An RCPP implementation of a multi-block multi-class sparse SVM

Yaya Zhai

Department of Computational Medicine and Bioinformatics,
University of Michigan

Abstract

Classification is an important task in machine learning. Support Vector Machine method is a famous classification method for both its beautiful and simple idea and good performance in practical problems. Our lab has published a sparse multi-block multi-class SVM method, which has been proven useful in variable selection when predictors form block structures [1, 2]. However, this method has only been implemented in MATLAB, which limits its application scope. Here this method was implemented with RCPP and tested on some simulations. The result suggests this method is great at variable selection and would impressively improve classification accuracy when used in combination with ordinary multi-class classification method.

1 Introduction

Multi-block multi-class sparse algorithms are useful when more than one sample are combined to make a decision. For example, combined use of each patient's baseline sample and test sample immediately before diagnosis could improve predictive accuracy, compared to using a single test sample. Another example could be samples from different part are jointly used. This kind of problems could be handled by a multi-block multi-class sparse algorithm proposed by Liu *et al.* [2]. Here this algorithm is implemented in RCPP, which allows for more convenient use on gene expression data.

2 Algorithm

2.1 Problem description

Suppose we have a dataset with n samples, $\{\mathbf{x}_i, y_i\}_{i=1}^n$, where $\mathbf{x}_i \in \mathcal{R}^{rp}$ are independent variables, and $y_i \in \{1, 2, \cdots, K\}$ is the dependent variable. Let p be the dimension of one block, and each $\mathbf{x}_i$ consists of r blocks. All r, p, K are positive integers. The goal is to learn the best classifier of label y_i given data $\mathbf{x}_i$.

2.2 Multi-class SVM without sparsity among blocks

Let's put the multi-block sparsity aside, and consider only a multi-class SVM classifier first. Crammer and Singer *et al.* proposed such an algorithm [3], and it has been implemented by Fan *et al.* [4]. The optimization problem is:

$$\begin{aligned} \min_{\mathbf{w}_k,\xi_i} \quad & \frac{1}{2}\sum_{k=1}^{K}\mathbf{w}_k^T\mathbf{w}_k + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & \mathbf{w}_{y_i}^T\mathbf{x}_i - \mathbf{w}_k^T\mathbf{x}_i \geq e_i^k - \xi_i, \forall i = 1,\cdots,n \end{aligned} \tag{1}$$

where

$$e_i^k = \begin{cases} 0 & \text{if } y_i = k \\ 1 & \text{if } y_i \neq k \end{cases}$$

And the decision function is

$$\hat{y} = \arg\max_{k=1,\cdots,K} \mathbf{w}_k^T\mathbf{x}$$

The dual problem of (1) is:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & f(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{k=1}^{K}\|\mathbf{w}_k\|^2 + \sum_{i=1}^{n}\sum_{k=1}^{K} e_i^k\alpha_i^k \\ \text{subject to} \quad & \sum_{k=1}^{K}\alpha_i^k = 0, \forall i = 1,\cdots,n \\ & \alpha_i^k \leq C_{y_i}^k, \forall i = 1,\cdots,n,\ k = 1,\cdots,K \end{aligned} \tag{2}$$

where

$$\mathbf{w}_k = \sum_{i=1}^{n}\alpha_i^k\mathbf{x}_i, \forall k = 1,\cdots,K, \quad \boldsymbol{\alpha} = [\alpha_1^1,\cdots,\alpha_1^k,\cdots,\alpha_n^1,\cdots,\alpha_n^k]^T \tag{3}$$

and

$$C_{y_i}^k = \begin{cases} 0 & \text{if } y_i \neq k \\ C & \text{if } y_i = k \end{cases} \tag{4}$$

The optimization problem (1) and its dual problem (2) could be solved by a coordinate descent method, where the sub-prblem is minimizing the following function for each sample (indexed by i):

$$\begin{aligned} \min_{\boldsymbol{\alpha}_i} \quad & \sum_{k=1}^{K}\frac{1}{2}A_i(\alpha_i^k)^2 + B_i^k\alpha_i^k \\ \text{subject to} \quad & \sum_{k=1}^{K}\alpha_i^k = 0, \\ & \alpha_i^k \leq C_{y_i}^k, k = 1,\cdots,K \end{aligned} \tag{5}$$

where

$$A_i = \mathbf{x}_i^T \mathbf{x}_i, \text{ and } B_i^k = \mathbf{w}_k^T \mathbf{x}_i + e_i^k - A_i \alpha_i^k \tag{6}$$

