NUANCES OF MACHINE LEARNING IN PLANT BREEDING

Alencar Xavier
Research Scientist, Corteva Biostatics
Adjunct Faculty, Purdue University
Johnston, Iowa, USA 50131
alenxav@gmail.com

April 20, 2021

ABSTRACT

The decision making process in plant breeding revolves around data-driven analytics, where machine learning represents a set of powerful tools that enables extracting useful information from data. However, there is still a lack of understanding about the underlying assumptions of these methods, their strengths and pitfalls. Machine learning has two main branches, supervised and unsupervised learning, that are utilized in routine analysis for breeding. This short review provides an insight on both branches from the plant breeding perspective, and further emphasize technical as aspects of supervised methods. The manuscript covers key topics for the proper deployment of machine learning, such as (a) the concept of target signals, (b) designing cross-validation to understand information gain, (c) general purpose breeding optimizations, (d) inner works of the algorithm implementations, and (e) some limitations of the machine learning framework.

***K*eywords** Statistical learning · Plant breeding · Genomics prediction · Algorithms

1 Introduction

The modern plant breeding relies on our ability to identify and manage superior germplasm to enable genetic improvements overtime. Within the analytical space, machine learning is a power tool for data-driven decision making. Nonetheless, the recipe for a successful breeding analysis lies in the intersection between plant breeding and statistical systems, that is, to understand the problems and tools that provide the solution.

The key motivation for deploying machine learning in the field of genetics aroused for the increase volume of data being generated through genotyping and high-throughput phenotyping, among other omics [1, 2], coupled with external sources of data, such as climate and environment [3, 4]. More data comes with the need for a change in analytical mindset, as traditional statistical method may not be suitable for sizable dataset. The technical issues encountered in the omic realm are commonly related to data dimensionality, cross-validation designs, biases, structured and unbalanced data [5]. More sophisticated methods become necessary to extract meaningful patterns, and a better understanding of such methods is important to ensure the interpretability and quality of the results. Knowing the various types of machine helps breeders and geneticist to decide which method should be used and when, as well as providing an insight on how mechanistic knowledge can be infuse into the machine [6].

In this manuscript we attempt to revisit concepts of machine learning through the lenses of plant breeding. The first section recapitulates how the branches of machine learning are currently applied in plant breeding. In the second section, cross-validation methods are conceptualized in terms of prediction and selection, with emphasis on the genetic signal, genetic information and applications to genomic prediction. The third section discuss the core idea of machine learning, *optimization*, how optimization connects to routine analysis, main types of optimization, and the optimization of breeding operations. The forth section covers technical details of the black boxes of supervised machine learning, as it provides details of algorithmic implementations in simple language. The fifth section revisits some known problems and limitation of machine learning.

2 Types of learning

As a field of research, machine learning is perceived and described differently by the various sciences it comprises. It is expected that computer scientists and engineers tend to described machine learning objectively as an optimization problem [7], whereas statisticians and data scientists prefer contextualizing the machinery as statistical learning [8]. Though it consensus that the two main branches of machine learning are the supervised an unsupervised machine learning. Other branches, including semi-supervised learning and reinforcement learning, fall into the intermediate spectrum where the approach is neither fully supervised or fully unsupervised. The general discrimination between supervised an unsupervised machinery is the availability and use of a response variable. Supervised learning aims to model y as a function of a set of parameters (or *features*), whereas the unsupervised learning goal is to find association patterns within a set of parameters X. The main supervised procedures in plant breeding is the generation of genetic values (BLUPs and GEBVs), and the most common unsupervised procedures include germplasm and environmental classification, and genotype-by-environment biplots. Other examples in plant breeding are provided in Figure 1.

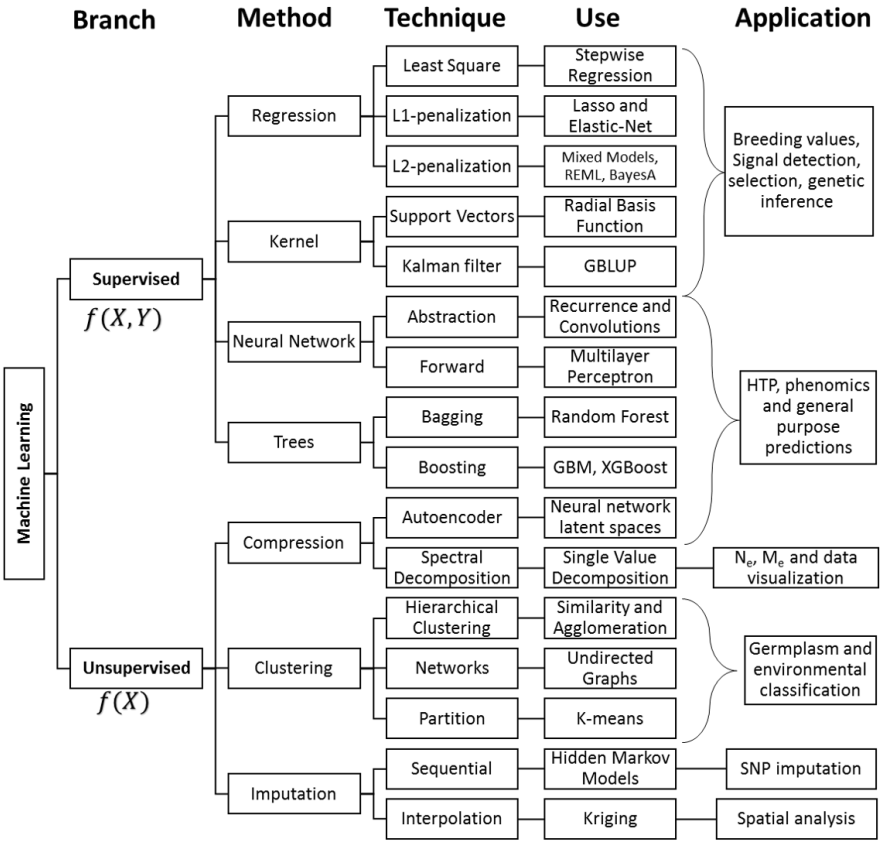

Figure 1: Map of machine learning divisions relevant to plant breeding.

Under the deployment of supervised learning, the breeder or data analyst must mind the target signal that suits the target analysis. Thus, the plant breeder's main concern entails selecting the data, selecting the correct model, and perform adequate validation. These topics are discussed in more detail in the sections "Cross-validation" and "Optimization". In broad terms, data selection regards performing the analysis within meaningful boundaries, as those may alter the interpretation of the results. For instance, breeding values can be generated in different combinations of population, year and location, which will translate in models with different meanings. In addition, different models, parameterization and machines will differ with respect to the signal being captured.

Figure 2 illustrates types of signal that are targeted by different breeding operations. In this case, the selection of lines that will parent upcoming generations must capitalize on additive genetics, as the additive genetic components are heritable and will be transferred to the offspring whereas dominance and epistasis are not. In contrast, models that estimate the genetic merit of lines advancing towards becoming products may benefit from machine learning approaches that capture as much signal as possible, regardless of its genetic nature. For the final product placement, modeling

elements of genotype-by-environment interaction enable targeting lines and hybrids to locations where these perform best or display some other competitive advantage.

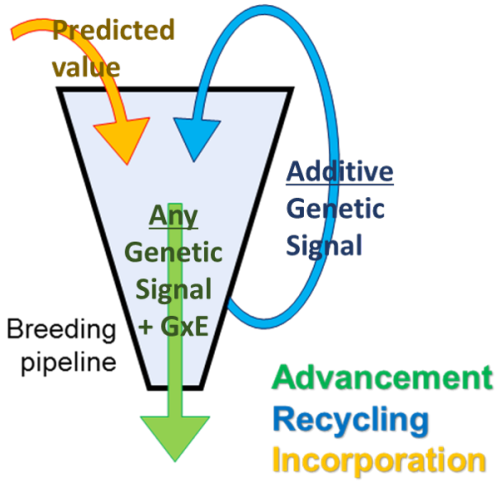

Figure 2: Target genetic signal by breeding operation.

3 Cross-Validation

Cross-validation refers to the procedure of evaluating prediction machines by training models with a partial dataset while holding a fraction of the dataset to be predicted back, with the intuition of assessing the predictive properties of the machine. Main variations of cross validation utilized in the field of plant breeding include:

- K-fold cross-validation.
- Leave-one-out (LOO) cross-validation.
- Holdout cross-validation.

The different cross-validation methods are used for different purposes. In the context of genomic prediction in plant breeding, the k-fold scheme describes the scenario where the dataset is split at random and X% of the data predicts (100-X)%. It provides a general assessment of the predictive power, it is used for comparing methods, and it gives a sense for the prediction on new individuals for an observed set of environments.

Cross-validation schemes based on LOO are useful to assess the structure of the data, including the prediction of unobserved population and genotype-by-environment interactions. It mitigates structural contamination the contamination that random data splits do not account for. For instance, K-fold cross-validation analysis with multiple combinations of traits and environments lead to the multivariate contamination [9], where the genotype-by-environment interactions are captured in the training data through correlated traits, causing the models to yield unrealistic prediction ability [10].

Holdout schemes usually aim to replicate the true application of the prediction machine. An common example of holdout is utilizing data from previous seasons to predict the most recent season, thus recreating the scenario where predictions are applied to the upcoming season. Besides representing an unobserved environments, predicting a new season also implies on new breeding cycle with unobserved families that display linkage disequilibrium patterns not seen in the training set [11].

Once the cross-validation design is chosen, the next step concerns the metric of success. Key metrics utilized to assess prediction method in plant breeding are:

- Predictability
- Spearman Correlation
- Jaccard Coefficient
- Prediction Error

These metrics can be briefly summarized as follows: (1) Predictability is measured the Pearson correlation observed and predicted values. The use of Pearson correlation is utilized because it is a simple metric that captures differences both ranking and magnitude. If the scope of the prediction is restricted to the data at hand, predictability can be re-scaled into accuracy by dividing it by the squared root of the heritability [12]. (2) Spearman rank-order correlation is an alternative to the predictability, which does not capture magnitude, but it is a direct measure of ranking. In plant breeding the Spearman correlation is used to measure the overall alignment of two selection method [13]. (3) Jaccard coefficient, or Czekanowski coefficient, concern the intersect between the predicted top X% and observed top X% and it is utilized as a metric of selection accuracy [14]. (4) Prediction error, or mean squared prediction error (MSPE), is not commonly applied in plant breeding but it provides a sense of how well predictions correspond to the observed value. In addition, MSPE can be analytically computed for linear models without explicit need for cross-validation [15].

