Iranian Society of Animal Science

Journal of Livestock Science and Technologies

Shahid Bahonar University of Kerman

ISSN: 2322-3553 (Print) ISSN: 2322-374X (Online)

Paper type: Original Research

Comparing the performance of *xgboost*, Gradient Boosting and GBLUP models under different genomic prediction scenarios

Farhad Ghafouri-Kesbi[*]

Department of Animal Science, Faculty of Agriculture, Bu-Ali Sina University, Hamedan, Iran

[*]Corresponding author,
E-mail address:
f.ghafouri@basu.ac.ir

Received: 15 Dec 2023,
Received in revised form: 02 Mar 2024,
Accepted: 13 Mar 2024,
Published online: 14 Mar 2024,
© The authors, 2024.

ORCID
Farhad Ghafouri-Kesbi
0000-0002-2219-055X

Abstract The aim of this study was to study the performance of *xgboost* algorithm in genomic evaluation of complex traits as an alternative for Gradient Boosting algorithm (GBM). To this end, genotypic matrices containing genotypic information for, respectively, 5,000 (S1), 10,000 (S2) and 50,000 (S3) single nucleotide polymorphisms (SNP) for 1000 individuals was simulated. Beside *xgboost* and GBM, the GBLUP which is known as an efficient algorithm in terms of accuracy, computing time and memory requirement was also used to predict genomic breeding values. *xgboost*, GBM and GBLUP were run in R software using xgboost, gbm and synbreed packages. All the analyses were done using a machine equipped with a Core i7-6800K CPU which had 6 physical cores. In addition, 32 gigabyte of memory was installed on the machine. The Person's correlation between predicted and true breeding values ($r_{p,t}$) and the mean squared error (MSE) of prediction were computed to compare predictive performance of different methods. While GBLUP was the most efficient user of memory, GBM required a considerably high amount of memory to run. By increasing size of data from S1 to S3, GBM went out from the competition mainly due to its high demand for memory. Parallel computing with *xgboost* reduced running time by 99% compared to GBM. The *speedup ratio*s (the ratio of the GBM runtime to the time taken by the parallel computing by *xgboost*) were 444 and 554 for the S1 and S2 scenarios, respectively. In addition, parallelization efficiency (*speed up ratio*/number of cores) were, respectively, 74 and 92 for the S1 and S2 scenarios, indicating that by increasing the size of data, the efficiency of parallel computing increased. The *xgboost* was considerably faster than GBLUP in all the scenarios studied. Accuracy of genomic breeding values predicted by *xgboost* was similar to those predicted by GBM. While the accuracy of prediction in terms of $r_{p,t}$ was higher for GBLUP, the MSE of prediction was lower for *xgboost*, specially for larger datasets. Our results showed that *xgboost* could be an efficient alternative for GBM as it had the same accuracy of prediction, was extremely fast and needed significantly lower memory requirement to predict the genomic breeding values.

Keywords: genomic evaluation, parallel computing, computing time, SNP

Introduction

Smith (1967) was the first one who suggested the idea of using genetic markers for improvement of the quantitative traits. However, it took 22 years until Fernando and Grossman (1989) showed the practical implementation of genetic markers for this purpose. Their method was termed the marker assisted selection (MAS). The MAS has a successful history in improving the less complicated traits such as tolerance in plants, which are controlled by a small number of genes. However, it was hardly used to improve the complex traits in crops and animals, traits which are regulated by a large number of genes. The MAS needed accurate estimation of marker effects which was computationally difficult. In addition, genotyping was economically affordable only for a few markers. Meuwissen et al. (2001) introduced the genomic selection (GS) to overcome the MAS drawbacks. The main part of GS was whole genome sequencing techniques providing genome-wide information needed for genomic evaluation. Instead of using few markers, the GS has the ability of using genetic information of all markers distributed along the genome. Accordingly, the genomic breeding values (GBV) of selection candidates can be estimated by adding the effects of all markers underlying the trait and therefore a much more comprehensive and reliable selection is expected compared to MAS. For the first time, GS was applied by the US dairy industry in August 2003 and its formal results were published by Van Raden et al. (2008). In plants, the first results of applying genomic selection in maize were published by Bernardo and Yu (2007). By increasing the accuracy and intensity of selection and decreasing the generation interval, the GS can double the rate of genetic gain for economically important traits in livestock (Matthews et al., 2019).

