Mixed-Ridge and Fast CV methods

Shi Binbin
Chen Xupeng

0.1 How Mixed-Ridge works

FaST-LMM algorithm is a powerful method using SNPs to predict traits. It can eliminate the influence of genetic similarity between samples when selecting features and use the relevence when predicting traits. However, it is a little low in efficiency, which is not perfect when we use big data and try to fine tune the model. We would like to develop a new model called Mixed-Ridge which can use the similar strength of FaST-LMM to overcome the problems traiditional machine learning algorithms meet. It also has other benefits too. For it is a linear model, we can apply our highly efficient k-fold cross validation method to have much more quick feedback. Also in linear model, it is easy to use iteration to optimize the loss function. What's more, it is more convenient to use L1 or L2 regularizations in linear model rather than probabilistic model. Here is the explanation of our model. We use the function

$$\mathbf{y} = X_1\vec{\boldsymbol{\beta}}_1 + \gamma(\Lambda_1\vec{\boldsymbol{\beta}}_2) + \boldsymbol{\beta}_0 + \epsilon \tag{1}$$

y is the trait we want to predict.X_1 are SNPs we select. Λ_1 is the matrix from singular value decomposition(SVD) . We random select 100,000 SNPs as X_2. First we did SVD to X_2 to generate U_1

$$X_2 = US^{\frac{1}{2}}V^T = \begin{bmatrix} U_1 & U_2 \end{bmatrix} S^{\frac{1}{2}}V^T \approx U_1 S^{\frac{1}{2}} V_1^T \tag{2}$$

U1 is constructed from columns of U that corresponds to singular values above a threshold. Currently we set the threshold for singular values to 0.1 (we set $S_{ii} > 0.1$). U_1 is a matrix has 6210 columns and we truncate it into 270 columns. So the dimension of U_1 is 100,000 * 270. In FaST-LMM, they use $\mathbf{X}_2\mathbf{X}_2^T$ as K_0 to generate U_1. It can be easily proved that both U_1 are exactly the same one:

$$X_2 = US^{\frac{1}{2}}V^T \tag{3}$$

$$K_0 = X_2X_2^T = USU^T \tag{4}$$

We do SVD to X_2 matrix's and spectral decomposition to K_0. They have different time complexity. The time complexity is O($n^3 + n^2m$) for K_0 and O(**min**(m^2n, n^2m)) for X_2 because the time complexity for matrix multiplication $(X_2X_2^T)$ is $O(n^2m)$. In this issue n equals to 100,000 and m equals to 6210, so we use X_2 instead of K_0 to save time.
Then we have Λ_1 from U_1:

$$\Lambda_1 = US^{\frac{1}{2}} \tag{5}$$

And $\vec{\boldsymbol{\beta}}_1, \gamma, \vec{\boldsymbol{\beta}}_2, \boldsymbol{\beta}_0$ are the parameters we need to optimize.
Here are our main steps to find the best parameters in equation(1):

Algorithm 1 Mixed-Ridge

1: Step 1. We need to optimize the parameters $\vec{\boldsymbol{\beta}}_1, \gamma, \vec{\boldsymbol{\beta}}_2, \boldsymbol{\beta}_0$ in the function:

$$\mathbf{y} = X_1\vec{\boldsymbol{\beta}}_1 + \gamma(\Lambda_1\vec{\boldsymbol{\beta}}_2) + \boldsymbol{\beta}_0 + \epsilon \tag{6}$$

To get a better prediction of **y**, we define a loss function like below:

$$\min||\mathbf{y} - X_1\vec{\boldsymbol{\beta}}_1 - \gamma(\Lambda_1\vec{\boldsymbol{\beta}}_2) - \boldsymbol{\beta}_0||_2^2 + \lambda||\boldsymbol{\beta}_1||_2^2 + \lambda||\boldsymbol{\beta}_2||_2^2 \tag{7}$$

The coefficients can be found by minimizing the squared-error and L_2 regularization term.