Cross-validation schemes must be tailored to the specific application of the prediction machine. Guidelines to benchmark genomic prediction targeting additive genetics were formalized by Daetwyler et al. [16]. Cross-validation designs are summarized in terms of testing purpose by Crossa et al. [17] as:

- CV00: Untested genotypes, unobserved environments.
- CV0: Tested genotypes, unobserved environments.
- CV1: Untested genotypes, observed environments.
- CV2: Tested genotypes, observed environments.

Cross-validations can provide an insight upon the population stratification as validation are performed within and across sub-populations. Werner et al. [18] presents a theoretical repercussions and discrepancies in prediction accuracy when validations are performed across-families or nested within-family. When the stratification factors are known, such as family or other population factors, the cross-validation enable the investigation of predictive ability in multiple levels [19]. The expected genetic information captured by different the validation structures can be summarized as:

- Single-family: Linkage.
- Within-family: Linkage and LD.
- Across-family: Relationship, Linkage and LD.
- Leave-family-out: Relationship and LD.

The theory surrounding the information captured by molecular marker was formalized by Habier et al. [20]. The three sources of information are: (1) Relationship, which refers to the overall population structure and infinitesimal polygenic effect; (2) LD, which regards to association between marker and QTL as measured across unrelated individuals; and (3) Linkage or co-segregation, which captured the information of regions inherited from common set of parents, sometimes referred to as haplotype effect.

There are practical implications that have been derived from understanding sources of genetic information. For example, validation structure information postulates that within-family predictions must yield lower prediction ability than across-families because within-family predictions do not benefit from relationship information [21]. In the machine learning literature, the discrepancy in accuracy reported across-population and within-populations is referred to as the Simpson paradox [22].

The various prediction machines are likely to differ with regards to how well they capture the different sources of information. In addition, the nature of the information can change according to the parameterization. Molecular marker can be informative of the multiple types of information from different genetic effects, namely additive, dominance and epistasis. It was noted by Howard et al. [23] and Pérez-Rodríguez et al. [24] that non-parametric prediction methods may be able to capture non-additive information even when the markers are not explicitly coded to capture other types of variation. In systems where the genetic architecture is predominantly additive, semi-parametric methods (e.g. deep neural networks) will fail to provide an improvement in accuracy over simpler additive machines [25].

Figure 3 illustrates the various types of information captured by the different types of cross-validation. In addition, splitting the validation set into tested and untested environments provides an insight on the non-predictable component of genotype-by-environment interactions. In Figure 3, grain yield in soybeans from the SoyNAM dataset (40 families, 5349 individuals, 4408 markers). The model was trained on BLUPs based on data from 2012 and validated on BLUPs of individuals not included into the training for set collected in 2012 (tested environment) and 2013 (untested environment). The models were REML-based ridge regression (BLUP), fast Laplace model (FLM, [26]), Gaussian kernel for reproducing kernel Hilbert spaces (RKHS, [27]), BayesB, random forest regression (RFR), extreme gradient boosting machine (XGB), and a two hidden-layers deep neural network (DNN). Data is available in the R package

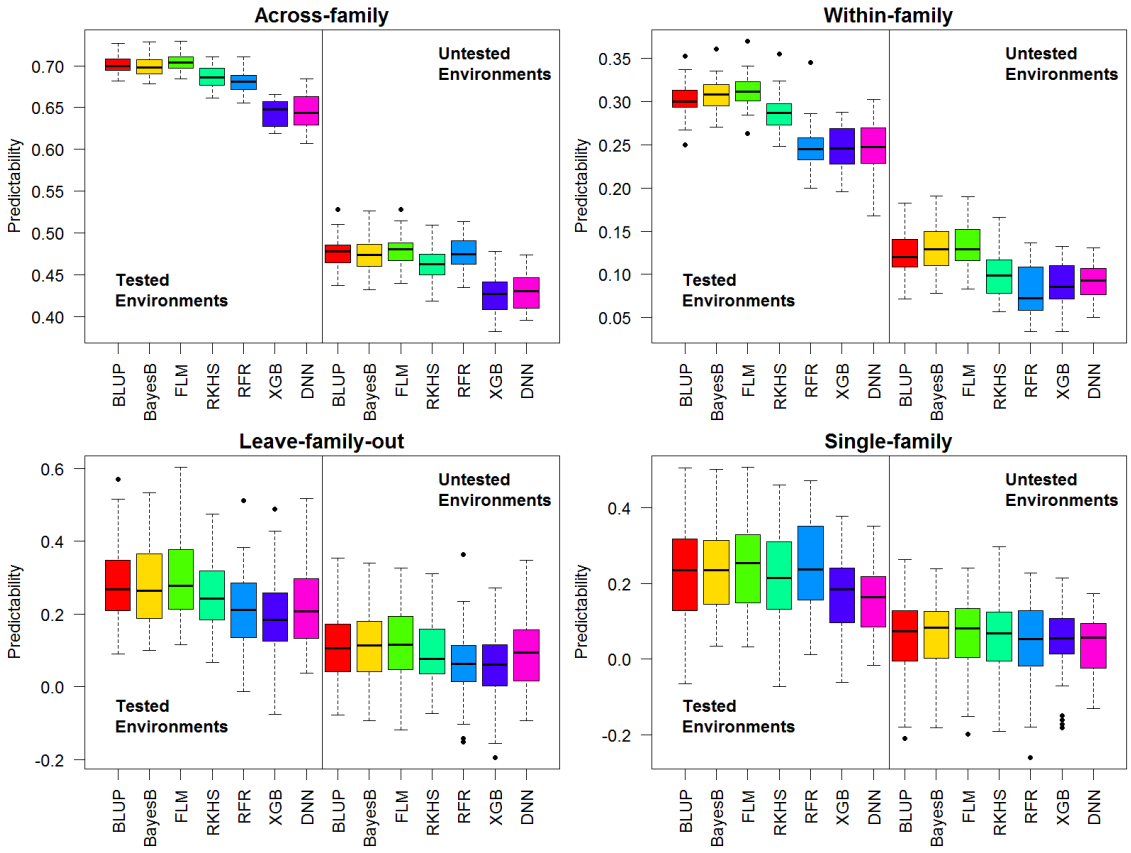

Figure 3: Four cross-validation schemes illustrating predictability of various methods utilized for genomic prediction. Grain yield models from the SoyNAM population, validated upon unobserved individuals from tested and untested environments.

SoyNAM [28], models are implemented in the R package bWGR [29], BGLR [30], ranger [31], xgboost [32] and keras [33]. Validations for single-family, within-family and across-family were performed as 5-fold random cross validation repeated 20x. Results illustrate that the across-family predictability is considerably higher within-family for all methods, as relationship is being captured. Likewise, within-family predictions are more accurate than single-family predictions because the former captures linkage disequilibrium in addition to linkage. In all scenarios, predictability is lower when validations are performed over untested environments.

4 Optimization

Linear optimization: A general interpretation of machine learning is treating statistical tasks as optimization problems, using data to find parameters that best describe the process under evaluation. A simple example is the generation of genetic value using the best linear unbiased predictors (BLUP). BLUPs are a product of a linear mixed model with unique solution due to a convex loss function, what makes the BLUP to be a BEST LINEAR predictor [34]. Whereas linear models can be statistically UNBIASED when all factor levels are balanced, the BLUP coefficients are always biased towards zero, or "shrunken", by a factor λ [35]. From the statistical standpoint, the model is called "mixed" because it contains both fixed and random terms. From the machine learning standpoint, the model is referred to "mixed" because it contains mixed penalization in the objective functions. The objective function of a mixed model utilized to compute BLUPs is $min(\sum \epsilon^2 + \lambda \sum \beta^2)$, where ϵ is vector of residuals and β are the BLUPs, or random effect coefficients. The role of the machinery is to find a set of regression coefficients from fixed and random terms that minimize that loss function. Regression coefficients from the fixed terms are not explicitly penalized by the loss function and, therefore, are not shrunken, but still play a role on minimizing the residual sum of squares. In the BLUP example, a secondary objective function can be utilized to optimize λ when this parameter is not treated as known a priori. For instance, the model bias parameter λ can be optimized to minimize prediction error using cross-validation or to maximize the Gaussian restricted (log) likelihood [36].

Optimization systems: For any task optimization, the problem has to be translated into a well-defined mathematical function. For the BLUP example aforementioned, the machinery goal is to produce accurate estimates of genetic merit and the mathematical function for this task was fitting a model that jointly minimized the squared sum of residuals ($\sum \epsilon^2$) and squared sum of coefficients ($\lambda \sum \beta^2$). Yet, machine learning must handle different functions for different problems. Once the problem is defined by a function the optimization takes place. The optimization is usually either performed by first- or second-order optimization methods. First-order methods involve first derivatives and the function is optimized one parameter at a time, some example include Expectation-Maximization (EM), Gibbs sampling and other Markov Chain Monte Carlo (MCMC) methods, Coordinate and Gradient descent. Second-order optimization are based on Newton and Quasi-Newton methods, using the exact or approximate second-derivative matrix (i.e. "Hessian") to precondition the problem and optimize multiple parameters at once, which considerably speeds up the convergence at a higher computational cost. In plant breeding, the classical example of first and second-order methods is the variance components estimation to calculate heritability, where the mathematical function of choice is the restricted maximum likelihood (REML). The variance components that optimize REML has been historically computed using EM [37], and more recently via average-information (AI) [38], named after the type of Hessian approximation performed as the average between the exact solution computed via Newton-Raphson and Fisher-Scoring matrices. A practical implication, besides convergence rate and computational requirements, is that each variance components estimated by a first-order method has an independent equation, whereas second-order methods present a single equation that estimates all variance components at once.