One of the important issues in genomic selection is the estimation of the effect of SNPs for which different parametric and nonparametric methods have been developed. These methods differ in accuracy of prediction, memory requirement and computing time (Wang et al., 2018). By lowering the cost of genotyping, today, we can genotype more animals for many more markers. For example, at the present, high density marker panels (777K; Matukumalli et al., 2011) are available for GS in dairy cows. Increasing the number of high-density genotyped animals increases the computational requirements of genomic prediction models. For some models, such as single step GBLUP (ssGBLUP), handling of big genotypic matrices becomes practically impossible when the number of animals exceeds 100,000, and for models such as Bayesian variable selection models, if not impossible, will be very time consuming (Ødegård et al., 2018). In addition, the use of cheap next-generation sequencing (NGS) technologies in animal breeding has been started. The NGS allows genotyping of large numbers of individuals and usually produces huge amounts of data. Analyzing such large genomic datasets in a reasonable time requires more efficient methods. Parallel computing (Figure 1) can be a promising strategy to deal with such large genomic datasets (Guo et al., 2018). The basis for parallel computing is the Hyper-Threading (HT) technology of the Intel company (Intel® Hyper-Threading Technology, 2003) by which the maximum processing power of the central processing unit (CPU) of the machine is exploited to analysis the data. Using a CPU with 10 cores, because of HT technology, the job is divided to 10 parts and then executed simultaneously on 10 cores of the CPU; hereby, decreasing the computing time. New parallel algorithms are able to use different numbers of cores for doing analysis based on the size of the input data and, with the aim of supercomputers, they could analyze all the available information at once which are important for researchers when real-time results are needed (Orozco-Arias et al., 2017). It is worthwhile to mention that parallel computing does not enable the machine to do what it was previously unable to do; rather it enables the machine to do what it was previously able to do much faster.

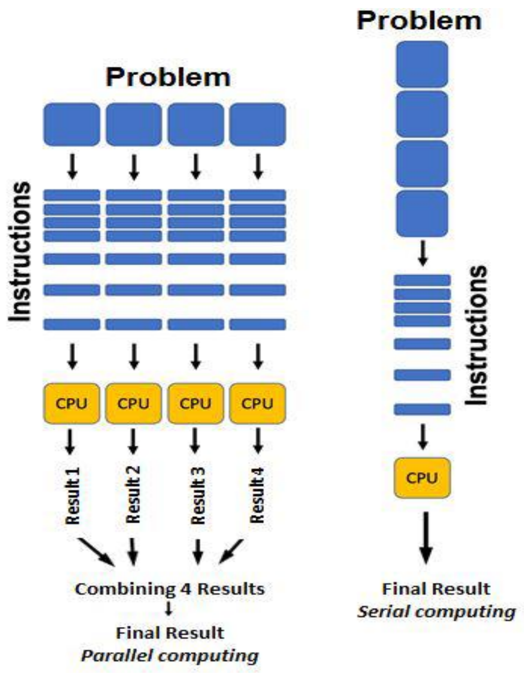

Figure 1. Parallel vs. serial computing (https://pythonnumericalmethods.berkeley.edu/notebooks/chapter13.01- 232 Parallel-Computing-Basics.html)

The Gradient Boosting algorithm (GBM) (Shapire, 2003) is a machine learning algorithm which is used for both classification and regression problems. It has been applied in genomic studies for gene function prediction, analysis of gene expression, and evaluation of genomic susceptibility (Kim and Kim, 2017). While GBM has been used for genomic selection (González-Recio et al., 2014; Ghafouri-Kesbi et al., 2017), this method is memory expensive and extremely slow (Ghafouri-Kesbi et al., 2017). Recently, a variant of GBM has been introduced through *xgboost* package (Chen et al., 2019) which automatically does parallel computing. Therefore, this study was conducted to compare *xgboost* with GBM algorithms and GBLUP in terms of computing time, memory requirement, and the accuracy of prediction.

