LEONARDO SIQUEIRA GLÓRIA

# ASSESSMENT OF GENOME-WIDE PREDICTION BY USING BAYESIAN REGULARIZED NEURAL NETWORKS

Thesis presented to the Universidade Federal de Viçosa as part of the requirements of Genetics and Breeding Graduate Program for the achievement of the title of *Doctor Scientiae*.

VIÇOSA
MINAS GERAIS – BRASIL
2015

**Ficha catalográfica preparada pela Biblioteca Central da Universidade Federal de Viçosa - Câmpus Viçosa**

T
G562a
2015

Glória, Leonardo Siqueira Glória, 1987-
Assessment of genome-wide prediction by using bayesian regularized neural networks / Leonardo Siqueira Glória Glória. – Viçosa, MG, 2015.
60f. : il. ; 29 cm.

Inclui apêndice.
Orientador: Fabyano Fonseca e Silva.
Tese (doutorado) - Universidade Federal de Viçosa.
Inclui bibliografia.

1. Genômica. 2. Genética - Parâmetros. 3. Redes neurais (Computação). I. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-graduação em Genética e Melhoramento. II. Título.

CDD 22. ed. 572.86

LEONARDO SIQUEIRA GLÓRIA

**ASSESSMENT OF GENOME-WIDE PREDICTION BY USING BAYESIAN REGULARIZED NEURAL NETWORKS**

Thesis presented to the Universidade Federal de Viçosa as part of the requirements of Genetics and Breeding Graduate Program for the achievement of the title of *Doctor Scientiae*.

APPROVED: May 25th, 2015.

Moysés Nascimento

Alexandre Pio Viana

Cosme Damião Cruz
(Co-adviser)

Ricardo Augusto Mendonça Vieira
(Co-adviser)

Fabyano Fonseca e Silva
(Adviser)

## BIOGRAPHY

Leonardo Siqueira Glória, son of Valeria Pinto Siqueira Glória and Manoel Luís Vieira Glória, was born on September 15, 1987, in Campos dos Goytacazes, Rio de Janeiro, Brazil.

In March 2007, he joined the Animal Science course at the Universidade Estadual do Norte Fluminense Darcy Ribeiro in the city of Campos dos Goytacazes - RJ where he studied and joined several research groups and projects. He was a junior research who earned a student scholarship from March 2008 to February 2012 at the same institution.

In March 2012, he joined the Masters Course in Animal Science at Universidade Estadual do Norte Fluminense Darcy Ribeiro completing the established requisites by February 2014 when he received a Master's degree in Animal Science.

In March 2014, he started his Doctorate Course in the area of Genetics and Breeding at the Universidade Federal de Viçosa. Leonardo completed and defended his Thesis on May 25, 2015 in order to obtain the title of Doctor Scientiae in Genetics and Breeding.

SUMMARY

## RESUMO

GLÓRIA, Leonardo Siqueira, D.Sc., Universidade Federal de Viçosa, maio de 2015. **Avaliação de predições genômicas utilizando redes neurais com regularização Bayesiana.** Orientador: Fabyano Fonseca e Silva. Coorientadores: Cosme Damião Cruz e Ricardo Augusto Mendonça Vieira.

Recentemente, há um aumento de interesse na utilização de métodos não paramétricos, tais como redes neurais artificiais (RNA), na área de seleção genômica ampla (SGA). Uma classe especial de RNA é aquela com regularização Bayesiana, a qual não exige um conhecimento a priori da arquitetura genética da característica, tais como outros métodos tradicionais de SGA (RR-BLUP, Bayes A, B, C$\pi$, BLASSO). O objetivo do presente estudo foi aplicar a RNA baseado em regularização Bayesiana na predição de valores genéticos genômicos utilizando conjuntos de dados simulados a fim de selecionar os marcadores SNP mais relevantes por meio de dois métodos diferentes. Objetivou-se ainda estimar herdabilidades para as características consideradas e comparar os resultados da RNA com dois métodos tradicionais (RR-BLUP e Lasso Bayesiano). A arquitetura mais simples da rede neural com regularização Bayesiana obteve os melhores resultados para as duas características avaliadas, os quais foram muito similares às metodologias tradicionais RR-BLUP e Lasso Bayesiano (BLASSO). A identificação de importância dos SNPs baseada nas RNA apresentaram correlações entre os efeitos verdadeiros e simulados de 0,61 e 0,81 para as características 1 e 2, respectivamente. Estas foram maiores do que aquelas produzidas pelo método tradicional BLASSO (0,55 e 0,71, para característica 1 e 2 respectivamente). Em relação a herdabilidade (assumindo o valor verdadeiro igual a 0,35), a RNA mais simples obteve valor de herdabilidade igual a 0,33, enquanto os métodos tradicionais a subestimaram (com média igual igual a 0,215).

## ABSTRACT

GLÓRIA, Leonardo Siqueira, D.Sc., Universidade Federal de Viçosa, May, 2015. **Assessment of genome-wide prediction by using Bayesian regularized neural networks**. Adviser: Fabyano Fonseca e Silva. Co-Advisers: Cosme Damião Cruz and Ricardo Augusto Mendonça Vieira.

Recently there is an increase interest to use nonparametric methods, such as artificial neural networks (ANN). In animal breeding, an especial class of ANN called Bayesian Regularized Neural Network (BRNN) has been preferable since it not demands a priori knowledge of the genetic architecture of the characteristic as assumed by the most used parametric methods (RR-BLUP, Bayes A, B, C$\pi$, BLASSO). Although BRNN has been shown to be effective for genomic enable prediction. The aim of the present study was to apply the ANN based on Bayesian regularization to genome-enable prediction regarding simulated data sets, to select the most relevant SNP markers by using two proposed methods, to estimate heritabilities for the considered traits, and to compare the results with two traditional methods (RR-BLUP and BLASSO). The simplest Bayesian Regularized Neural Network (BRNN) model gave consistent predictions for both traits, which were similar to the results obtained from the traditional RR-BLUP and BLASSO methods. The SNP importance identification methods based on BRNN proposed here showed correlation values (0.61 and 0.81 for traits 1 and 2, respectively) between true and estimated marker effects higher than the traditional BLASSO (0.55 and 0.71, respectively for traits 1 and 2) method. With respect to $h^2$ estimates (assuming 0.35 as true value), the simplest BRNN recovered 0.33 for both traits, thus outperforming the RR-BLUP and BLASSO, that, in average, estimated $h^2$ equal to 0.215.

# CHAPTER 1

## General Review

### 1. Whole-genome marker-enabled prediction (WGP)

Genomic information for livestock animals (Wang *et al.* 2005) has been rapidly changing and expanding since the 2000's, when methods for DNA sequencing became available.

The generation of genomic information and its availability to the scientific community has increased the opportunities for research and commercial uses, and this is affecting the Animal Science knowledge area. Additionally, an ambitious aim for researchers in the field of Animal Science is to achieve appropriate technology (statistical techniques and computing algorithms) to access decision in breeding programs. Thus, companies together with public organizations have invested in produce data for commercial uses of genomic information, including personalized prediction of future outcomes in livestock (Vazquez 2010).

Modern animal breeding schemes select individuals based on predictions of genetic merit. Rapid genetic progress requires such predictions to be accurate and produced early in the of domestic animals life. Molecular markers allow to describe the genome of individuals at a large number of loci, and this opens possibilities to derive accurate predictions of genetic values early in life. The first attempts to incorporate marker information into predictions were based on the assumption that one can localize causative mutations underlying genetic variation (de Los Campos*, et al.* 2013). This approach, known as quantitative trait loci (QTL) mapping (Soller 1978, Soller and Plotkin-Hazan 1977), led to the discovery of few genes associated to genetic differences of traits of commercial interest.

There is a consensus that large numbers of small-effect genes affects most traits (Buckler*, et al.* 2009) and that the prediction of complex traits requires the consideration of a large number of variants concurrently. The continuous advances of high-throughput genotyping and sequencing technologies allowed the discovery of hundreds of thousands of genetic markers (e.g., single nucleotide polymorphisms, SNPs) in the genome of humans and several animal species. Such dense panels of molecular markers allow the prediction of genetic values exploiting multi locus linkage disequilibrium (LD) between QTL and genome-wide markers (e.g., SNPs). Although earlier contributions exist (Haley and Visscher 1998, Nejati-Javaremi*, et al.* 1997, Whittaker*, et al.* 2000), the foundations of genome-enabled selection (GS) were largely defined in

the groundbreaking article by Meuwissen *et al.* (2001), which proposed to incorporate dense molecular markers into models using a simple, but powerful idea: regress phenotypes on all available markers using a linear model. Moreover, in recent years this approach has gained ground both in animal (VanRaden*, et al.* 2009) and plant breeding (Bernardo and Yu 2007, Crossa *et al.* 2010).

With high-density SNP panels the number of markers (p) can vastly exceed the number of records (n) for analysis, and fit this large-p with-small-n regression requires using some type of variable selection or shrinkage estimation procedure. Owing to developments of penalized and Bayesian estimation procedures, as well as advances in the field of nonparametric regressions, several shrinkage estimation methods have been proposed and used for whole-genome regression and prediction (WGP) of phenotypes or breeding values. However, the relationships between these methods have not been fully addressed and many important topics emerging in empirical applications have been often overlooked (de Los Campos *et al.* 2013).

WGP methods use LD between markers and trait-causing loci in order to estimate the joint contribution of loci across the genome to the trait. It is essential to these methods the prediction of genetic values and phenotypes, instead of the identification of specific genes, which has been the central focus of Genome-wide association study (GWAS). WGP methods, which lay on the Quantitative genetic theory (Falconer and Mackay 1996), are appropriate to study complex traits because such methods may assume that a large number of small-effect, possibly interacting, genes may affect traits.

Specifically, the WGP method consists of to regressing phenotypes on genotypes markers using a linear regression model. An advantage of dense markers maps is that as the number of markers largely exceeds the number of samples, predictions can be accurate even when the estimated effect of each marker is subjected to large uncertainty (de Los Campos*, et al.* 2013). On the other hand, an undesired consequence is that the estimation of marker effects is not feasible with ordinary least squares methods. Penalized and Bayesian estimation methods, in which estimates of marker effects estimates are shrunk typically to zero, have been developed to overcome this and other problems, and such methods have been reviewed elsewhere (de los Campos *et al.* 2010). For instance, the BayesB method uses the prior knowledge that a high proportion of SNPs (typically denoted as $\pi$) have a zero effect and that the effects of the remaining fraction of markers ($1–\pi$) will follow the chosen distribution (Meuwissen *et al.* 2001).

