Applying Machine Learning To Maize Traits Prediction

Binbin Shi
Department of Life Science
Tsinghua University
Beijing, 100084
ltbyshi@gmail.com
,

Xupeng Chen
Department of Life Science
Tsinghua University
Beijing, 100084
xp-chen14@mails.tsinghua.edu.cn

Abstract

Heterosis is the improved or increased function of any biological quality in a hybrid offspring. We have studied yet the largest maize SNP dataset for traits prediction. We develop linear and non-linear models which consider relationships between different hybrids as well as other effect. Specially designed model proved to be efficient and robust in prediction maize's traits.

1 Introduction

1.1 Heterosis and Materials

Heterosis, hybrid vigor, or outbreeding enhancement, is the improved or increased function of any biological quality in a hybrid offspring. Nearly all field corn (maize) grown in most developed nations exhibits heterosis. Modern corn hybrids substantially outyield conventional cultivars and respond better to fertilizer.

Scientists and breeding experts have spent more than one hundred years on finding high-yielding corn varieties. The traditional way is to do filed experiment to find the better hybrids. It is time consuming and expensive. Thanks to the development of sequencing, scientists have developed several statistical and computational models to predict quantitative traits using genomic data including SNP (Single Nucleotide Polymorphism). Using computational model to predict and determine which hybrids are better can save a lot of time and expenses on breeding. However, the existing model still have many problems to predict traits accurately due to some reasons including lack of big data and their own defect.

Scientists recently accomplish the largest sequencing project in maize. This work provides the largest datasets available to help to develop a model. The project design and basic information of materials can be seen in Figure 1. We have 30 elite inbred lines as male and 1404 lines (from 24 best lines in China) as female. After hybridization there are 42,120 hybrids. We have the whole genome sequence data of all the 1404 plus 30 parents lines and 6210 hybrids. These data can be used as training set of the model. Also there are 2808 hybrids from the hybridization of two males and 1404 females as test sets. The general goal is using processed SNP data to predict each maize's three traits: height, flowering time and yield.

1.2 Trait Predicting Model in Breeding

There is an important problem in breeding: nature versus nurture. For phenotype such as height of plants or intelligence quotient of a person, how much of the phenotype is inherited and how much is determined by environment is hard to determine. This question was made precise by Fisher and Wright almost a century ago: Given observations of a phenotype from a population of individuals, what is the fraction of variance of the phenotype that is caused by inherited factors relative to the total variance of the phenotype due to both inherited and environmental factors? This fraction is termed heritability. For example, a phenotype in a population where environmental factors have large variation will have a

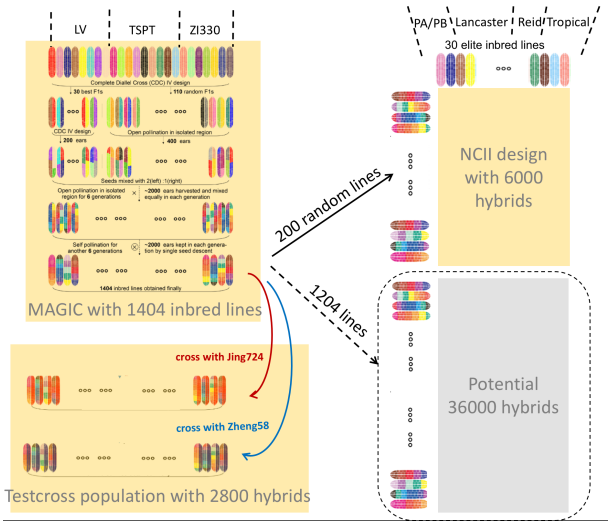


Figure 1: Materials: training and test datasets from hybrids of parental lines

smaller heritability than in an otherwise similar population where environmental factors have a small variation.

Over the years, many approaches have been developed to estimate heritability from data. The most successful Whole-Genome Regression (WGR) models are based on Linear or Bayesian model. For example, one of the approach uses a linear mixed model (LMM), a form of multivariate regression of the genomic and environmental factors on the phenotype. It can use genomic data like SNP to predict the quantitative traits.

We can add some more designs to the model to do more work. For example the model can also be applied to predict the traits considering environment conditions. Another important question is how to determine the most important SNPs associated with traits. Using the weights of feature in the model and significance test, we can pick up the most significant SNP by its p-value. Then we can find the related genes or non-coding region. However, the current methods usually considers the feature(SNP) seperately instead considering them jointly.

1.3 Machine Learning Methods

Machine Learning is a field of computer science that gives computer systems the ability to ”learn” and ”recognize” the hidden pattern of a specific problem. The specially designed model progressively improve performance on a specific task with the help of data. The model doesn't need to be being explicitly programmed, it relies on the data(usually big) to learn the hidden rules. With the development of machine learning methods (like deep learning) and the data accumulation, machine learning has becoming much more powerful in solving problems in many fields. By using a few prior knowledge and enough data, machine learning models have outperform many rule-based models in different fields. The breeding fields have used linear model for a long time for some reasons: using complex machine learning model to process very high dimension data is time-consuming, linear model is easy to explain, it is easier to design and apply.

However, current models still have some drawbacks. A not specially designed model performs bad on some hard to predict samples. A specially designed model has some chance to perform good on the hard to predict sample, but also has unbearable high variance in prediction. Current models including breeding models and general machine learning models can't overcome all the drawbacks to achieve high predicting accuracy as well as high robustness. Although we have collected the largest datasets available, it is still not big enought to train a machine learning model without further design.

We hope to construct a machine learning model doesn't rely a lot on the special rules of a specific field (like breeding). After many exploration on the project, we have discovered that machine learning model without special design fails to predict well on some samples. When examine it carefully, we find out that the model generally predict the traits by learning the relationship of parents. It fails to predict well on a relatively distantly related hybrids. This kind of overfitting have been considered to design a special model considering causal SNP (several hundreds of very important SNP), parents relationship and environmental effect together(Like Linear mixed model). But such kind of model also has serious drawbacks: since the training sample is very small, it is hard to select the causal SNP. So by chance the model can select a set of SNPs very suitable for prediction, but most of the time the prediction results turn out to be very bad. It is also very hard to solve such kind of overfitting problem.