Materials and methods

Data simulation

The genome and population were simulated using the package *hypred* in R (Technow, 2013). In the framework of genomic selection, we need a reference population and a validation population. A reference population comprised of 1000 individuals was simulated for which genotypic and phenotypic information were available. Subsequently, the validation population was generated from the reference population. Animals in the validation population only had genotypic information but their phenotypic information was unknown. This validation population comprised of selection candidates whose genomic breeding values had to be estimated. Heritability was set to 0.5. The QTL effects were sampled from a gamma distribution, with shape and scale parameters as 0.4 and 1.66, respectively (Meuwissen et al., 2001). Using prediction equations, the effects of all SNPs were estimated in the reference population by combining the genotypic and phenotypic information of all individuals. Then, the genomic breeding values of selection candidates were estimated by summing the effects of all SNPs they carry according to SNPs effects previously estimated in the reference population. In different scenarios, different genomes were simulated in terms of the number of chromosomes and number of SNP markers. The aim was to hold a competition between different methods in terms of the computing time and amount of data they could handle. To do this, three scenarios of size of data were considered. A genome comprised of 5 and 10 and 25 chromosomes were simulated on which, respectively, 5,000 (S1), 10,000 (S2) and 50,000 SNP (S3) was uniformly distributed. The covariate for each genotype with alleles A1 and A2 was set to 1 for A1A1, -1 for A2A2 and 0 for A1A2 or A2A1. Minor allele frequency was set to 0.05.

Method of genomic prediction

Gradient Boosting algorithm

The Gradient Boosting belongs to a family of machine learning algorithms that convert the weak learners to strong learner. Here, regression trees were the weak learners. The GBM adds regression trees to the residual (misclassified inputs) of the previously decision tree in such a way that by adding the new trees, the error function decreased (Hastie et al., 2009; Oguto et al., 2011):

$$f(x) = \sum_{m=1}^{M} \beta_m b(x;\gamma_m)$$

where, β_m , m=1,2,.., M are basis expansion coefficient, and $b(x;\gamma_m)$ are simple functions of the multivariate argument, with a set of parameters $\gamma = (\gamma_1,\gamma_2,\dots,\gamma_M)$. Prediction is accomplished by weighting the ensemble outputs of all the regression trees. Both the serial and parallel GBM were considered. The serial GBM was carried out using the package *gbm* (Greenwell et al., 2019). The tuning parameters in GBM were the number of tree (*ntree*), tree depth or tree complexity (*tc*) and shrinkage rate or learning rate (*lr*). A series of values for each parameter was specified and the performance of the model with each combination of the tuning parameters was evaluated. In the model with highest predictive performance, *ntree* = 1500, *tc* = 7 and *lr* = 0.02.

xgboost

Chen et al. (2019) developed the *xgboost* package for parallel tree GBM. Like GBM, *xgboost* uses the method of additive training by which it grows regression trees on the residual of the previous tree sequentially. But it doesn't mean that it cannot be run in parallel. Of course, *xgboost* doesn't run multiple trees in parallel because it needs predictions of each tree to update the gradients. Rather, it does the parallelization within a single tree to create branches independently. Searching for optimal splits in the 'weak' decision trees can be streamlined by utilizing large number of cores. Contrary to the GBM codes which are executed serially on CPU of machine, the *xgboost* codes are automatically executed in parallel on all cores of the CPU. Assessing the model's performance in terms of computing time was performed by calculating the *speedup ratio* as follows:

Speed up ratio= T-Serial/T-Parallel

where, T-Serial was the execution time of the sequential algorithm and T-parallel was the execution time of the parallelized algorithm. The *Speedup ratio* was then used to calculate the parallelization efficiency as: *Speedup ratio*/number of cores.