WGP methods are often referred as genomic selection for their initial and most common application to the selection of individuals for traits, typically continuous, of economic interest. For instance, genomic selection has been utilized in cattle for milk related, meat quality and other traits of economic interest (Garrick 2010; Harris *et al.* 2009; Hayes *et al.* 2009; VanRaden *et al.* 2009), in chicken for body weight as well as food consumption and mortality rates (González-Recio *et al.* 2008; Long *et al.* 2010). The accuracy of genomic genetic measures can be used to compare different methods proposed to GWP. Studies focusing on species of economic interest typically assess the accuracy or reliability of WGP methods as the correlation between the true value of the phenotype and the genomic prediction (commonly called genomic estimates of breeding values or GEBV).

## 2. The Challenge

From a statistical and computational stand view, a challenge presented when analyzing genomic information is the enormous number of parameters, (such as SNP effects, or effects of a chromosome section) to be estimated from a much smaller number of phenotypic observations (n), i.e., p>>n. For example in QTL analysis or gene mapping, the degrees of freedom provided by the sample are not enough to estimate all marker effects when using standard statistical methods (Lande and Thompson 1990), and marker-by-marker testing leads to multiple testing problems (Manolio *et al.* 2009), as well as bias in the estimates. A common approach is to treat markers as random effects and estimate their effects by applying some shrinkage methodology. Alternatively, dimension-reduction approaches include filtering non-important SNPs, or a compromise between shrinkage and filtering (Vazquez 2010) can be also used to this aim.

Some popular frequentist parametric approaches include stepwise regression and marker selection, with non-significant markers dropped out of the model (Meuwissen *et al.* 2001); and best linear unbiased prediction (BLUP) of allelic effects (or random regression BLUP), where the variance of SNP effects is assumed to be homogeneous (Meuwissen *et al.* 2001). The Bayesian procedure has been also very used in GWP. The models are postulated at three levels, one at the level of the data, a second one at the level of marker effects, and with a third tier at the level of the SNP-effect variances. Bayes A, B and C, consisting of variations on the prior distribution on SNP effects or on their variances (Habier *et al.* 2007); and the Bayesian LASSO, in which marker effects

are modeled using a double exponential distribution, with a high peak at zero and heavy tails that accommodate SNPs with larger effects (Park and Casella 2008).

Other approaches are based on non-parametric methods including support vector machines (Vapnik and Vapnik 1998) and neural networks (Bishop 2006). There has been a growing interest in semi-parametric and nonparametric methods for prediction of quantitative traits based on reproducing kernel Hilbert space regression on markers (de Los Campos *et al.* 2010) and radial basis functions models (Long *et al.* 2010) or related approaches (Ober*, et al.* 2011).

Artificial neural networks (ANN) provide an interesting alternative because these learning machines can act as universal approximators of complex functions (Alados*, et al.* 2004, Bishop 2006). Those networks can capture non-linear relationships between predictors and responses and learn about functional forms in an adaptive manner, because a series of transformations called activation functions are driven by parameters. The artificial can be viewed as a computer based system composed of many processing elements (neurons) operating in parallel (Lamontagne and Marchand 2006), and as a scheme of Kolmogorov's theorem for representation of multivariate functions (Pereira and Rao 2009). An artificial neural network is determined by the network structure, represented by the number of layers and of neurons, by the strength of the connections (akin to non-parametric regression coefficients) between inputs, neurons and outputs, and by the type of processing performed at each neuron, represented by a linear or non-linear transformation: the activation function (Gianola*, et al.* 2011).

## 3. Neural network

### 3.1. A brief history of Neural Networks

The beginning of Neurocomputing is often taken to be the seminal paper of McCulloch and Pitts published in 1943, which showed that even simple types of neural networks could, in principle, compute any arithmetic or logical function. Other researchers, principally Norbert Wiener and von Neumann, wrote a book and a research manuscript (Von Neumann 1951, Von Neumann 1956) in which the suggestion was that the research into the design of brain-like or brain-inspired computers might be interesting.

In 1949, Hebb wrote a book entitled The Organization of Behavior, which pursued the idea that classical psychological conditioning is ubiquitous in animals because it is a property of individual neurons. This idea was not itself new, but Hebb

took it further than anyone else did at that time by proposing a specific learning law for the synapses of neurons. Hebb then used this learning law to build a qualitative explanation of some experimental results in the psychology field. Although there were many other researchers examining the issues surrounding the neurocomputing in the 1940s and early 1950s, their studies have more effect on setting the stage for later developments than actually causing those developments.

Typical of this era was the construction of first neurocomputer (the Snark) by Marvin Minsky in 1951. The Snark did operated successfully from a technical standpoint but it never actually carried out any particularly interesting information processing functions (Yadav*, et al.* 2015). The first successful neuro-computer (the Mark I perceptron) was developed during 1957 and 1958 by Frank Rosenblatt, Charles Wightman, and others. As we know it today, Rosenblatt as the founder of Neurocomputing. His primary interest was pattern recognition. Besides inventing the perceptron, Rosenblatt also came up with a new book on Neurocomputing, Principles of Neurodynamics (Rosenblatt). Slightly later than Rosenblatt, but studying the same subject, Bernard Widrow developed a different type of neural network processing element called ADALINE, which was equipped with a powerful new learning law which, unlike the perceptron leaning law, is still in widespread use. Widrow and his students applied the ADALINE successfully to a large number of toy problems, and produced several films of their successes. Besides Rosenblatt and Widrow, there were a number of other researches during the late 1950s and early 1960s who had substantial success in the development of neural network architectures and implementation concepts (Yadav *et al.* 2015).

By the early 1980s many Neurocomputing researchers became bold enough to begin submitting proposals to explore the development of neuro-computers and of neural network applications. In the years 1983–1986 John Hopfield, an established physicist of worldwide reputation had become interested in neural networks a few years earlier. Hopfield wrote two highly readable papers on neural networks in 1982 (Hopfield, 1982) and 1984 (Hopfield, 1984) and these, together with his many lectures all over the world, persuaded hundreds of highly qualified scientists, mathematicians, and technologists to join the emerging field of neural networks. In 1986, with the publication of the "PDP books" (Rumelhart*, et al.* 1986), the field exploded. In 1987, the first open conference on neural networks in modern times, the IEEE International Conference on Neural Networks was held in San Diego, and the International Neural

Network Society (INNS) was formed. In 1988, the INNS journal Neural Networks was founded followed by Neural Computation in 1989 and the IEEE Transactions on Neural Networks in 1990.

### 3.2. Components of Artificial Neural Network

According to (Yadav*, et al.* 2015) an artificial neural network (ANN) is an information–processing system that has certain performance characteristics in common with biological neural networks. Artificial neural networks have been developed as generalizations of mathematical models of human cognition or neural biology, based on the assumptions that: Information processing occurs at many simple connections called neurons; signals are passed between neurons over connection links; Each connection link has an associated weight, which in a typical neural net, multiplies the signal transmitted; each neuron applies an activation function to its net input to determine its output signal.

A neural network based model for the solution of differential equations provides the following advantages over the standard numerical methods (Yadav*, et al.* 2015): the neural network based solution of a differential equation is differentiable and is in closed analytic form that can be used in any subsequent calculation. On the other hand, most other techniques offer a discrete solution or a solution of limited differentiability; the neural network based method to solve a differential equation provides a solution with very good generalization properties; computational complexity does not increase quickly in the neural network method when the number of sampling points is increased while in the other standard numerical methods computational complexity increases rapidly as we increase the number of sampling points in the interval; the method is general and can be applied to the systems defined either on orthogonal box boundaries or on irregular arbitrary shaped boundaries; model based on neural network offers an opportunity to tackle in real time difficult differential equation problems arising in many sciences and engineering applications; the method can be implemented on parallel architectures.

The basic component of an artificial neural network is the artificial neuron similar to the biological neuron in biological neural networks. A biological neuron may be modeled artificially to perform computation and then the model is termed as artificial neuron. A neuron is the basic processor or processing element in a neural network. Each neuron receives one or more input over these connections (i.e., synapses) and produces

only one output. In addition, this output is related to the state of the neuron and its activation function. This output may fan out to several other neurons in the network. The inputs are the outputs i.e. activations of the incoming neurons multiplied by the connection weights or synaptic weights. Each weight is associated with an input of a network. The activation of a neuron is computed by applying a threshold function (popularly known as activation function) to the weighted sum of the inputs plus a bias.

### 3.3. Mathematical modeling of Artificial Neural Network

Bishop (2006) explained that linear models for regression are based on linear combinations of fixed nonlinear basis functions $\phi_j(x)$ and take the form:

$$y(x,w)=f\left(\sum_{j=1}^{M} w_j\phi_j(x)\right), \quad (1)$$

In which f(·) is a nonlinear activation function is the identity in the case of regression. To extend this model by making the basis functions, $\phi_j(x)$, depend on parameters and then to allow these parameters to be adjusted along with the coefficients $\{w_j\}$ during training. There are, of course, many ways to build parametric nonlinear basis functions. Neural networks use basis functions that follow the same form as this equation, so that each basis function is itself a nonlinear function of a linear combination of the inputs, where the coefficients in the linear combination are adaptive parameters.

This leads to the basic neural network model, which can be described as a series of functional transformations. First, we construct $M$ linear combinations of the input variables $x_1, \ldots, x_D$ in the form

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}, \quad (2)$$

where j = 1, . . . , M, and the superscript (1) indicates that the corresponding parameters are in the first "layer" of the network. We shall refer to the parameters $w_{ji}^{(1)}$ as weights and the parameters $w_{j0}^{(1)}$ as biases. The quantities $a_j$ are known as activations. Each of them is then transformed using a differentiable, nonlinear activation function h(·) to give:

$$z_j = h(a_j). \quad (3)$$

These quantities correspond to the outputs of the basis equation in (1) that, are called hidden units. The nonlinear functions h(·) are generally chosen to be sigmoidal functions such as the logistic or the hyperbolic tangent function. Following (1), these values are again linearly combined to give output unit activations:

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}, \quad (4)$$

∀ k=1,…, $K$ as the number of outputs. This transformation corresponds to the second layer of the network, and again the $w_{k0}^{(2)}$ are the bias parameters. Finally, the output unit activations are transformed using an appropriate activation function to give a set of network outputs $a_k$. The choice of activation function is determined by the nature of the data and the assumed distribution of target variables.