Our main innovation is to develop a machine learning model which can overcome the problems discussed in the previous paragraphs. The basic and most important feature of the model is to predict better on nearly all

samples. Then it can be further designed to solve other problems. The general goals of our research are as follows:

- Constructing a robust Machine Learning model to predict three traits of 2808 maizes more accurately and find hybrids with heterosis.
- Using the robust model to find casual SNPs from millions of SNPs and its related gene.
- Using environment related data to predict environment variance and find robust hybrids.

2 Experimental Design

We have tried several machine learning models and designed another two models. One is an improved linear model based on currently the best model, another is a non-linear model aims to solve the small sample problem. Since the two newly-designed models still have some drawbacks in solving the problem, we also propose some other methods to solve the problems.

2.1 Simply Applied Machine Learning Model

Generally a machine learning problem can be splitted into two parts: (1) Feature Selection and Dimension Reduction. (2) Classification or Regression. We have tried several feature selection methods and machine learning models (without special design). They achieve good performance in average, but fails to predict on distantly related hybrids. These kinds of model easily overfits and have no power to predict a new material whose parents not shown in the training set before(which means the model is lack of generalization ability).

We use feature selection methods including Analysis of variance(ANOVA), Random projection and some GWAS methods to pick up features. We do not consider traditional breeding methods because it is merely the linear combination and transformation of the data, which can also be considered by the machine learning model. Here we use the characterization of the data(Normally distribution) to optimize the ANOVA algorithm and achieve higher computational speed.

We simply apply the existing machine learning models to predict the traits using SNP data. The models include Random Forest, Support Vector Machine Regression(SVR), Gradient Boosting(XGBoost), Gaussian Process Regression, K Nearest Neighbor(KNN) and Multilayer Perceptron(MLP). We do 10-fold cross validation to tune the model and find best parameters of each model, then we test the model on seperate test sets.

2.2 Mixed-Ridge Model

Since we found out the general machine learning model fails to predict well on distantly related hybrids, we turned to the classic model in breeding field to find some insipiration. We found out that the classic model considers the problem of only relying on parents relationship. The model aims to consider some casaul SNPs, parents relationship and environment effect together. It has beed proved that several hundreds of SNP data with data represents relationship is enough to make a very good prediction. And when the sample is distantly related to the training set(especially some hybrids whose parents are both unknown), the model will rely more on the causal SNP data to predict. So the classic model partially solves the problems of predicting samples having distant relationship. But it is still hard to pick the causal SNP since the training sample isn't enough. We improve the linear mixed model in several ways to achieve better results.

2.2.1 Improved From LMM

FaST-LMM (originated from LMM) is the most powerful model using SNP or other high dimension data to predict quantitative trait. 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. It also has other disadvantages which leads to high variance and overfitting.

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. Whats more, it is more convenient to use L1 or L2 regularizations in linear model rather than probabilistic model.

Here is the simple comparison of FaST-LMM and Mixed-Ridge to show the improvement

- F-L optimize parameters by optimizing h^2, M-R directly optimize β and γ.
- F-L is time consuming, M-R generate Λ_1 more quickly to use more SNPs.
- F-L has no regularization when optimizing, M-R has L_2 regularization to reduce colinearity and regularize parameters to achieve good results.
- F-L's feature selection isnt enough, M-R use a newly designed Fast-CV algorithm to select feature efficiently.

2.2.2 Principle and Similar Effect Provement

Principle

Here is the general explanation of the 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 $100000 \cdot 270$. 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.

The details of algorithm can be seen in appendices.

Similar Effect Provement

If we can prove that our Mixed-Ridge model have the similar effect 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. Omits the details of deduction, we can have the expression of Mixed-Ridge and FaST-LMM's mean value.

M - R:

$$\vec{y^*} = \vec{x^*}^T \begin{bmatrix} X & \gamma\Lambda \end{bmatrix} (XX^T + \gamma\Lambda\Lambda^T + \alpha I)^{-1}\vec{y} \tag{6}$$

LMM:

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

Clearly they have the same form, so we

prove that the two models have the similar effect. Note that since the two models have the similar effect, the newly designed model still has problems to avoid overfitting.

2.2.3 Mixed-Ridge with Fast-CV

One of the most important problem is to find the best SNP set(casual SNPs) to use. The FaST-LMM model use each single SNP to predict and evaluate its significance. The methods isn't suitable to find a satisfying set of SNPs. Here we use the feature of linear model to design a special SNP-selection methods.

First, it is worth noticing that the data isn't independent, different regions of datasets have different prediction difficulty. Also further tests show that the traits is more dominated by male parent. So we can do some special dataset split instead of totally random(Shown in Figure2). The linear model also has some special quality : the weights of different features are independent. So we can only fit the model to the data once, and we can get results of different "cross validation" at once, no matter how you change the dataset splitting way. It allows us to do dozens or even more cross validation very quickly instead of fitting the data each time the training set changes. So we can select the causal SNP by cross validation in a wide range.

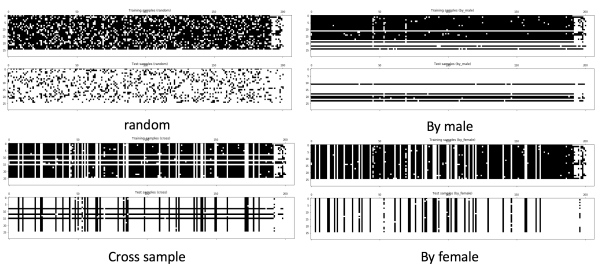


Figure 2: Different cross validation ways

Although Mixed-Ridge makes several improvement compared to Mixed-Ridge, it turns out the results aren't satisfying enough. After hundreds of testing, we find out the Linear model, although specially designed, still fails to sovle the overfitting problems totally.