This sub-problem as described in (5) and (6) have been discussed [3–5] and the solution is:

$$\alpha_i^k = \begin{cases} \min(0, \frac{\beta - B_i^k}{A_i}) & \text{if } k \neq y_i \\ \frac{\beta - B_i^{y_i}}{A_i^{y_i}} & \text{if } k = y_i \end{cases} \tag{7}$$

where

$$\begin{aligned} A_i^{y_i} &= A_i + \frac{1}{2} C \\ \beta &= \frac{\frac{A_i}{A_i^{y_i}} B_i^{y_i} + \sum_{k:\alpha^k < 0} B_i^k}{\frac{A_i}{A_i^{y_i}} + |\{k | \alpha_i^k < 0\}|} \end{aligned} \tag{8}$$

This coordinate descent method is described in Algorithm 1.
To get stopping conditions, let's define

$$G_i^k = \frac{\partial f(\boldsymbol{\alpha})}{\alpha_i^k} = \mathbf{w}_k^T \mathbf{x}_i + e_i^k, \forall i, m \tag{9}$$

and the program should stop when

$$\max_k G_i^k - \min_{k:\alpha_i^k < C_i^k} G_i^k = 0, \forall i \tag{10}$$

In addition, as G_i^k is defined as in (9), we will also have

$$B_i^k = G_i^k - A_i \alpha_i^k \tag{11}$$

2.3 Multi-block Multi-class sparse SVM

To get a multi-block sparse classifier, define a loss function as following:

$$\begin{aligned} &\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} I(y_i \neq k)[\mathbf{w}_k^T \mathbf{x}_i + 1]_+ + \lambda \sum_{j=1}^{p} \|\tilde{\mathbf{w}}_{(j)}\|_2 \\ &\tilde{\mathbf{w}}_{(j)} = [\mathbf{w}_{(j)}; \mathbf{w}_{(j+p)}; \cdots; \mathbf{w}_{(j+(r-1)p)}] \end{aligned} \tag{12}$$

where $\mathbf{w}_k \in \mathcal{R}^{rp}$ refers to the weights for class k, $\mathbf{w}_{(j)} \in \mathcal{R}^k$ the weights for j-th feature in first block, and $\tilde{\mathbf{w}}_{(j)} \in \mathcal{R}^{k \times p}$ the weights in all blocks for j-th feature. A regularization term like this can enforce a sparse model over different blocks. Variable splitting method is utilized to solve this optimization problem. This algorithm claims that if we want to solve an optimization problem of $\min_{\upsilon} f_1(\upsilon) + f_2(\upsilon)$, we could regard a different variable in one of the functions, and penalize

Algorithm 1: Coordinate descent method for multi-class SVM by Crammer and Singer

Initialize $\boldsymbol{\alpha}$, $\mathbf{w}$, $\mathbf{A} = \{A_1, \cdots, A_n\}$
while $\boldsymbol{\alpha}$ *is not optimal (stopping condition is not satisfied, as in (10))* **do**
  Random permute sample index $\{1, \cdots, n\}$ to $\pi(1), \cdots, \pi n$ **for** $i = \pi(1), \cdots, \pi(n)$ **do**
    if $\boldsymbol{\alpha}_i$ *is active and* $\mathbf{x}_i^T \mathbf{x}_i \neq 0$ **then**
      get G_i as in (9)
      get B_i as in (11)
      solve the sub-problem as solutions in (7) and (8)
      update $\mathbf{w}$ with new $\boldsymbol{\alpha}_i$
    end
  end
end

for their difference, which is equivalent to the optimization of $\min_{\upsilon,\omega} f_1(\upsilon) + f_2(\omega) + \frac{\mu}{2}\|\upsilon - \omega\|_2^2$. The solution is listed in Algorithm 2.
According to Algorithm 2, we need to perform the following two steps iteratively:

$$W_{t+1} = \arg\min_W \frac{1}{n}\sum_{i=1}^{n} \xi_i + \frac{\mu}{2}\|W - M_t - D_t\|_F^2 \quad \text{subject to } (\mathbf{w}'_{y_i}\mathbf{x}_i) + \delta_{y_i,k} - \mathbf{w}'_k\mathbf{x}_i \geq 1 - \xi_i,\ \forall i, k \tag{13}$$

$$M_{t+1} = \arg\min_M \lambda \sum_{j=1}^{p} \|\tilde{\mathbf{M}}_{(j)}\|_2 + \frac{\mu}{2}\|W_{t+1} - M - D_t\|_F^2 \tag{14}$$

2.3.1 Step one in variable splitting: solving (13)

The dual of eq13 is exactly the same with (2), except that

$$\mathbf{w}_k = \sum_{i=1}^{n} \alpha_i^k \mathbf{x}_i + \mathbf{m}_{t,k} + \mathbf{d}_{t,k} \tag{15}$$

Solving it by coordinate descent and we will again have (5) and (6).

Algorithm 2: Variable splitting method

Initialize $t = 0$, $\mu > 0$, υ_0, ω_0, d_0
while *stopping condition is not satisfied* **do**
  $\upsilon_{t+1} = \arg\min_\upsilon f_1(\upsilon) + \frac{\mu}{2}\|\upsilon - \omega_t - d_t\|_2^2$
  $\omega_{t+1} = \arg\min_\omega f_2(\omega) + \frac{\mu}{2}\|\upsilon_{t+1} - \omega - d_t\|_2^2$
  $d_{t+1} = d_t - \upsilon_{t+1} + \omega_{t+1}$
  t = t+1
end

2.3.2 Step two in variable splitting: solving (14)

(14) has a closed form solution [6]. Let $C = W_{t+1} - D_t$, then

$$\tilde{\mathbf{m}}_{(j)} = [\|\tilde{\mathbf{c}}_{(j)}\|_2 - \frac{\lambda}{\mu}]_+ \frac{\tilde{\mathbf{c}}_{(j)}}{\|\tilde{\mathbf{c}}_{(j)}\|_2} \tag{16}$$

2.4 Implementation

As analyzed above, step one in variable splitting can be adapted from standard multi-class SVM by initializing $\mathbf{w}$ to be the sum of the corresponding terms in $\mathbf{m}$ and $\mathbf{d}$, and the term of $\sum_{i=1}^{n} \alpha_i^k \mathbf{x}_i$ will be added by coordinate descent starting with the initial $\mathbf{w}$. The source code of R package LiblineaR (main functions written in C++) was utilized and adapted to perform this optimization. An RCPP program is written to take in R input, set up data and parameters, and perform Algorithm 2 by iteratively calling adapted multi-class SVM and solving (16).

3 Results and discussion

It is difficult to simulate some multi-block multi-class data that is similar to some practical problem. Here we just test the algorithm on a five-class simulation model under a small-n large-p setting. In this model, the independent predictor has a dimension of $p = 1000$, in which the first two variables are generated by $N(\boldsymbol{\mu}_k, \sigma_1^2 I_2)$, where $\boldsymbol{\mu}_k = 2(\cos([2k-1]\pi/5), \sin([2k-1]\pi/5))$, $k = 1, 2, 3, 4, 5$. The remaining $p - 2$ dimensions are independently generated by $N(0, \sigma_2^2)$ ($\sigma_1 = \sqrt{2}$ and $\sigma_2 = 1$). Each class has 50 samples and a visualization of first two dimensions and class is shown in Figure 1.