2: Step 2. We use iterations to optimize the loss function. At first we fix γ and optimize the loss function to search for γ. After t steps we can get:$\hat{\vec{\boldsymbol{\beta}}}_1^{(t)}, \hat{\boldsymbol{\beta}}_2^{(t)}, \hat{\boldsymbol{\beta}}_0^{(t)}$

3: Step 3. We then fix $\hat{\vec{\boldsymbol{\beta}}}_1^{(t)}, \hat{\boldsymbol{\beta}}_2^{(t)}, \hat{\boldsymbol{\beta}}_0^{(t)}$. We can have:

$$\begin{aligned} \hat{\vec{\boldsymbol{y}}}_1^{(t)} &= X_1 \hat{\vec{\boldsymbol{\beta}}}_1^{(t)} + \beta_0 \\ \hat{\vec{\boldsymbol{y}}}_2^{(t)} &= \Lambda_1 \hat{\vec{\boldsymbol{\beta}}}_2^{(t)} \end{aligned} \tag{8}$$

Then we can use the loss function to search for γ:

$$\min||\hat{\vec{\boldsymbol{y}}} - [(1-\gamma)\hat{\vec{\boldsymbol{y}}}_1^{(t)} + \gamma \hat{\vec{\boldsymbol{y}}}_2^{(t)}]||_2^2 \tag{9}$$

The function is equal to:

$$\min||(\hat{\vec{\boldsymbol{y}}} - X_1 \hat{\vec{\boldsymbol{\beta}}}_1^{(t)} - {\beta_0}^t) + \gamma(\hat{\boldsymbol{\beta}}_2^{(t)} - X_1 \hat{\vec{\boldsymbol{\beta}}}_1^{(t)})||_2^2 \tag{10}$$

4: Step 4. We can repeat the first three steps many times to select best SNPs. The optimization method is gradient descent or alternating least squares(ALS) or grid search.

0.2 Mixed-Ridge has the similar effect with FaST-LMM

If we can prove that our Mixed-Ridge model have the same fact as the FaST-LMM model, we can use our new model to enjoy the benefits above. Here we prove that the mean value of the two models distribution are same. The models both have the assumption that the distribution are normal which is determined by the parameters mean and variance, and the prediction only relys on mean value. So we can prove the two models have the same effect by proving their means are the same.

0.2.1 LMM

The distribution of test set is $N(\vec{y}|X\vec{\beta}; \sigma_g^2 K + \sigma_e^2 I)$

$$P(\vec{y} \mid X, \sigma_g^2, K) = \int N(\vec{y}|X\vec{\beta}; \sigma_g^2 K + \sigma_e^2 I)\ p(\vec{\beta}|\ )\ d\vec{\beta} \tag{11}$$

$$= N(\vec{y} \mid 0; \sigma_g^2 K + \sigma_e^2 I) \tag{12}$$

We assume the distribution of LMM model is Normal, so we have the predictive distribution of a Gaussian process:$N(\vec{y^*} \mid m(z^*), var(x^*))$
m means mean and var means variance of the sample. Now we will calculate $m(z^*)$:

$$\begin{aligned} m(z^*) &= {k^*}^T C^{-1} \vec{y} \\ \vec{z^*} &= \begin{bmatrix} \vec{x^*} & \vec{a^*} \end{bmatrix} \end{aligned} \tag{13}$$

a_i and x_i are elements of **X**:

$$\text{X=} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

$$k_j^* = \sigma_g^2 k(\vec{z^*}, \vec{z_i}) + \vec{x^*}^T \vec{x_i} \tag{14}$$

$$\begin{aligned} k &= \frac{1}{N} G G^T \\ k(\vec{z^*}, \vec{z_i}) &= \vec{g^*} \vec{g_i} \end{aligned} \tag{15}$$

So we have $m(z^*)$:

$$m(z^*) = \vec{k^*}(\sigma_g^2 K + \sigma_e^2 I + \alpha X X^T)^{-1} \vec{y} \tag{16}$$

$$= \begin{bmatrix} \vec{x^*} \\ \vec{a} \end{bmatrix}^T \begin{bmatrix} \alpha X & \sigma_g^2 \Lambda \end{bmatrix} (\sigma_g^2 K + \sigma_e^2 I + \alpha X X^T)^{-1} \vec{y} \tag{17}$$

0.2.2 Mixed-Ridge

We should optimize the following function:

$$||\mathbf{y} - X_1 \vec{\boldsymbol{\beta}_1} - \gamma(\Lambda_1 \vec{\boldsymbol{\beta}_2}) - \boldsymbol{\beta}_0||_2^2 + \lambda ||\boldsymbol{\beta}_1||_2^2 + \lambda ||\boldsymbol{\beta}_2||_2^2 \tag{18}$$

