ABSTRACT

In the next few years, with the advent of high-density single nucleotide polymorphism (SNP) arrays and genome sequencing, genomic evaluation methods will need to deal with a large number of genetic variants and an increasing sample size. The boosting algorithm is a machine-learning technique that may alleviate the drawbacks of dealing with such large data sets. This algorithm combines different predictors in a sequential manner with some shrinkage on them; each predictor is applied consecutively to the residuals from the committee formed by the previous ones to form a final prediction based on a subset of covariates. Here, a detailed description is provided and examples using a toy data set are included. A modification of the algorithm called “random boosting” was proposed to increase predictive ability and decrease computation time of genome-assisted evaluation in large data sets. Random boosting uses a random selection of markers to add a subsequent weak learner to the predictive model. These modifications were applied to a real data set composed of 1,797 bulls genotyped for 39,714 SNP. Deregressed proofs of 4 yield traits and 1 type trait from January 2009 routine evaluations were used as dependent variables. A 2-fold cross-validation scenario was implemented. Sires born before 2005 were used as a training sample (1,576 and 1,562 for production and type traits, respectively), whereas younger sires were used as a testing sample to evaluate predictive ability of the algorithm on yet-to-be-observed phenotypes. Comparison with the original algorithm was provided. The predictive ability of the algorithm was measured as Pearson correlations between observed and predicted responses. Further, estimated bias was computed as the average difference between observed and predicted phenotypes. The results showed that the modification of the original boosting algorithm could be run in 1% of the time used with the original algorithm and with negligible differences in accuracy and bias. This modification may be used to speed the calculus of genome-assisted evaluation in large data sets such as those obtained from consortiums.

Key words: genomic evaluation, boosting, machine learning, predictive ability

INTRODUCTION

In the last years, several methods have been proposed to incorporate high-density marker information in genetic evaluations (Meuwissen et al., 2001; Gianola et al., 2006; Gonzalez-Recio et al., 2008; Aguilar et al., 2010). These methods are based on either linear regression on the marker effects [e.g., Bayes-B, Bayesian LASSO (least absolute shrinkage and selection operator)] or in genomic covariance between genotyped individuals [e.g., genomic (G)-BLUP, single-step G-BLUP]. These methods are supposed to deal with the curse of dimensionality problem, although concerns have been raised about their convenience in analyzing high-dimensional data (Gianola et al., 2010). Nonparametric models from the machine-learning repository have been proposed as an alternative in genome-assisted evaluations because they are able to extract hidden relationships from large, noisy, and redundant data and do not follow a particular parametric design. For instance, reproducing kernel Hilbert spaces (RKHS; Gonzalez-Recio et al., 2008), radial basis functions (Long et al., 2010), random forest (Gonzalez-Recio and Forni, 2011), neural networks (Gianola et al., 2011), or the boosting algorithm (Gonzalez-Recio et al., 2010) have already been implemented in this context. In general, previous results showed that nonparametric methods have similar or better predictive accuracy than regression on SNP and genomic relationship matrices. Further, machine-learning methods are attractive and flexible for the implementation of genome-assisted evaluation using high-density SNP arrays: SNP chips include increasing numbers of SNP, and sequence data may soon be available increasing the computation requirements. Thus, new strategies need to be developed to deal with reference population samples with a larger number of
genotyped individuals with chips including an increasing number of SNP.

The gradient boosting algorithm (BOOST) is an interesting alternative in a genome-assisted evaluation context when many more animals and markers are genotyped or sequenced, because it performs variable selection, uses simple regression models in an additive fashion, and is computationally fast and easy. The gradient boosting algorithm is a machine-learning algorithm classified as an ensemble method. It was first proposed by Freund and Schapire (1996) for classification problems and was known as AdaBoost. Since then, it has been used in many fields and shows similar or higher predictive accuracy than traditional methods, in both classification and regression problems. The boosting algorithm has been previously used in the genome-wide prediction of genetic merit and disease susceptibility in animal breeding (González-Recio et al., 2010; González-Recio and Forni, 2011) and showed similar or higher accuracy than other methods, such as Bayes A or Bayesian LASSO. The algorithm uses a reference data set to find a predictive model that, given some genotype markers (e.g., SNP), predicts the most likely genetic merit for individuals yet to be observed. It does not assume any particular mode of inheritance or parametric model, and as noted above, is suitable for analyzing very high dimensional, redundant, and fuzzy data such as high-density SNP chips.

Nonetheless, BOOST, like any other method used in a genome-assisted evaluation context, has yet to deal with the estimation of regression equation on markers when several thousand genotyped animals are used in the reference population (Van Raden et al., 2011), such as in the case of the EuroGenomics consortium, in which more than 22,000 genotypes are already available as a reference population. These methods need to be adapted or modified to be implemented in the new era of genomic evaluations, with many more genotypes and phenotypes, to predict genetic merit of young sires and cows in an accurate manner with minimum computer requirements.

The objective of this article is to provide a comprehensive description of the boosting algorithm in a genome-assisted genetic evaluation context, and to propose modifications thereof to deal with the larger number of genotypes and phenotypes in genomic evaluations. The paper is organized as follows: first, we provide a brief description of ensemble methods and then we detail the gradient boosting algorithm in the context of a genome-assisted evaluation. The implementation of gradient boosting is illustrated in a toy data set example using 2 different base regression functions: ordinary least square (OLS) and RKHS regression.

We propose a modification of the algorithm for implementation in genome-assisted evaluations with many more phenotypes and genotypes. Finally, we apply this modification to a real data set and compare it with the original BOOST. Comparison with other methods commonly used in this context is provided in the companion paper (Jiménez-Montero et al., 2013) in a real genomic evaluation problem.