2.3 Non-Linear Model

2.3.1 Metric Regressor

We design a new network for small dataset prediction problem. It has some advantages to avoid overfitting. We also use some methods to reduce the complexity and speed up algorithms. However, it still needs more optimization considering the computation complexity. In ridge regression, the mean squared error (MSE) and L_2 regularization term is minimized:

$$\text{minimize}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2 \qquad (7)$$

For datasets with small number of samples, the model easily overfits to the data and generalizes poorly.

Here we omits the details of formula derivation and only show the overall training procedure. It can be viewed as minimizing the expected mean squared error on the test dataset:

$$\text{argmin}_{\mathbf{A}}\mathbb{E}_{(\mathbf{X}_1,\mathbf{y}_1,\mathbf{X}_2,\mathbf{y}_2)}\|\mathbf{y}_2 - \hat{\mathbf{y}}_2(\mathbf{X}_1,\mathbf{y}_1)\|_2^2 \qquad (8)$$

We use Figure 3 to illustrate how Metric Regressor works. In general Metric Regressor works by learning a distance metric for regression on small samples.

Note that the overall time complexity for computation of the gradients of MSE is approximately $O(N_1^3 + N_1(N_1 + N_2)pq)$. This is much more computationally extensive than ordinary ridge regression, but can be reduced

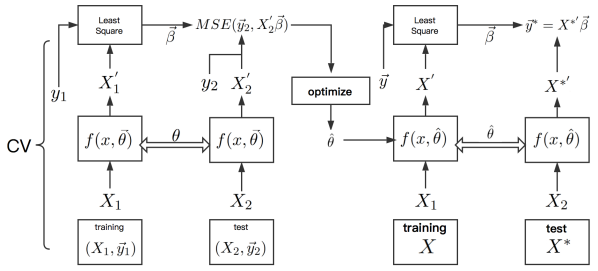


Figure 3: Metric Regressor Model

by choosing a smaller N_1 and N_2. The time complexity also grows proportionally with the number of features p and low-rank dimension q. For datasets with a large number of features, the time complexity can be reduced by use a sparse weight matrix for **A**. A certain portion of elements in **A** can be set to zeros which can be ignored during feature transformation and evaluation of the gradients with respect to **A**.

In the training process, a small number of samples (usually fewer than 20) is taken from the whole dataset and then splitted into a training dataset and a test dataset. Then a ridge regression model is fitted on the training dataset and the gradients of the mean squared error is evaluated on the test dataset. The weight matrix **A** is optimized by stochastic gradient descent (SGD) or other variants.

Other transformations

To speed up the algorithm, we use two other functions for $f(x_j, \theta)$

The original one:

$$f(x_j, \theta) = XA$$
$$A = \theta, X \in \mathbb{R}^{n\times p}, A \in \mathbb{R}^{p\times q}$$

It is dense and takes a little long time to run. So we design another two functions:

$$f(x_j, \theta) = X(A \circ B)$$
$$A = \theta, X \in \mathbb{R}^{n\times p}, A \in \mathbb{R}^{p\times q}$$

$$f(x_j, \theta) = \vec{\omega} \circ \vec{x} \qquad \vec{\omega} \in \mathbb{R}^p, \vec{x} \in \mathbb{R}^p$$
$$f(x_j, \theta) = [\vec{\omega} \circ \vec{x}]_j = \omega_j x_j$$

B is used for producing sparse matrix to reduce running time. We use these two functions to speed up our algorithms.

The algorithm details of Metric Regressor can be seen in appendices.

2.3.2 Other Designs

We propose some ideas to find causal SNPs better. Use some similar ideas from Mixed-Ridge, the key point of prediction is the model shouldn't relies on relationship, but also needs some causal SNPs to predict. So we can calculate the residual of hybrids (which is called SCA in breeding). This will eliminate the relationship effect. And the model should use features to predict residuals well. We have tested all the general machine learning models and found that the models have no ability to predict residuals at all, which is the reason why these models will overfit on distant related samples. Since we already know the parent traits value, the final prediction(predicted-residuals plus GCA part) will be easier. But it won't help to find heterosis, so we still have to find a model to predict residuals.

In other words, if we can find features to predict residuals well, we can find some important SNPs related to quantitative traits. One way is to design an "Expert Model": it can combine basic models and use different

weights for different samples when predicting.

We will also try Multiple-Trait combined prediction. At first we will test the correlation of different traits. (there are 20 total traits). The highly correlated traits may be predicted jointly. Since multiple traits can provide more information. It is another way to augment the datasets. Also a feature good for multiple traits may be more important than considering the traits seperately.

textbfBetter Feature Selection Methods Normal feature selection methods have the same drawbacks with machine learning model in breeding problems. Feature selection based on FaST-LMM consider SNPs seperately and proved to be useless. We propose to use another feature selection methods based on Mixed-Ridge and Lasso. It will consider feature combination by test the SNPs by sets. Also since Mixed-Ridge can eliminate genetic similarities, it may have the ability to find causal SNPs which we need.(Figure 4)

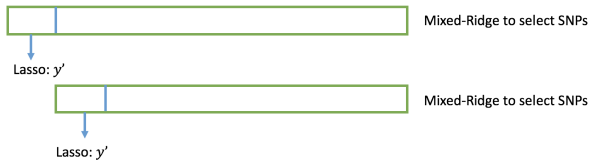


Figure 4: Feature Selection using Mixed-Ridge

Each iteration we use Mixed-Ridge to fit on a subset of whole SNPs and predict. Then we use F-test to find features with lower p-value. Then we use Lasso to construct a residual model. In the next iteration, Mixed-Ridge wil predict the residual and also leaves some important features. The iteration will stop when MSE(mean square error) doesn't change. We can apply some methods to speed up the computation. In this methods, the number of SNPs in each iteration and the regularization parameter in Lasso are hyperparameters to optimize.