We have:

$$\begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}^T = (X^T X + \gamma \Lambda^T \Lambda + \alpha I)^{-1} \begin{bmatrix} X & \Lambda \end{bmatrix}^T \vec{y} \tag{19}$$

$$= \begin{bmatrix} X & \Lambda \end{bmatrix} (X X^T + \gamma \Lambda \Lambda^T + \alpha I)^{-1} \vec{y} \tag{20}$$

Then we have the prediction $\vec{y^*}$

$$\vec{y^*} = \vec{x^*}^T \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}^T \tag{21}$$

$$= \vec{x^*}^T \begin{bmatrix} X & \gamma \Lambda \end{bmatrix} (X X^T + \gamma \Lambda \Lambda^T + \alpha I)^{-1} \vec{y} \tag{22}$$

We can compare Mixed-Ridge and LMM's mean value's form:

$$\begin{aligned} &\textbf{M - R:}\ \vec{y^*} = \vec{x^*}^T \begin{bmatrix} X & \gamma \Lambda \end{bmatrix} (X X^T + \gamma \Lambda \Lambda^T + \alpha I)^{-1} \vec{y} \\ &\textbf{LMM:}\ m(z^*) = \begin{bmatrix} \vec{x^*} \\ \vec{a} \end{bmatrix}^T \begin{bmatrix} \alpha X & \sigma_g^2 \Lambda \end{bmatrix} (\sigma_g^2 K + \sigma_e^2 I + \alpha X X^T)^{-1} \vec{y} \end{aligned} \tag{23}$$

Clearly they have the same form, so we prove that the two models have the similar effect.

0.3 Fast CV: Efficient Computation of K-fold Cross-validation Error for Linear Models

We have developed a new model which has same effects with FaST-LMM and has some other benefits. One of its great advantage is its less time consuming. We develop a highly efficient new K-fold Cross-validation computation algorithm to speed up the computation. For ridge regression, the coefficients are found by minimizing the squared-error and L_2 regularization term:

$$\text{minimize} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_2^2 \tag{24}$$

The solution to ridge regression is:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \tag{25}$$

The estimate of $\mathbf{y}$ is:

$$\hat{\mathbf{y}} = \mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \tag{26}$$

We can also write $\hat{\mathbf{y}}$ as:

$$\hat{\mathbf{y}} = \mathbf{S}\mathbf{y} \tag{27}$$

where $\mathbf{S}$ is a smoother matrix of $\mathbf{y}$: $\mathbf{S} = \mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T$.
The leave-one-out cross-validation error of linear regression can be computed efficiently:

$$\text{LOOCV}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N}[y_i - \hat{y}^{-i}(x_i)]^2 = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \hat{f}(x_i)}{1 - S_{ii}}\right]^2 \tag{28}$$

where S_{ii} is the *i*-th diagonal element of $\mathbf{S}$.
The GCV approximation of the leave-one-out cross-validation is:

$$\text{GCV}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \hat{f}(x_i)}{1 - \text{trace}(\mathbf{S})/N}\right]^2 \tag{29}$$

where $\text{trace}(\mathbf{S})$ is the effective number of parameters.
For k-fold cross-validation, the cross-validation error is:

$$\text{CV}(\hat{f}) = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{N_k}\sum_{i=1}^{N_k}[y_{ki} - \hat{f}^{-k}(\mathbf{x}_{ki})]^2 \tag{30}$$

where N_k is the number of test samples in the *k*-th part of the dataset. $(\mathbf{x}_{ki}, y_{ki})$ is the *i*th sample in the *k*-th part of the dataset. $\hat{f}^{-k}(\mathbf{x}_{ki}$ is the fitted function on the dataset with the *k*-th part removed.
The smoother matrix of the training samples is:

$$\mathbf{S}_k = \mathbf{X}_k\mathbf{A}^{-1}\mathbf{X}_k^T \tag{31}$$

where $\mathbf{A} = \mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$
The estimate of *k*th part of the test samples by the function fitted on the full dataset is:

$$\hat{\mathbf{f}}(\mathbf{X}_k) = \mathbf{X}_k\mathbf{A}^{-1}\mathbf{X}^T\mathbf{y} \tag{32}$$