MATERIALS AND METHODS

Brief Description of Ensemble Methods

Ensemble methods are a linear combination of some models instead of using a single fit of the model (Hastie et al., 2009; Seni and Elder, 2010), which can be expressed in the form

\[ y = c_0 + c_1 h_1(y;X) + c_2 h_2(y;X) + \ldots + c_m h_m(y;X) + \ldots + c_M h_M(y;X) + e, \]

where \( y \) is the vector of observed phenotypes, \( h_m(y;X)(m \in \{1,\ldots,M\}) \) is some sort of model or function implemented on the phenotypes and genotypes in some specified manner, \( c_0 \) is the population mean, and \( c_m(m \in \{1,\ldots,M\}) \) are the coefficients or weights for each model, and \( e \) is the vector of the corresponding residuals. Each model \( h_m(y;X) \) is usually called “weak learner” because they are simple models that are supposed to perform slightly better than random guessing. It is important to point out that little improvement would be gained with a strong learner, and computation time would increase significantly. The ensemble methods form a “committee” of predictors with potentially greater predictive ability than that of any of the individual predictors. They became popular as a relatively simple device to improve the predictive performance of a base procedure. Random forest, bagging, and boosting are examples of ensemble methods. They have been used in different fields and may be implemented in studies using large amount of genomic information.

Gradient Boosting

Gradient boosting is considered an ensemble method (Hastie et al., 2009). This algorithm combines different predictors in a sequential manner with some shrinkage on them (Friedman, 2001). It also performs variable selection.
Gradient boosting, as an ensemble method, may be described as follows:

\[ y = \mu + \sum_{m=1}^{M} v_{h_m} (y, X) + e, \]

where \( y \) is the vector of observed phenotypes, \( \mu \) is a population mean, \( v \) is a shrinkage factor, \( h_m \) is a predictor model, \( y_{\text{m}} \) is the matrix of corresponding genotypes, and \( e \) is the vector of residuals. Each predictor \( h_{\text{m}} (y, X) \) for \( m \in [1, M] \) is added in a sequential manner, and is applied consecutively to the residuals from the committee formed by the previous ones, weighted by \( c_{\text{m0}} = v \). This algorithm can be calculated using importance sampling learning ensembles as described here:

\[(\text{Initialization}): \text{Given data} (y, X), \text{let the prediction of phenotypes be} \ F_0 = \bar{y}. \text{Then, for} \ m \in \{1 \text{ to } M\}, \text{with} \ M \text{ being large, calculate the loss function} \ (L) \text{ for} \ \left[ y_i, F_{m-1} (x_i) + h (y_i, x_i, p_m) \right], \text{where} \ p_m \text{ is the SNP \ (only 1 SNP is selected at each iteration) that minimizes} \ \sum_{i=1}^{n} L \left[ y_i, F_{m-1} (x_i) + h (y_i, x_i, p_m) \right] \text{ at iteration} \ m; \]

\[ h (y_i, x_i, p_m) \text{ is the prediction of the observation using learner} \ h (\cdot) \text{ on SNP} \ p_m. \text{ Selection of SNP} \ p_m \text{ may be based on the minimization of the loss function} \ L (\cdot) \text{ in the training set or in a tuning set previously put aside in an} \ n \text{-fold cross-validation scenario.} \]

Next, update the predictions at iteration \( m \) in the form \( F_m (x_i) = F_{m-1} (x_i) + v \cdot h (y_i, x_i, p_m), \) with \( v \in (0, 1) \) being some shrinkage factor; for example, \( v = 0.01 \).

Each subsequent model is trained on the residuals of the previous model, which are actually residual estimates \( (\hat{e}) \). These \( \hat{e} \) are expected to be identical and independently distributed as \( \hat{e} \sim N \left( 0, \sigma_{\varepsilon_m}^2 \right) \), where \( \sigma_{\varepsilon_m}^2 \) is the residual variance for model \( m \). Therefore, the larger \( M \), the smaller \( \sigma_{\varepsilon_m}^2 \). This means that for larger \( m \), the contribution of the selected SNP at \( m \) is expected to be smaller. The shrinkage parameter \( v \) aims to control this tradeoff between number of models and importance of the SNP. When \( v \) is smaller, less variance from the residuals is subtracted at each iteration; therefore, new (or the same) SNP are allowed to explain the remaining residual variance.

Note that a large variety of learners \( h_{\text{m}} (y, X) \) and loss functions \( \left\{ L (y, F_{\text{m}} (x_i)) \right\} \) may be proposed, each leading to a different boosting model. For instance, classification and regression trees, generalized least squares regression, or nonparametric kernel regression may be used as weak learners. A quadratic error term, the exponential \( L_1 \) loss function, the Gini index, or the Huber loss function are some examples of loss functions that may be implemented within the algorithm.

The choice of the number of iterations, \( M \), is a model comparison problem that may be overcome in many different ways (Friedman, 2001; Hastie et al., 2009; González-Recio et al., 2010). This parameter may control the complexity of the ensemble and the overfitting caused in the training set. A simple manner of choosing \( M \) is stopping the algorithm when the decrease in error rate or mean squared error in a tuning set is not relevant during a large enough number of iterations (e.g., 100). Once the coefficient and the weak learners have been estimated, predictions for yet-to-be-observed records may be calculated as

\[ \hat{y}_i = \hat{F}_m (x_i) = \hat{\mu} + \sum_{m=1}^{M} v_{h_m} (x_i). \]

More details on gradient boosting can be found in Freund and Schapire (1996) and Friedman (2001), and its implementation on genomic prediction is described in González-Recio et al. (2010).

Following below is a toy data example to describe the procedure to compute predicted genomic merit of genotyped individuals using 2 different weak learner models: OLS and RKHS regression.