### 3.4. ANN models for genome-enabled prediction

There is a wide range of ANNs utilization to genome enabled predictions; in general ANNs could be implemented in a non-penalized, penalized and also in a Bayesian framework (de Los Campos *et al.* 2013). The most used ANN in genome-enabled prediction of complex traits is the multilayer feed-forward network, also called multilayer feed-forward perceptron (MLP). The central idea is to extract linear combinations of the inputs as derived features in the hidden layer and then model the target as a function of these features in the output layer (Ehret *et al.* 2014).

In the context of genomic enable prediction the hidden layer of an ANN has an input matrix $(x_{ij})$ of individual $i$ $(i = (1,2, \dots n)$ with $j$ $(j = 1,2, \dots, p)$ genomic covariates (e.g. marker genotypes). These information are linearly combined in the hidden layer with a vector of weights, $w_{1j}^{[t]}$, plus an intercept, or "bias", $a_t$, where $t = 1,2, \dots, s$ represents one of $s$ neurons. The resultant linear combination of this process is then transformed using an activation function, $f(\cdot)$, generating the output of the hidden neuron $t$ for individual $i$:

$$z_i^{[t]} = f_t\left(\sum_{j=1}^{p} w_{1j}^{[t]} x_{ij} + a_t\right), \quad (5)$$

The activation function $f_t(\cdot)$ can be either linear or non-linear and has to be monotonically increasing. A non-linear activation function in the hidden layer gives the neural network a greater flexibility than standard linear regression models (Bishop

2006) because the transformation is adaptive via $w$ the parameters. In the output layer, the $s$ data-derived scores and an output intercept ("bias") $b$ are combined and transformed by the use of another activation function to yield the predicted phenotype:

$$y_i = g\left(\sum_{t=1}^{s} w_{2t} z_i^{[t]} + b\right) + e_i, \quad (6)$$

In which, $w_{21}, w_{2t}, .., w_{2s}$ are the weights connecting the $s$ derived functions with the output, $g(\cdot)$ is another activation function, which can be linear or non-linear although an identity function is used in most cases, when is a continuous-valued phenotype.

The back-propagation algorithm is powerful and simple. From a practical perspective, this learning rule and its variations are flexible enough to cover highly complex interactions between predictor variables. Flexibility reaches a limit when the network architecture is not optimally chosen, or when the number of predictors is very large compared to the data cases available. In this case there is a drastic increase in model complexity and, hence, in number of parameters to estimate. This is prone to occur when multi-layer perceptron (MLP) are used for genome-enabled prediction problems in which p>>n. It also leads to a very slow convergence of the back-propagation algorithm and can cause over-fitting. For an empirical control of model complexity, the distribution of the connection strengths (i.e., weights and biases) of the network can be taken as an indicator of the extent of the regularization attained.

Typically, weight values decrease as model complexity increases (Gianola*, et al.* 2011). A natural and automated approach of dealing with model complexity is to use penalized Bayesian (mostly Bayesian), where a compromise between model goodness of fit and complexity of the network is naturally attained without the need of artificial constraints (Okut*, et al.* 2011). This leads to a subgroup of MLPs, called Bayesian regularized artificial neural networks (BRNN). This class of ANN were first introduced by Neal (1992) and Mackay (1992). The main difference with respect to the MLPs, in which an optimal set of weights is chosen by minimizing a cost function is that the Bayesian approach involves a prior probability distribution of the network weights (Ehret *et al.* 2014). Regularization is obtained by treating the weights as random effects following a specified prior distribution, given the observed data (Lampinen and Vehtari 2001). Gianola *et al.* (2011) adapted BRNN methodology to prediction of complex phenotypes with genomic data.

### 3.5. Bayesian regularized artificial neural network

A Bayesian Regularized Artificial Neural Network (BRANN) is a feed-forward network based on the maximum a posteriori approach in which the regularizer is the logarithm of a prior density distribution (Bishop 2006). This model assigns a probability distribution to the network weights and biases, so that predictions are made in a Bayesian framework and generalization is improved over predictions made without Bayesian regularization (Felipe*, et al.* 2014).

In BRANN, in addition to the loss function, given by the sum of squared errors, a penalty is also included in order to have a regularization (i.e., shrink to zero non-relevant weights). Thus, the objective function is given by:

$$f = \gamma E_D(D|w,M) + \alpha E_w(w|M), \quad (7)$$

where $E_D(D|w,M)$ is the residuals sum of squares in which D represents the observed data (input data and target variable), $w$ are the weights and M represents the architecture of the neural network. Further, $E_w(w|M)$ is known as weight decay that is calculated as the sum of squares of weights of the network, and α and γ are the regularization parameters that control the trade-off between goodness of fit and smoothing. Following the Bayes theorem, the posterior distribution of w given α, γ, D and M is:

$$P(w|D,\alpha,\gamma,M)=\frac{P(D|w,\gamma,M)P(w|\alpha,M)}{P(D|\alpha,\gamma,M)}, \quad (8)$$

where $P(D|w,\gamma,M)$is the likelihood function, $P(w|\alpha,M)$ is the prior distribution on weights under the chosen architecture, and $P(D|\alpha,\gamma,M)$ is the normalization factor (Gianola*, et al.* 2011), also called marginal likelihood.

The advantage of BRANN is that the models are robust in relation to the validation process (Hawkins*, et al.* 2003). This family of networks automatically solves a number of important problems that arise in modeling, such as choice of model, robustness of model, choice of validation set, size of validation effort, and optimization of network architecture. Bayesian regularized neural networks have additional advantages (Burden and Winkler 2009): they difficult overtraining beacause an evidence procedure provides an objective criterion for stopping training and removes the need for a separate validation set to detect the onset of overtraining; they are difficult to overfit because they calculate and train on the effective number of parameters (essentially the number of nontrivial weights in the trained neural network). The effective number of parameters is considerably smaller than the number of weights

in a standard fully connected back-propagation neural network. Bayesian neural networks essentially incorporate Occam's razor, automatically and optimally penalizing excessively complex models. As the architecture is made more complex (e.g., by increasing the number of hidden-layer neurodes), the number of effective parameters converges to a constant. This provides an optimum balance between bias and variance (where the model is excessively complex and fits the noise); Bayesian neural networks are inherently insensitive to the architecture of the network, as long as a minimal architecture has been provided; it has been shown mathematically that they do not strictly need a test set, as they produce the best possible model most consistent with the data. This has the advantage that a parsimonious and optimal model is provided, all available data may be used in the model (an advantage where data are scarce and expensive to acquire), and the validation effort is removed, which could be demanding for large data sets.

Overfitting problems can arise if too many neurons (in neural networks called "neurode") are used, since each additional neurode introduces $N_v + 2$ new weights ($N_v$ being the number of independent variables or molecular descriptors). For instance, for a regression using 20 molecular descriptors and 5 hidden neurodes there will be $(20 + 1) \times 5 + (5 + 1) = 111$ weights, including input and hidden-layer bias neurodes. The addition of a sixth neurode adds more 22 weights to the model. Since the number of weights generally should not exceed half the number of samples (molecules), care needs to be taken to not use too many neurodes (Burden and Winkler 2009). For ANNs, it is not always clear when the optimum number of neurodes has been employed because the addition of new neurodes may cause the training error to decrease, but overfitting may occur decreasing the prediction success of the net. A large number of iterations is needed for training an ANN in order to reach the minimum in the validation set error, and many repetitions need to be made with alternative validation sets. If the net architecture is modified in attempting to find the optimum number of hidden neurodes, this training and validation effort is magnified (Beale*, et al.* 2010).

BRANNs avoid overfitting because the regularization shrinks unnecessary weights towards zero, effectively eliminating them. In the case of BRANNs, since there is no need for validation set during training, the necessary number of neurodes is easily found to the extent that if too many are used the number of effective parameters barely changes. This is effectively a pruning of the network in that many of the weights are set to near zero (Gianola*, et al.* 2011). Generally, the BRANN method is far more robust,

parsimonious and efficient than a simple ANN. Also, its weights are more useful because they are not obtained from a committee of networks as commonly is done when multiple validation sets are used.

## 4. References


Alados, I., Mellado, J. A., Ramos, F. and Alados-Arboledas, L. (2004) Estimating UV erythemal irradiance by means of neural networks. Photochem Photobiol,80, 351-358.

Beale, M. H., Hagan, M. T. and Demuth, H. B. (2010) Neural Network Toolbox 7. User's Guide, MathWorks.

Bernardo, R. and Yu, J. M. (2007) Prospects for genomewide selection for quantitative traits in maize. Crop Science,47, 1082-1090.

Bishop, C. M. (2006) *Pattern recognition and machine learning*. springer New York.

Buckler, E. S., Holland, J. B., Bradbury, P. J., Acharya, C. B., Brown, P. J., Browne, C., Ersoz, E., Flint-Garcia, S., Garcia, A., Glaubitz, J. C., Goodman, M. M., Harjes, C., Guill, K., Kroon, D. E., Larsson, S., Lepak, N. K., Li, H., Mitchell, S. E., Pressoir, G., Peiffer, J. A., Rosas, M. O., Rocheford, T. R., Romay, M. C., Romero, S., Salvo, S., Sanchez Villeda, H., da Silva, H. S., Sun, Q., Tian, F., Upadyayula, N., Ware, D., Yates, H., Yu, J., Zhang, Z., Kresovich, S. and McMullen, M. D. (2009) The genetic architecture of maize flowering time. Science,325, 714-718.

Burden, F. and Winkler, D. (Yeaer) Bayesian Regularization of Neural Networks. In: Livingstone, D. (ed.), *Artificial Neural Networks*, Humana Press, pp. 23-42.

Crossa, J., de los Campos, G., Perez, P., Gianola, D., Burgueno, J., Araus, J. L., Makumbi, D., Singh, R. P., Dreisigacker, S., Yan, J. B., Arief, V., Banziger, M. and Braun, H. J. (2010) Prediction of Genetic Values of Quantitative Traits in Plant Breeding Using Pedigree and Molecular Markers. Genetics,186, 713-U406.

de los Campos, G., Gianola, D. and Allison, D. B. (2010) Predicting genetic predisposition in humans: the promise of whole-genome markers. Nature reviews. Genetics,11, 880-886.

de Los Campos, G., Gianola, D., Rosa, G. J., Weigel, K. A. and Crossa, J. (2010) Semi-parametric genomic-enabled prediction of genetic values using reproducing kernel Hilbert spaces methods. Genetics Research,92, 295-308.

de Los Campos, G., Hickey, J. M., Pong-Wong, R., Daetwyler, H. D. and Calus, M. P. (2013) Whole-genome regression and prediction methods applied to plant and animal breeding. Genetics,193, 327-345.