Coordinate descent and gradient descent: Coordinate and gradient descent are two key first-order algorithms utilized in machine learning optimization to solve linear and non-linear problems with good computational performance [39]. A common application of coordinate and gradient descents entails finding regression coefficients for mixed models and deep neural networks, respectively [40]. In the plant breeding context, a practical analogy to illustrate coordinate descent and gradient descent optimizations can be used to describe multiple-trait selections, where the objective (function) is to maximize the population means for multiple traits [41]. The *tandem method* can be interpreted as a coordinate descent approach to the selection of multiple-traits, where the population is selected for a single trait per generation, alternating traits over cycles, such that the optimization of every traits is conditional to the population previously selected for every other trait. Conversely, multiple-trait selection performed via *independent culling* can be interpreted as an example of gradient descent because all traits are selected at the same time with disregard to the multicollinearity among traits. Moreover, the selection intensity in this example corresponds to the machine learning parameter *learning rate*: High selection intensity leads to faster improvements at first with rapid depletion of genetic variance to a suboptimal point, or the convergence to local maximum point. Whereas low selection intensity translates into low and steady gains, likely to take more breeding cycles to deplete the genetic variance but yielding better results, which corresponds to the convergence to the global maximum point.

Stochastic batching: A stochastic processes are those based on randomness. For example, Monte Carlo is an optimization via stochastic simulation because it uses sampling techniques to estimate parameters. Randomness also can be introduced into optimization problems is through batching the observed data, which is particularly used in *big data* applications when not all the data can be used at once. This strategy is commonly referred to as *lazy loading*. There are few studies where batches are applied in the context of linear models [42], but the use of batches is important for deep learning [43] and XGBoost [32], where utilizing subsets plays an important role on improving computational and predictive performance by using only chunks of data in each iteration, which also helps with mitigating over-fitting the data [44]. When subsets of the data are used in each iteration in deep learning, the optimization normally done via gradient descent is renamed "stochastic gradient descent", as batching adds the stochastic element to the computation of gradients. The technical term *epoch* is reserved to a whole-set iteration when batching is utilized during the optimization. For example, if the complete dataset set has one thousand observations but the optimization is being with batches of one hundred, one epoch corresponds to ten iterations using batches.

Breeding pipeline optimization: Beside the typical regression-and-classification problems, multiple optimization problems need the attention of breeders. The optimization of the data fed into prediction machines is an addition concern that can be tackled numerically and may generate improved predictions and breeding decisions. As long as a objective can be defined mathematically, it can be optimized. With the routine collection of genomic information, breeders have the information necessary to optimize breeding scheme as a whole [45], from the selection of crosses to product placement. For instance, decisions involving the selection of individuals to be utilized as parents and which cross combinations to performed can be defined by one (or more) objectives. Akdemir et al. [46] proposed a multi-objective optimization with goal of offering a set of solutions that maximize the breeding value of upcoming generations while preserving the genetic variance to preclude fast depletion of genetic variation and, therefore, enabling long-term gains. Another example entails optimization of training populations to maximize predictive ability are tight to the deterministic computation of accuracy, aiming the optimization of training sets that maximize accuracy of predictions within- and across-family when then prediction targets are known [47, 48, 49].

Maximizing accuracy: Deterministic accuracy are based on the known properties the mixed models [50]. Accuracy (a) is defined as the correlation between values predicted from the calibration set (u_c) and true values from the prediction target (u_p). Considering the model $y = Xb + Zu + e$ with $E[y] = Xb$ and $Var(y) = V_c = ZGZ'\sigma_g^2 + I\sigma_e^2$, the outcome accuracy is given by:

$$a = Cor(u_c, u_p) = \frac{Cov(u_c, u_p)}{\sqrt{Var(u_c)Var(u_p)}} = \sqrt{g_p^{-1} g'_{g,c} Z' V_c^{-1} Z g_{c,g} \sigma_{c,g}} \tag{1}$$

Where $g_{c,g}$ is a vector with the genomic relationship between any given individual to be predicted and all the individuals in the calibration data; g_p^{-1} is the diagonal element of the genomic relationship matrix corresponding to the individual being predicted; $\sigma_{c,g}$ is the genetic covariance between calibration and prediction set, which under the assumption that the statistical model is the same as the true model, then $\sigma_{c,g} = \sigma_c^2 \rho$ as ρ would correspond to the "GxE" correlation that informs the similarity between where the dataset c was collected and where the prediction target p would have been collected. From the equation above, if variance components (σ_g^2, σ_e^2, ρ) and relationship kernel (G) are known, then the accuracy of any individual can be predicted, regardless of whether or not that individual is a part of the calibration set. This equation can be utilized to maximizing accuracy by varying which individuals will be included into the calibration set by changing the composition of G. The same equation may aid the optimization of experimental designs by changing, for instance, the number of replications (changing Z). The equation can be easily expanded to accommodate multiple random effects (composition of V_c) and to measure the impact of variance components into the accuracy.

5 Supervised algorithms

Prediction methods described by the machine learning perspective can appear divergent from traditional statistics. This section intends to provide an operational sense of how the main machines work using an algebraic notation that is more familiar to the plant breeding community, by simplifying and adapting the description from books focused on the mathematical [51, 52, 53] and applied side [8, 7] of machine learning. The main supervised machine learning algorithms pertinent to plant breeding predictions are summarized in this section.

(1) Linear models: Linear machines comprises the set of methods that describe the response variable as a linear combination of features. Yet, the feature may or may not be linear. Such models are mostly defined as:

$$y = Xb + e \tag{2}$$

Where y is a vector of length n that serves as the response variable, X is the design matrix of feature with dimension n rows (observations) by p columns (parameters), b is a vector with length p with the corresponding coefficients, and e is the vector of residuals. Most linear systems can be efficiently solved via coordinate descent, as aforementioned. Coordinate descent marginalizes complex models to resolve one dimension at a time. The marginal model for the j^{th} parameter of Xb can be derived as follows:

$$\begin{aligned} y &= Xb + e \\ y &= X_{-j}b_{-j} + x_j b_j + e \\ y - X_{-j}b_{-j} &= x_j b_j + e \\ y_j &= x_j b_j + e \end{aligned} \tag{3}$$

Where y_j is the response variable y conditioned to all model parameters except for the j^{th} parameter. The conditioning is based on isolating the j^{th} parameter from other set of parameters, that occurs by separating Xb into two terms, $X_{-j}b_{-j}$ and $x_j b_j$.

Machine learning methods based on regression differ based on their univariate solution. Linear machines include least squares, likely the oldest prediction machine, dating back to 1722 [54], the modern ridge regression from 1970 [55], as well as newer machines such as the least absolute shrinkage and selection operator "LASSO", introduced in 1996 [56], and the elastic-net introduced in 2005 [57]. The various solution for the regression coefficient b_j are summarized below.

The least squares and least absolute solutions are not penalized by any parameter and cause model over-fit as the number of parameters surpass the number of observations. The least absolute has no analytical derivation and multiple solution have been proposed overtime, generally based on estimators that utilize median [58]. Ridge regression penalizes the model by inflating the denominator causing it to shrink the regression coefficient. By contrast, LASSO shrinks

Table 1: Solution of coefficient b for univariate linear systems: $y = xb + e$.

Method	Least squares (LS)	Least absolute (LAD)	Ridge (L_2)	LASSO (L_1)	Elastic-Net (L_1L_2)
Solution (b)	$\frac{Cov(x_j,y_j)}{Var(x_j)} = \frac{x_j'y_j}{x_j'x_j}$	$\frac{Median(x_j \# y_j)}{Var(x_j)}$	$\frac{x_j'y_j}{x_j'x_j+\lambda}$	$\frac{\lvert x_j'y_j\rvert-\lambda}{x_j'x_j}$	$\frac{x_j'y_j-\lambda_1}{x_j'x_j+\lambda_2}$
Minimizes	$e'e$	$\sum \lvert e\rvert$	$e'e + \lambda b'b$	$e'e + \lambda \sum \lvert b\rvert$	$e'e + \lambda b'b + \sum \lvert b\rvert \lambda_1$
Unique	Yes	No	Yes	No	No

the coefficient by deflating the nominator and removing it from the model altogether if $\lambda > |x_j'y_j|$. Elastic-net was proposed as the combination of ridge and LASSO solutions.

For the ridge regression, coordinate descent can be efficiently computed through the Gauss-Seidel via residual update to avoid the explicit conditioning $(y - X_{-j}b_{-j})$ which reduces the problem to a univariate. The Gauss-Seidel in question was described by Legarra and Misztal [59]. It operates by updating the j^{th} coefficient with subsequent update of residuals:

$$\begin{aligned} b_j^{t+1} &= \frac{x_j'e^t + x_j'x_jb_j^t}{x_j'x_j+\lambda} \\ e^{t+1} &= e^t - x_j(b_j^{t+1} - b_j^t) \end{aligned} \tag{4}$$

A same linear model may accommodate multiple losses if a subset of parameters is subjective to one loss, whereas another set of parameters is subjective to another loss. The example aforementioned is the mixed model, where fixed effects are solved via least squares and random effects as ridge regression. Moreover, each random effect is subjective to its own penalization λ. Under traditional machine learning settings, the penalization parameter λ is commonly tuned via cross-validation. An exception is found for the ridge regularized models containing one or multiple λ parameters, where statisticians and geneticist infer λ using Bayesian and likelihood methods, lambda has analytical solution through variance components as $\lambda_r = \sigma_e^2\sigma_b^{-2}$ [60], and more details are provided in the subsection 5.

(2) Kernel methods: Kernels can be interpreted as a re-parameterization of linear machines. Such methods are mostly based on stationary kernels (K) that use the model features to defined the similarity among observations [61]. When the kernel is based on linear relationship ($K = XX'$), it yields the exact same solution as linear machines. In addition, any (non-linear) kernels can be linearly decomposed and solved as linear machines [27], which may involve simple algebraic methods such as Cholesky or Eigendecomposition. Kernel methods are powerful prediction machines, but they are also referred to as "lazy learners" because it is not possible to store the model from a kernel machine and the training data must be available during the predictions.

Kernels are popular for de-noising data and prediction. Kernel methods commonly applied in modeling include kernel ridge regression "KRR" or Kalman filters (e.g. GBLUP), kriging, splines, stationary auto-regressive processes, reproducing kernel Hilbert spaces "RKHS", support-vector regression "SVR", radial basis function "RBF", and reduced ranking kernels, such as principal components regression. The nomenclature of multiple methods is often inter-exchangeable, as most kernel method have their origins in different field of science and engineering but share the same analytical solution, and generally differing based on how the kernel itself is built.