**Illustrations**

Let \( y = \begin{bmatrix} 21.08 & 16.13 & 18.41 & 20.50 & 12.95 \end{bmatrix} \) be the vector of observed phenotypes for \( n = 5 \) individuals. Each individual is genotyped for \( p = 3 \) SNP codified as 0, 1, or 2 if they share 0, 1, or 2 copies, respectively, of the most frequent allele in the population (as an arbitrary coding example). Let the corresponding \( X \) matrix be

\[
X = \begin{bmatrix}
1 & 2 & 0 \\
0 & 2 & 1 \\
2 & 0 & 1 \\
1 & 2 & 2 \\
1 & 2 & 2
\end{bmatrix}.
\]

The mean estimate \( \bar{y} \) for trait \( y \) is 17.81. The algorithm is initialized, setting \( F_0 = 17.81 \) for all individuals. Note that other environmental effects may be included to adjust the phenotype. Let the loss function be the mean squared error (MSE) and the shrinkage coefficient \( v = 0.9 \). This value is used only for illustra-
tion purposes and a smaller value (e.g., v = 0.10) is usually desired.

**Illustration 1: OLS**

Suppose that the weak learner \([h(\cdot)]\) is the OLS regression and the MSE was assumed as loss function \(L(\cdot)\). The ensemble will be constructed adding the results of several of these regressions.

The first model, \(m = 1\), is estimated as follows:

The heuristic search begins by trying \(p = 3\) models in the form

\[
y = F_0 + h_1(y; x_s, s) + e, \text{ with } s \in \{1, 2, p = 3\}.
\]

In this case, \(h(\cdot)\) is a linear regression on SNP \(s\), and the model becomes

\[
y = F_0 + a_1 + b_1 x_i + e,
\]

where \(x_i\) is the column vector of the genotype codes for SNP \(s\).

For simplicity, here the model was solved by least squares estimates, although other estimators such as \(\text{squares estimates, although other estimators such as }\) for SNP \(s = 1\):  

\[
\begin{array}{c|c|c}
\text{SNP} & \hat{y} & \text{MSE} \\
\hline
1 & 21.08 & 17.81 + 0.9 \cdot (-1.137) + 0.9 \cdot 1.137 \\
2 & 16.13 & 17.81 + 0.9 \cdot (-1.137) + 0.9 \cdot 1.137 \\
3 & 16.06 & 17.81 + 0.9 \cdot (-1.137) + 0.9 \cdot 1.137 \\
4 & 20.50 & 17.81 + 0.9 \cdot (-1.137) + 0.9 \cdot 1.137 \\
5 & 12.95 & 17.81 + 0.9 \cdot (-1.137) + 0.9 \cdot 1.137 \\
\end{array}
\]

The heuristic search begins again by trying \(p = 3\) models in the form

\[
y_1 = h_2(y; x_s, s) + e, \text{ with } s \in \{1, 2, p = 3\}.
\]

As before, \(h(\cdot)\) is a linear regression on SNP \(s\), and for the heuristic search the is then \(r_1 = a_2 + b_2 x_i + e\), where \(x_i\) is the column vector of the genotype codes for SNP \(s\).

The solutions for each SNP would be

For \(s = 1\), \(\hat{a}_2 = -0.114\) and \(\hat{b}_2 = 0.114\) with MSE = 8.44;

For \(s = 2\), \(\hat{a}_2 = -0.431\) and \(\hat{b}_2 = 0.269\) with MSE = 8.40;

For \(s = 3\), \(\hat{a}_2 = 2.336\) and \(\hat{b}_2 = -1.946\) with MSE = 6.33.

The SNP \(s = 3\) is selected because it was the one minimizing the MSE. The new estimates are

\[
\begin{array}{c|c|c}
\text{SNP} & \hat{y} & \text{MSE} \\
\hline
1 & 21.08 & 17.81 + 0.9 \cdot (-1.137) + 0.9 \cdot 1.137 \\
2 & 16.13 & 17.81 + 0.9 \cdot (-1.137) + 0.9 \cdot 1.137 \\
3 & 16.06 & 17.81 + 0.9 \cdot (-1.137) + 0.9 \cdot 1.137 \\
4 & 20.50 & 17.81 + 0.9 \cdot (-1.137) + 0.9 \cdot 1.137 \\
5 & 12.95 & 17.81 + 0.9 \cdot (-1.137) + 0.9 \cdot 1.137 \\
\end{array}
\]

The SNP selected at each iteration were \([1, 3, 3, 2, 1, 2, 1, 2, 1, 2, 1]\). The predicted genomic merits of individuals in the toy data set were

\[
\begin{array}{c|c|c}
\text{SNP} & \hat{y} & \text{MSE} \\
\hline
1 & 21.08 & 17.81 + 0.9 \cdot (-1.137) + 0.9 \cdot 1.137 \\
2 & 16.13 & 17.81 + 0.9 \cdot (-1.137) + 0.9 \cdot 1.137 \\
3 & 16.06 & 17.81 + 0.9 \cdot (-1.137) + 0.9 \cdot 1.137 \\
4 & 20.50 & 17.81 + 0.9 \cdot (-1.137) + 0.9 \cdot 1.137 \\
5 & 12.95 & 17.81 + 0.9 \cdot (-1.137) + 0.9 \cdot 1.137 \\
\end{array}
\]

As before, \(h(\cdot)\) is a linear regression on SNP \(s\), and for the heuristic search the is then \(r_1 = a_2 + b_2 x_i + e\), where \(x_i\) is the column vector of the genotype codes for SNP \(s\).