Genomic Best Linear Unbiased Prediction (GBLUP)

The GBLUP was fitted as follow:

$$\mathbf{y=1\mu+Zg+e,}$$

where, **y** is the vector of phenotypic observations, **Z** is the design matrix associating the phenotypic observations to GBVs, **g** is the vector of genomic breeding values, assuming that $\mathbf{g}\sim\mathbf{N}(0,\mathbf{G}\delta_g^2)$, where δ_g^2 is the additive genetic variance, and **G** is the genomic relationship matrix whose elements were estimated based on the allelic similarity between individuals (VanRaden, 2008). The R package synbreed (Auinger et al., 2018) was used to run GBLUP.

Comparison of methods

Two criteria were used to measure the prediction accuracy of methods: 1) the Person's correlation between the predicted and true breeding values ($r_{p,t}$). 2) The mean squared error of prediction (MSE). Larger estimates of $r_{p,t}$ show more reliable predictions. The MSE of prediction provides a measure of overall fit of the model. Each scenario which was a combination of data size and prediction method was analyzed 10 times and the average computing time, memory requirement, $r_{p,t}$ and MSE of each scenario were presented.

Memory requirement and computing time

The package *pryr* was used to record memory usage for each method (Wickham, 2018). It records the amount of memory occupied by the objects created by executing a function in R. The computing time in each scenario was monitored and recorded with an R function. The computing time was measured as the time consumed for executing the codes of the methods studied, and did not include the time consumed for the simulation of population and genome. All the analyses were done using a machine with the CPU model Intel Core i7-6800K. This CPU has 6 physical cores and a base frequency of 3.40 gigahertz. In addition, 32 gigabyte of memory was installed on the machine.

Results

Figure 2 shows that GBM was an inefficient user of memory. In the S1 scenario, the total size of objects on the memory during execution of the GBM codes was 138 megabytes, while for GBLUP it was only 331 kilobytes. The *xgboost* were ranked as second. The difference between *xgboost* and GBM was significant (38.5 megabytes vs. 138 megabytes). Figure 2 also shows that GBM only could handle the S1 and S2 data and dealing with S3 scenario of data size, it went out from the competition.

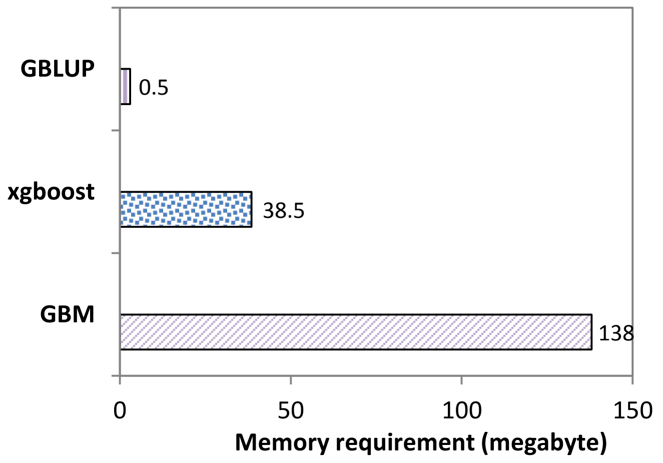

Figure 2. Memory requirement for the studied methods using S3 dataset

Figure 3 shows the differences between methods in terms of computing time. The *xgboost* was superior to other methods studied. In the scenario of S1, in which both GBM and *xgboost* were present, the *xgboost* which runs in parallel, decreased the computing time by 99.7% compared to GBM which runs serially (1.49 seconds vs. 661 seconds). The *speedup ratio* was 444 with parallelization efficiency as 74. In addition, in the S2 scenario, *xgboost* decreased the computing time by 99.8% compared to GBM (2.55 seconds vs. 1384.3 seconds). The corresponding *speedup ratio* and parallelization efficiency were 554 and 92, respectively.