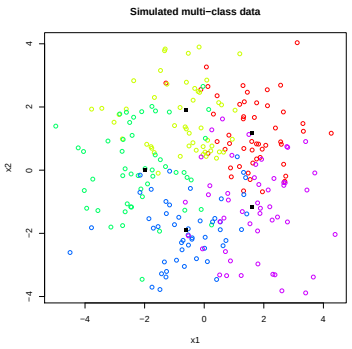

Figure 1: Simulated data visualization

Although there are no blocks in the simulated data, we can still build a model setting number of blocks to be 2. Four models are compared: **M1** ordinary multi-class SVM (MCSVM) by Crammer and Singer (R package LiblineaR) on complete independent variable; **M2** MCSVM on first 2 dimensions; **M3** multi-block multi-class SVM (MBMCSVM) setting Nblock = 1; **M4** multi-block multi-class SVM (MBMCSVM) setting Nblock = 2. A series of cost values are evaluated and training and test errors are reported using a 5-fold cross validation. The entire experiment is repeated for 10 trials. The averaged train and test errors are shown in Table 1.

From the table we can see that:

Cost		5E-07	1E-06	5E-06	1E-05	5E-05	1E-04	5E-04	0.001	0.005	0.01	0.05	0.1
MCSVM	Train	0	0	0	0	0	0	0	0	0	0	0	0
(p=1000)	Test	0.72	0.73	0.73	0.73	0.73	0.73	0.73	0.73	0.73	0.73	0.73	0.73
MCSVM	Train	0.41	0.41	0.41	0.41	0.41	0.41	0.41	0.41	0.41	0.41	0.37	0.33
(p=2)	Test	0.49	0.49	0.49	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.44	0.37
MBMCSVM	Train	0.81	0.80	0.81	0.81	0.80	0.80	0.81	0.81	0.80	0.81	0.81	0.81
(Nblock=1)	Test	0.79	0.79	0.79	0.79	0.79	0.79	0.79	0.79	0.79	0.79	0.79	0.79
	Nvar	93.1	93	92.94	93	92.7	92.68	92.46	91.82	66.2	55.74	46.6	45.58
MBMCSVM	Train	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.80	0.80	0.80	0.80	0.80
(Nblock=2)	Test	0.80	0.80	0.80	0.81	0.81	0.80	0.80	0.79	0.79	0.80	0.80	0.80
	Nvar	374.92	375.2	375.6	374.92	374.44	373.16	374	96.72	81.08	81.88	70.08	69.92

Table 1. Simulation result

- **M1** clearly has an overfitting, due to too many dimensions of the independent variable.
- Although **M3** and **M4** performs slightly worse than **M1**, they are not only optimized towards best classification result and they use much fewer predictors (as shown in their Nvar row). In addition, their performance can be improved by choosing an optimal λ. Here I just arbitrarily set a λ due to time limit.
- It is worth noting that they choose the two correct dimension in all 600 runs (12 cost * 5 CV * 10 trail), except that **M4** missed one dimension only once. Of course in **M4**, the corresponding dimensions of 1 and 2 in the other block are also selected. Dimensions other than the correct ones have a maximum selection time to be ¡ 200.
- If we perform MCSVM using the two dimensions that stand out in **M3** and **M4**, we will have our best model **M2**.
- Larger values of cost are not tested here because they would require more running time.

4 Conclusion

Multi-class classification is a difficult problem, but here we provide a multi-block multi-class sparse SVM classifier, which enforces sparsity over blocks. It has good performance on variable selection in small-n large-p case and performing variable selection prior to ordinary SVM helps improve accuracy.

References

[1] Tzu-Yu Liu, Ami Wiesel, and Alfred O Hero. A sparse multi-class classifier for biomarker screening. In *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pages 77–80. IEEE, 2013.

[2] Tzu-Yu Liu, Thomas Burke, Lawrence P Park, Christopher W Woods, Aimee K Zaas, Geoffrey S Ginsburg, and Alfred O Hero. An individualized predictor of health and disease using paired reference and target samples. *BMC bioinformatics*, 17(1):1, 2016.

[3] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research*, 2(Dec):265–292, 2001.

[4] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[5] Ching-Pei Lee and Chih-Jen Lin. A study on l2-loss (squared hinge-loss) multiclass svm. *Neural computation*, 25(5):1302–1323, 2013.

[6] Arnau Tibau Puig, Ami Wiesel, and Alfred O Hero. A multidimensional shrinkage-thresholding operator. In *2009 IEEE/SP 15th Workshop on Statistical Signal Processing*, pages 113–116. IEEE, 2009.