The solutions for each SNP would be

For \(s = 1\), \(\hat{a}_2 = -0.114\) and \(\hat{b}_2 = 0.114\), with MSE = 8.44;

For \(s = 2\), \(\hat{a}_2 = -0.431\) and \(\hat{b}_2 = 0.269\), with MSE = 8.40;

For \(s = 3\), \(\hat{a}_2 = 2.336\) and \(\hat{b}_2 = -1.946\), with MSE = 6.33.
For generalization, it can be shown that the non-parametric genomic merit of any individual can be captured using OLS regression as weak learner is
\[
\hat{y} = x^\top \hat{a} + \hat{b}^T x
\]
with \( \hat{a} \) and \( \hat{b} \) being the intercept and slope coefficient estimates, respectively, in model \( m \), and \( x \) the vector for the corresponding genotypes codes for SNP selected at model \( m \).

Here, the intercept estimates can be added to compute a global intercept \((\hat{a}_g)\) that may be interpreted as a bias corrector: \( \hat{a}_g = v \sum a \hat{a} + \sum \hat{b} \hat{m} \), where \( \hat{b} = (\hat{b}_1, \hat{b}_2, ..., \hat{b}_m, ..., \hat{b}_M) \) is a row vector of \( M \) dimensions containing the slope estimates at each model \( m \in \{1, ..., M\} \).

Then, SNP contribution to the genomic merit \((b_j x)\) of the individual may be expressed as \( \hat{b}_j x = \hat{b}^\top \hat{m} x \), where \( \hat{b} = (\hat{b}_1, \hat{b}_2, ..., \hat{b}_m, ..., \hat{b}_M) \) is a row vector of \( M \) for the genomic breeding value of a given individual would be \( \hat{y}_{g, j} = \hat{a}_g + \hat{b}_j x \).

The global coefficient estimate \((\hat{b}_j)\) for SNP \( j \) is the sum of the slope estimates in the model in which the SNP \( j \) was selected, as
\[
\hat{b}_j = \sum_{m=1}^{M} A_m \hat{b}_m,
\]
where \( A_m \) is an indicator function equal to 1 if the SNP is selected at model \( m \), and 0 otherwise, and \( \hat{b}_m \) is the slope estimate from model \( m \).

It is clear that \( \hat{b}_j \) is a row vector containing the global coefficient estimates for each SNP in the form
\[
\hat{b}_j = \hat{b}^\top \hat{A} = v (\hat{b}_1, \hat{b}_2, ..., \hat{b}_m, ..., \hat{b}_M).
\]

It must be pointed out that although each \( \hat{b}_m \) is calculated from a linear function, the sum of all \( \hat{b}_m \) lacks a linear interpretation because each is calculated from previously corrected phenotypes. Predictions of new genomic breeding values for young genotyped individuals can be easily calculated using the regression equations obtained as described above.

Illustration 2: Kernel Regression

Assume now that the weak learner \( b(\cdot) \) is a nonparametric regression (kernel or RKHS) as described in Kimeldorf and Wahba (1971):
\[
K' y = (K' K + \lambda K) \alpha,
\]
where \( y \) is the vector of phenotypes, \( K = (k_{ij}) \) is an \( n \times n \) matrix of kernels, \( \lambda \) is a smoothing parameter that may be interpreted as the inverse of the variance explained by the kernel matrix, and \( \alpha = (\alpha_1, \alpha_2, ..., \alpha_n) \) is a column vector of \( n \) nonparametric coefficients.

Following the reparameterization I in de los Campos et al. (2009), the model equation can be written as follows:
\[
K + \frac{\sigma^2}{\lambda} I \alpha = y,
\]
with \( \sigma^2 \) being some residual variance. For equivalences between RKHS and BLUP, see de los Campos et al. (2009). Both the residual variance and \( \lambda \) must be estimated in an RKHS scenario. Maximum likelihood or Bayesian estimates from these parameters may be obtained using standard procedures. Further, the model may be simplified using a kernel regression model, as described in Gianola et al. (2006), without needing the estimation of these parameters, using the Nadaraya-Watson estimator (Nadaraya, 1964; Watson, 1964).

Here, for convenience, an RKHS model is proposed but the ratio between \( \sigma^2 \) and \( \lambda \) was assumed equal to 1.

Then, a kernel matrix must be constructed for each SNP. Each matrix \((K^s, s \in \{1, 2, 3\})\) must be semipositive definite and contains the set of quantitave values representing genomic similarities between pairs of individuals \( (k_{ij}^s) \) at a given locus \( s \). A large variety of kernels has proved useful for genomic data (Gonzalez-Recio et al., 2008, 2009; Schaid, 2010). Here, again for simplicity, the allele match kernel was used as illustration; the kernel score assays the number of common alleles between the locus \( s \) of 2 individuals \( i \) and \( j \). The score is 4 if the genotypes of the individuals are the same; 2 if one is a heterozygote and the other is a homozygote, and 0 if they do not share any common allele (i.e., a molecular relationship).

Therefore, the matrix \( K^s = (k_{ij}^s) \) for each SNP \( s \in \{1, 2, 3\} \) would be
function is the MSE, and the shrinkage coefficient \( v = 0.9 \).

The first model, \( m = 1 \), is estimated as follows:

The heuristic search begins by trying \( p = 3 \) models in the form \( \mathbf{y} = \mathbf{F}_0 + \mathbf{h}_1(\mathbf{y}; \mathbf{K}^s, s) + \mathbf{e} \), with \( s \in \{1, 2, 3 \} \), with \( \mathbf{h}(\cdot) \) being the RKHS with \( \mathbf{K}^s \) as kernel matrix. The model is \( \mathbf{y} = \mathbf{F}_0 + \mathbf{K}^s \mathbf{\alpha} + \mathbf{e} \), where \( \mathbf{K}^s \) is the kernel matrix corresponding to locus \( s \), and \( \mathbf{\alpha} \) is the vector of nonparametric coefficients for model \( m = 1 \).