Table 2: Computation of dense kernels for linear systems: $y = K(X) + e$.

Kernel	Linear	Gaussian	Arc-Cosine
Solution $K(i,j)$	$\alpha^{-1}x_i'x_j$	$\exp\left[-\theta \sum (x_i - x_j)^2\right]$	$\pi^{-1}(x_i'x_i)(x_j'x_j)sin(\theta_{ij}) + (\pi - \theta_{ij})cos(\theta_{ij})$
	where $\alpha = \sum_i var(x_i)$	where θ is tuned	where $\theta_{ij} = cos^{-1}Cor(x_i, x_j)$
Used for	Kalman Filter, GBLUP	RKHS, RBF, SVR	Deep Kernel
Reference	[62]	[63]	[64, 65]

The outcome kernel matrix is a $n \times n$ matrix that express the pairwise relationship among observations, based on the feature space (X). Kernel methods may provide a convenient way to compute linear interactions, which is often necessary to parameterize epistasis in genetic analysis [66]. For example, dataset with 100 SNPs would increase the

number of features to 10,000 to accommodate the pairwise interaction among SNPs in a linear machine, whereas the interaction kernel can be simply computed as $K_i = XX'\#XX'$, using the Hadamard product (element-wise multiplication) on the linear kernel.

In spite the non-linear nature of most kernels, the solutions and predictions from kernel methods are found through linear system of equations [67]. The linear systems aim to find the predictor g, which contains the signal from y as extracted by K. Factorization is often involved to speed up the computation of the coefficients. Different solutions for kernel regression with $L2$ penalization are presented below. These are regularized by the same λ parameter presented on linear machines, which is found through cross-validation or estimated from variance components.

Table 3: Solving kernel linear systems $y = g + e$, where $g = K(X)$.

Factorization	None	Inverse	Eigen	Cholesky
Transformation	$f(K) = K$	$f(K) = K^{-1}$	$f(K) = UDU'$	$f(K) = LL'$
Solution	$(K + I\lambda)g = Ky$	$(I + \lambda K^{-1})g = y$	$(I + \lambda D^{-1})g = U'y$	$(L'L + \lambda I)g = Ly$

Reduction of dimensionality is attained through the 'Eigen' strategy. If the kernel K is decomposed into eigenvectors (matrix U) and their respective eigenvalues (diagonal matrix D), one may reconstruct K without all eigen-pairs as those are sorted by the amount of variance explained. Furthermore, it has been suggested that using enough eigenpair to capture 98% of the total variation of the kernel, as defined by the eigenvalues, may be beneficial for predictions and computation [68]. Reduction of dimensionality is also the key concept of principal components regression, where least squares regression is deployed on the first few eigenvectors of any given kernel.

The inverse kernel (K^{-1}) is referred to as the *precision matrix*. In many cases, it is possible to compute the precision matrix directly, without having to build the kernel itself, with the bonus of building a sparse matrix that would save a lot of computational resources. A classic example in genetic analysis is the Henderson method [69] to build the inverse relationship matrix from pedigree information. Another important example commonly utilized in plant breeding analysis is the row-column spatial adjustment AR(1)×AR(1) utilized for field trials, where the auto-regressive matrix itself is a dense matrix but its inverse is a easy-to-compute "thick-diagonal" sparse matrix. In fact, the construction of sparse precision matrices directly from the features matrix is an important subject in machine learning [70, 71].

(3) Neural network: Artificial neural networks (NN) are generalizations of linear machines where non-linear eigenvectors are created to transform the initial set of parameters into informative features that do not necessarily have a meaning by itself but may serve as good predictors. These transformations in the parameter space are based on the so-called *activation function* (α). Deep neural network (DNN) contain two or more hidden layers of parameter transformation, which might allow the network to increase the complexity and non-linearity of the eigenvectors. If a single-layer neural network is deployed with linear eigenvectors, the corresponding output is mathematically equivalent to a partial least square (PLS) regression, where the number of nodes into the hidden layer of the network corresponds to the number of components in the PLS. A progression can be drawn from a linear model (LM) to a deep neural network as follows:

$$\begin{array}{ll} y = Xb + e & LM \\ y = (XB_1)b_2 + e & PLS \\ y = \alpha(XB_1)b_2 + e & NN \\ y = \alpha(\alpha(XB_1)B_2)b_3 + e & DNN \end{array} \tag{5}$$

The network coefficients (B_1, B_2, b_3) are estimated via gradient descent, as there is no single analytical solution for the joint estimation of parameters across layers. Gradient descent, although not regarded as a good optimizer, is particularly preferred for DNNs due to computational convince. Within the realm of a linear systems of equations ($y = Xb + e$), gradient descent solves for regression coefficients as follows:

$$b^{t+1} = b^t - \frac{2r}{n}(X'(y - Xb) - \lambda b^t) \tag{6}$$

Where r is the rate parameter (or *learning rate*), n is the number of observations, e corresponds to the vector *gradients* (or residuals, since $e = y - Xb$), and λ is the penalization parameter utilized for the particular case of ridge regression. The learning rate must be set a priori and it will control the convergence rate, often dictated by the degree of multicollinearity among the features of X. When transitioning into the solution of DNNs, the residuals must be *back-propagated* to the inner layers. It is imperative that networks have random values as starting point for the regression coefficients, otherwise it is simply not possible to propagate the residuals. Fitting a DNN involves three steps:

- Fit neural network
$$H_1 = \alpha(XB_1)$$
$$H_2 = \alpha(H_1B_2)$$
$$h_3 = H_2b_3$$
- Compute gradient
$$e_3 = y - h_3$$
$$E_2 = \alpha(e_3B_3')$$
$$E_1 = \alpha(E_2B_2')$$
- Update coefficients
$$b_1^{t+1} = b_1^t + \frac{2r_1}{n}(X'(E_1 - H_1) - \lambda b_1^t)$$
$$b_2^{t+1} = b_2^t + \frac{2r_2}{n}(H_1'(E_2 - H_2) - \lambda b_2^t)$$
$$b_3^{t+1} = b_3^t + \frac{2r_3}{n}(H_2'(e_3 - h_3) - \lambda b_3^t)$$

Here the intercepts (or *bias*) have been omitted for simplicity. Otherwise an additional column filled with 1's must be added to the feature matrices: X, H_1 and H_2. The learning rate (r_1, r_2, r_3) may vary across parameters and change overtime, few algorithms have become the gold standard to track and update rates, including Adam (adaptive momentum [72]) and RMSProp (root mean square propagation [73]). Tracking the learning rates increases the computational cost of fitting DNNs because gradients must be stored twice, but the cost is offset by faster convergence. When stochastic methods are deployed (e.g. stochastic gradient descent), a different fraction or subset of the data ("batch") is utilized for the three steps in each iteration, which limits the amount of data flowing to the network at a time and considerably increases the computation performance.

Neural networks are prone to overfit the data when the coefficients are not penalized and scientists are often encouraged to prepare a "testing set", some data not utilized for training that serves the purpose of determining the convergence based on the decay in predictive power. Another consideration about neural networks regard the activation functions (α), which are simply element-wise transformation. Nowadays, hyperbolic tangent function (Tanh) and rectifier linear (ReLU) are the two most commonly utilized transformations for regression problems [25]. Activation functions must be easy to compute as they are utilized to fit the network (forwards) and estimate the gradients (backwards). Tanh squeezes values into a [-1,1] space where any value below approximately -2 or above 2 are set as -1 and 1, respectively. Thus, Tanh can be sensitive to scale and may benefit from normalizing the response. ReLU is a "half-linear" transformation, as it sets negative values to zero without altering positive values, creating a breakout (or *fold*) of the parameter space. The "leaky" version of ReLU is closer to a linear system, as it linearly shrinks negative values by a pre-defined constant instead of eliminating them altogether. In spite of being virtually linear, the ReLU activation is believed to provide universal approximation for any function [74], as ReLU achieves more complex folds of the parameter space as the number of nodes and layers increases. In addition, ReLU activation is particularly known to benefits from "dropout" [75], which simply means to ignore a random proportion of the parameters in each iteration with the hope of increasing predictive power by mitigating co-adaptation among transformed features [76].

(4) Decision trees: Unlike linear methods, kernels and neural networks, the models based on tree is largely algorithmic-driven and does not involve explicit algebra. Tree-based machines include the regression trees per se [77], adaptive boosting (AdaBoost [78]), gradient boosting machine (GBM [79]), extreme gradient boosting (XGBoost [32]), and random forest [80]. Tree-methods, particularly random forest, have become popular for genetic analysis of signal detection and prediction [81, 82, 83, 31].

Regression trees are regarded as *weak learners* and benefit from stochastic ensembles, as the collective of weak learners creates a strong learner. For this reason, the generation and aggregation of multiple trees through resampling (bagging and boosting) is essential for robust predictions and has become standard practice for this type of machine. Hierarchically, the implementation of tree-based machines is based on three steps:

- Step 1. Split function
- Step 2. Tree building
- Step 3. Ensemble method

Split function: The split function defines the breakdown of x that divides the response variable y based on some information metric to be minimized. The greedy search for the best split point is referred to as the "CART" algorithm, which generally outperform rule-based algorithms, namely ID3 and C4.5. The squared error is a suitable information metric for the modeling of continuous response, whereas Gini impurity is preferred for classification problems. CART's

greedy search for splitting point screens the parameter space of continuous variables, and searches for combination of levels for categorical variables. At its simplest, a tailored greedy search for diploid SNP markers coded as 012 have only two possible binary splits, ≤ 1 and < 1, representing (AA,Aa)/(aa) and (AA)/(Aa,aa), respectively. Yet, the same SNP may undergo further split in the node containing the heterozygous classes. The output of the split function must include the best split point and the squared error attributed to it, as both information is utilized to build trees.