2.3.3 Environment Model

The goal of environment model is to predict phenotype variance in different environment and find robust hybrids in different locations. Since there are only 5 data points (5 locations). We should consider by samples. Each sample from 6210 hybrids have 5 locations traits. So the feature is a three dimensional tensor instead of a matrix. We can try to use tensor decomposition for multiple-location traits variance prediction.

3 Expeiment Results

3.1 Collected Data

We have collected several kinds of data now, and there will be some more data to use in future.

Data collected now:

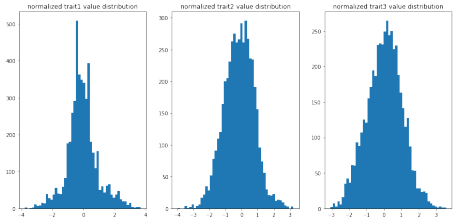
1. 6210 hybrids:
- 1*a*) SNP data using new preprocessing pipeline. Each sample has 5.88M SNP point.
- 1*b*) Traits: DTT, PH, EW(height, flowering time and yield). continuous and normaly distributed.

2. 30+1404 parents:
- 2*a*) SNP data using new preprocessing pipeline. Each sample has 5.88M SNP point.
- 2*b*) Traits: DTT, PH, EW(height, flowering time and yield). continuous and normaly distributed. (Figure 5)

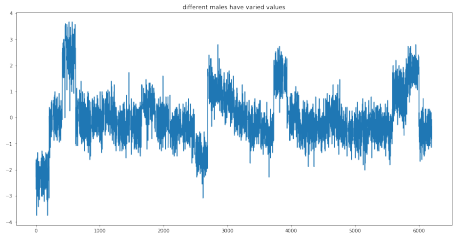
3. Environment data: 5 locations hybrid and parents traits (Figure 6)

Data to be collect in future:

1. All 30*1404 hybrid:



(a) Continous trait value after normalization



(b) Hybrids from different males have varied trait values

Figure 5: Traits overview

1*a*) SNP data using new preprocessing pipeline. Each sample has 5.88M SNP point.

1*b*) Traits: DTT, PH, EW(height, flowering time and yield). continuous and normaly distributed.

1*c*) Whole genome data

2. Environment condition data

3. RNA-seq data:

3*a*) Parents

3*b*) Hybrids

3.2 Prediction Results

Previously we do training and test on datasets shown in Figure 7.

We have analyzed the genral machine learning model by doing cross validation 1,000 times to test. Here we show one of the models' results (Figure 8). The criterion we use to evaluate the prediction performance is pearson correlation coefficient: $pcc = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$. It can be seen in figure that a general machine learning model can achieve good performance in average, but it fails to predict the relatively distantly related hybrids, which are crucial in our research.

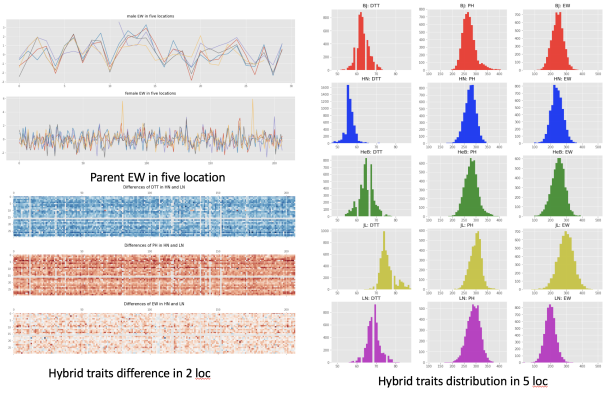


Figure 6: Environment data

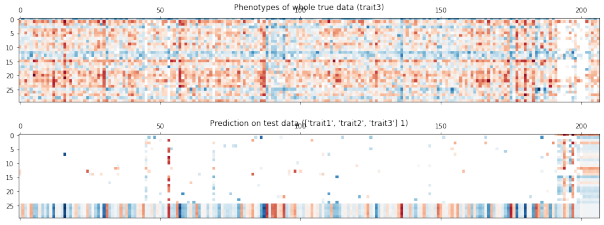
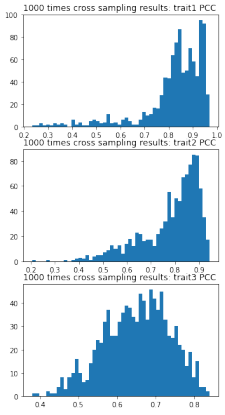


Figure 7: Previous training and test datasets

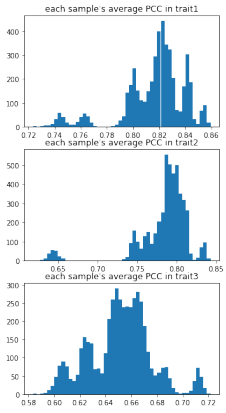
References

[1] Jer-Ming Chia, Chi Song, Peter J Bradbury, Denise Costich, Natalia De Leon, John Doebley, Robert J Elshire, Brandon Gaut, Laura Geller, Jeffrey C Glaubitz, et al. Maize hapmap2 identifies extant variation from a genome in flux. *Nature genetics*, 44(7):803, 2012.

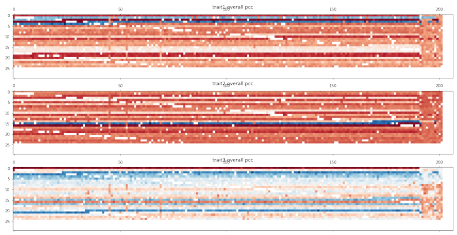
[2] Arnis Druka, Elena Potokina, Zewei Luo, Ning Jiang, Xinwei Chen, Mike Kearsey, and Robbie Waugh. Expression quanti-



(a) Average PCC in 1000 times cross validation



(b) Average PCC of each samples after 1000 times cross validation



(c) Average PCC of each samples aranged according to parents

Figure 8: 1000 times cross validation using Random Projection and GPR

tative trait loci analysis in plants. *Plant biotechnology journal*, 8(1):10–27, 2010.