The solutions for each SNP would be as follows:

For \( s = 1 \), \( \hat{\mathbf{\alpha}}_1 = \begin{bmatrix} 3.00 & -0.45 & 0.00 & 2.42 & -5.13 \end{bmatrix} \), with MSE = 8.29;

For \( s = 2 \), \( \hat{\mathbf{\alpha}}_1 = \begin{bmatrix} 3.41 & -1.54 & 0.12 & 2.83 & -4.73 \end{bmatrix} \), with MSE = 8.88;

For \( s = 3 \), \( \hat{\mathbf{\alpha}}_1 = \begin{bmatrix} 0.85 & -1.38 & 0.89 & 3.77 & -3.79 \end{bmatrix} \), with MSE = 6.40.

The SNP \( s = 3 \) produced the smallest MSE and was therefore selected in this case. The new estimates are \( \hat{\mathbf{F}}_1 = \mathbf{y} + v \hat{\mathbf{h}}_1(\mathbf{y}; \mathbf{K}^3) \), with \( \hat{\mathbf{h}}_1(\mathbf{y}; \mathbf{K}^3) = \mathbf{K}^3 \hat{\mathbf{\alpha}}_1 \). The prediction for animal \( i \) now becomes \( \hat{y}_i = \mathbf{y} + \mathbf{k}^3_i \hat{\mathbf{\alpha}}_1 \), with \( v \) being the shrinkage coefficient, \( \hat{\mathbf{\alpha}}_1 = \begin{bmatrix} 0.85 & -1.38 & 0.89 & 3.77 & -3.79 \end{bmatrix} \), and \( \mathbf{k}^3_i = \{ \mathbf{k}^3 \}_i \) containing the vector with the genomic similarities between the individual \( i \) and each individual with record at locus 3.

As with the OLS learner, the SNP minimizing the MSE in the same data set as the one used to estimate \( a \) and \( b \) was selected, but a tuning set may be used, as stated previously.

A second model \( m = 2 \) is then added to the ensemble as \( \mathbf{y} = \hat{\mathbf{F}}_1 + h_2(\mathbf{y}; \mathbf{X}, s) + \mathbf{e} \), with \( s \in \{1, 2, 3 \} \). The model may be written as \( \mathbf{r}_1 = \mathbf{y} - \hat{\mathbf{F}}_1 = h_2(\mathbf{y}; \mathbf{X}, s) + \mathbf{e} \), and the dependent variables in the second model are the residuals obtained from \( m = 1 \):

\[
\begin{bmatrix}
1.09 \\
-1.41 \\
0.86 \\
3.66 \\
-3.90 \\
\end{bmatrix} = \begin{bmatrix}
21.08 \\
16.13 \\
18.41 \\
20.50 \\
12.95 \\
\end{bmatrix} + \begin{bmatrix}
17.81 + 0.9 \cdot -2.42 \\
17.81 + 0.9 \cdot -0.30 \\
17.81 + 0.9 \cdot -0.30 \\
17.81 + 0.9 \cdot -1.08 \\
17.81 + 0.9 \cdot -1.08 \\
\end{bmatrix}.
\]

This yields the new solution

\[
\mathbf{r}_1 = \mathbf{y} - (\hat{\mathbf{F}}_0 + \hat{\mathbf{F}}_1).
\]
The heuristic search begins again by trying \( p = 3 \) models in the form

\[
r_i = h_2 (y; K^*, s) + e = K^* \alpha_i + e,
\]

with \( s \in \{1, 2, p = 3\} \). The solutions for each SNP would be:

- For \( s = 1 \), \( \hat{\alpha}_2 = [0.87, -0.36, 0.10, 3.44, -4.12] \), with MSE = 5.93;
- For \( s = 2 \), \( \hat{\alpha}_2 = [1.22, -1.28, 0.17, 3.79, -3.77] \), with MSE = 6.34;
- For \( s = 3 \), \( \hat{\alpha}_2 = [0.31, -1.25, 1.02, 3.82, -3.74] \), with MSE = 6.25.

The SNP \( s = 1 \) is selected in \( m = 2 \) because it minimized the MSE. The new estimates are \( \hat{F}_2 = \hat{F}_1 + v \hat{h}_2 (y; K^1) \), with \( \hat{h}_2 (y; K^1) = K^1 \hat{\alpha}_2 \). The prediction for animal \( i \) now becomes \( \hat{y}_i = \hat{y} + v k_{i2}^1 \hat{\alpha}_1 + v k_{i2}^2 \hat{\alpha}_2 \), with \( \hat{\alpha}_2 = [0.87, -0.36, 0.10, 3.44, -4.12] \) and \( k_{i2}^1 = \{k_{i2}^m\} \) containing the vector with the genomic similarities between the individual \( i \) and each individual with record at locus 1.

As described previously, subsequent models are added to the residuals of the previous ensemble until a convergence criterion is reached. In this case, the algorithm converged at the second decimal in the MSE (5.71) for \( M = 7 \). The SNP selected at each iteration were \( [3, 1, 3, 1, 1, 3, 1] \). The predicted genomic merits of individuals in the toy data set were

\[
\hat{y} = \hat{F}_7 = \begin{bmatrix} 21.08 \\ 16.16 \\ 18.38 \\ 16.72 \\ 16.72 \end{bmatrix}.
\]

For generalization, it can be shown that the nonparametric genomic merit of any individual using RKHS as weak learner is

\[
\hat{y} = \hat{y} + v \left( \sum_{m=1}^{M} k_{i2}^m \hat{\alpha}_m \right) = \hat{y} + v \sum_{m=1}^{M} k_{i2}^m \hat{\alpha}_m,
\]

with \( \hat{\alpha}_m \) being the nonparametric coefficient estimates at model \( m \), and \( k_{i2}^m = \{k_{i2}^m\} \) the vector containing the genomic similarities between the individual \( i \) and each individual with record at the locus selected at model \( m \). Hence, if the \( \hat{\alpha}_m \) are estimated at each model using the residuals of the previous model they will differ between models, whereas the \( K \) matrix remains constant. Hence, if the phenotype of a new individual has to be predicted, the nonparametric coefficient estimates and the pairs of the genomic similarity between it and the individuals with observation should be computed once and electronically stored. A single text file may be stored for each individual, containing the genomic similarity at each marker position with each individual in the reference population. The algorithm does not need to be run again, and the predictive equations can be computed in a straightforward manner, as with linear regression models.