Ehret, A., Tusell, L., Gianola, D. and Thaller, G. (2014) Artificial neural networks for genome-enabled prediction in animal and plant breeding: A review. Artificial Neural Networks for Genome-Enabled Prediction in Cattle: Potential And Limitations, 17.

Falconer, D. S. and Mackay, T. F. (1996) Introduction to quantitative genetics. Harlow. UK: Longman.

Garrick, D. The nature, scope and impact of some whole-genome analysis in beef cattle. In: Proceedings of the Proceedings of the 9th World Congress on Genetics Applied to Livestock Production: 1-6 August 2010; Leipzig. 2010. Copyright Publisher|, Place Published.

Gianola, D., Okut, H., Weigel, K. A. and Rosa, G. J. (2011) Predicting complex quantitative traits with Bayesian neural networks: a case study with Jersey cows and wheat. BMC Genetics,12, 87.

González-Recio, O., Gianola, D., Long, N., Weigel, K. A., Rosa, G. J. and Avendano, S. (2008) Nonparametric methods for incorporating genomic information into genetic evaluations: an application to mortality in broilers. Genetics,178, 2305-2313.

Habier, D., Fernando, R. and Dekkers, J. (2007) The impact of genetic relationship information on genome-assisted breeding values. Genetics,177, 2389-2397.

Haley, C. S. and Visscher, P. M. (1998) Strategies to utilize marker-quantitative trait loci associations. Journal of Dairy Science,81 Suppl 2, 85-97.

Harris, B., Johnson, D., Spelman, R. and Sattler, J. (Year) Genomic selection in New Zealand and the implications for national genetic evaluation. In: Proceedings of the Identification, Breeding, Production, Health and Recording of Farm Animals. Proceedings of the 36th ICAR Biennial Session, Niagara Falls, USA, 16-20 June, 2008., 2009. Copyright Publisher, Place Published.

Hawkins, D. M., Basak, S. C. and Mills, D. (2003) Assessing model fit by cross-validation. Journal of chemical information and computer sciences,43, 579-586.

Hayes, B. J., Bowman, P. J., Chamberlain, A. J. and Goddard, M. E. (2009) Invited review: Genomic selection in dairy cattle: progress and challenges. Journal of Dairy Science,92, 433-443.

Hebb, D. (1949) The organization of behavior; a neuropsychological theory.

Hopfield, J. J. (1982) Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences of the United States of America,79, 2554-2558.

Hopfield, J. J. (1984) Neurons with graded response have collective computational properties like those of two-state neurons. Proceedings of the national academy of sciences,81, 3088-3092.

Lamontagne, L. and Marchand, M. (2006) Advances in Artificial Intelligence: 19th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2006, Quebec City, Quebec, Canada, June 7-9, Proceedings. Springer Science & Business Media.

Lampinen, J. and Vehtari, A. (2001) Bayesian approach for neural networks—review and case studies. Neural networks,14, 257-274.

Lande, R. and Thompson, R. (1990) Efficiency of marker-assisted selection in the improvement of quantitative traits. Genetics,124, 743-756.

Long, N. Y., Gianola, D., Rosa, G. J. M., Weigel, K. A., Kranis, A. and Gonzalez-Recio, O. (2010) Radial basis function regression methods for predicting quantitative traits using SNP markers. Genetics Research,92, 209-225.

Mackay, D. J. C. (1992) A Practical Bayesian Framework for Backpropagation Networks. Neural Computation,4, 448-472.

Manolio, T. A., Collins, F. S., Cox, N. J., Goldstein, D. B., Hindorff, L. A., Hunter, D. J., McCarthy, M. I., Ramos, E. M., Cardon, L. R., Chakravarti, A., Cho, J. H., Guttmacher, A. E., Kong, A., Kruglyak, L., Mardis, E., Rotimi, C. N., Slatkin, M., Valle, D., Whittemore, A. S., Boehnke, M., Clark, A. G., Eichler, E. E., Gibson, G., Haines, J. L., Mackay, T. F., McCarroll, S. A. and Visscher, P. M. (2009) Finding the missing heritability of complex diseases. Nature,461, 747-753.

McCulloch, W. S. and Pitts, W. (1943) A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics,5, 115-133.

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

Neal, R. M. (Yeaer) Bayesian training of backpropagation networks by the hybrid Monte Carlo method. Citeseer.

Nejati-Javaremi, A., Smith, C. and Gibson, J. P. (1997) Effect of total allelic relationship on accuracy of evaluation and response to selection. Journal of Animal Science,75, 1738-1745.

Ober, U., Erbe, M., Long, N., Porcu, E., Schlather, M. and Simianer, H. (2011) Predicting genetic values: a kernel-based best linear unbiased prediction with genomic data. Genetics,188, 695-708.

Okut, H., Gianola, D., Rosa, G. J. M. and Weigel, K. A. (2011) Prediction of body mass index in mice using dense molecular markers and a regularized neural network. Genetics Research,93, 189-201.

Park, T. and Casella, G. (2008) The Bayesian Lasso. Journal of the American Statistical Association,103, 681-686.

Pereira, B. and Rao, C. (2009) Data mining using neural networks: A guide for statisticians. State College, Pennsylvania.

Quilez Oliete, J. (2012) Application of genome-wide single-nucleotide polymorphism arrays to understanding dog disease and evolution.

Rosenblatt, F. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. 1962. Washington DC: Spartan.

Rumelhart, D., McClelland, J. L. and Group, P. R. (Yeaer) Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Psychological and Biological Models, Vol. 2. Cambridge: MIT Press.

Soller, M. (1978) The use of loci associated with quantitative effects in dairy cattle improvement. Animal production,27, 133-139.

Soller, M. and Plotkin-Hazan, J. (1977) The use marker alleles for the introgression of linked quantitative alleles. Theoretical and Applied Genetics,51, 133-137.

VanRaden, P. M., Van Tassell, C. P., Wiggans, G. R., Sonstegard, T. S., Schnabel, R. D., Taylor, J. F. and Schenkel, F. S. (2009) Invited review: reliability of genomic predictions for North American Holstein bulls. J Dairy Sci,92, 16-24.

Vapnik, V. N. and Vapnik, V. (1998) *Statistical learning theory*. Wiley New York.

Vazquez, A. I. (2010). Statistical modeling of genomic data: applications to genetic markers and gene expression (Doctoral dissertation, University of Wisconsin-Madison).

Von Neumann, J. (1951) The general and logical theory of automata. Cerebral mechanisms in behavior, 1-41.

Von Neumann, J. (1956) Probabilistic logics and the synthesis of reliable organisms from unreliable components. Automata studies,34, 43-98.

Wang, W. Y., Barratt, B. J., Clayton, D. G. and Todd, J. A. (2005) Genome-wide association studies: theoretical and practical concerns. Nature reviews. Genetics,6, 109-118.

Whittaker, J. C., Thompson, R. and Denham, M. C. (2000) Marker-assisted selection using ridge regression. Genetical Research,75, 249-252.

Yadav, N., Yadav, A., & Kumar, M. (2015). An Introduction to Neural Network Methods for Differential Equations. SpringerBriefs in Applied Sciences and Technology.

# CHAPTER 2

Manuscript submitted to Journal of Animal Breeding and Genetics (A2 from Qualis CAPES)

**Signaling for SNP effects and heritability estimates in genome prediction based on Bayesian regularized neural networks**


## Abstract

Recently there is an increase interest to use nonparametric methods, such as artificial neural networks (ANN). In animal breeding, a especial class of ANN called Bayesian Regularized Neural Network (BRNN) has been preferable since it does not demand a priori knowledge of the genetic architecture of the characteristic (assumption about markers effects) as assumed by the most used parametric methods (RR-BLUP, Bayes A, B, C$\pi$, BLASSO, and many others). Although BRNN has been shown to be effective for genomic enable prediction, its results approaching marker effects and genetic parameter estimates are still been scarcely used. The aim of the present study was to apply the ANN based on Bayesian regularization to genome-enable prediction regarding simulated data sets, to select the most relevant SNP markers by using two proposed methods, to estimate heritabilities for the considered traits, and to compare the results with two traditional methods (RR-BLUP and BLASSO). The simplest Bayesian Regularized Neural Network (BRNN) model gave consistent predictions for both traits, which were similar to the results obtained from the traditional RR-BLUP and BLASSO methods. The SNP importance identification methods based on BRNN proposed here showed correlation values (0.61 and 0.81 for traits 1 and 2, respectively) between true and estimated marker effects higher than the traditional BLASSO (0.55 and 0.71, respectively for traits 1 and 2) method. With respect to $h^2$ estimates (assuming 0.35 as true value), the simplest BRNN found 0.33 for both traits, thus outperforming the RR-BLUP and BLASSO, that in average estimated $h^2$ equal to 0.215.



**Keywords:** Genomic selection, neural networks, genetic parameters.

**Introdution**

The importance of Genome-enabled selection (GS) have increased in animal breeding area for the last few years due to the rapid development of DNA sequencing technologies that have allowed large-scale genotyping of thousands of genetic markers in animal species. Hence, the number of markers (p) typically runs into thousands and often far exceeds the number of phenotypes (n), leading to the classic p>>n problem. That is just one of other statistics and computation challenges.

Over the years, many parametric methods have been proposed to solution these challenges, some of these are: RR-BLUP, Elastic net (EN), Bayesian LASSO (BLASSO), Bayes A, B and C$\pi$ (de Los Campos*, et al.* 2013). However recently, there is an increase interest to use semi-parametric methods such as kernel-based methods (González-Camacho*, et al.* 2012), support vector machines (Moser*, et al.* 2009) and artificial neural networks (ANN) (Gianola*, et al.* 2011).

In the context of genome-enable prediction, the high dimensional nature of high-throughput SNP-marker data sets has required the versatility of regularization methods, such as the Bayesian Regularized Neural Network (de Los Campos*, et al.* 2013). Many studies using this method have been proposed in the recently years (Gianola*, et al.* 2011, González-Camacho*, et al.* 2012, Okut*, et al.* 2011). This method is characterized as universal approximators of complex functions (Hornik*, et al.* 1989), which are based on the use of a wide range of arbitrary functions (called "activation functions") able to intuitively exploit linear and non-linear relationships between a response variable (phenotype) and several other predictor variables (the genotypes). This is possible using training algorithms, which updated specific “weights” (divided into different levels called “neurons”), for each explanatory variable in different levels of learning (called "layers"). Thus, in animal breeding, this regularized Bayesian viewpoint is preferable

because the regularization is obtained by treating the weights as random effects following a specified prior distribution, given the observed data (Felipe*, et al.* 2014). Furthermore, this perspective presents the advantage of not demand a priori knowledge of the genetic architecture of the characteristic (assumption about the markers effects) as recommended by most used methods in genome enable prediction, like Bayes A, B, C$\pi$ and LASSO.