[3] Nicholas A Furlotte and Eleazar Eskin. Efficient multiple trait association and estimation of genetic correlation using the matrix-variate linear mixed-model. *Genetics*, pages genetics–114, 2015.

[4] David Heckerman, Deepti Gurdasani, Carl Kadie, Cristina Pomilla, Tommy Carstensen, Hilary Martin, Kenneth Ekoru, Rebecca N Nsubuga, Gerald Ssenyomo, Anatoli Kamali, et al. Linear mixed model for heritability estimation that explicitly addresses environmental variation. *Proceedings of the National Academy of Sciences*, 113(27):7377–7382, 2016.

[5] Christoph Lippert, Jennifer Listgarten, Ying Liu, Carl M Kadie, Robert I Davidson, and David Heckerman. Fast linear mixed models for genome-wide association studies. *Nature methods*, 8(10):833, 2011.

[6] Christian Riedelsheimer, Angelika Czedik-Eysenberg, Christoph Grieder, Jan Lisec, Frank Technow, Ronan Sulpice, Thomas Altmann, Mark Stitt, Lothar Willmitzer, and Albrecht E Melchinger. Genomic and metabolic prediction of complex heterotic traits in hybrid maize. *Nature genetics*, 44(2):217, 2012.

[7] Frank Technow, Tobias A Schrag, Wolfgang Schipprack, Eva Bauer, Henner Simianer, and Albrecht E Melchinger. Genome properties and prospects of genomic prediction of hybrid performance in a breeding program of maize. *Genetics*, pages genetics–114, 2014.

[8] Matthias Westhues, Tobias A Schrag, Claas Heuer, Georg Thaller, H Friedrich Utz, Wolfgang Schipprack, Alexander Thiemann, Felix Seifert, Anita Ehret, Armin Schlereth, et al. Omics-based hybrid prediction in maize. *Theoretical and Applied Genetics*, 130(9):1927–1939, 2017.

[9] Jian Yang, Jian Zeng, Michael E Goddard, Naomi R Wray, and Peter M Visscher. Concepts, estimation and interpretation of snp-based heritability. *Nature genetics*, 49(9):1304, 2017.

Appendices

A Algorithm

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:

$$\mathbf{y} = X_1\vec{\boldsymbol{\beta}}_1 + \gamma(\Lambda_1\vec{\boldsymbol{\beta}}_2) + \boldsymbol{\beta}_0 + \epsilon \tag{9}$$

To get a better prediction of $\mathbf{y}$, we define a loss function like below:

$$\min\|\mathbf{y}-X_1\vec{\boldsymbol{\beta}}_1-\gamma(\Lambda_1\vec{\boldsymbol{\beta}}_2)-\boldsymbol{\beta}_0\|_2^2+\lambda\|\boldsymbol{\beta}_1\|_2^2+\lambda\|\boldsymbol{\beta}_2\|_2^2 \tag{10}$$

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:

$$\begin{aligned} \hat{\vec{\boldsymbol{y}}}_1^{(t)} &= X_1\hat{\vec{\boldsymbol{\beta}}}_1^{(t)} + \beta_0 \\ \hat{\vec{\boldsymbol{y}}}_2^{(t)} &= \Lambda_1\hat{\vec{\boldsymbol{\beta}}}_2^{(t)} \end{aligned} \tag{11}$$

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

$$\min\|\hat{\vec{\boldsymbol{y}}} - [(1-\gamma)\hat{\vec{\boldsymbol{y}}}_1^{(t)} + \gamma\hat{\vec{\boldsymbol{y}}}_2^{(t)}]\|_2^2 \tag{12}$$

The function is equal to:

$$\min\|(\hat{\vec{\boldsymbol{y}}} - X_1\hat{\vec{\boldsymbol{\beta}}}_1^{(t)} - {\beta_0}^t) + \gamma(\hat{\boldsymbol{\beta}}_2^{(t)} - X_1\hat{\vec{\boldsymbol{\beta}}}_1^{(t)})\|_2^2 \tag{13}$$

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.

Algorithm 2 Mixed-Ridge with Fast CV

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

$$X = USV^T \tag{14}$$

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$\Lambda_1 = U_1 S^{\frac{1}{2}}$

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:

$$\mathbf{y} = X_1\vec{\boldsymbol{\beta}}_1 + \gamma(\Lambda_1\vec{\boldsymbol{\beta}}_2) + \boldsymbol{\beta}_0 + \epsilon \tag{15}$$

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.

Algorithm 3 Metric Regressor

1: Step 1. As in the left network, we do cross validation on training set to get

$\hat{\theta}$ for prediction. We have four ways to divide training set for CV. We use $(P(X_1, \vec{y}_1, X_2, \vec{y}_2))$ to stand for division. The four ways include three ways in the form of final test set:P_{s0}, P_{s1m}, P_{s1f} and one common dividing method:P_{rand} for random selection. For example $P_{s1f}(X_1, \vec{y}_1, X_2, \vec{y}_2|N_1, N_2)$ stands for dividing the training set in the form of s1f, N_1, N_2 are sample size of training and test sets. Considering the complexity of the algorithm we set N less than 100.