**Modification of the Boosting Algorithm: Random Boosting**

The purpose of this modification was to speed up the algorithm for large data sets or learners that are time consuming. We proposed to sample \( m_{try} \) covariates, with \( m_{try} \) being the percentage of covariates sampled at random out of the \( p \) SNP at each iteration and select the SNP among the \( m_{try} \) that minimizes the given loss function. Therefore, computation time may be reduced in the order of \( O \left( \frac{m_{try}}{p} \right) \) regarding the original algorithm, as only a small percentage of SNP are tested for minimization of the loss function at each iteration. The parameter \( m_{try} \) may be tuned in the random boosting modification. Studies of similar strategies used in the random forest algorithm showed that a value for \( m_{try} \) of 0.1 \( \times \) \( p \) might achieve satisfactory results (Goldstein et al., 2010).

The boosting algorithm with this modification would work as follows.

(Initialization): Given data \( \Psi = (y, X) \), let the prediction of phenotypes be \( F_0 = \hat{\mu} \).

Then, for \( m \) in \( \{1 \to M\} \), with \( M \) being large, proceed as follows:

1. Draw \( m_{try} \) out of \( p \) covariates from the original training set to construct a reduced training covariate matrix \( \Psi^{(b)} = (y, X_{m_{try}}) \) to train the algorithm in iteration \( m \).
2. Calculate the loss function \( L[y_i, F_{m-1}(x_i) + h(y_i; x_i, m_{try})] \) for all \( m_{try} \) SNP and select that mini-
mizing \( \sum_{i=1}^{n} L[y_i, F_{m-1}(x_i)] + h(y_i; x_i, mtry_m) \) in the tuning set at iteration \( m \), with \( h(y_i; x_i, mtry_m) \) being the prediction of the observation \( i \) in the tuning set using the learned parameters or coefficients of \( h(\cdot) \) on the SNP \( mtry_m \). These parameters or coefficients are learned using the training set as in the original algorithm. Note that if the resulting tuning set is not large enough, it may be recommended to select the SNP that minimizes the loss function in the training set, without leaving a set aside set.

Step 3. Update predictions at iteration \( m \) in the form \( F_m(x_i) = F_{m-1}(x_i) + v \cdot h(y_i; x_i, mtry_m) \), with \( v \) being some shrinkage factor (e.g., \( v = 0.10 \)).

Step 4. Update the residuals to be used in the next iteration as \( y_i = y_i - F_m(x_i) \).

Repeat steps 1 to 4 a large number of times (\( M \)).

This modification changes the order in which SNP are selected in the algorithm with respect to the original boosting, as not all SNP will be tested at each iteration. However, the boosting algorithm is considered as a small-step gradient descent technique (Bühlmann, 2006); therefore, for a sufficiently small \( v \), it is expected that the effect of the order in which the covariates are used to reduce the residual estimates has a small or null effect on the final predictions. Nonetheless, a small data set might yield different results for smaller \( mtry \) and fewer iterations.

**Case Study**

The algorithm and the proposed modification were implemented in a real data set composed of 1,859 genotyped bulls. Full details on genotypes and the editing procedure can be found in Jiménez-Montero et al. (2013). After quality control, 39,714 SNP were kept in the analyses. Sires born before 2005 were used as a training sample (1,601 and 1,574 individuals for production and type traits, respectively), whereas younger sires were used as a testing sample to evaluate predictive ability of the algorithm on yet-to-be-observed phenotypes. Deregressed proofs (DRP) of 4 productive traits: milk yield, fat yield, protein yield, and fat percentage, and 1 type trait: udder depth from January 2009 routine evaluations were used as dependent variables. The DRP were obtained following Jairath et al. (1998). Note that bulls in the testing set did not have progeny test proofs at that time. For convenience, the OLS regression and MSE were chosen as weak learner and loss function, respectively, as set up in the first illustration example above. A 10-fold cross-validation scenario was implemented in the training set. In each fold, nine-tenths of the training data were used to calculate the regression coefficient estimates \( \hat{a}_m \) and \( \hat{b}_m \), and the remaining one-tenth of records were used as a tuning set to choose the SNP minimizing the MSE.

The respective DRP from the December 2011 routine evaluations were used to calculate the predictive ability of the predictions for sires in the testing set. Only sires with more than 20 effective daughter contribution were kept in the testing set (258 and 235 for production and type, respectively). The predictive accuracy was evaluated using Pearson correlation between predicted and observed (December 2011 DRP) response. The predicted bias was also calculated as \( \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i) \), with \( n \) being the number of validation bulls.

The random boosting was applied to this data using a grid of values for \( mtry \) (1, 5, 10, and 50%) and compared with the original boosting (\( mtry = 100\% \)). Further, different values for the smoothing parameter were tested (\( v = 0.01, 0.10, \) and \( 0.20 \)).