Although ANN has been shown to be effective for genomic enable prediction, even producing results similar to, or better than, the traditional methods in the areas of animal (Perez-Rodriguez*, et al.* 2013, Shahinfar*, et al.* 2012) and plant breeding (Crossa*, et al.* 2014, Heslot*, et al.* 2012), biological interpretation from the marker effects (like QTL detection) and genetic parameter estimates (like heritability) are still been scarcely used. Thus, studies taking into account the advantages in the predictive ability of ANN approaching the identification of relevant chromosome regions as well as the genetic parameter estimates can be seen as an interesting research field in Genetics and Breeding.

The aim of the present study was to apply the ANN based on Bayesian regularization to genome-enable prediction regarding simulated data sets, to select the most relevant SNP markers by using two proposed methods, to estimate heritabilities for the considered traits, and to compare the results with two traditional methods (RR-BLUP and BLASSO).

## Materials and methods

### Simulated data set

An outbred population was simulated for the 16$^{th}$ QTLMAS Workshop (2012). A base population (G0) of 1,020 unrelated individuals (20 males and 1,000 females)

was generated. Each of the next four generations (G1-G4) consisted of 20 males and 1,000 females, and was originated from the previous one by randomly mating each male with 50 females. All the females produced one female individual, except for 20 of them, which generated two individuals, one male and one female. During the simulation, generations did not overlap by themselves.

The pedigree of all 4100 individuals, including the individual identity, sire, dam, sex and generation were identified. Also, they were saved individual´s genotypes wich were composed by bi-allelic SNP (single nucleotide polymorphisms) genetic markers with known location on each chromosome.

The genome simulated comprised five chromosomes, each one with 100 Mb size and 2000 equally distributed SNPs (total of 10,000 SNPs per genome). Fifty QTL positions were sampled from among the even SNPs. The allele substitution effects of the QTLs were drawn from a gamma distribution with scale parameter 5.4 and shape parameter 0.42 (Hayes and Goddard 2001). The effects were standardized and their sign were sampled to be positive or negative with probability 0.5. Two milk production quantitative traits, expressed only in females individuals, were simulated.

True breeding values (TBV) for the two simulated traits were calculated as the sum of the additive effects of the 50 QTLs in each individual. Random residuals were drawn from normal distributions with mean zero and trait-specific residual variances to simulate heritability of 0.35 for milk yield (T1) and fat yield (T2). The phenotypes, given as individual yield deviations, were determined only for the 3,000 females from G1 to G3. The remainder 1,020 genotyped individuals, from the fourth generation, were used as the validation set to estimate the genomic estimated breeding value (GEBV).

**RR-BLUP and Bayesian Lasso**

With respect to the GWS, the phenotypic outcomes, which are denoted by $y_i$ (i =1, 2,…, to 3000*),* were regressed on the marker covariates $x_{ik}$ (k=1, 2, …, and 10,000) following the regression model that was previously proposed by (Meuwissen*, et al.* 2001):

$$y_i = \mu + \sum_{k=1}^{10,000} x_{ik}\beta_k + e_i \quad (1)$$

where $y_i$ is the phenotypic observation of animal $i$, μ is the general mean, $\beta_k$ is the effect of marker k, and $e_i$ is the residual term $e_i \sim N(0,\sigma_e^2)$. In this model, $x_{ik}$ equals a value of 2, 1 or 0 that represents the homozygous, heterozygous and the other homozygous for SNP genotypes representing the allele copy number at each locus k.

In the RR-BLUP (Meuwissen*, et al.* 2001) method, $\beta_k$ is the random marker effect $\beta_k \sim N(0,\sigma^2_{\beta_k})$, which assumes that $\sigma^2_{\beta_1} = \sigma^2_{\beta_2} = ... = \sigma^2_{\beta_{2,500}} = \sigma^2_{\beta}$ (i.e., each locus explains an equal amount of genetic variation). This method was implemented using the R software (R Development Core Team, 2011) package *rrBLUP* (Endelman 2011).

The Bayesian Lasso (BL) (Park and Casella 2008) method is a penalised Bayesian regression procedure whose general estimator is given by $\hat{\boldsymbol{\beta}} = \arg\min_{\beta}\left\{(\hat{\mathbf{y}} - \mathbf{X}\boldsymbol{\beta})'(\hat{\mathbf{y}} - \mathbf{X}\boldsymbol{\beta}) + \lambda\sum_{k=1}^{10,000}|\beta_k|\right\}$, where λ is the regularisation parameter. The BL method was implemented in the package *BLR* (Perez-Rodriguez*, et al.* 2013) of the R software using 30.000 posteriors samples of a 100,000 MCMC chain size, sampled each two iterations after a burn-in of 40,000 iterations.

**Feed-Forward Neural Networks**

In order to better understanding, consider a network with three layers, as shown in Figure 1. Each layer has a weight matrix *W*, a bias vector *b*, an output vector *a*, and

an input vector *p*. To distinguish between the weight matrices for the input or the output in each layer, a label was set as input weight (IW) or layer weight (LW) for the weight matrix of the input and the output, respectively. The number of the layer is appended as a superscript to the variable of interest. Equations for the results of the each layer ($a^1$, $a^2$, $a^3$), are presented as follow:

$$a^1=f^1(IWp+b^1); \quad (2)$$

$$a^2=f^2(LW^1a^1+b^2); \quad (3)$$

$$a^3=f^3(LW^2a^2+b^3). \quad (4)$$

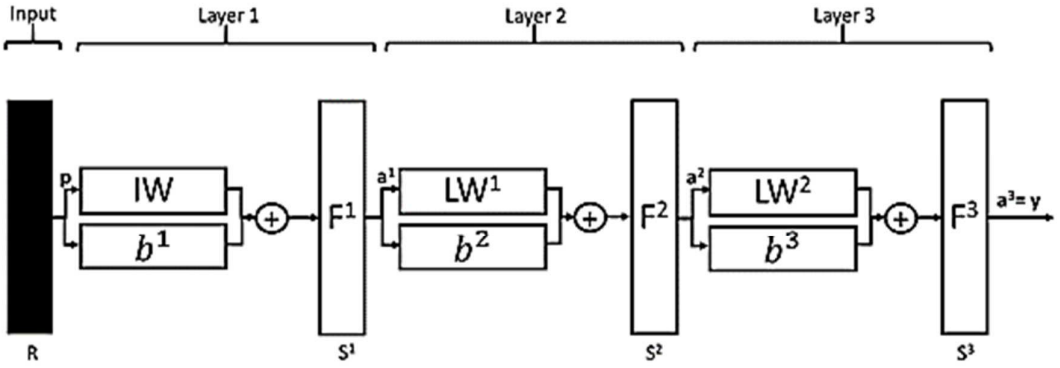



Figure 1- Illustration of a neural network considering a three layers architecture.

In the studied dataset, there were 10,000 SNPs (R) in 3,000(n) animals used as inputs ($p_{R,n}$) for each trait in the training set. Each $p_{R,n}$ is connected to $S^1$ neurons via coefficients $IW_{R,S}^1$. Each neuron has a bias parameter $b_j$ (j denotes neuron). Input variable (p, $a^1$, $a^2$) are combined using a F(.) activation function (linear or non-linear), resulting in a set of inferred weights (W).

The network shown in Figure 1 has R inputs, $S^1$ neurons in the first layer, $S^2$ neurons in the second layer and $S^3$ neurons in the third layer. Note that the outputs of each intermediate layer are the inputs to the following one. Thus, layer 2 can be seen as

a one-layer network with $S^1$ inputs, $S^2$ neurons, and an $S^2 \times S^1$ weight matrix $LW^1$. The input to layer 2 is $a^1$; the output is $a^2$ that will be the input for the layer 3. All the vectors and matrices of layer 2 been identified, it can be treated as a single-layer network on its own. This approach can be done with any layer of the network.

The feed-forward network generates random initial weights and transforms input information (in this case, genotype codes) through each connected neuron in the hidden layer using a linear or non-linear activation function. The result will be sent to the next layer using another activation (transformation) function generating the output or predicted value. Next, the results are back-propagated (non-linear least-squares) in order to update weights and biases using derivatives. Therefore, no assumptions about the relationship between genotypes (input) and phenotypes (target) are made in this model. The predicted values obtained after train a three layers neural network are:

$$a^3=F^3(LW^2(F^2(LW^1(F^1(IW\ p+b^1))+b^2))+b^3) \quad (5)$$

If the $F^1$ is a tangent hyperbolic, $F^2$ a logistic e and $F^3$ an identity activation function, the predict value $a^3$ will be given by:

$$F^1 = \frac{2}{1+\exp(-2IW^p + b^1)} - 1 \quad (6)$$

$$F^2 = \frac{1}{1+\exp(-2LW^{1S} + b^2)} \quad (7)$$

$$F^3 = LW^{2s} + b^3 \quad (8)$$

$$a^3 = \frac{2\sum_{s=1}^{S} LW^{2s}}{1+\exp(-2b^2 - \frac{2\sum_{r=1}^{R} LW^{1r}}{1+\exp(-b^1 - \sum_{p=1}^{P} IW^p X_i)})} + b^3 - \sum_{s=1}^{S} LW^{2s} + e, \quad (9)$$

where P, R and S are the number of the correspondent variable; IW, $LW^1$ and $LW^2$ are the weights; $b^1$, $b^2$ and $b^3$ are the "bias" in the first, second and third layer, respectively; $e$ is a vector of prediction errors.

In this study, we used the standard error of prediction (SEP) to evaluate the prediction ability (Hervás, *et al.* 2001):

$$\text{SEP}=\sqrt{\frac{\sum(\text{True GEBV}-\text{predicted GEBV})^2}{n}}, \quad (10)$$

where, "True GEBV" is the real simulated genomic breeding value and "predicted GEBV" is the breeding value provided by the neural network; "n" is the number of animals in the validation dataset.