Tree building: Nowadays, trees are built from sequential recursive splits based on CART algorithm, which performs a greedy search for the best split points as opposed to its predecessor rule-based algorithms, namely ID3 and C4.5. Given a set of parameters (X), the tree is built by checking which parameters reduced most of the squared error using the split function. That parameter will define the first split of the tree to provide an optimal topology. The parameter space X is reassessed within each primary split to generate the second split. The process is repeated within each split until the stopping point, where further splits do not provide additional information. The regularization of individual trees is performed (1) by limiting its depth, (2) by the number of cases in each split, (3) by the minimum information gain and (4) by penalizing the information gain afterwards via "cost-complexity pruning", where splits are removed if not enough information gain is provided.

Ensemble method: The ensemble is done in one of the two ways: bootstrapping aggregation (or *bagging*) and boosting. Bagging is the framework that leads to random forest, whereas boosting is the key for AdaBoost, GBM and XGBoost. Bagging consists of creating a predefined number of trees (500 trees is a common default) from random samples of individuals and a random subset of parameters, and the final random forest prediction is the average result from all trees. The sampling of individuals is usually based on bootstrap sample that preserve the original size of the data but sub-sampling without replacement is also acceptable, especially for large datasets. The sub-sample of parameters is usually done on a small fraction of parameters (e.g. $\sqrt{p}$), but that may also depend on the number of features available and the amount of information carried by each feature, as datasets range from having continuous variables (e.g. environmental factors) to thousands of indicator variables (e.g. factors and SNP data). Boosting consists of running one tree at a time, using the residuals (or *gradients*) computed from the current set of the trees to build and annex the next tree. Similar to random forest, boosting benefits from creating tree on subsets of the data available, both at observation and parameter level. Thus, boosting is guided by the following five steps: 1. Sample a random subset of individuals, analogous to a batch in a neural network; 2. Estimate their predicted value from the current set of trees; 3. Calculate the residuals ($y - \hat{y}$); 4. sample a subset of parameters to construct the new tree; 5. fit and prune a new tree. Repeat. Since both ensemble methods are controlled by multiple parameters, both at tree and ensemble level, there multiple opportunities for regularization. For instance, more trees with fit with less parameters benefits the ensemble by increasing stochasticity of the model, whereas reducing the depth trees may benefit boosting by mitigating over-fitting. Post-ensemble, the linear description of the tree-machine is formulated as:

$$y = n_t^{-1} \sum T(X_{p \in P}) + e \tag{7}$$

Where y is the response variable, e is the vector of residuals, T is the function that represents a single tree, and $X_{p \in P}$ represents the design matrix of exploratory variables, where each tree contains a random subset (p) of all the features available (P). The linear description above is acceptable for both bagging and boosting ensembles.

(5) Analytical tuning: Prediction machine based on L_2 penalization, particularly ridge regression and kernel methods, are subject to a shrinkage parameter λ that, as aforementioned, has an analytical solution based on variance components and does not necessarily require cross-validation. The solution is $\lambda_r = \sigma_e^2 \sigma_r^{-2}$. In fact, most genetic analysis prefer inferring variance components over cross-validation-based tuning since the variance components are meaningful parameters and are utilized to estimate heritability. The most common function being minimized for variance components, for both linear ridge regression and kernel methods, corresponds to the REML function:

$$L(b, \sigma^2_{u(1)}, ..., \sigma^2_{u(r)}, \sigma^2_e) = -\frac{1}{2} ln|V| - \frac{1}{2} ln|X'V^{-1}X| + -\frac{1}{2}(y - Xb)'V^{-1}(y - Xb) \tag{8}$$

The function above suits the linear $y = Xb + Z_1u_1 + ... + Z_ru_r + e$, where the response variable (y) is modeled by non-penalized features (fixed effects, Xb) and differently penalized set of parameters (random effects, $Z_1u_1, ..., Z_ru_r$). Where $V = Z_1K_1Z_1'\sigma^2_{u(1)} + ... + V = Z_rK_rZ_r'\sigma^2_{u(r)} + R\sigma^2_e$, e is the vector of residuals, R is the residual correlation kernel usually set as $R = I$, the matrices Z_r and K_r correspond to the feature set and the kernel matrix of the r^{th} term, and the design matrix X contains the intercept and other non-penalized features. REML is a convex and considerably more complex function to optimize when compared to most loss functions, but it enables analytical solution for models containing non-penalized parameters and multiple independent ridge regressions. For context, the expectation of a variance component, derived from the first derivative, has the following solution:

$$\sigma_i^2 = \frac{yPV_iPy}{tr(PV_i)} \tag{9}$$

Where P is the mixed model null-space projection matrix ($P = V^{-1} - V^{-1}X(X'V^{-1}X)^{-1}X'V^{-1}$) and V_i is the first derivative of $\partial V/\partial\sigma_i^2$, such that $V_e = R$ for the residuals and $V_r = Z_rK_rZ_r'$ for other terms. The projection P can also be though as the joint absorption of fixed and random effects $P = V^{-1}S$ where S is the projects into the null-space of fixed effects, $S = I - X(X'X)^{-1}X$. The null-space projections can be intuitively perceived as functions that remove signal, as $Sy = y - Xb$ and $Py\sigma_e^2 = y - Xb - Zu$, and it is used for mathematical simplifications. Due to the important role of mixed model in plant breeding, more technical detail about model specification and variance components are provided in the appendix.

6 Limitations

Machine learning provides a powerful set of tools for predictive analytics, but some limitations of deploying these black-box models are not well understood. Multiple problems also arise at the implementation level. Whereas coding machine learning systems can be an objectively simple task, other components of the whole system are not trivial, particularly for validating, implementing, productionizing and curating the machinery. A series of deployment checks is proposed by Sculley et al. [84], where authors discuss the use of questionable data or non-reproducible features, bad coding and implementation practices that may create problems with maintaining the system, and the need for updating models and continuous monitoring the results. In addition, a common concern of data scientists regards epistemological problems of machine learning [85], as complex algorithms are predisposed to a problem referred to as cognitive mismatch, that is, when the machine does not learn what the programmer think it is learning.

Robust machine learning methods have their application scope limited by the quality and quantity of the data. Good models are still prone to carry biases from the data. Problems model from with biased data have been previously identified on machine learning applied to judicial decisions, mortgage lending, and heath care [86]. Use of biased data in plant breeding obviously does not have the same societal high stakes, but it may lead to poor decision-making with economic consequences. Bias data in plant breeding comes from both genotypes and phenotypes. **A**) Genotypic bias is attributed to the germplasm structure due to domestication bottlenecks and ongoing selection. Selection for specific alleles creates large linkage disequilibrium blocks surrounding the target region referred to as signatures of selection [87]. The population structure is proportional to the accumulation of genomic signatures. For signal detection, such as genome-wide association analysis, it is imperative to mitigate the influence of population structure [88] while preserving enough allele representation to capture the correct signal (see Beavis effect [89]). In contrast, population structure is necessary for genomic prediction across-family. **B**) Phenotypic bias reflects systematic problems with data collection. This may include (1) the over-representation of certain individuals, such as experimental checks that out-weights the information from selection candidates; (2) censorial bias from opportunistic notes where data collecting is restricted to when events occur, such as lodging and disease scores; (3) extrapolation of sampling space, which occurs when products are placed outside the geographical scope of where the breeding trials were located; and (4) inherited bias, common problem in multi-stage analysis when the output of a model is the input of another model and the biases are propagated through the system [90].

7 Conclusions

Plant breeding can benefit greatly from data-driven solutions that identify and manage superior germplasm through machine learning, especially with the availability of phenotypic and genotypic data collected with high-throughput technologies. The various prediction approaches differ with regards to the types of information they capture, and cross-validation can bu utilized to optimize the prediction framework when the validation design is tailored to the specific application of the prediction machine. In addition to prediction, machine learning also enables the optimization of any statistical task as long as the problem has to be translated into a well-defined mathematical function, such as the deterministic accuracy of genomic prediction. When the volume of data is large, the optimization can be done in stochastic manner using sampling techniques.

Supervised machines are primarily based on linear models, kernels, neural networks and tree ensembles. In the field of genetics, linear and kernel methods are commonly deployed under the scope of mixed models, where no tuning is required. Neural networks and tree ensembles are general-purpose prediction machines that may require tuning, but those can handle a larger amount of data. Supervised machines are powerful tools, but these do not come without problems. In addition to known issues attributed to statistical properties of the data, there are other steps to implement a successful framework, from model training and validation to the follow up monitoring of the results.

References

[1] Eugene Lin and Hsien-Yuan Lane. Machine learning and systems genomics approaches for multi-omics data. *Biomarker research*, 5(1):1–6, 2017.

[2] Nikolaos Perakakis, Alireza Yazdani, George E Karniadakis, and Christos Mantzoros. Omics, big data and machine learning as tools to propel understanding of biological mechanisms and to discover novel diagnostics and therapeutics. *Metabolism-Clinical and Experimental*, 87:A1–A9, 2018.

[3] Mohsen Shahhosseini, Guiping Hu, Isaiah Huber, and Sotirios V Archontoulis. Coupling machine learning and crop modeling improves crop yield prediction in the us corn belt. *Scientific reports*, 11(1):1–15, 2021.

[4] Germano Costa-Neto, Roberto Fritsche-Neto, and José Crossa. Nonlinear kernels, dominance, and envirotyping data increase the accuracy of genome-based prediction in multi-environment trials. *Heredity*, 126(1):92–106, 2021.

[5] Andrew E Teschendorff. Avoiding common pitfalls in machine learning omic data science. *Nature materials*, 18(5):422–427, 2019.

[6] Guido Zampieri, Supreeta Vijayakumar, Elisabeth Yaneske, and Claudio Angione. Machine and deep learning meet genome-scale metabolic modeling. *PLoS computational biology*, 15(7):e1007084, 2019.

[7] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[8] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[9] Daniel Runcie and Hao Cheng. Pitfalls and remedies for cross validation with multi-trait genomic prediction methods. *G3: Genes, Genomes, Genetics*, 9(11):3727–3741, 2019.

[10] Jin Sun, Jessica E Rutkoski, Jesse A Poland, José Crossa, Jean-Luc Jannink, and Mark E Sorrells. Multitrait, random regression, or simple repeatability model in high-throughput phenotyping data improve genomic prediction for wheat grain yield. *The plant genome*, 10(2):plantgenome2016–11, 2017.

[11] M Pszczola and MPL Calus. Updating the reference population to achieve constant genomic prediction reliability across generations. *Animal*, 10(6):1018–1024, 2016.