2: Step 2. We use the function f to calculate $X_1^{'}, X_2^{'}$ from X_1, X_2. They share the same parameters θ, the function $f(x_j, \theta)$ is defined as follows:

$$f(x_j, \theta) = XA \qquad A = \theta, X \in \mathbb{R}^{n\times p}, A \in \mathbb{R}^{p\times q} \tag{16}$$

3: Step 3. We use $X_1^{'}, y_1$ to calculate the parameter $\vec{\beta}$,the method is least square. Then $\vec{\beta}$ can be used for $X_2^{'}, y_2$

4: Step 4. Calculate the mean square error of $\vec{y}_2, X_2^{'}\vec{\beta}$, i.e. $MSE(\vec{y}_2, X_2^{'}\vec{\beta})$. Through optimization we can have parameter $\hat{\theta}$ by minimizing the expected mean square error:

$$\hat{\vec{\theta}} = \arg\min_{\vec{\theta}} \mathbb{E}_{(X_1,\vec{y}_1,X_2,\vec{y}_2)} MSE(\vec{y}_2, X_2^{'}\vec{\beta}) \tag{17}$$

We use gradient descent as optimization method:

$$\frac{\partial MSE(\vec{y}_2, X_2^{'}\vec{\beta})}{\partial \vec{\theta}} \tag{18}$$

We can use methods like SGD, Adam, Adagrad, Deltaadam. to get $\hat{\theta}$

5: Step 5. The right part are similar to the left. We use the whole known samples as training set and unknown samples as test set. The $\hat{\theta}$ in step 4 will be used to calculate $\vec{\beta}$, and then $\vec{y}^*$

B Formula

B.1 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.

B.1.1 LMM

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

$$P(\vec{y} \mid X, \sigma_g^2, K) = \int N(\vec{y}|X\vec{\beta}; \sigma_g^2 K + \sigma_e^2 I)\ p(\vec{\beta}|\ )\ d\vec{\beta} \tag{19}$$

$$= N(\vec{y} \mid 0; \sigma_g^2 K + \sigma_e^2 I) \tag{20}$$

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 $m(z^*)$:

$$\begin{aligned} m(z^*) &= k^{*T} C^{-1} \vec{y} \\ \vec{z^*} &= \begin{bmatrix} \vec{x^*} & \vec{a^*} \end{bmatrix} \end{aligned} \tag{21}$$

a_i and x_i are elements of $\mathbf{X}$:

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

$$k_j^* = \sigma_g^2 k(\vec{z^*}, \vec{z_i}) + \vec{x^*}^T \vec{x_i} \tag{22}$$

$$\begin{aligned} k &= \frac{1}{N} G G^T \\ k(\vec{z^*}, \vec{z_i}) &= \vec{g^*} \vec{g_i} \end{aligned} \tag{23}$$

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

$$m(z^*) = \vec{k^*}(\sigma_g^2 K + \sigma_e^2 I + \alpha X X^T)^{-1} \vec{y} \tag{24}$$

$$= \begin{bmatrix} \vec{x^*} \\ \vec{d} \end{bmatrix}^T \begin{bmatrix} \alpha X & \sigma_g^2 \Lambda \end{bmatrix} (\sigma_g^2 K + \sigma_e^2 I + \alpha X X^T)^{-1} \vec{y} \tag{25}$$

B.1.2 Mixed-Ridge

We should optimize the following function:

$$||\mathbf{y} - X_1 \vec{\boldsymbol{\beta}_1} - \gamma(\Lambda_1 \vec{\boldsymbol{\beta}_2}) - \boldsymbol{\beta}_0||_2^2 + \lambda||\boldsymbol{\beta}_1||_2^2 + \lambda||\boldsymbol{\beta}_2||_2^2 \tag{26}$$

We have:

$$\begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}^T = (X^T X + \gamma \Lambda^T \Lambda + \alpha I)^{-1} \begin{bmatrix} X & \Lambda \end{bmatrix}^T \vec{y} \tag{27}$$

$$= \begin{bmatrix} X & \Lambda \end{bmatrix} (X X^T + \gamma \Lambda \Lambda^T + \alpha I)^{-1} \vec{y} \tag{28}$$

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

$$\vec{y^*} = \vec{x^*}^T \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}^T \tag{29}$$

$$= \vec{x^*}^T \begin{bmatrix} X & \gamma \Lambda \end{bmatrix} (X X^T + \gamma \Lambda \Lambda^T + \alpha I)^{-1} \vec{y} \tag{30}$$

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

$$\begin{aligned} &\textbf{M - R:}\ \vec{y^*} = \vec{x^*}^T \begin{bmatrix} X & \gamma \Lambda \end{bmatrix} (X X^T + \gamma \Lambda \Lambda^T + \alpha I)^{-1} \vec{y} \\ &\textbf{LMM:}\ m(z^*) = \begin{bmatrix} \vec{x^*} \\ \vec{d} \end{bmatrix}^T \begin{bmatrix} \alpha X & \sigma_g^2 \Lambda \end{bmatrix} (\sigma_g^2 K + \sigma_e^2 I + \alpha X X^T)^{-1} \vec{y} \end{aligned} \tag{31}$$

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

B.2 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:

$$\text{minimize}||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_2^2 \tag{32}$$

The solution to ridge regression is:

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

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

$$\hat{\mathbf{y}} = \mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \tag{34}$$

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

$$\hat{\mathbf{y}} = \mathbf{S}\mathbf{y} \tag{35}$$

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:

$$\text{LOOCV}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N}[y_i - \hat{y}^{-i}(x_i)]^2 \quad (36)$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \hat{f}(x_i)}{1 - S_{ii}}\right]^2 \quad (37)$$

where S_{ii} is the i-th diagonal element of $\mathbf{S}$.

The GCV approximation of the leave-one-out cross-validation is:

$$\text{GCV}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \hat{f}(x_i)}{1 - \text{trace}(\mathbf{S})/N}\right]^2 \quad (38)$$

where trace($\mathbf{S}$) is the effective number of parameters.