The Neural Network Toolbox™ of MATLAB® (Beale, *et al.* 2010) was used for the analysis, and the "trainbr" function in this tool was used for Bayesian regularization in all cases, using 1000 epochs of algorithm run for each network. Six combinations of activation function, number of layers and number of neurons in each layer were tested, Net1 - is the neural network with 1 layer with one neuron and identity activation function. Net2 - is the neural network with two layers, first logistic and second identity activation function with two and one neuron respectively. Net3 - is the neural network with two layers and two identity activation function with two and one neuron respectively. Net4 is the neural network with two layers first tangent hyperbolic and second identity activation function with two and one neuron respectively. Net5 - is the neural network with tree layers first logistic, second tangent hyperbolic, and the last identity activation function with two, two and one neuron respectively. Net6 - is the neural network with tree layers first tangent hyperbolic, second logistic, and the last identity activation function with two, two and one neuron respectively.

**Heritability and SNP effects estimation**

The estimates of SNP effects in the genomic enable prediction using the different proposed neural networks were based in the methods proposed by Garson (1991) and Dimopoulos*, et al.* (1995).

The procedure is based on partitioning the connection weights to determine the relative importance of the various inputs as discusses by Goh (1995). In summary, this method essentially involves partitioning the hidden-output connection weights of each hidden neuron into components associated with each input neuron.

For each hidden neuron h, it is divided the absolute value of the input-hidden layer connection weight by the sum of the absolute value of the input-hidden layer connection weight of all input neurons.

$$Q_{ih} = \frac{|W_{ih}|}{\sum_{i=1}^{ni}|W_{ih}|} \tag{11}$$

For each input neuron $i$, it is divided the sum of the $Q_{ih}$ for each hidden neuron by the sum for each hidden neuron of the sum for each input neuron of $Q_{ih}$. The relative importance of all output weights attributable to the given input variable is then obtained.

$$RI_i = \frac{\sum_{h=1}^{nh} Q_{ih}}{\sum_{h=1}^{nh}\sum_{i=1}^{ni} Q_{ih}} \tag{12}$$

The second method proposed by Dimopoulos*, et al.* (1995) considers the fact that the link between the modification of inputs ($x_j$), and the variation of outputs ($y_j = f(x_j)$) is the Jacobian matrix $dy/dx^T = [\vartheta y/\vartheta x]$. It represents the sensitivity of the network outputs according to small input perturbations. For a network with $n$ layers the Jacobian is:

$$dy/dx^T = D_n W_n D_{n-1} W_{n-1} \dots D_i W_i \dots D_1 W_1, \tag{13}$$

where, $D_i$ is a diagonal matrix containing the elements given by the first partial derivatives of each activation function $\phi$ with respect to its input. In this context, the

upper bound of the Jacobian norm measures the sensitivity of the network, and this norm depends on the number of weights, which is related to the number of layers and the number of nodes per layer (Fu and Chen 1993). Matsuoka (1990) proposed that this norm is added to the function $J$ modifying the learning rule of the back-propagation. For a network with $n$ inputs, one hidden layer with $ni$ nodes, and one output (i.e. $m = 1$), the gradiant vector of $y_j$ with respect to $x_j$ is $d_j = \left[d_{j1}, \ldots, d_{je}, \ldots, d_{jn}\right]^T$, with:

$$d_{je} = s_j \sum_{i=1}^{ni} w_{is} I_{ij} (1 - I_{ij}) w_{ei} \quad (14)$$

(if assuming a logistic sigmoid function to be used for the activation). When $s_j$ is the derivative of the output node with respect to its input, $I_{ij}$ is the output of the *i*th hidden node for the input $x_j$. The scalars $w_{is}$ and $w_{ei}$ are the weights between the output node and the *i*th hidden node, and between the *e*th input node and the *i*th hidden node, respectively.

**SNP effects from neural network using matrix of weights**

The methodology to estimate the SNP effects using the methods showed earlier was adapted from Wang*, et al.* (2012).

Let the animal effects of genotyped animals ($a_g$) are a function of SNP effects:

$$a_g = Zu, \quad (15)$$

where, Z is a matrix relating genotypes of each locus and u is a vector of SNP marker effects. Thus, the variance of the animal effects is:

$$var(a_g) = var(Zu) = ZDZ'\sigma_u^2 = G^*\sigma_\alpha^2, \quad (16)$$

where, D is a diagonal matrix of weights for variances of SNPs (D=I for GBLUP), $\sigma_u^2$ is the additive genetic variance captured by each SNP marker when no weights are present

and $G^*$ is the weighted genomic relationship matrix. In this study, the obtained values using the two methods were used to make the diagonal matrix D.

Therefore, the equation for predicting SNP effects that uses weighted diagonal matrix D becomes:

$$\hat{u} = DZ'[ZDZ]^{-1}\hat{\alpha}_g, \quad (17)$$

where, $\hat{\alpha}_g$, is the genomic estimated breeding value by the neural network.

The $\sigma_\alpha^2$ used was the simplest variance of genomic estimated breeding values from the neural network ($var(\hat{\alpha}_g)$).

Thus, the heritability was calculated as:

$$h^2 = \frac{\sigma_\alpha^2}{\sigma_e^2}, \quad (18)$$

where, $\sigma_e^2$ is the variance of the difference between the phenotypes and genomic estimated breeding values.

## Results

### Predictive performance

In the evaluation of the methods, the following quantities were subjected to comparisons: correlation with the true breeding values (the best are the highest ones, figure 2); Standard error of prediction (the best are the minor ones, figure 3) and heritability estimates (figure 4).

In the figure 2 we can note that the first architecture of neural network (net1), RR-BLUP and the BLASSO provided very closed correlations, being there values, respectively 0.73, 0.73 and 0.75 for trait 1 (1a) and 0.77, 0.77 and 0.80 for trait 2. These mentioned methods outperformed the another one represented by more parameterized neural network.

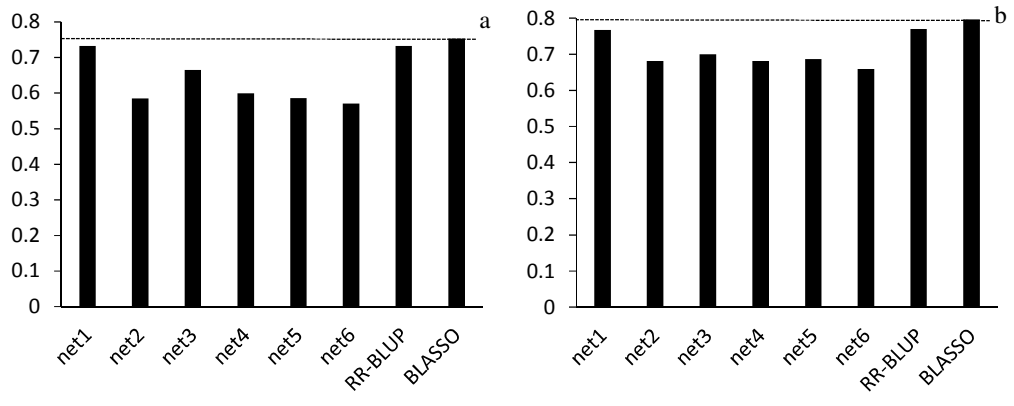



Figure 2. Correlations between simulated genomic value for trait 1 (a) and trait 2 (b). The "net1" is the neural network with 1 layer and identity activation function; "net2" is the neural network with two layers first logistic and second identity activation function; "net3" is the neural network with two layers and two identity activation function; "net4" is the neural network with two layers first tangent hyperbolic and second identity activation function; "net5" is the neural network with tree layers first logistic, second tangent hyperbolic, and the last identity activation function; "net6" is the neural network with tree layers first tangent hyperbolic, second logistic, and the last identity activation function.

**Standard error of prediction (SEP)**

The standard error of prediction (SEP) for trait 1 (Figure 3a) and trait 2 (Figure 3b) obtained by net1, RR-BLUP and BLASSO presented lower values than other methods, thus according with the results of predictive performance presented in figure 1. When we increased the complexity of the neural network by adding a layer ("net3"), the SEP increased too, probably due to overfitting.

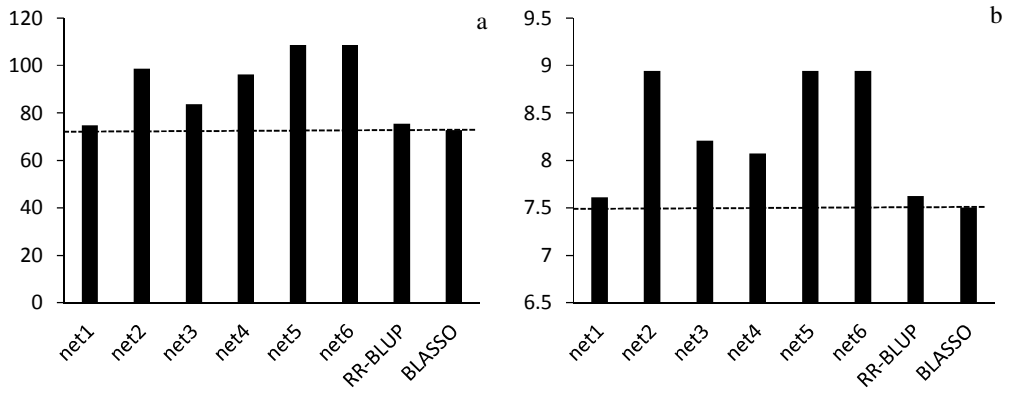



Figure 3. Standard error of prediction for trait 1 (a) and trait 2 (b). The "net1" is the neural network with 1 layer and identity activation function; "net2" is the neural network with two layers first logistic and second identity activation function; "net3" is the neural network with two layers and two identity activation function; "net4" is the neural network with two layers first tangent hyperbolic and second identity activation function; "net5" is the neural network with tree layers first logistic, second tangent hyperbolic, and the last identity activation function; "net6" is the neural network with tree layers first tangent hyperbolic, second logistic, and the last identity activation function.

**Heritability estimates**

For both traits, the best method to estimate heritability (for which the true value was equal to "0.35") was the architecture of neural network with one layer and identity activation function ("net1"), that provided a value equal to 0.33 (Figure 4). The RR-BLUP and BLASSO underestimated the heritabilities, providing respectively the values 0.18 and 0.20 for trait 1, and 0.22 and 0.26 for trait 2.