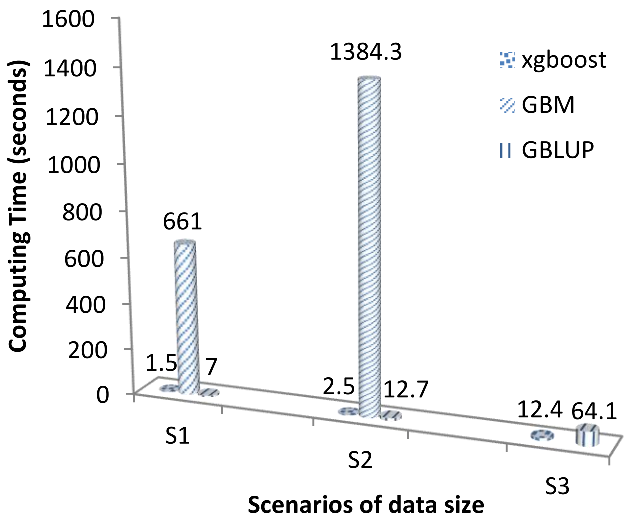

Figure 3. Computing time of the studied methods in different scenarios of data size

Figures 4 and 5 show the prediction accuracy and MSE of prediction, respectively. In all methods studied, by increasing the number of markers, the accuracy of prediction ($r_{p,t}$) decreased and mean square error of prediction (MSE) increased. For *xgboost* and GBLUP, $r_{p,t}$ decreased by 18% and 20% following the increase in the number of SNPs from 5,000 to 50,000. On the other hand, the MSE of prediction increased 4 times by increasing the number of SNPs from 5,000 to 50,000. Regarding the accuracy of prediction measured by $r_{p,t}$, GBLUP predicted the GBVs with higher accuracy compared to other methods. But regarding MSE, in all the scenarios studied, the MSE of predictions for *xgboost* was smaller than other methods, although in most cases its differences with GBM were not statistically significant (P>0.05).

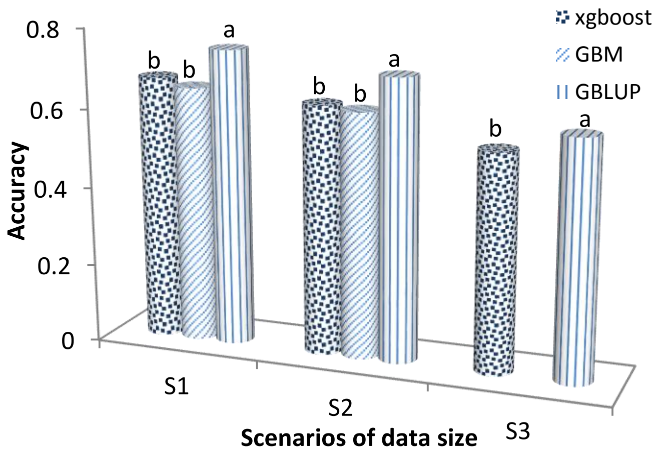

Figure 4. Accuracy of the studied methods in different scenarios of data size

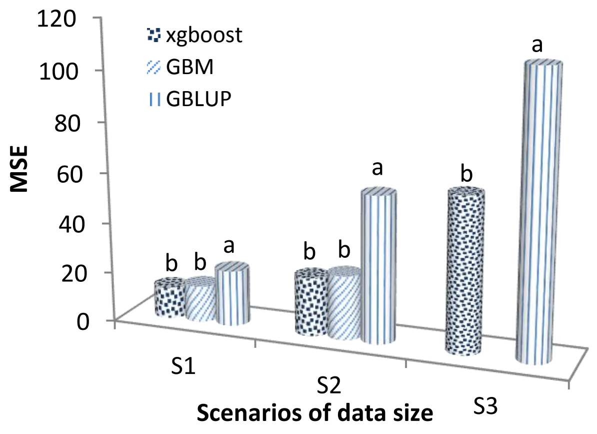

Figure 5. MSE of prediction for the studied methods in different scenarios of data size

Discussion

The results showed that GBLUP was the most efficient user of memory. Therefore, GBLUP could handle much larger datasets compared to other methods. The amount of memory required by a method depends on the mathematical algorithm used and the software on which it run. For example, implementation of some methods such as rrBLUP requires inverting large matrices that take up a lot of space on the memory and makes the operation to be very memory expensive. In addition, some packages in R such as *gbm* (Greenwell et al., 2019) which was used to run GBM suffer from the "memory leak" or "space leak" problem. Because of this problem, a package cannot correctly manage memory allocation in a way that memory which is no longer needed is not released and therefore there won't be free room on the memory for execution of remained codes (Singh et al., 2007).

