Impelemtation Details for SuSiE Trend Filtering

Kaiqian Zhang

May 2, 2019

1 Notations

We now describe the notation used in this text. We denote matrices by boldface uppercase letters ($\mathbf{A}$), vectors are denoted by boldface lowercase letters ($\mathbf{a}$), and scalars are denoted by non-boldface letters (a or A). All vectors are column-vectors. Lowercase letters may represent elements of a vector or matrix if they have subscripts. For example, a_{ij} is the (i,j)th element of $\mathbf{A}$, a_i is the ith element of $\mathbf{a}$, and $\mathbf{a}_i$ is either the ith row or ith column of $\mathbf{A}$. For indexing, we will generally use capital non-boldface letters to denote the total number of elements and their lowercase non-boldface versions to denote the index. For example, $i = 1, \ldots, I$. We let $\mathbf{A}_{n\times p}$ denote that $\mathbf{A} \in \mathbb{R}^{n\times p}$. We denote the matrix transpose by $\mathbf{A}^T$, the matrix inverse by $\mathbf{A}^{-1}$, and the matrix determinant by $\det(\mathbf{A})$. Finally, sets will be denoted by calligraphic letters ($\mathcal{A}$).

2 *SuSiE* for trend filtering

2.1 Overview

Trend filtering is a useful statistical tool for nonparametric regression. Kim et al. (2007) first proposed ℓ_1 trend filtering for estimating underlying piecewise linear trends in time series data. This idea can be further extended to fit piecewise polynomial of degree k to the data. In their paper, Kim et al. showed the equivalence between the ℓ_1 trend filtering and the ℓ_1-regularized least squares problem. This motivates us to think about the connection between trend filtering and sparse approximation in general.

2.2 Trend filtering and sparse regression

Trend filtering problem is defined mathematically as follows. For a given integer $k \geq 0$, the kth order trend filtering is defined by a penalized least squares optimization problem,

$$\hat{\boldsymbol{b}} = \underset{\boldsymbol{b}}{\operatorname{argmin}} \frac{1}{2}||\boldsymbol{y} - \boldsymbol{b}||_2^2 + \frac{n^k}{k!}\lambda||D_{k+1}\boldsymbol{b}||_1, \tag{2.1}$$

where $\boldsymbol{y} = [y_1 \ldots y_n]^T$ is an n vector of observations, λ is a tuning parameter, and D_{k+1} is the discrete difference operator of order k. When order $k = 0$, D is defined

$$D_1 = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ \vdots & \ddots & & & & \\ 0 & 0 & 0 & \dots & -1 & 1 \end{bmatrix} \in \mathbb{R}^{(n-1)\times n}. \tag{2.2}$$

In this case, the components of the trend filtering estimate form a piecewise constant structure, with break points corresponding to the nonzero entries of $D_1\hat{\boldsymbol{b}} = (\hat{b}_2 - \hat{b}_1, \dots, \hat{b}_n - \hat{b}_{n-1})$ Tibshirani (2014). And when $k \geq 1$, the operator D_{k+1} is defined recursively,

$$D_{k+1} = D_1 \cdot D_k \in \mathbb{R}^{(n-k-1)\times n}, \tag{2.3}$$

where the dot product is matrix multiplication. Notice that D_1 here in 2.3 is the $(n-k-1)\times(n-k)$ version of D_1 described in 2.2.

Now we want to transform the trend filtering problem into a sparse regression problem. Let $\boldsymbol{\beta} = D_{k+1}\boldsymbol{b}$. Then if D_{k+1} were invertibe, we could write $\boldsymbol{b} = (D_{k+1})^{-1}\boldsymbol{\beta}$ and the above problem would become

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{2}||\boldsymbol{y} - (D_{k+1})^{-1}\boldsymbol{\beta}||_2^2 + \frac{n^k}{k!}\lambda||\boldsymbol{\beta}||_1. \tag{2.4}$$

We can consider this a sparse regression with ℓ_1 regularization problem, where the design matrix is $X_{k+1} = (D_{k+1})^{-1}$.