**RESULTS AND DISCUSSION**

Tables 1 and 2 show the Pearson correlation and bias, respectively, between predicted and observed phenotype in the testing set, regarding the smoothing parameter \( v \) and \( mtry \) for each trait. In general, the predictive ability of the algorithm was very similar regardless of \( mtry \), with differences of 1 to 2 points in Pearson correlation. Fat percentage showed better predictive ability at larger \( mtry \) values. The known major genes (e.g., \( DGAT1 \)) controlling this trait may partly explain this behavior, as sampling a small proportion of SNP at each iteration may miss markers in these hot spots, hampering the predictive ability of the algorithm. Pearson correlation for \( v = 0.10 \) and \( 0.20 \) were very similar, although \( v = 0.10 \) showed equal or higher Pearson correlation than \( v = 0.20 \) in all the analyses, excepting for udder depth with \( mtry \) equal to 5 and 10%. In terms of bias, the value of \( mtry \) did not show a clear trend, and differences were negligible. Convergence was slower for smaller values of \( v \), because higher shrinkage is done on each coefficient estimate, and a larger number of covariates is needed to explain the variance of the observed phenotypes. Nonetheless, the best combination of \( v \) and \( mtry \) was trait dependent.

As a general recommendation, the random boosting algorithm may be used to speed up the calculus of genome-assisted evaluation without a relevant effect on predictive ability, and in some cases with higher Pearson
correlation between predicted and observed phenotypes in the testing set than using the original algorithm. Smaller values of mtry may be used without decreasing the predictive ability and with a significant reduction in the computation time. Nonetheless, mtry is dependent on the genetic architecture, and a large value is recommended to analyze traits with known major genes, as in the case of fat percentage. The choice of mtry and v is under discussion, and cross-validation is currently the standard procedure. A more formal strategy with statistical properties could be studied in the future.

The original gradient boosting algorithm performed the complete genome-assisted evaluation (10-fold) in 171.67 h with v = 0.01, 69.17 h with v = 0.10, and 50 h with v = 0.20 (Table 3). The computation time was substantially reduced using the modification of the algorithm with mtry = 0.01. The times were 1.5, 0.83, and 0.67 h for mtry = 0.01 and v = 0.01, v = 0.10, and v = 0.20, respectively. These computing times make random boosting feasible for running frequent routine genome-assisted evaluations with large data sets without impairing the predictive accuracy.

### Table 1. Pearson correlation\(^1\) between predicted and observed responses in the testing set using the original gradient boosting algorithm (mtry = 100\%) or its modified version, “random boosting,” for different values of percentage of SNP sampled at each iteration (mtry) and smoothing parameter (v)

<table>
<thead>
<tr>
<th>Trait</th>
<th>mtry (%)</th>
<th>1</th>
<th>5</th>
<th>10</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Milk yield</td>
<td>0.01</td>
<td>0.495</td>
<td>0.502</td>
<td><strong>0.508</strong></td>
<td>0.507</td>
<td>0.507</td>
</tr>
<tr>
<td></td>
<td>0.10</td>
<td>0.487</td>
<td>0.500</td>
<td>0.503</td>
<td><strong>0.508</strong></td>
<td>0.503</td>
</tr>
<tr>
<td></td>
<td>0.20</td>
<td>0.483</td>
<td>0.503</td>
<td>0.503</td>
<td>0.503</td>
<td><strong>0.504</strong></td>
</tr>
<tr>
<td>Fat yield</td>
<td>0.01</td>
<td>0.552</td>
<td>0.561</td>
<td>0.559</td>
<td>0.559</td>
<td>0.559</td>
</tr>
<tr>
<td></td>
<td>0.10</td>
<td>0.567</td>
<td>0.565</td>
<td><strong>0.569</strong></td>
<td>0.556</td>
<td>0.556</td>
</tr>
<tr>
<td></td>
<td>0.20</td>
<td>0.551</td>
<td>0.554</td>
<td>0.562</td>
<td>0.550</td>
<td>0.551</td>
</tr>
<tr>
<td>Protein yield</td>
<td>0.01</td>
<td>0.454</td>
<td>0.443</td>
<td>0.440</td>
<td>0.433</td>
<td>0.443</td>
</tr>
<tr>
<td></td>
<td>0.10</td>
<td><strong>0.466</strong></td>
<td>0.441</td>
<td>0.445</td>
<td>0.444</td>
<td>0.444</td>
</tr>
<tr>
<td></td>
<td>0.20</td>
<td>0.465</td>
<td>0.437</td>
<td>0.429</td>
<td>0.434</td>
<td>0.428</td>
</tr>
<tr>
<td>Fat percentage</td>
<td>0.01</td>
<td>0.746</td>
<td>0.753</td>
<td>0.748</td>
<td>0.763</td>
<td><strong>0.768</strong></td>
</tr>
<tr>
<td></td>
<td>0.10</td>
<td>0.741</td>
<td>0.746</td>
<td>0.748</td>
<td>0.761</td>
<td>0.765</td>
</tr>
<tr>
<td></td>
<td>0.20</td>
<td>0.728</td>
<td>0.737</td>
<td>0.740</td>
<td>0.753</td>
<td>0.767</td>
</tr>
<tr>
<td>Udder depth</td>
<td>0.01</td>
<td>0.496</td>
<td>0.504</td>
<td>0.502</td>
<td>0.509</td>
<td>0.503</td>
</tr>
<tr>
<td></td>
<td>0.10</td>
<td>0.496</td>
<td>0.502</td>
<td>0.507</td>
<td>0.505</td>
<td>0.505</td>
</tr>
<tr>
<td></td>
<td>0.20</td>
<td>0.490</td>
<td>0.505</td>
<td><strong>0.510</strong></td>
<td>0.502</td>
<td>0.507</td>
</tr>
</tbody>
</table>

\(^1\)Highest value for each trait is in bold.