Significant decrease in computing time for *xgboost* was in agreement with Thompson and Charnigo (2015) who reported that machine learning algorithms which account for heterogeneous correlation structures could benefit significantly from parallelization. Increases in *speedup ratio* and parallelization efficiency following an increase in the data size indicated that by increasing the size of data, the efficiency of parallel computing increased. In other words, scenarios of using small datasets could not well represent the advantage of parallel computing over serial computing. Carlborg et al. (2001) used serial and parallel computing to detect the quantitative trait loci (QTL) and reported that parallel computing decreased the computing time by 85% compared to serial computing (13.7 hours vs. 2.1 hours). Ma et al. (2008) showed that by parallelizing a GWAS including 50000 SNPs and 2000 individuals the computing time decreased from 10 hours to 0.2 hours. Wu et al. (2012) applied the Bayesian method in parallel using Hidden Markov Chain for genome-enabled prediction of the rib Eye in a population of beef cattle and reported 87% decrease in computing time compared to serial computing (7.78 minutes vs. 1386 minutes). Guo et al. (2018) used Hidden Markov Chain to do parallel computing with Bayesian methods for genomic predictions of four quantitative traits in a Simmental population and reported, depending on the method used, 3 to 13 fold decreases in computing time compared with serial computing. The *xgboost* not only was superior to its serial form (GBM), but also it was significantly faster than GBLUP (Figure 3). For small datasets, the decrease in runtime may not be very important, but when analyses include large datasets, decrease in runtime is valuable for researchers. Especially with recent advances in DNA technology which allow the rapid collection of up to a few million genetic markers of thousands individuals, the use of big data in genomic evaluation of livestock will be inevitable. Under such situations, parallelization of genomic prediction models will be an effective strategy to obtain results of genomic evaluation as soon as possible. In addition, the extent to which *xgboost* can accelerate the process of genomic selection depends on the number of cores in the CPU of the machine. Here, using a CPU equipped with 6 cores, *xgboost* was considerably faster than GBLUP. Certainly with a CPU equipped with higher number of cores, it will be much faster than GBLUP. With a cluster of thousands cores, a serial computing-GBLUP which takes hours to complete, would be finished in a few second by *xgboost*. The superiority of *xgboost* over other methods showed that *xgboost* is highly efficient for high dimensional data.

Following the increase in the number of markers from 5,000 to 50,000 markers, a decrease in predictive performance of prediction methods shown by $r_{p,t}$ and MSE was noticed. This was expected, because by increasing the number of markers, the computation burden increases significantly as the number of unknown parameters (*p*, number of markers) that have to be estimated by using a fixed number of known parameters (*n*, number of individuals) increases (Zhang et al., 2019). This is termed "Big-p, Little-n" or "p >> n" problem. A major problem with p >> n problem when using machine learning models is overfitting the training dataset. Given the lack of samples, most models are unable to generalize and instead learn the statistical noise in the training data. Therefore, the model performs perfectly on training set, while fitting poorly on testing set (Ying, 2019). Considering the GBLUP, the problem with more predictors than phenotypic values is that there will be no unique solution to a standard linear regression problem. Therefore, once the coefficients for *n* of the predictors are estimated, the coefficients for the other (p−n) predictors can be expressed as arbitrary linear combinations of those first *n* predictors. As a result, by increasing the number of markers, the number of phenotyped animals in the reference population should be increased to prevent a decrease in accuracy. The current results demonstrated that while parallel computing accelerated the analyses, it did not affect the accuracy of prediction" and add them to the beginning of the next paragraph.