Denote the fitted function with the *k*th part removed by $\hat{\mathbf{f}}^{-k}(\mathbf{X}_k)$.

$$\hat{\mathbf{f}}^{-k}(\mathbf{X}_k) = \mathbf{X}_k(\mathbf{X}^T\mathbf{X} - \mathbf{X}_k^T\mathbf{X}_k + \lambda\mathbf{I})^{-1}(\mathbf{X}^T\mathbf{y} - \mathbf{X}_k^T\mathbf{y}_k) \tag{33}$$

$$= \mathbf{X}_k(\mathbf{A} - \mathbf{X}_k^T\mathbf{X}_k)^{-1}(\mathbf{X}^T\mathbf{y} - \mathbf{X}_k^T\mathbf{y}_k) \tag{34}$$

Following the properties of inverse of a block matrix:

$$(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} \tag{35}$$

we can separate $\mathbf{X}_k$ from $(\mathbf{A} - \mathbf{X}_k^T\mathbf{X}_k)^{-1}$:

$$(\mathbf{A} - \mathbf{X}_k^T\mathbf{X}_k)^{-1} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{X}_k(\mathbf{I} - \mathbf{X}_k\mathbf{A}\mathbf{X}_k^T)^{-1}\mathbf{X}_k^T\mathbf{A}^{-1} \tag{36}$$

$$= \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{X}_k(\mathbf{I} - \mathbf{S}_k)^{-1}\mathbf{X}_k^T\mathbf{A}^{-1} \tag{37}$$

Plugging in $(\mathbf{A}-\mathbf{X}_k^T\mathbf{X}_k)^{-1}$ into the calculation of $\hat{\mathbf{f}}^{-k}(\mathbf{X}_k)$:

$$\begin{aligned}\hat{\mathbf{f}}^{-k}(\mathbf{X}_k) &= \mathbf{X}_k[\mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{X}_k(\mathbf{I}-\mathbf{S}_k)^{-1}\mathbf{X}_k^T\mathbf{A}^{-1}](\mathbf{X}^T\mathbf{y}-\mathbf{X}_k^T\mathbf{y}_k)\\ &= \mathbf{X}_k\mathbf{A}^{-1}\mathbf{X}^T\mathbf{y}\\ &\quad +\mathbf{X}_k\mathbf{A}^{-1}\mathbf{X}_k^T\mathbf{y}_k\\ &\quad +\mathbf{X}_k\mathbf{A}^{-1}\mathbf{X}_k(\mathbf{I}-\mathbf{S}_k)^{-1}\mathbf{X}_k^T\mathbf{A}^{-1}\mathbf{X}^T\mathbf{y}\\ &\quad +\mathbf{X}_k\mathbf{A}^{-1}\mathbf{X}_k(\mathbf{I}-\mathbf{S}_k)^{-1}\mathbf{X}_k^T\mathbf{A}^{-1}\mathbf{X}_k^T\mathbf{y}_k\\ &= \hat{\mathbf{f}}(\mathbf{X}_k)+\mathbf{S}_k\mathbf{y}_k+\mathbf{S}_k(\mathbf{I}-\mathbf{S}_k)^{-1}\hat{\mathbf{f}}(\mathbf{X}_k)+\mathbf{S}_k(\mathbf{I}-\mathbf{S}_k)^{-1}\mathbf{S}_k\\ &= [\mathbf{I}+\mathbf{S}_k(\mathbf{I}-\mathbf{S}_k)^{-1}][\hat{\mathbf{f}}(\mathbf{X}_k)-\mathbf{y}_k]+\mathbf{y}_k\end{aligned}$$

Then the cross-validated residual on the test samples can be written as:

$$\mathbf{y}_k-\hat{\mathbf{f}}^{-k}(\mathbf{X}_k)=[\mathbf{I}+\mathbf{S}_k(\mathbf{I}-\mathbf{S}_k)^{-1}][\mathbf{y}_k-\hat{\mathbf{f}}(\mathbf{X}_k)] \tag{38}$$

The cross-validated squared error of the *k*th part of the dataset is:

$$\frac{1}{N_k}||\mathbf{y}_k-\hat{\mathbf{f}}^{-k}(\mathbf{X}_k)||_2^2=[\mathbf{y}_k-\hat{\mathbf{f}}(\mathbf{X}_k)]^T\mathbf{B}_k^T\mathbf{B}_k[\mathbf{y}_k-\hat{\mathbf{f}}(\mathbf{X}_k)] \tag{39}$$

where $\mathbf{B}_k=\mathbf{I}+\mathbf{S}_k(\mathbf{I}-\mathbf{S}_k)^{-1}$.