2.3 Modification on D

As we have seen, the trend filtering problem becomes a sparse regression with ℓ_1 regularization if we consider the design matrix $X_{k+1} = (D_{k+1})^{-1}$. However, $D_1 \in \mathbb{R}^{(n-1)\times n}$ is not invertible, so is D_{k+1} for $k = 1, 2, \dots$. By observation, we complete D_1 as a square and symmetric matrix

$$\hat{D}_1 = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ \vdots & \ddots & & & & \\ 0 & 0 & 0 & \dots & -1 & 1 \\ 0 & 0 & 0 & \dots & 0 & -1 \end{bmatrix} \in \mathbb{R}^{n\times n}. \tag{2.5}$$

And for $k \geq 1$, we obtain

$$\hat{D}_{k+1} = \hat{D}_1 \cdot \hat{D}_k \in \mathbb{R}^{n\times n}. \tag{2.6}$$

We notice that, by this modification, $\hat{D}_{k+1}$ has k more rows added at the bottom without changing any previous entry. With this modification, we are able to invert $\hat{D}_{k+1}$ and consider the inverse matrix as a design matrix in the sparse regression.

2.4 Special structure on $\hat{D}^{-1}$

After determining the design matrix X in the sparse regression problem, we could apply *SuSiE* algorithm to help us find a possible fit. Here, we denote $X_{k+1} = (\hat{D}_{k+1})^{-1}$, where k is the order of trend filtering. Rather than generating $\hat{D}^{-1}$ and set this as an X input, we exploit the special structure of $\hat{D}^{-1}$ and perform *SuSiE* on this specific trend filtering problem with $O(n)$ complexity. We will talk about how to make different computations linear in complexity by utilizing the special structure respectively.

2.5 Computation on Xb

In the trend filtering application, since $X_{k+1} = (\hat{D}_{k+1})^{-1}$, we obtain

$$X_{k+1}\boldsymbol{b} = (\hat{D}_{k+1})^{-1}\boldsymbol{b} = (\underbrace{\hat{D}_1 \dots \hat{D}_1}_{k+1})^{-1}\boldsymbol{b} \tag{2.7}$$

$$= \underbrace{(\hat{D}_1)^{-1} \dots (\hat{D}_1)^{-1}}_{k+1}\boldsymbol{b} \tag{2.8}$$

$$= \underbrace{X_1 \dots X_1}_{k+1}\boldsymbol{b}. \tag{2.9}$$

We notice that since

$$X_1 = \begin{bmatrix} -1 & -1 & -1 & \dots & -1 & -1 \\ 0 & -1 & -1 & \dots & -1 & -1 \\ \vdots & \ddots & & & & \\ 0 & 0 & 0 & \dots & -1 & -1 \\ 0 & 0 & 0 & \dots & 0 & -1 \end{bmatrix} \in \mathbb{R}^{n\times n}, \tag{2.10}$$

$$X_1\boldsymbol{b} = -1 \cdot [b_1 + b_2 + \dots + b_n, b_2 + \dots + b_n, \dots, b_{n-1} + b_n, b_n]^T \tag{2.11}$$

$$= -1 \cdot \text{cumsum}(\text{reverse}(\boldsymbol{b})). \tag{2.12}$$

Let $f : \mathbb{R}^n \to \mathbb{R}^n$ such that $f(\boldsymbol{x}) = -\text{cumsum}(\text{reverse}(\boldsymbol{x}))$ for any $\boldsymbol{x} \in \mathbb{R}^n$. Then

$$X_{k+1}\boldsymbol{b} = \underbrace{X_1 \dots X_1}_{k+1}\boldsymbol{b} = f^{(k+1)}(\boldsymbol{b}), \tag{2.13}$$

where k is the order of trend filtering.

2.6 Computation on $X^T y$

We consider $X_{k+1}^T\boldsymbol{y}$. Here $X_{k+1} = (\hat{D}_{k+1})^{-1}$ in the trend filtering problem, and $\boldsymbol{y}$ is an n vector. We have

$$X_{k+1}^T \boldsymbol{y} = ((\hat{D}_{k+1})^{-1})^T \boldsymbol{y} = ((\underbrace{\hat{D}_1 \dots \hat{D}_1}_{k+1})^{-1})^T \boldsymbol{y} \tag{2.14}$$

$$= \underbrace{((\hat{D}_1)^{-1})^T \dots ((\hat{D}_1)^{-1})^T}_{k+1} \boldsymbol{y} \tag{2.15}$$

$$= \underbrace{X_1^T \dots X_1^T}_{k+1} \boldsymbol{y}. \tag{2.16}$$

Similarly, we observe that since

$$X_1^T = \begin{bmatrix} -1 & 0 & 0 & \dots & 0 & 0 \\ -1 & -1 & 0 & \dots & 0 & 0 \\ \vdots & \ddots & & & & \\ -1 & -1 & -1 & \dots & -1 & 0 \\ -1 & -1 & -1 & \dots & -1 & -1 \end{bmatrix} \in \mathbb{R}^{n \times n}, \tag{2.17}$$