### Table 2. Estimated prediction bias\(^1\) (measured as average difference between predicted and observed responses in standard deviation units) in the testing set using the original gradient boosting algorithm (mtry = 100\%) or its modified version, “random boosting,” for different values of percentage of SNP sampled at each iteration (mtry) and smoothing parameter (v)

<table>
<thead>
<tr>
<th>Trait</th>
<th>mtry (%)</th>
<th>1</th>
<th>5</th>
<th>10</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Milk yield</td>
<td>0.01</td>
<td>−0.040</td>
<td>−0.047</td>
<td>−0.037</td>
<td>−0.039</td>
<td>−0.039</td>
</tr>
<tr>
<td></td>
<td>0.10</td>
<td>−0.044</td>
<td>−0.042</td>
<td>−0.041</td>
<td>−0.035</td>
<td>−0.038</td>
</tr>
<tr>
<td></td>
<td>0.20</td>
<td>−0.032</td>
<td>−0.014</td>
<td><strong>−0.008</strong></td>
<td>−0.029</td>
<td>−0.026</td>
</tr>
<tr>
<td>Fat yield</td>
<td>0.01</td>
<td>−0.113</td>
<td>−0.107</td>
<td>−0.104</td>
<td>−0.104</td>
<td>−0.104</td>
</tr>
<tr>
<td></td>
<td>0.10</td>
<td>−0.121</td>
<td>−0.107</td>
<td><strong>−0.090</strong></td>
<td>−0.100</td>
<td>−0.099</td>
</tr>
<tr>
<td></td>
<td>0.20</td>
<td>−0.095</td>
<td>−0.114</td>
<td>−0.103</td>
<td>−0.092</td>
<td>−0.095</td>
</tr>
<tr>
<td>Protein yield</td>
<td>0.01</td>
<td>−0.049</td>
<td>−0.061</td>
<td>−0.071</td>
<td>−0.067</td>
<td>−0.070</td>
</tr>
<tr>
<td></td>
<td>0.10</td>
<td>−0.029</td>
<td>0.062</td>
<td>−0.046</td>
<td>−0.047</td>
<td>−0.058</td>
</tr>
<tr>
<td></td>
<td>0.20</td>
<td><strong>−0.025</strong></td>
<td>−0.034</td>
<td>−0.056</td>
<td>−0.068</td>
<td>−0.075</td>
</tr>
<tr>
<td>Fat percentage</td>
<td>0.01</td>
<td>0.039</td>
<td>0.051</td>
<td>0.053</td>
<td>0.046</td>
<td>0.045</td>
</tr>
<tr>
<td></td>
<td>0.10</td>
<td>0.030</td>
<td>0.053</td>
<td>0.053</td>
<td>0.042</td>
<td>0.040</td>
</tr>
<tr>
<td></td>
<td>0.20</td>
<td>0.032</td>
<td>0.048</td>
<td>0.055</td>
<td><strong>0.010</strong></td>
<td>0.041</td>
</tr>
<tr>
<td>Udder depth</td>
<td>0.01</td>
<td>−0.234</td>
<td>−0.233</td>
<td>−0.232</td>
<td>−0.219</td>
<td>−0.238</td>
</tr>
<tr>
<td></td>
<td>0.10</td>
<td><strong>−0.217</strong></td>
<td>−0.226</td>
<td>−0.232</td>
<td>−0.233</td>
<td>−0.231</td>
</tr>
<tr>
<td></td>
<td>0.20</td>
<td>−0.219</td>
<td>−0.220</td>
<td>−0.229</td>
<td>−0.241</td>
<td>−0.234</td>
</tr>
</tbody>
</table>

\(^1\)Lowest value for each trait is in bold.
parallelization of the code can be implemented at step 2 described above when searching for the SNP minimizing the loss function. The parallelization would drastically decrease the computation time of the algorithm (not implemented in this study).

**CONCLUSIONS**

Incorporating high-density markers into models for prediction of genetic values poses important statistical and computational challenges. Machine learning algorithms can be used to deal with the curse of dimensionality and computational limitations when a large number of individuals have genotypic information. In particular, the boosting algorithm provides an efficient strategy to calculate additive genomic breeding values using high-density SNP information. We have provided here a comprehensive description of the mechanisms of the algorithm and show that it can be viewed as an additive gradient descent method that may be implemented as a SNP regression model. A modification of the algorithm has been also proposed to speed up computation of genomic breeding values, with a minimum effect on predictive ability. The companion study by Jiménez-Montero et al. (2013) provides comparison of boosting and random boosting with other methods commonly used in the genome-assisted evaluations.

**ACKNOWLEDGMENTS**

The authors acknowledge funds from the project CDTI-P080250866 UPM and the agreement INIA-CC10-046, and to CONAFE, ASCOL (Asturias, Spain), ABEREKIN (Bizcaia, Spain), XENETICA FONTAO (Lugo, Spain), GENETICAL (Leon, Spain) for providing genotypes and phenotypes used in this study.

**REFERENCES**


---

**Table 3.** Computation time1 (hours) to run 10-fold cross-validations (a complete genomic-assisted evaluation cycle) regarding the value of the smoothing parameter (v) and the proportion of SNP sampled at each iteration (mtry) for 10 iterations.

<table>
<thead>
<tr>
<th>Smoothing parameter (v)</th>
<th>mtry (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0.01</td>
<td>1.50</td>
</tr>
<tr>
<td>0.10</td>
<td>0.83</td>
</tr>
<tr>
<td>0.20</td>
<td>0.67</td>
</tr>
</tbody>
</table>

1In an Intel Xeon QuadCore E5430 (4 × 2.66 GHz) processor with 8 Gb of RAM under a Linux operating system.

²This value of mtry is equivalent to the original gradient boosting.