The current results demonstrated that while parallel computing accelerated the analyses, it did not affect the accuracy ofmprediction. Guo et al. (2018) reported that parallel computing for genomic evaluation of Simental cow provided predictions with similar accuracy as serial computing. The GBLUP superiority to other methods, regarding the accuracy of prediction, was in agreement with other studies including Neves et al. (2012), Abdollahi-Arpanahi et al. (2013) and Zhang et al. (2019) who reported higher $r_{p,t}$ for GBLUP compared to other methods studied. However, the predictions with *xgboost* and GBM had significantly smaller MSE ($P<0.05$). There is no consensus among researchers regarding the use of $r_{p,t}$ or MSE for model comparison. The $r_{p,t}$ is the most commonly used criterion to evaluate the predictive ability of genomic prediction models. While larger estimates of $r_{p,t}$ show more reliable predictions, it does not address the bias of predictions. The MSE of prediction is equal to the squared prediction bias plus the variance of the prediction error and, therefore, it includes both the predictive accuracy and bias of predicted genomic breeding values. Small values of MSE show that the predictor is precise and accurate (Gonzales Racio et al., 2014). Gonzales Racio et al. (2014) compared the MSE with $r_{p,t}$ and reported that MSE was more informative than $r_{p,t}$ for comparing genomic prediction methods. They demonstrated that methods with similar $r_{p,t}$, may have different MSE and suggested using MSE as the main metric for model comparison. Comparison of the long-term genomic selection schemes based on GBVs evaluated by $r_{p,t}$ or MSE could help to resolve the ambiguities in this area.

Conclusions

In conclusion, *xgboost* significantly speeded up the process of genomic evaluation compared with GBM. Although, the accuracy of prediction in terms of $r_{p,t}$ was higher for GBLUP, the MSE of prediction was smaller for *xgboost*. Taking into account that big genomic data are being generated for use in genomic evaluation procedure, developing packages for parallelization of accurate genomic methods such as GBLUP is recommended.

References

Abdollahi-Arpanahi, R., Pakdel, A., Nejati-Javaremi, A, Moradi Shahre Babak, M., 2013. Comparison of different methods of genomic evaluation in traits with different genetic architecture. *Journal of Animal Production* 15, 65-77 (In Farsi).

Auinger, H.S., Wimmer, V., Auinger, H.J., Albrecht, T., Schoen, C.C., Schaeffer, L., Erbe, M., Ober, U., Reimer, C., Badke, Y., VandeHaar, P., 2018. Framework for the analysis of genomic prediction data using R (synbreed). Available at https://cran.rproject. org/web/packages/synbreed/index.html.

Bernardo, R., Yu, J., 2007. Prospects for genome-wide selection for quantitative traits in maize. *Crop Science* 47, 1082-1090.

Carlborg, Ö., Andersson-Eklund, L., Andersson, L., 2001. Parallel computing in interval mapping of quantitative trait loci. *Journal of Heredity* 92, 449-451.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., Li, Mu., Xie, J., Lin, M., Geng, Y., Li, Y., 2019. xgboost: Extreme Gradient Gradient Boosting. Available at: https://cran.r-project.org/web/packages/xgboost/index.html.

Fernando, RL., Grossman, M., 1989. Marker-assisted selection using best linear unbiased prediction. *Genetic Selection Evolution* 2, 246-477.

Ghafouri-Kesbi, F., Rahimi-Mianji, G., Honarvar, M., Nejati-Javaremi, A., 2017. Predictive ability of random forests, Gradient Boosting, support vector machines and genomic best linear unbiased prediction in different scenarios of genomic evaluation. *Animal Production Science* 57, 229-236.

González-Recio, O., Rosa, GJM., Gianola, D., 2014. Machine learning methods and predictive ability metrics for genome-wide prediction of complex traits. *Livestock Science 166* 217-231.

Greenwell, B., Bradley, B., Cunningham, J., 2019. *gbm*: Generalized Boosted Regression Models. Available at: https://cran.r-roject.org/web/packages/gbm/index.html

Guo, P., Zhu, B., Niu, H., Wang, Z., Liang, Y., Chen, Y., Zhang, L., Ni, H., Guo, Y., El Hamidi, AH., Gao, X., Gao, H., Wu, X., Xu, L., Li, J., 2018. Fast genomic prediction of breeding values using parallel Markov chain Monte Carlo with convergence diagnosis. *BMC Bioinformatics* 19, 3.

Hastie, T.J., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning. 2nd ed., Springer, New York, USA.