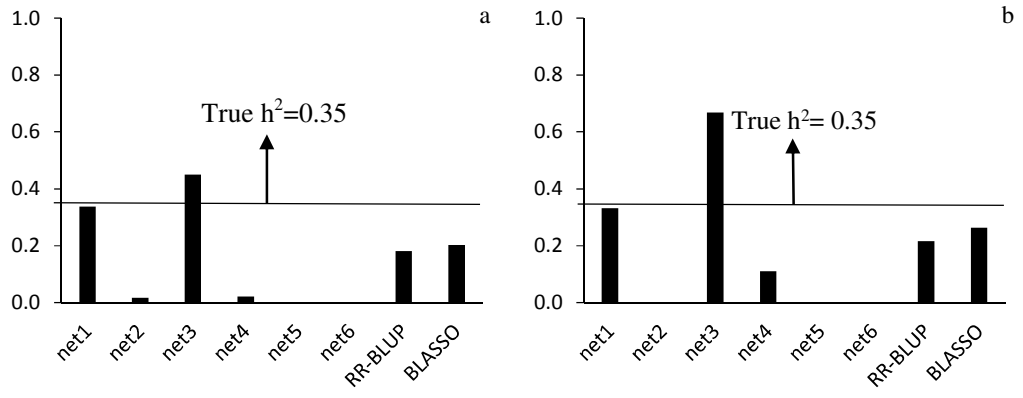



Figure 4. Heritability estimate for trait 1 (a) and trait 2 (b). The "net1" is the neural network with 1 layer and identity activation function; "net2" is the neural network with two layers first logistic and second identity activation function; "net3" is the neural network with two layers and two identity activation function; "net4" is the neural network with two layers first tangent hyperbolic and second identity activation function; "net5" is the neural network with tree layers first logistic, second tangent hyperbolic, and the last identity activation function; "net6" is the neural network with tree layers first tangent hyperbolic, second logistic, and the last identity activation function.

**SNP importance**

The effects of the markers were distributed throughout the five chromosomes for the traits. The Manhattan plots are presented in the figure 5, 6, 7 and 8 calculated using the best prediction with the bayesian regularized neural network with the two methods (Dimopoulus "a" and Garson method "b") for trait 1 and 2 (Figure 5 and 6), and RR-BLUP method (a) and Bayesian Lasso (b) for trait 1 and 2 (Figure 7 and 8). For both traits the BLASSO method penalized more than the other methods (RR-BLUP and net1), thus the number of SNP with values equal 0 is higher in the BLASSO, and the number of QTL discovered using the other methods was higher than the BLASSO

method. In addition, for the trait 1, the correlation between the true top fifty SNP (most relevant) effects and the estimate from Garson (1991), Dimopoulos*, et al.* (1995), RR-BLUP, and the BLASSO, was 0.61, 0.60, 0.60 and 0.55, respectively. For the trait 2 these correlation were 0.81, 0.81, 0.81 and 0.71, respectively.

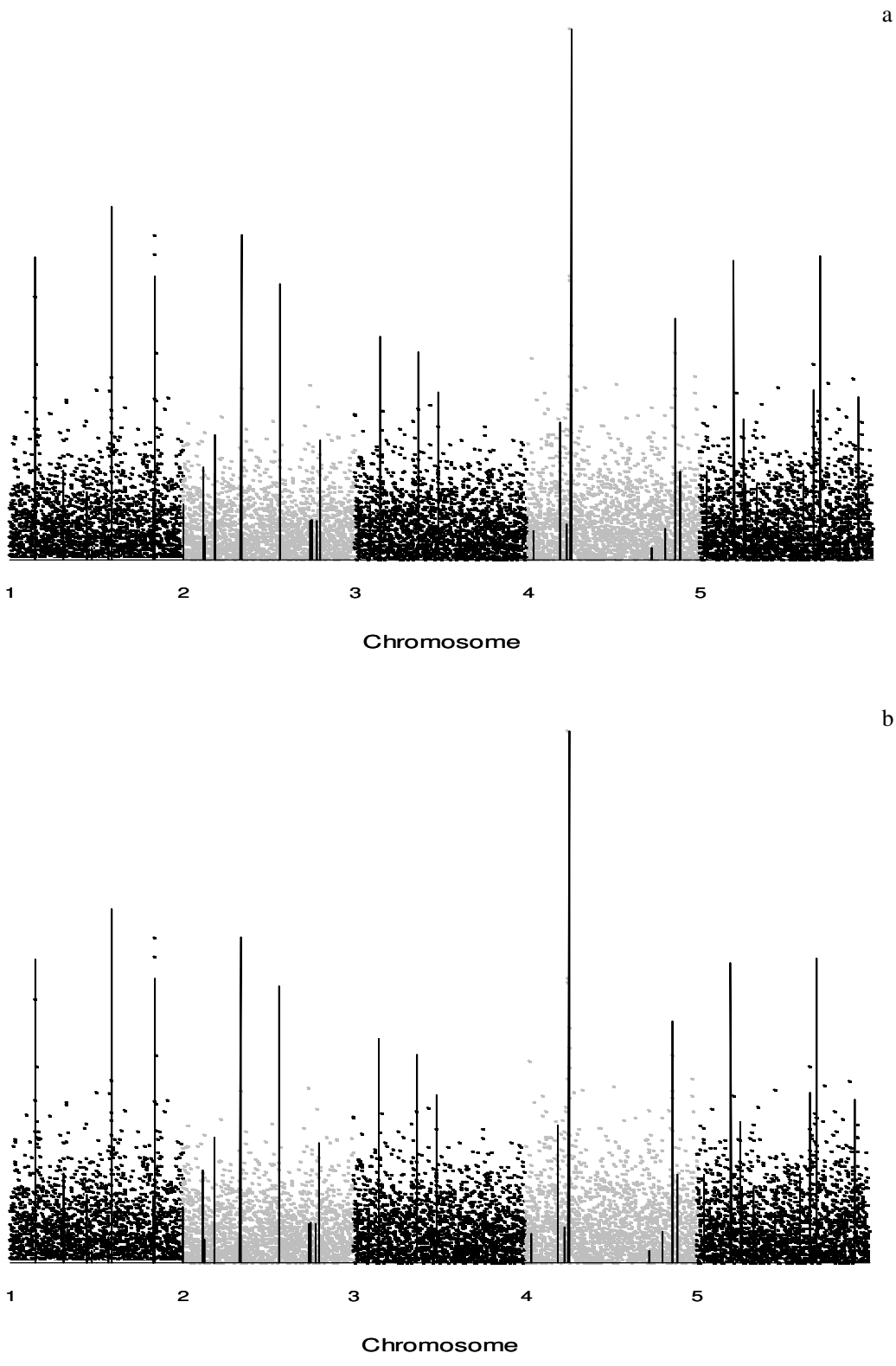
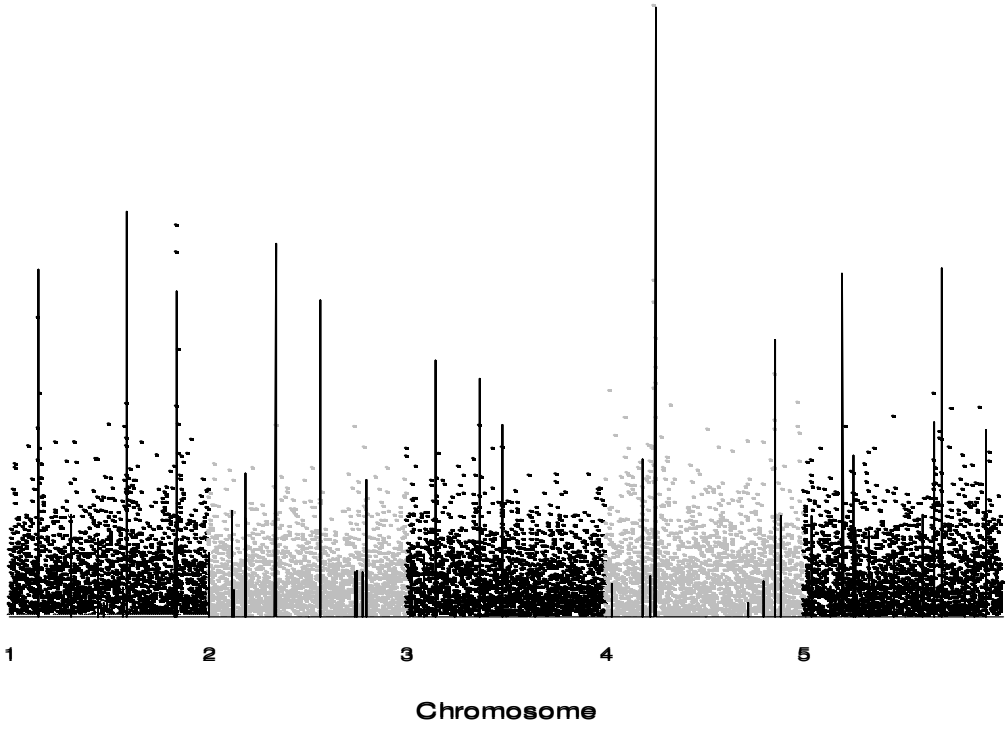



Figure 5. SNP effects calculate for trait 1 with Dimopoulos (a) and Garson (b).

a




b

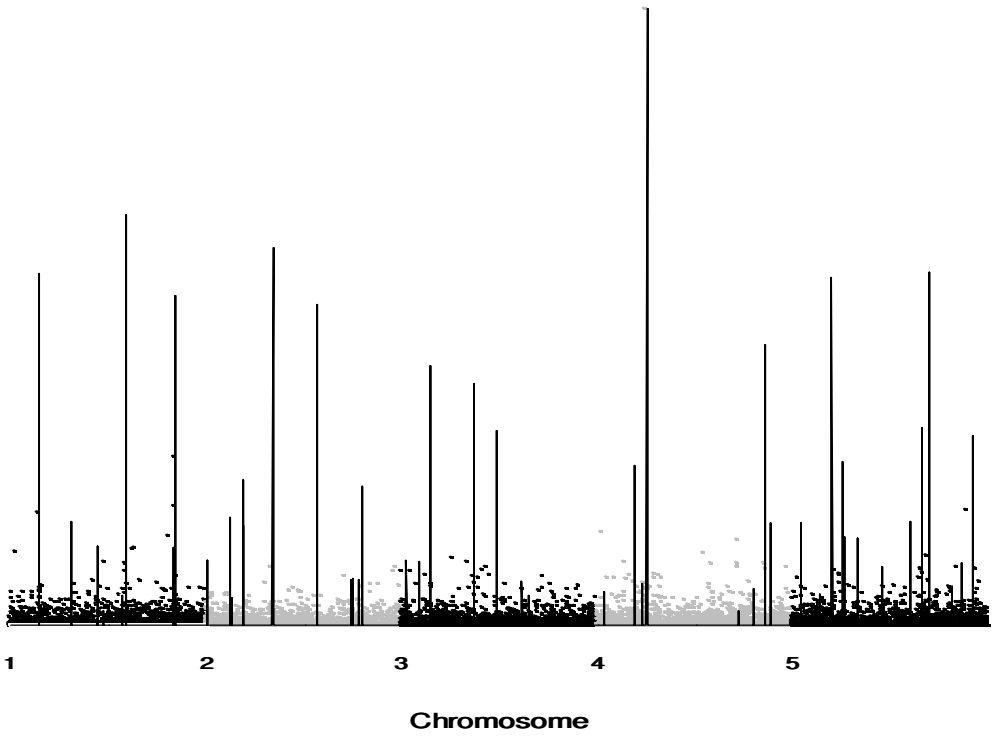



Figure 6. SNP effects calculate for trait 1 with RR-BLUP (a) and BLASSO (b).