$\mathbf{S}_k$ can be approximated by only considering the diagonal elements. Then the cross-validation error on the *k*th part can be approximated:

$$\frac{1}{N_k}||\mathbf{y}_k-\hat{\mathbf{f}}^{-k}(\mathbf{X}_k)||_2^2\approx\frac{1}{N_k}\sum_{i=1}^{N_k}\left[\frac{y_{ki}-\hat{f}(\mathbf{x}_{ki})}{1-S_{ki}}\right]^2 \tag{40}$$

where S_{ki} is the *i*th diagonal element of $\mathbf{S}_k$.

Through Fast CV method, we can run Mixed-Ridge model really fast. Fast CV algorithm helps us do large scale quick serch for parameters and SNP, increasing efficiency tens of times. We can quickly test many different SNP and parameters. The main algorithm of Mixed-Ridge with Fast CV are as follows:

Algorithm 2 Mixed-Ridge with Fast CV

1: Step 1. Randomly select 100,000 SNP from whole SNP. Do SVD(Singular-value decomposition).

$$X=USV^T \tag{41}$$

Rank singular value in descending order. Cut off them with the threshold 0.1. Retain the corresponding eigenvector in U as U_1, then calculate$\Lambda_1 = U_1S^{\frac{1}{2}}$

2: Step 2. Select 200 SNP from the whole SNP randomly as X_1. Concatanate Λ_1 and X_1.

3: Step 3. As shown in Algorithme 1. We will optimize the function:

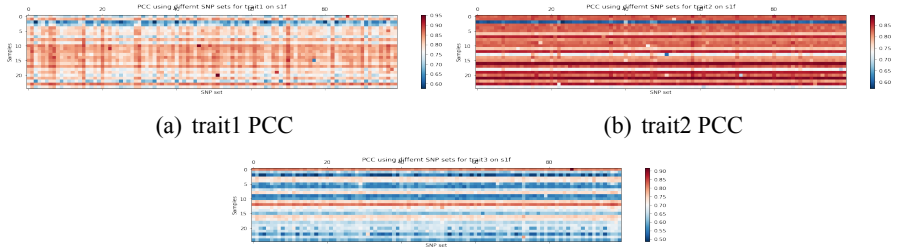
$$\mathbf{y}=X_1\vec{\boldsymbol{\beta}_1}+\gamma(\Lambda_1\vec{\boldsymbol{\beta}_2})+\boldsymbol{\beta}_0+\epsilon \tag{42}$$

And the U_1 and X_1 are from step2.

We do Mixed-Ridge following Algorithm 1's steps. use grid search or gredient descent method to optimize and search for the bestγ andλ on our designed cross validation dataset(to best predict s1f area). By using our Fast CV method, we can do this steps very quickly. We do 25-fold cross validation. For each fold, we use offsprings that share a male parent for test and the remaining samples for training.

4: Step 4. Repeat step 2 and 3 200 times. Each time use different X_1.

Through Fast CV, we do 25-fold cross validation for every SNP choice to select best parameters γ andλ. The time cost is only 2 minutes. The calculation is parrallel so our Mixed-Ridge algorithm can reach very fast speed for feature selection and prediction. Through large scale search, we calculate different SNP choices's PCC on validation set. For example, as shown in figure 6, we divide the dataset in the form of s1f for cross validation and calculate PCC on validation set.



(a) trait1 PCC (b) trait2 PCC

(c) trait3 PCC

Figure 1: Summary of PCC on s1f-form cross validation

0.4 Prediction for new samples

As for new samples we don't have before, we use the following method to transform the SNP to Λ:

G is the matrix of 4754 samples *100000 SNP.We have:

$$K = GG^T = USU^T = \Lambda\Lambda^T \tag{43}$$

We can have that G = $US^{\frac{1}{2}}V^T$ from SVD. And when we have a new sample V, we can do GV = $US^{\frac{1}{2}} = \Lambda$. Then we can use the Mixed ridge model to predict as well as the competitions's test set sample.