[12] Christina Lehermeier, Valentin Wimmer, Theresa Albrecht, Hans-Jürgen Auinger, Daniel Gianola, Volker J Schmid, and Chris-Carolin Schön. Sensitivity to prior specification in bayesian genome-based prediction models. *Statistical applications in genetics and molecular biology*, 12(3):375–391, 2013.

[13] Nicolas Heslot, Hsiao-Pei Yang, Mark E Sorrells, and Jean-Luc Jannink. Genomic selection in plant breeding: a comparison of models. *Crop science*, 52(1):146–160, 2012.

[14] CG Qiao, KE Basford, IH DeLacy, and M Cooper. Evaluation of experimental designs and spatial analyses in wheat breeding trials. *Theoretical and Applied Genetics*, 100(1):9–16, 2000.

[15] Shizhong Xu. Predicted residual error sum of squares of mixed models: an application for genomic prediction. *G3: Genes, Genomes, Genetics*, 7(3):895–909, 2017.

[16] Hans D Daetwyler, Mario PL Calus, Ricardo Pong-Wong, Gustavo de Los Campos, and John M Hickey. Genomic prediction in animals and plants: simulation of data, validation, reporting, and benchmarking. *Genetics*, 193(2):347–365, 2013.

[17] José Crossa, Paulino Pérez-Rodríguez, Jaime Cuevas, Osval Montesinos-López, Diego Jarquín, Gustavo de los Campos, Juan Burgueño, Juan M González-Camacho, Sergio Pérez-Elizalde, Yoseph Beyene, et al. Genomic selection in plant breeding: methods, models, and perspectives. *Trends in plant science*, 22(11):961–975, 2017.

[18] Christian R Werner, R Chris Gaynor, Gregor Gorjanc, John M Hickey, Tobias Kox, Amine Abbadi, Gunhild Leckband, Rod J Snowdon, and Andreas Stahl. How population structure impacts genomic selection accuracy in cross-validation: Implications for practical breeding. *Frontiers in plant science*, 11:2028, 2020.

[19] Alencar Xavier and Katy M Rainey. Quantitative genomic dissection of soybean yield components. *G3: Genes, Genomes, Genetics*, 10(2):665–675, 2020.

[20] David Habier, Rohan L Fernando, and Dorian J Garrick. Genomic blup decoded: a look into the black box of genomic prediction. *Genetics*, 194(3):597–607, 2013.

[21] Andrés Legarra, Christèle Robert-Granié, Eduardo Manfredi, and Jean-Michel Elsen. Performance of genomic selection in mice. *Genetics*, 180(1):611–618, 2008.

[22] Carem C Fabris and Alex A Freitas. Discovering surprising patterns by detecting occurrences of simpson's paradox. In *Research and Development in Intelligent Systems XVI*, pages 148–160. Springer, 2000.

[23] Réka Howard, Alicia L Carriquiry, and William D Beavis. Parametric and nonparametric statistical methods for genomic selection of traits with additive and epistatic genetic architectures. *G3: Genes, Genomes, Genetics*, 4(6):1027–1046, 2014.

[24] Paulino Pérez-Rodríguez, Daniel Gianola, Juan Manuel González-Camacho, José Crossa, Yann Manès, and Susanne Dreisigacker. Comparison between linear and non-parametric regression models for genome-enabled prediction in wheat. *G3: Genes| Genomes| Genetics*, 2(12):1595–1605, 2012.

[25] Osval Antonio Montesinos-López, Abelardo Montesinos-López, Paulino Pérez-Rodríguez, José Alberto Barrón-López, Johannes WR Martini, Silvia Berenice Fajardo-Flores, Laura S Gaytan-Lugo, Pedro C Santana-Mancilla, and José Crossa. A review of deep learning applications for genomic selection. *BMC genomics*, 22(1):1–23, 2021.

[26] Alencar Xavier. Efficient estimation of marker effects in plant breeding. *G3: Genes, Genomes, Genetics*, 9(11):3855–3866, 2019.

[27] Gustavo de Los Campos, Daniel Gianola, Guilherme JM Rosa, Kent A Weigel, and José Crossa. Semi-parametric genomic-enabled prediction of genetic values using reproducing kernel hilbert spaces methods. *Genetics Research*, 92(4):295–308, 2010.

[28] Alencar Xavier et al. Soynam: Soybean nested association mapping dataset. 2021. R package version 1.6.1.

[29] Alencar Xavier, William M Muir, and Katy M Rainey. bWGR: Bayesian whole-genome regression. *Bioinformatics*, 36(6):1957–1959, 10 2019.

[30] Paulino Pérez and Gustavo de Los Campos. Genome-wide regression and prediction with the bglr statistical package. *Genetics*, 198(2):483–495, 2014.

[31] Marvin N Wright, S Wager, and P Probst. Ranger: A fast implementation of random forests. *R package version 0.5. 0, URL https://CRAN. R-project. org/package= ranger*, 2016.

[32] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[33] Taylor B Arnold. kerasr: R interface to the keras deep learning library. *Journal of Open Source Software*, 2(14):296, 2017.

[34] Timothy L White and Gary R Hodge. Best linear unbiased prediction: Applications. In *Predicting Breeding Values with Applications in Forest Tree Improvement*, pages 300–327. Springer, 1989.

[35] George K Robinson et al. That blup is a good thing: the estimation of random effects. *Statistical science*, 6(1):15–32, 1991.

[36] Ronald de Vlaming and Patrick JF Groenen. The current and future use of ridge regression for prediction in quantitative genetics. *BioMed research international*, 2015, 2015.

[37] David A Harville. Maximum likelihood approaches to variance component estimation and to related problems. *Journal of the American statistical association*, 72(358):320–338, 1977.

[38] DL Johnson and Robin Thompson. Restricted maximum likelihood estimation of variance components for univariate animal models using sparse matrix techniques and average information. *Journal of dairy science*, 78(2):449–456, 1995.

[39] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *arXiv preprint arXiv:1606.04474*, 2016.

[40] Eduard Gorbunov, Filip Hanzely, and Peter Richtárik. A unified theory of sgd: Variance reduction, sampling, quantization and coordinate descent. In *International Conference on Artificial Intelligence and Statistics*, pages 680–690. PMLR, 2020.

[41] Weikai Yan and Judith Frégeau-Reid. Breeding line selection based on multiple traits. *Crop Science*, 48(2):417–423, 2008.

[42] Alencar Xavier, Shizhong Xu, William Muir, and Katy Martin Rainey. Genomic prediction using subsampling. *BMC bioinformatics*, 18(1):1–7, 2017.

[43] Bo Pang, Erik Nijkamp, and Ying Nian Wu. Deep learning with tensorflow: A review. *Journal of Educational and Behavioral Statistics*, 45(2):227–248, 2020.

[44] Tomoumi Takase. Dynamic batch size tuning based on stopping criterion for neural network training. *Neurocomputing*, 429:1–11, 2021.

[45] Mark Henryon, P Berg, and AC Sørensen. Animal-breeding schemes using genomic information need breeding plans designed to maximise long-term genetic gains. *Livestock Science*, 166:38–47, 2014.

[46] Deniz Akdemir, William Beavis, Roberto Fritsche-Neto, Asheesh K Singh, and Julio Isidro-Sánchez. Multi-objective optimized genomic breeding strategies for sustainable food improvement. *Heredity*, 122(5):672–683, 2019.

[47] Renaud Rincent, Alain Charcosset, and Laurence Moreau. Predicting genomic selection efficiency to optimize calibration set and to assess prediction accuracy in highly structured populations. *Theoretical and applied genetics*, 130(11):2231–2247, 2017.

[48] Pascal Schopp, Dominik Müller, Yvonne CJ Wientjes, and Albrecht E Melchinger. Genomic prediction within and across biparental families: means and variances of prediction accuracy and usefulness of deterministic equations. *G3: Genes, Genomes, Genetics*, 7(11):3571–3586, 2017.

[49] Yvonne CJ Wientjes, Piter Bijma, and Mario PL Calus. Optimizing genomic reference populations to improve crossbred performance. *Genetics Selection Evolution*, 52(1):1–18, 2020.

[50] Yvonne CJ Wientjes, Roel F Veerkamp, Piter Bijma, Henk Bovenhuis, Chris Schrooten, and Mario PL Calus. Empirical and deterministic accuracies of across-population genomic prediction. *Genetics Selection Evolution*, 47(1):1–14, 2015.

[51] Alan Julian Izenman. Modern multivariate statistical techniques. *Regression, classification and manifold learning*, 10:978–0, 2008.

[52] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[53] Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. *Mathematics for machine learning*. Cambridge University Press, 2020.

[54] Stephen M Stigler. Gauss and the invention of least squares. *the Annals of Statistics*, pages 465–474, 1981.

[55] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[56] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[57] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.

[58] Yinbo Li and Gonzalo R Arce. A maximum likelihood approach to least absolute deviation regression. *EURASIP Journal on Advances in Signal Processing*, 2004(12):1–8, 2004.

[59] Andres Legarra and I Misztal. Computing strategies in genome-wide selection. *Journal of dairy science*, 91(1):360–366, 2008.

[60] Alencar Xavier, William M Muir, Bruce Craig, and Katy Martin Rainey. Walking through the statistical black boxes of plant breeding. *Theoretical and applied genetics*, 129(10):1933–1949, 2016.

[61] Marco Signoretto and Johan A. K. Suykens. *Kernel Methods*, pages 577–605. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[62] Paul M VanRaden. Efficient methods to compute genomic predictions. *Journal of dairy science*, 91(11):4414–4423, 2008.

[63] JM González-Camacho, G de Los Campos, P Pérez, D Gianola, JE Cairns, G Mahuku, R Babu, and J Crossa. Genome-enabled prediction of genetic values using radial basis function neural networks. *Theoretical and Applied Genetics*, 125(4):759–771, 2012.

[64] Jaime Cuevas, Osval Montesinos-López, Philomin Juliana, Carlos Guzmán, Paulino Pérez-Rodríguez, José González-Bucio, Juan Burgueño, Abelardo Montesinos-López, and José Crossa. Deep kernel for genomic and near infrared predictions in multi-environment breeding trials. *G3: Genes, Genomes, Genetics*, 9(9):2913–2924, 2019.