For k-fold cross-validation, the cross-validation error is:

$$\text{CV}(\hat{f}) = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{N_k}\sum_{i=1}^{N_k}[y_{ki} - \hat{f}^{-k}(\mathbf{x}_{ki})]^2 \quad (39)$$

where N_k is the number of test samples in the k-th part of the dataset. $(\mathbf{x}_{ki}, y_{ki})$ is the ith 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:

$$\mathbf{S}_k = \mathbf{X}_k\mathbf{A}^{-1}\mathbf{X}_k^T \quad (40)$$

where $\mathbf{A} = \mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$

The estimate of kth part of the test samples by the function fitted on the full dataset is:

$$\hat{\mathbf{f}}(\mathbf{X}_k) = \mathbf{X}_k\mathbf{A}^{-1}\mathbf{X}^T\mathbf{y} \quad (41)$$

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

$$\hat{\mathbf{f}}^{-k}(\mathbf{X}_k) = \mathbf{X}_k(\mathbf{X}^T\mathbf{X} - \mathbf{X}_k^T\mathbf{X}_k + \lambda\mathbf{I})^{-1}(\mathbf{X}^T\mathbf{y} - \mathbf{X}_k^T\mathbf{y}_k) \quad (42)$$

$$= \mathbf{X}_k(\mathbf{A} - \mathbf{X}_k^T\mathbf{X}_k)^{-1}(\mathbf{X}^T\mathbf{y} - \mathbf{X}_k^T\mathbf{y}_k) \quad (43)$$

Following the properties of inverse of a block matrix:

$$(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} \quad (44)$$

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

$$(\mathbf{A} - \mathbf{X}_k^T\mathbf{X}_k)^{-1} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{X}_k(\mathbf{I} - \mathbf{X}_k\mathbf{A}\mathbf{X}_k^T)^{-1}\mathbf{X}_k^T\mathbf{A}^{-1} \quad (45)$$

$$= \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{X}_k(\mathbf{I} - \mathbf{S}_k)^{-1}\mathbf{X}_k^T\mathbf{A}^{-1} \quad (46)$$

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:

$$\mathbf{y}_k - \hat{\mathbf{f}}^{-k}(\mathbf{X}_k) = [\mathbf{I} + \mathbf{S}_k(\mathbf{I} - \mathbf{S}_k)^{-1}][\mathbf{y}_k - \hat{\mathbf{f}}(\mathbf{X}_k)] \quad (47)$$

The cross-validated squared error of the kth part of the dataset is:

$$\frac{1}{N_k}\|\mathbf{y}_k - \hat{\mathbf{f}}^{-k}(\mathbf{X}_k)\|_2^2 = [\mathbf{y}_k - \hat{\mathbf{f}}(\mathbf{X}_k)]^T\mathbf{B}_k^T\mathbf{B}_k[\mathbf{y}_k - \hat{\mathbf{f}}(\mathbf{X}_k)] \quad (48)$$

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 kth part can be approxi-

mated:

$$\frac{1}{N_k}\|\mathbf{y}_k - \hat{\mathbf{f}}^{-k}(\mathbf{X}_k)\|_2^2 \approx \frac{1}{N_k}\sum_{i=1}^{N_k}\left[\frac{y_{ki} - \hat{f}(\mathbf{x}_{ki})}{1 - S_{ki}}\right]^2 \tag{49}$$

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

B.3 Metric Regressor works by learning a distance metric for regression on small samples

In ridge regression, the mean squared error (MSE) and L_2 regularization term is minimized:

$$\text{minimize}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2 \tag{50}$$

However, for problems with small number of samples and large number of samples, the linear regression is under-determined and there is no unique solution to the ordinary least-squares problem. The L_2 regularization term solves the problem by penalize the regression coefficients to generate unique solution.

For datasets with small number of samples, the model easily overfits to the data and generalizes poorly.

For datasets with a large number of features, we can transform the features into a low-dimensional space by linear combination of original features:

$$\mathbf{x}_i' = \mathbf{A}\mathbf{x}_i \tag{51}$$

where $\mathbf{x}_i$ is a p-dimensional vector and $\mathbf{x}_i'$ is a q-dimensional vector. $\mathbf{A}$ is a $p \times q$ weight matrix.

The whole data matrix $\mathbf{X} \in \mathbb{R}^{N\times p}$ with rows as samples can be transformed to low dimension $\mathbf{X}' \in \mathbb{R}^{N\times q}$:

$$\mathbf{X}' = \mathbf{X}\mathbf{A} \tag{52}$$

Both training samples $\mathbf{X}_1 \in \mathbb{R}^{N_1\times p}$ and test samples $\mathbf{X}_2 \in \mathbb{R}^{N_1\times p}$ by the same set of weights:

$$\mathbf{X}_1' = \mathbf{X}_1\mathbf{A} \tag{53}$$

$$\mathbf{X}_2' = \mathbf{X}_2\mathbf{A} \tag{54}$$

On the transformed training samples $\mathbf{X}_1'$, a ridge regression model can be fitted by minimizing the cost function:

$$\text{minimize}\|\mathbf{y}_1 - \mathbf{X}_1'\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2 \tag{55}$$

The coefficients $\boldsymbol{\beta}$ that minimize the cost function can be written as:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}_1'^T\mathbf{X}_1' + \lambda\mathbf{I}_q)^{-1}\mathbf{X}_1'^T\mathbf{y}_1 \tag{56}$$

From the properties of pseudoinverse, the coefficients can also be written as:

$$\hat{\boldsymbol{\beta}} = \mathbf{X}_1'^T(\mathbf{X}_1'\mathbf{X}_1'^T + \lambda\mathbf{I}_{N_1})^{-1}\mathbf{y}_1 \tag{57}$$

This form can reduce computational complexity because the matrix $(\mathbf{X}_1'\mathbf{X}_1'^T + \lambda\mathbf{I}_q) \in \mathbb{R}^{N_1\times N_1}$ that needs to be inverted is usually much smaller than $(\mathbf{X}_1'^T\mathbf{X}_1' + \lambda\mathbf{I}_{N_1}) \in \mathbb{R}^{q\times q}$.