$$X_1^T \boldsymbol{y} = -1 \cdot [y_1, y_1 + y_2, \dots, y_1 + y_2 + \dots + y_n]^T \tag{2.18}$$

$$= -1 \cdot \text{cumsum}(\boldsymbol{y}). \tag{2.19}$$

Let $g : \mathbb{R}^n \to \mathbb{R}^n$ such that $g(\boldsymbol{x}) = -\text{cumsum}(\boldsymbol{x})$ for any $\boldsymbol{x} \in \mathbb{R}^n$. Then

$$X_{k+1}^T \boldsymbol{y} = \underbrace{X_1^T \dots X_1^T}_{k+1} \boldsymbol{y} = g^{(k+1)}(\boldsymbol{y}), \tag{2.20}$$

where k is the order of trend filtering.

2.7 Computation on $(X^2)^T 1$ (i.e. colSums(X^2))

To compute $(X_{k+1}^2)^T \mathbf{1}$, let's first explore the special structure of $X_{k+1} = (\hat{D}^{(k+1)})^{-1}$ for $k = 0, 1, 2$.

$$X_1 = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & \dots \\ 0 & -1 & -1 & -1 & -1 & \dots \\ 0 & 0 & -1 & -1 & -1 & \dots \\ 0 & 0 & 0 & -1 & -1 & \dots \\ \vdots & \ddots & & & & \end{bmatrix} \in \mathbb{R}^{n \times n}, \tag{2.21}$$

$$X_2 = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & \dots \\ 0 & 1 & 2 & 3 & 4 & 5 & \dots \\ 0 & 0 & 1 & 2 & 3 & 4 & \dots \\ 0 & 0 & 0 & 1 & 2 & 3 & \dots \\ \vdots & \ddots & & & & & \end{bmatrix} \in \mathbb{R}^{n \times n}, \tag{2.22}$$

$$X_3 = \begin{bmatrix} -1 & -3 & -6 & -10 & -15 & \dots \\ 0 & -1 & -3 & -6 & -10 & \dots \\ 0 & 0 & -1 & -3 & -6 & \dots \\ 0 & 0 & 0 & -1 & -3 & \dots \\ \vdots & \ddots & & & & \end{bmatrix} \in \mathbb{R}^{n\times n}, \tag{2.23}$$

Define a triangular rotate matrix $Q \in \mathbb{R}^{n\times n}$ such that
(i) For any $i, j \le n$, $Q_{ij} = 0$ if $i > j$.
(ii) For any $k < n$, $Q_{ab} = Q_{cd}$ if $b - a = d - c = k$.
We observe that if X is a triangular rotate matrix, then

$$X^T\mathbf{1} = \text{cumsum}(X_{1.}). \tag{2.24}$$

Since X^2 is still a triangular rotate matrix, we obtain

$$(X^2)^T\mathbf{1} = \text{cumsum}(X^2_{1.}). \tag{2.25}$$

Since $X_{k+1} = (\hat{D}_{k+1})^{-1}$ is a triangular rotate matrix,

$$(X^2_{k+1})^T\mathbf{1} = \text{cumsum}((X_{k+1})^2_{1.}). \tag{2.26}$$

And obviously, the first row of X_{k+1} is

$$(X_{k+1})_{1.} = \begin{cases} \mathbf{-1} & \text{if } k = 0 \\ g^{(k)}(\mathbf{1}) & \text{if } k > 0. \end{cases} \tag{2.27}$$

2.8 Computation on μ (i.e. column means)

For each column $j = 1, 2, \dots, n$,

$$\mu_j = E[X_{.j}], \tag{2.28}$$

and $\boldsymbol{\mu} = [\mu_1, \mu_2, \dots, \mu_n]^T \in \mathbb{R}^n$. Hence we get

$$\boldsymbol{\mu} = \frac{1}{n}X^T_{k+1}\mathbf{1} = \frac{1}{n}\text{cumsum}((X_{k+1})_{1.}), \tag{2.29}$$

where $(X_{k+1})_{1.}$ is defined above in 2.7.

