, \boldsymbol{P}_{\text{col}}$ and α. From those variables we can finde $\boldsymbol{B}_2$ and $\boldsymbol{B}_3$ as

Output: (alpha, P_row, P_col)

M = \sum alpha_i y_i X_i

$$\boldsymbol{B}_2 = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{P}_{\text{row}} \boldsymbol{P}_{\text{row}}^T \boldsymbol{X}_i$$

Can we conclude P_row, P_col are left/right singular vectors of M?

$$\boldsymbol{B}_3^T = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{P}_{\text{col}} \boldsymbol{P}_{\text{col}}^T \boldsymbol{X}_i^T.$$

I verified $\boldsymbol{B}_2 + \boldsymbol{B}_3 \approx \boldsymbol{B}$ and $\boldsymbol{B}_2 + \boldsymbol{B}_3$ is even better than $\boldsymbol{B}$ with respect to loss value. However, we still have $\boldsymbol{B}_2 \neq \boldsymbol{B}_3$. Detailed numerical results are as follow. simulation setting is the same as in Section 1 and $n = 50$ this time.

$$\boldsymbol{B} = \begin{pmatrix} 2.211869 & -1.977696 \\ 1.951516 & -1.744907 \end{pmatrix}, \quad L(B) = 25.41109.$$

$$\boldsymbol{B}_2 = \begin{pmatrix} 1.1287952 & -1.6637340 \\ 0.4933141 & -0.727096 \end{pmatrix}, \quad \boldsymbol{B}_3 = \begin{pmatrix} 1.172863 & -0.5698075 \\ 1.587180 & -0.771093 \end{pmatrix}$$

$$\boldsymbol{B}_2 + \boldsymbol{B}_3 = \begin{pmatrix} 2.301658 & -2.233541 \\ 2.080494 & -1.498190 \end{pmatrix}, \quad L(B2 + B3) = 23.4933.$$

So new algorithm works better than SMM algorithm in the sense that $L(B2 + B3) < L(B)$ but cannot guarantee to converge exactly to global minimum where $B_2 = B_3$.

Figure 1 plots the boundary of classification rules when SMM, SMM with symmetric trick and SMMK concatenated version are used. They all have the similar classification boundary.

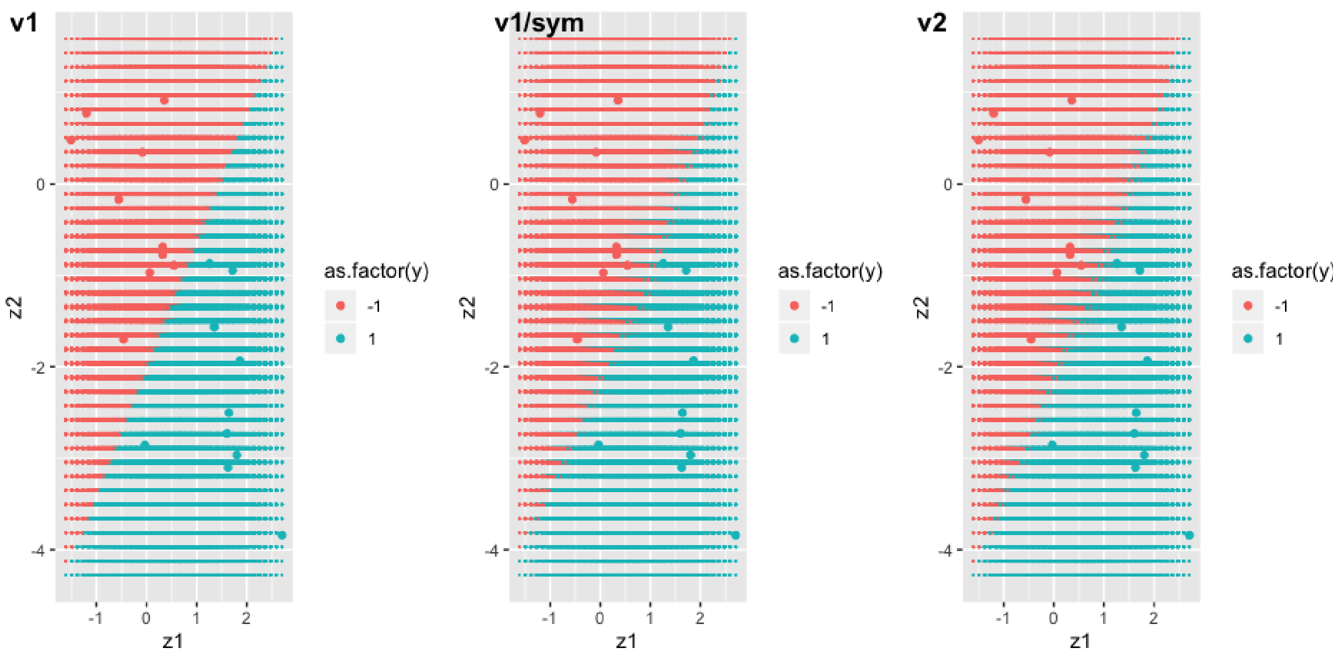

Figure 1: The left figure is a classification boundary when SMM is used. The middle figure plots the boundary of SMM with symmetric trick while the right figure is when SMMK concatenated version is utilized.