Intel® Hyper-Threading Technology., 2003. Technical User's Guide. Available at: http://www.cslab.ece.ntua. gr/courses/advcomparch/2007/material/readings/Intel %20Hyper-Threading%20Technology.pdf

Ma, L., Birali, Runesha, H., Dvorkin, D., Garbe, G.R., 2008. Parallel and serial computing tools for testing single-locus and epistatic SNP effects of quantitative traits in genome-wide association studies. *BMC Bioinformatics* 9, 315.

Kim, B., Kim, S., 2018. Prediction of inherited genomic susceptibility to 20 common cancer types by a supervised machine-learning method. *Proceedings of the National Academy of Sciences* 115, 1322-1327.

Matukumalli, L.K., Schroeder, S., DeNise, S.K., 2011. Analyzing LD blocks and CNV segments in cattle:

Novel genomic features identified using the Bovine HD BeadChip. Illumina Inc. San Diego, USA.

Matthews, D., Kearney, J.F., Cromie, AR., 2019. Genetic benefits of genomic selection breeding programmes considering foreign sire contributions. *Genetic Selection Evolution* 51, 40.

Meuwissen, T.H.E., Hayes, B.J., Goddard, M.E., 2001. Prediction of total genetic value using genome wide dense marker maps. *Genetics* 157, 1819-1829.

Neves, H.H.R., Carvalheiro, R., Queiroz, S.A., 2012. A comparison of statistical methods for genomic selection in a mice population. *BMC Genetics* 13,100.

Ødegård, J., Indahl, U., Strandén, I., Meuwissen, T.H.E., 2018. Large‐scale genomic prediction using singular value decomposition of the genotype matrix. *Genetic Selection Evolution* 50, 6.

Oguto, J.O., Piepho, H.P., Schulz-Streeck, T., 2011. A comparison of random forests, Gradient Boosting and support vector machines for genomic selection. *BMC Proceedings* 5, 11.

Orozco-Arias, S., Tabares-Soto, R., Ceballos, D., Guyot, R., 2017. Parallel Programming in Biological Sciences, Taking Advantage of Supercomputing in Genomics. *Advances in Computing* 735, 627-643.

R Core Team., 2022. R: A language and environment for statistical computing. Vienna, Austria. Available at: https://www.R-project.org/.

Singh, P.P., Nagpal R., Pal, R., Nagamani, V., Rao, B.B.P., 2007. MemHunt: Dynamic Memory Leak Analyzer and Garbage Collector. In Proceedings of the 2nd National Conference on Emerging Trends and Applications in Computer Engineering, Ajmir, India.

Smith, C., 1967. Improvement of metric traits through specific genetic loci. *Animal Production* 9, 349-358.

Technow, F., 2013. hypred: Simulation of genomic data in applied genetics. Available at: https://github. com/cran/hypred.

Thompson, K., Charnigo, R., 2015. Parallel Computing in Genome-Wide Association Studies. *Journal of Biometrics and Biostatistics* 6, 1000e131.

VanRaden, PM., 2008. Efficient methods to compute genomic predictions. *Journal of Dairy Science* 91, 4414-4423.

Wang, X., Xu, Y., Hu, Z., Xu, C., 2018. Genomic selection methods for crop improvement: Current status and prospects. *The Crop Journal* 6, 330-340.

Wickham, H., 2018. *pryr*: Useful tools to pry back the covers of R and understand the language at a deeper level. Available at: https://cran.r-project.org/web /packages/pryr/index.html.

Wu, XL., Sun, C., Beissinger, TM., Rosa, GJ., Weigel, KA., Gatti Nde, L., Gianola, D., 2012. Parallel Markov chain Monte Carlo bridging the gap to high-performance Bayesian computation in animal breeding and genetics. *Genetics Selection Evolution* 44, 29.

Ying, X., 2019. An overview of overfitting and its solutions. *Journal of Physics: Conference. Series* 1, 1168.

Zhang, H., Yin, L., Wang, M., 2019. Genomic selection for agricultural economic traits in maize, cattle, and pig populations. *Frontiers in Genetics* 10,189.