2.9 Computation on σ (i.e. column standard deviations)

For each column $j = 1, 2, \dots, n$,

$$\sigma_j = \sqrt{E[X_{.j}^2] - E[X_{.j}]^2} \tag{2.30}$$

$$= \sqrt{\frac{1}{n}\sum_{i=1}^{n} X_{ij}^2 - (\frac{1}{n}\sum_{i=1}^{n} X_{ij})^2}. \tag{2.31}$$

Hence, $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \ldots, \sigma_n]^T \in \mathbb{R}^n$ becomes

$$\boldsymbol{\sigma} = \sqrt{E[X^2] - E[X]^2} \tag{2.32}$$

$$= \sqrt{\frac{1}{n}\text{colSums}(X^2) - (\frac{1}{n}\text{colSums}(X))^2} \tag{2.33}$$

$$= \sqrt{\frac{1}{n}(X^2)^T\mathbf{1} + (\frac{1}{n}X^T\mathbf{1})^2}, \tag{2.34}$$

where the first term involves 2.7 and the second term is computed in 2.8. Note that in the algorithm, we set the column standard deviation 1 when the column has variance 0 for computation convenience.

2.10 Computation on $(\hat{X}^2)^T 1$ (i.e. colSums($\hat{X}^2$))

We want to compute colSums($\hat{X}^2$), where $\hat{X}$ is scaled by both column means $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_n]^T \in \mathbb{R}^n$ and column standard deviations $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \ldots, \sigma_n]^T \in \mathbb{R}^n$. We define $\hat{X} \in \mathbb{R}^{n\times n}$ such that for each $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n$,

$$\hat{X}_{ij} = \frac{X_{ij} - \mu_j}{\sigma_j} \tag{2.35}$$

where $X = X_{k+1} = (\hat{D}_{k+1})^{-1}$ if the order is k. We consider

$$\text{colSums}(\hat{X}^2) = \sum_{i=1}^{n} \hat{X}_{i.}^2 \tag{2.36}$$

$$= [\sum_{i=1}^{n} \hat{X}_{i1}^2, \ldots, \sum_{i=1}^{n} \hat{X}_{in}^2]^T \tag{2.37}$$

$$= [\sum_{i=1}^{n} (\frac{X_{i1} - \mu_1}{\sigma_1})^2, \ldots, \sum_{i=1}^{n} (\frac{X_{in} - \mu_n}{\sigma_n})^2]^T \tag{2.38}$$

$$= [\sum_{i=1}^{n} \frac{X_{i1}^2 - 2X_{i1}\mu_1 + \mu_1^2}{\sigma_1^2}, \ldots, \sum_{i=1}^{n} \frac{X_{in}^2 - 2X_{in}\mu_n + \mu_n^2}{\sigma_n^2}]^T \tag{2.39}$$

$$= \{\sum_{i=1}^{n} X_{i.}^2 - 2X^T\mathbf{1} \odot (\mu_1, \ldots, \mu_n) + n(\mu_1, \ldots, \mu_n)^2\} \oslash (\sigma_1, \ldots, \sigma_n)^2 \tag{2.40}$$

$$= (\text{colSums}(X^2) - 2X^T\mathbf{1} \odot \boldsymbol{\mu} + n\boldsymbol{\mu}^2) \oslash \boldsymbol{\sigma}^2, \tag{2.41}$$

where colSums(X^2) is computed by 2.7, $\odot$ is element-wise multiplication, $\oslash$ is element-wise division, $\boldsymbol{\mu}$ is an n vector of column means computed by 2.8, and $\boldsymbol{\sigma}$ is an n vector of column standard deviations computed by 2.9. Interestingly, because of the special structure of X_{k+1}, we also observe that

$$\text{colSums}(\hat{X}^2_{k+1}) = \begin{cases} [\underbrace{n-1, n-1, \ldots, n-1}_{n-1}, 0]^T & \text{if } k = 0 \\ [\underbrace{n-1, n-1, \ldots, n-1, n-1}_{n}]^T & \text{if } k \neq 0. \end{cases} \tag{2.42}$$

2.11 Conclusion

Computation details from section 2.5 to section 2.10 explain how we can benefit from the unique structure of matrices from trend filtering problem. As shown by our formula, we do not need to form any matrix and complete *SuSiE* algorithm with $O(n)$ complexity.

References

Kim, S.-J., K. Koh, S. Boyd, and D. Gorinevsky (2007). L1 trend filtering.

Tibshirani, R. J. (2014). Adaptive piecewise polynomial estimation via trend filtering. *The Annals of Statistics 42*(1), 285–323.