The prediction of target variable $\mathbf{y}_2$ is given by:

$$\hat{\mathbf{y}}_2 = \mathbf{X}_2'\mathbf{X}_1'^T(\mathbf{X}_1'\mathbf{X}_1'^T + \lambda\mathbf{I}_{N_1})^{-1}\mathbf{y}_1 \tag{58}$$

$\mathbf{X}_2'\mathbf{X}_1'^T$ and $\mathbf{X}_1'\mathbf{X}_1'^T$ are also called kernel matrices and we can denote them by $\mathbf{K}^*$ and $\mathbf{K}_1$: $\mathbf{K}^* = \mathbf{X}_2'\mathbf{X}_1'^T$ and $\mathbf{K}_1 = \mathbf{X}_1'\mathbf{X}_1'^T$.

Then the prediction of target variable on the test samples can also be written as:

$$\hat{\mathbf{y}}_2 = \mathbf{K}^*(\mathbf{K}_1 + \lambda\mathbf{I}_{N_1})^{-1}\mathbf{y}_1 \tag{59}$$

The objective is to find the best combination of weights that minimizes the mean squared er-

ror on the test samples:

$$\text{argmin}_{\mathbf{A}} L(\mathbf{A}; \mathbf{X}_1, \mathbf{y}_1, \mathbf{X}_2, \mathbf{y}_2) = \frac{1}{2}\|\mathbf{y}_2 - \hat{\mathbf{y}}_2\|_2^2 \tag{60}$$

The gradients of the loss function with respect to $\mathbf{A}$ is given by:

$$\frac{\partial L}{\partial A} = (\hat{\mathbf{y}}_2 - \mathbf{y}_2)^T \frac{\partial \hat{\mathbf{y}}_2}{\mathbf{A}} \tag{61}$$

$$= (\hat{\mathbf{y}}_2 - \mathbf{y}_2)^T \frac{\partial}{\partial \mathbf{A}} (\mathbf{K}^* (\mathbf{K}_1 + \lambda \mathbf{I}_{N_1})^{-1} \mathbf{y}_1) \tag{62}$$

$$= (\hat{\mathbf{y}}_2 - \mathbf{y}_2)^T \tag{63}$$

$$\left(\frac{\partial \mathbf{K}^*}{\mathbf{A}} (\mathbf{K}_1 + \lambda \mathbf{I}_{N_1})^{-1} + \mathbf{K}^* \frac{\partial (\mathbf{K}_1 + \lambda \mathbf{I}_{N_1})^{-1}}{\partial \mathbf{A}} \right) \mathbf{y}_1 \tag{64}$$

Each element of $\mathbf{K}^*$ and $\mathbf{K}$ can be evaluated as a kernel function between two samples:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{A}^T \mathbf{A} \mathbf{x}_j \tag{65}$$

$$[\mathbf{K}_1]_{ij} = k([\mathbf{x}_1]_i, [\mathbf{x}_1]_j) \tag{66}$$

$$[\mathbf{K}^*]_{ij} = k([\mathbf{x}_2]_i, [\mathbf{x}_1]_j) \tag{67}$$

The partial derivative of $k(\mathbf{x}_i, \mathbf{x}_j)$ with respect to $\mathbf{A}$ can be written as:

$$\frac{\partial k(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{A}} = \frac{\partial (\mathbf{x}_i^T \mathbf{A}^T \mathbf{A} \mathbf{x}_j)}{\partial \mathbf{A}} \tag{68}$$

$$= \mathbf{A}(\mathbf{x}_i \mathbf{x}_j^T + \mathbf{x}_j' \mathbf{x}_i'^T) \tag{69}$$

and the derivatives respect to each element in $\mathbf{A}$ is:

$$\left[\frac{\partial k(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{A}} \right]_{kl} = \sum_{m=1}^{q} A_{km}(x_{im} x_{jl} + x_{jm} x_{il}) \tag{70}$$

The partial derivative of the matrix inverse $(\mathbf{K}_1 + \lambda \mathbf{I}_{N_1})^{-1}$ can be evaluated for each element in A:

$$\frac{\partial (\mathbf{K}_1 + \lambda \mathbf{I}_{N_1})^{-1}}{\partial A_{kl}} = (\mathbf{K}_1 + \lambda \mathbf{I}_{N_1})^{-1} \frac{\partial \mathbf{K}_1}{\partial A_{kl}} (\mathbf{K}_1 + \lambda \mathbf{I}_{N_1})^{-1} \tag{71}$$

Note that the partial derivative $\frac{\partial \mathbf{K}_1}{\partial A_{kl}}$ needs to be evaluated for every element of $\mathbf{A}$.

The overall time complexity for computation of the gradients of the mean squared error is approximately $O(N_1^3 + N_1(N_1 + N_2)pq)$. This is much more computationally extensive than ordinary ridge regression, but can be reduced by choosing a smaller N_1 and N_2. The time complexity also grows proportionally with the number of features p and low-rank dimension q. For datasets with a large number of features, the time complexity can be reduced by use a sparse weight matrix for $\mathbf{A}$. A certain portion of elements in $\mathbf{A}$ can be set to zeros which can be ignored during feature transformation and evaluation of the gradients with respect to $\mathbf{A}$.

In the training process, a small number of samples (usually fewer than 20) is taken from the whole dataset and then splitted into a training dataset and a test dataset. Then a ridge regression model is fitted on the training dataset and the gradients of the mean squared error is evaluated on the test dataset. The weight matrix $\mathbf{A}$ is optimized by stochastic gradient descent (SGD) or other variants.

The overall training procedure can be viewed as minimizing the expected mean squared error on the test dataset:

$$\text{argmin}_{\mathbf{A}} \mathbb{E}_{(\mathbf{X}_1, \mathbf{y}_1, \mathbf{X}_2, \mathbf{y}_2)} \|\mathbf{y}_2 - \hat{\mathbf{y}}_2(\mathbf{X}_1, \mathbf{y}_1)\|_2^2 \tag{72}$$