[65] José Crossa, Johannes WR Martini, Daniel Gianola, Paulino Pérez-Rodríguez, Diego Jarquin, Philomin Juliana, Osval Montesinos-López, and Jaime Cuevas. Deep kernel and deep learning for genome-based prediction of single traits in multienvironment breeding trials. *Frontiers in genetics*, 10:1168, 2019.

[66] Shizhong Xu. Mapping quantitative trait loci by controlling polygenic background effects. *Genetics*, 195(4):1209–1222, 2013.

[67] I Misztal and A Legarra. Invited review: efficient computation strategies in genomic selection. *animal*, 11(5):731–736, 2017.

[68] I Pocrnic, DAL Lourenco, Y Masuda, A Legarra, and I Misztal. Limited dimensionality of genomic information and effective population size. In *Proceedings of the World Congress on Genetics Applied to Livestock Production*, volume 11, page 32, 2018.

[69] CR Henderson. Inverse of a matrix of relationships due to sires and maternal grandsires. *Journal of Dairy Science*, 58(12):1917–1921, 1975.

[70] Tony Cai, Weidong Liu, and Xi Luo. A constrained 1 minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, 106(494):594–607, 2011.

[71] Weidong Liu and Xi Luo. Fast and adaptive sparse precision matrix estimation in high dimensions. *Journal of multivariate analysis*, 135:153–162, 2015.

[72] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[73] Dongpo Xu, Shengdong Zhang, Huisheng Zhang, and Danilo P Mandic. Convergence of the rmsprop deep learning method with penalty for nonconvex optimization. *Neural Networks*, 2021.

[74] Boris Hanin. Universal function approximation by deep neural nets with bounded width and relu activations. *Mathematics*, 7(10):992, 2019.

[75] Pierre Baldi and Peter Sadowski. The dropout learning algorithm. *Artificial intelligence*, 210:78–122, 2014.

[76] Sangchul Hahn and Heeyoul Choi. Understanding dropout as an optimization trick. *Neurocomputing*, 398:64–70, 2020.

[77] Leo Breiman, JH Friedman, RA Olshen, and CJ Stone. Classification and regression trees, 1st edn, vol. 1, 1984.

[78] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[79] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[80] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[81] Daniel F Schwarz, Inke R König, and Andreas Ziegler. On safari to random jungle: a fast implementation of random forests for high-dimensional data. *Bioinformatics*, 26(14):1752–1758, 2010.

[82] Xi Chen and Hemant Ishwaran. Random forests for genomic data analysis. *Genomics*, 99(6):323–329, 2012.

[83] Vincent Botta, Gilles Louppe, Pierre Geurts, and Louis Wehenkel. Exploiting snp correlations within random forest for genome-wide association studies. *PloS one*, 9(4):e93379, 2014.

[84] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. *Advances in neural information processing systems*, 28:2503–2511, 2015.

[85] Manuel Carabantes. Black-box artificial intelligence: an epistemological and critical analysis. *AI & SOCIETY*, pages 1–9, 2019.

[86] Daniel Varona, Yadira Lizama-Mue, and Juan Luis Suárez. Machine learning's limitations in avoiding automation of bias. *AI & SOCIETY*, pages 1–7, 2020.

[87] John F Doebley, Brandon S Gaut, and Bruce D Smith. The molecular genetics of crop domestication. *Cell*, 127(7):1309–1321, 2006.

[88] Arthur Korte and Ashley Farlow. The advantages and limitations of trait analysis with gwas: a review. *Plant methods*, 9(1):1–9, 2013.

[89] Shizhong Xu. Theoretical basis of the beavis effect. *Genetics*, 165(4):2259–2268, 2003.

[90] Thomas Hellström, Virginia Dignum, and Suna Bensch. Bias in machine learning-what is it good for? In *International Workshop on New Foundations for Human-Centered AI (NeHuAI) co-located with 24th European Conference on Artificial Intelligence (ECAI 2020), Virtual (Santiago de Compostela, Spain), September 4, 2020*, pages 3–10. RWTH Aachen University, 2020.

Appendix: a general mixed model cheat sheet

Index

1) Model statement

A linear Gauss-Markov models is presented as:

$$\mathbf{y} = \mathbf{Xb} + \boldsymbol{\varepsilon}$$
$$\mathbf{y} \sim \mathrm{N}(\mathbf{Xb}, \mathbf{V})$$
$$\boldsymbol{\varepsilon} \sim \mathrm{N}(0, \mathbf{V})$$

And independent random terms other than the residuals:

$$\boldsymbol{\varepsilon} = \sum_{\mathrm{I}}^{\mathrm{i}=1} \mathbf{Z}_\mathrm{i}\mathbf{u}_\mathrm{i} + \mathbf{e}$$

And the variance-covariance matrix is defined by

$$\mathrm{E}[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}'] = \mathbf{V} = \sum_{\mathrm{I}}^{\mathrm{i}=1} \mathbf{Z}_\mathrm{i}\mathbf{K}_\mathrm{i}\mathbf{Z}_\mathrm{i}'\sigma_\mathrm{i}^2 + \mathbf{I}\sigma_\mathrm{e}^2$$

Reshape the model the model into:

$$\mathbf{y} = \mathbf{Xb} + \sum_{\mathrm{I}}^{\mathrm{i}=1} \mathbf{Z}_\mathrm{i}\mathbf{u}_\mathrm{i} + \mathbf{e}$$
$$\mathbf{u} \sim \mathrm{N}\left(0, \mathbf{Z}_\mathrm{i}\mathbf{K}_\mathrm{i}\mathbf{Z}_\mathrm{i}'\sigma_\mathrm{i}^2\right)$$
$$\mathbf{e} \sim \mathrm{N}(0, \mathbf{I}\sigma_\mathrm{e}^2)$$

2) Likelihood

Gaussian likelihood is defined as:

$$L = 2\pi^{-0.5\mathrm{n}}\,|\mathbf{V}|^{-0.5}\exp(-0.5(\mathbf{y}-\mathbf{Xb})'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{Xb}))$$

Log likelihood:

$$\ln L = -\frac{1}{2}[\ln|\mathbf{V}| + \mathbf{y}'\mathbf{P}\mathbf{y}]$$

And restricted log likelihood (log REML) is defined as:

$$\ln L = -\frac{1}{2}\left[\ln|\mathbf{V}| + \ln\left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right| + \mathbf{y}'\mathbf{P}\mathbf{y}\right]$$

Where the projection matrix $\mathbf{P}$ is defined by:

$$\mathbf{P} = \mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}$$

Properties and equalities of projection matrices ($\mathbf{S}$ and $\mathbf{P}$):

$$\mathbf{SX} = \mathbf{PX} = 0, \quad \mathbf{Py} = \mathbf{V}^{-1}(\mathbf{y}-\mathbf{Xb}), \quad \mathbf{PP} = \mathbf{P}$$
$$\mathbf{PVP} = \mathbf{P}, \quad \mathbf{y}'\mathbf{Py} = \mathbf{y}'\mathbf{e}\,\sigma_\mathrm{e}^{-2}, \quad \mathbf{P} = \mathbf{V}^{-1}\mathbf{S}$$
$$\mathbf{S} = \mathbf{I} - \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}, \quad \mathbf{Sy} = \mathbf{y} - \mathbf{Xb}, \quad \mathbf{SS} = \mathbf{S}$$
$$\mathbf{ZAZ}'\mathbf{Py} = \mathbf{Zu}\sigma_\mathrm{u}^{-2}, \mathbf{Py} = (\mathbf{y} - \mathbf{Xb} - \mathbf{Zu})\sigma_\mathrm{e}^{-2}$$
$$\mathbf{V}^{-1} = \mathbf{I} - \mathbf{Z}(\mathbf{Z}'\mathbf{Z} + \lambda\mathbf{K}^{-1})^{-1}\mathbf{Z}'$$

3) BLUE and BLUP

The solution for the best linear unbiased estimators ($\mathbf{b}$, *BLUE*) and best linear unbiased predictor ($\mathbf{u}$, *BLUP*) are commonly defined through the equations:

$$\mathbf{b} = (\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{y}$$
$$\mathbf{u} = \sigma_\mathrm{i}^2\mathbf{K}_\mathrm{i}\mathbf{Z}_\mathrm{i}'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{Xb})$$

Conditional to the random terms, BLUE coefficients can be estimated from solving the system of equation:

$$(\mathbf{X}'\mathbf{X})\,\mathbf{b} = \mathbf{X}'(\mathbf{y} - \textstyle\sum \mathbf{Z}_\mathrm{i}\mathbf{u}_\mathrm{i})$$

And coefficients of the i^{th} random effect, conditional to fixed effects and other random effects, as:

$$\left(\mathbf{Z}_\mathrm{i}'\mathbf{Z}_\mathrm{i} + \mathbf{K}_\mathrm{i}^{-1}\lambda_\mathrm{i}\right)\mathbf{u} = \mathbf{Z}_\mathrm{i}'(\mathbf{y} - \mathbf{Xb} - \textstyle\sum \mathbf{Z}_{-\mathrm{i}}\mathbf{u}_{-\mathrm{i}})$$

Where $\lambda_\mathrm{i} = \sigma_\mathrm{e}^2\sigma_\mathrm{i}^{-2}$.

4) General terms

In a more generalized notation, the mixed model equation is described as:

$$\mathbf{y} = \mathbf{Xb} + \sum_{\mathrm{I}}^{\mathrm{i}=1} \mathbf{Z}_\mathrm{i}\mathbf{u}_\mathrm{i} + \mathbf{e} = \mathbf{Wg} + \mathbf{e}$$

Where the design matrices and coefficients are unified as:

$$\mathbf{W} = \mathbf{X}|\mathbf{Z}_1|\mathbf{Z}_{\dots}|\mathbf{Z}_\mathrm{I}$$
$$\mathbf{g} = \mathbf{b}|\mathbf{u}_1|\mathbf{u}_{\dots}|\mathbf{u}_\mathrm{I}$$

And the mixed model can be written as:

$$\mathbf{y} = \mathbf{Wg} + \mathbf{e}$$
$$\mathbf{y} \sim \mathrm{N}(\mathbf{Wg}, \mathbf{I}\sigma_\mathrm{e}^2)$$

Where the coefficients are, in probabilistic terms, described as a normal distribution with a block diagonal variance:

$$\mathbf{g} \sim \mathrm{MVN}\left(\begin{bmatrix} b \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \infty & 0 & \cdots & 0 \\ 0 & \mathbf{K}_1\sigma_1^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{K}_\mathrm{I}\sigma_\mathrm{I}^2 \end{bmatrix}\right)$$

Thus, the mixed model equation is generalized as:

$$\mathbf{W}'\mathbf{W} + \boldsymbol{\Sigma} = \mathbf{W}'\mathbf{y}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & \mathbf{K}_1^{-1}\lambda_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{K}_\mathrm{I}^{-1}\lambda_\mathrm{I} \end{bmatrix}$$

And the mixed model equation makes the complex problem a simple "Ax = b" mathematic problem:

$$\mathbf{C} = \mathbf{gr}$$
$$\mathbf{C} \rightarrow \mathbf{W}'\mathbf{W} + \boldsymbol{\Sigma}$$
$$\mathbf{r} \rightarrow \mathbf{W}'\mathbf{y}$$

5) Coefficients

The conditional expectation of each regression coefficient, assuming $\mathbf{K_i} = \mathbf{I}$, is expressed as:

$$g_j = \frac{\mathbf{w}_j'(\mathbf{y} - \mathbf{W}_{-j}\mathbf{g}_{-j})}{\mathbf{w}_j'\mathbf{w}_j + \lambda_j} = \frac{\mathbf{w}_j'\tilde{\mathbf{y}}}{\mathbf{w}_j'\mathbf{w}_j + \lambda_j}$$

Where w_j from fixed effect coefficients assume $\lambda = 0$.

Conditional to all other terms via *Gauss-Seidel residual update* (GSRU), coefficients can be efficiently estimated in two steps:

$$g_j^{t+1} = \frac{\mathbf{w}_j'\mathbf{e}^t + \mathbf{w}_j'\mathbf{w}_j g_j^t}{\mathbf{w}_j'\mathbf{w}_j + \lambda_j}$$
$$\mathbf{e}^{t+1} = \mathbf{e}^t - \mathbf{w}_j(g_j^{t+1} - g_j^t)$$

6) Kernel trick

It is also possible to expand this framework to scenarios where $\mathbf{K_i} \neq \mathbf{I}$, that is through the diagonalization of $\mathbf{K}_i$ by spectral decomposition:

$$\mathbf{K}_i = \mathbf{U}_i\mathbf{D}_i\mathbf{U}_i'$$

Where $\mathbf{U}$ is a matrix of Eigenvectors and $\mathbf{D}$ is a diagonal matrix of Eigenvalues. With properties

$$\mathbf{K^{-1}} = \mathbf{UD^{-1}U'}$$
$$\mathbf{K'K} = \mathbf{UD^2U'}$$
$$\mathbf{D} = \mathbf{U'KU}$$
$$\mathbf{I} = \mathbf{U'U}$$
$$\mathbf{U'} = \mathbf{U^{-1}}$$
$$\mathrm{tr}(\mathbf{K}) = \mathrm{tr}(\mathbf{D})$$

Such that:

$$\mathbf{u_i} \sim \mathrm{N}(0, \mathbf{Z}_i\mathbf{K}_i\mathbf{Z}_i'\sigma_i^2)$$
$$\mathbf{u_i} \sim \mathbf{U}\,\mathrm{N}(0, \mathbf{Z}_i\mathbf{D}_i\mathbf{Z}_i'\sigma_i^2)$$
$$\mathbf{a_i} \sim \mathrm{N}(0, \mathbf{D}_i\sigma_i^2)$$
$$\mathrm{E}[\mathbf{u_i}] = \mathbf{U}_i\mathbf{a}_i$$
$$\mathbf{Z}_i\mathbf{u}_i = \mathbf{Z}_i\mathbf{U}_i\mathbf{a}_i$$

Assume the notation:

$$\mathbf{B}_i = \mathbf{Z}_i\mathbf{U}_i$$

Then, for i^{th} random effect and j^{th} Eigen-pair, the regression coefficient has expectation (GSRU):

$$g_{ij} = \frac{\mathbf{b'}\tilde{\mathbf{y}}}{\mathbf{b'b} + \lambda_i d_j^{-1}}, \qquad \mathbf{b} = \mathbf{Z}_i\mathbf{U}_{ij}$$

7) Information matrix

$$\boldsymbol{V}_{(\mathrm{Zu})} = \frac{\partial \mathbf{V}}{\partial \sigma_i^2} = \mathbf{Z_i K_i Z_i} \quad \text{and} \quad \boldsymbol{V}_{(\mathrm{e})} = \frac{\partial \mathbf{V}}{\partial \sigma_e^2} = \mathbf{I}$$

$$\text{Information} = \mathrm{P}\frac{\partial \mathbf{V}}{\partial \sigma_i^2}\mathrm{P}\frac{\partial \mathbf{V}}{\partial \sigma_j^2} = \mathrm{PV_iPV_j}$$

8) Variance components

First derivative 1: Expectation-maximization

$$\sigma_e^2 = \frac{\mathbf{y'Py}}{\mathrm{tr}(\mathbf{P})} = \frac{\mathbf{e'e} + \mathrm{tr}(\mathbf{WC^{-1}W'})\sigma_e^{2^{t-1}}}{n} = \frac{\mathbf{y'e}}{n-r}$$
$$\sigma_i^2 = \frac{\mathbf{u}_i'\mathbf{K_i^{-1}u}_i + \mathrm{tr}(\mathbf{K}_i^{-1}\mathbf{C^{ii}})\sigma_e^2}{q_i} = \frac{\mathbf{u}_i'\mathbf{K_i^{-1}u}_i}{q_i - \lambda\,\mathrm{tr}(\mathbf{K}_i^{-1}\mathbf{C^{ii}})}$$

n = number of observations
r = rank of $\mathbf{X}$ (number of columns of $\mathbf{X}$)
q_i = number of columns of $\mathbf{Z}_i$

First derivative 2: Gibbs sampling (flat priors: $S = 0, \nu = -2$)

$$\sigma_e^2 \sim \frac{\mathbf{e'e} + S_e\nu_e}{\chi^2_{n+\nu_e}} \quad \text{and} \quad \sigma_i^2 \sim \frac{\mathbf{u}_i'\mathbf{K_i^{-1}u}_i + S_i\nu_i}{\chi^2_{q_i+\nu_i}}$$

First derivative 3: Pseudo-expectation

$$\sigma_e^2 = \frac{\mathbf{y'e}}{n-r} \quad \text{and} \quad \sigma_i^2 = \frac{\mathbf{y'S'Z}_i\mathbf{u}_i}{\mathrm{tr}(\mathbf{SZ}_i\mathbf{K}_i\mathbf{Z}_i)}$$

Method of moments: MIVQUE ($\mathbf{A} = \mathbf{PVP}$)

$$\begin{bmatrix} \mathrm{tr}(\mathbf{PZKZ'PZKZ'}) & \mathrm{tr}(\mathbf{PZKZ'P}) \\ \mathrm{tr}(\mathbf{PZKZ'P}) & \mathrm{tr}(\mathbf{P}) \end{bmatrix} \begin{bmatrix} \sigma_u^2 \\ \sigma_e^2 \end{bmatrix} = \begin{bmatrix} \mathbf{y'PZKZ'Py} \\ \mathbf{y'Py} \end{bmatrix}$$

Second derivative 1: Fisher scoring

$$\mathbf{v}^{t+1} = \mathbf{v}^t - \mathbf{I}^{-1}\mathbf{s}, \qquad \mathbf{v} = \begin{bmatrix} \sigma_u^2 \\ \sigma_e^2 \end{bmatrix}$$

$$\mathbf{I} = \begin{bmatrix} \mathrm{tr}(\mathbf{PZKZ'PZKZ'}) & \mathrm{tr}(\mathbf{PZKZ'P}) \\ \mathrm{tr}(\mathbf{PZKZ'P}) & \mathrm{tr}(\mathbf{P}) \end{bmatrix}$$

$$\mathbf{s} = \begin{bmatrix} \mathrm{tr}(\mathbf{PZKZ'}) - \mathbf{y'PZKZ'Py} \\ \mathrm{tr}(\mathbf{P}) - \mathbf{y'Py} \end{bmatrix}$$

Second derivative 2: Average information

$$\mathbf{v}^{t+1} = \mathbf{v}^t - \mathbf{AI}^{-1}\mathbf{s}, \qquad \mathbf{v} = \begin{bmatrix} \sigma_u^2 \\ \sigma_e^2 \end{bmatrix}$$

$$\mathbf{AI} = 0.5 \times \begin{bmatrix} \mathbf{y'PZKZ'PZKZ'Py} & \mathbf{y'PZKZ'Py} \\ \mathbf{y'PZKZ'Py} & \mathbf{y'Py} \end{bmatrix}$$

$$\mathbf{s} = \begin{bmatrix} \mathrm{tr}(\mathbf{PZKZ'}) - \mathbf{y'PZKZ'Py} \\ \mathrm{tr}(\mathbf{P}) - \mathbf{y'Py} \end{bmatrix}$$

Derivation-free method: EMMA

$$\mathbf{ZKZ'} = \mathbf{UDU'}$$

$$\mathbf{b} = (\mathbf{X'V^{-1}X})^{-1}\mathbf{X'V^{-1}y}$$
$$\sigma_e^2 = \frac{(\mathbf{y} - \mathbf{Xb})'\mathbf{V}^{-1}(\mathbf{y} - \mathbf{Xb})}{n-r}$$

$$\mathrm{argmax}\, L(\lambda) = -0.5\left[\ln|\mathbf{V}| + \ln\left|\mathbf{X'V^{-1}X}\right| + \mathbf{y'Py}\right]$$

$$\ln|\mathbf{V}| = \sum \ln(d_j\lambda^{-1} + 1)$$
$$\ln|\mathbf{X'V^{-1}X}| = \ln\left|\mathbf{X'U}(\mathbf{D^{-1}}\lambda^{-1} + \mathbf{I})\mathbf{U'X}\right|$$
$$\mathbf{y'Py} = \mathbf{y'e}\,\sigma_e^2$$