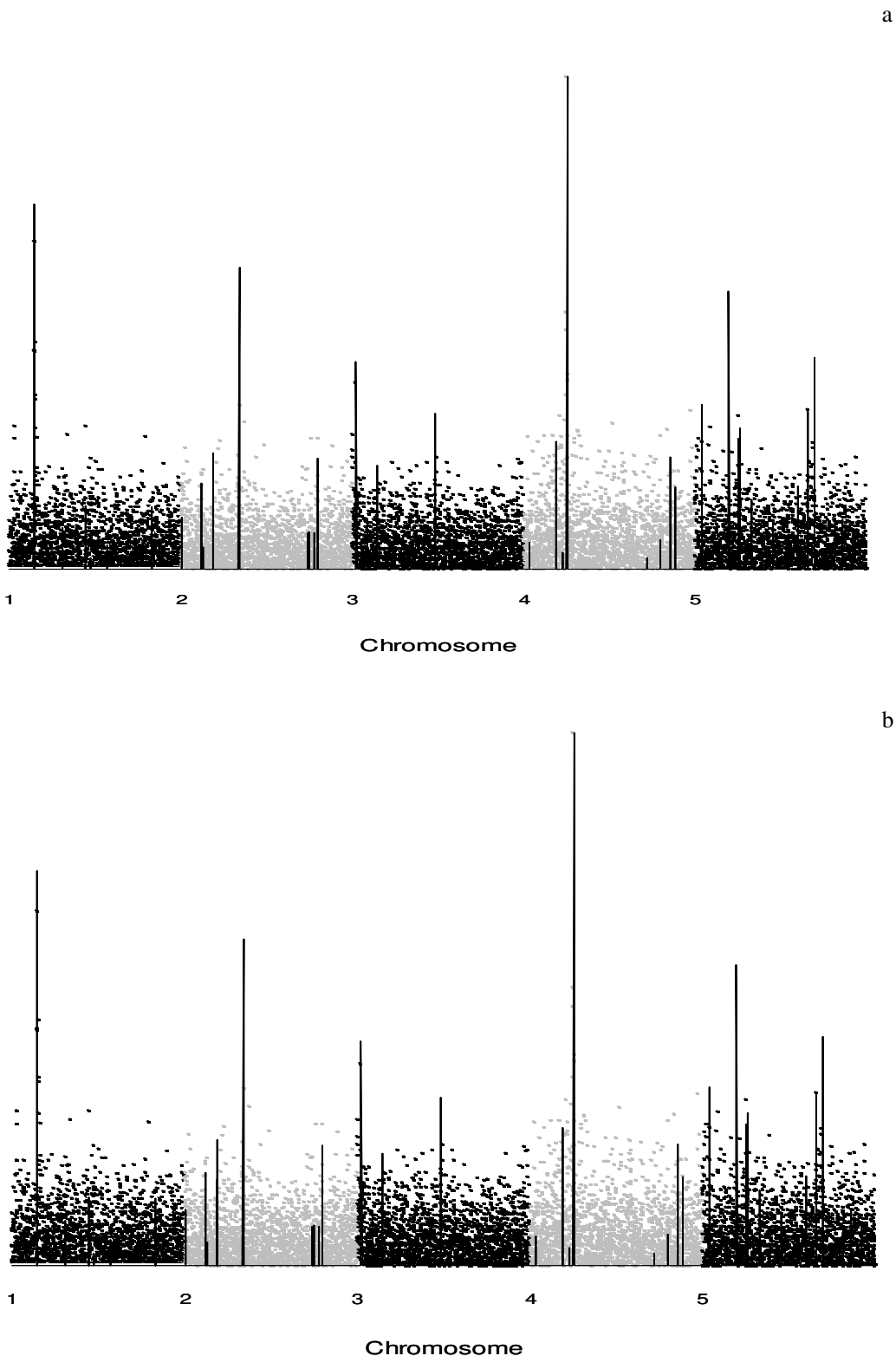



Figure 7. SNP effects calculate for trait 2 with Dimopoulos (a) and Garson (b).

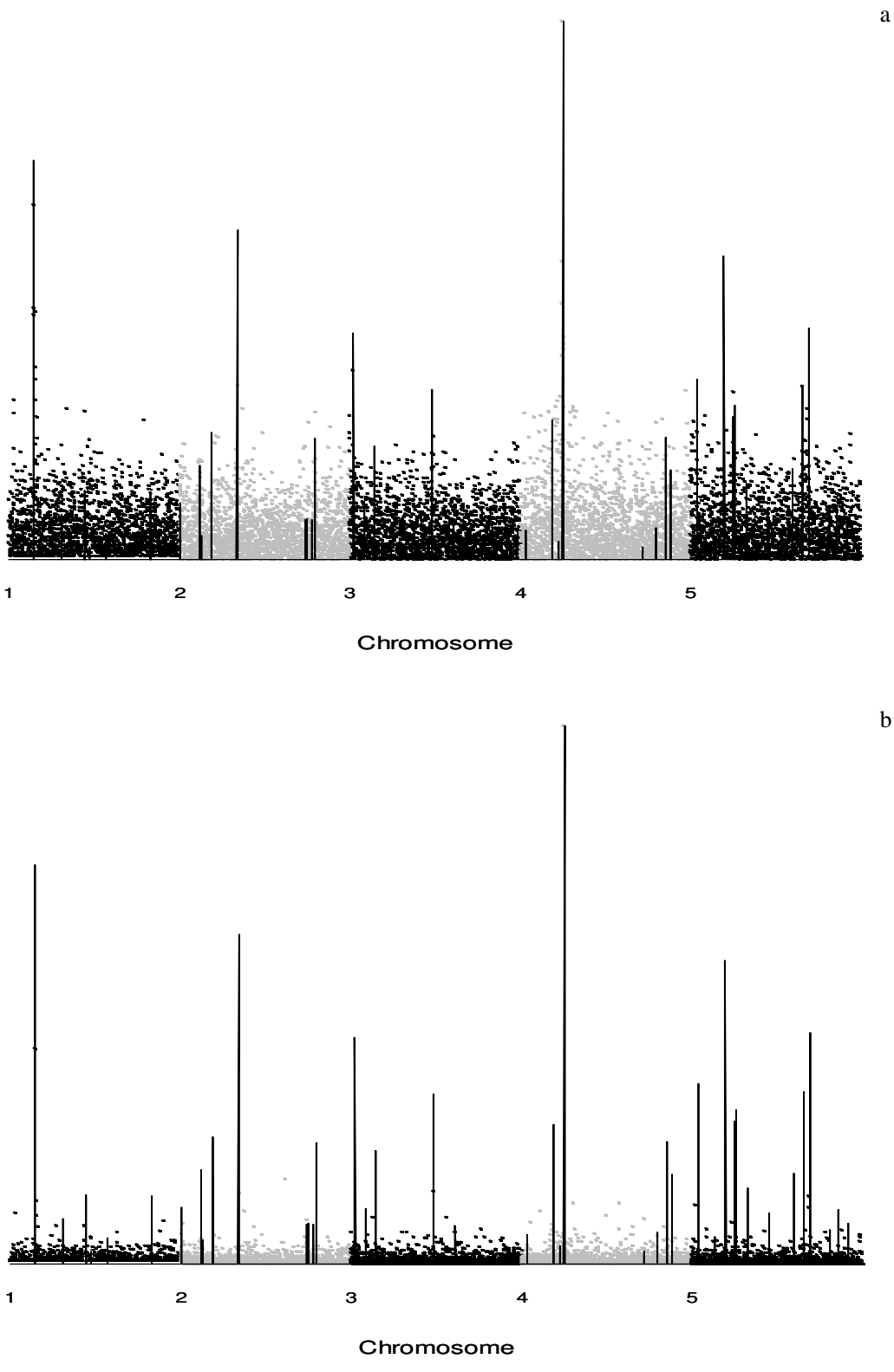



Figure 8. SNP effects calculate for trait 2 with RR-BLUP (a) and BLASSO (b).

## Discussion

We have shown in this study (Figure 2) that the simplest Bayesian Regularized Neural Network (BRNN) was able to predict accurately the genomic breeding values from an outbred population based on 3,000 and 1,020 individuals, respectively in the training and validation data set. Although the BRNN model gave consistent predictions for both traits, similar results were reported for traditional RR-BLUP and BLASSO methods. Therefore, one interesting point to be highlighted is that under a BRNN approach no assumptions about the relationship between genotypes (input) and phenotypes (target) are required, like the other regarded methods. The RR-BLUP assumes, a priori, that all loci explain an equal amount of genetic variation, while the BLASSO allows the assumption that each locus explains its own amount of this variation. According to Okut*, et al.* (2011), the BRNN may be at least as useful as other methods for high-dimensional genome-enabled prediction, with the advantage of its potential ability of capturing non-linear relationships, which may be useful in the study of quantitative traits under complex gene action.

According to (Gianola*, et al.* 2011), the results obtained from BRNN applied to genomic prediction using real data in dairy cattle suggest that this method may be useful for predicting complex traits using high-dimensional genomic information, a situation where the number of coefficients that need to be estimated exceeds sample size. Furthermore, BRNN have the ability of capturing nonlinearities, and do so adaptively, which may be useful in the study of quantitative traits under complex gene action. When applied to real data of litter size in pigs (Tusell*, et al.* 2013), the non-parametric models gave similar predictions to the parametric counterparts, but BRNN giving the best prediction (r=0.31), leading the authors to conclude that this method showed some

promising results under certain scenarios. On the other hand, most parameterized BRNN presented worse predictive ability maybe because the overfitting as mentioned in several other studies applying neural networks for genomic predictions.

According to González-Recio*, et al.* (2014), none of the metrics currently used for comparing genome-assisted evaluation models provide a full representation of predictive ability, so several criteria must be used jointly, with emphasis on those that give the best fit to a particular case. In this context, the standard error of prediction (SEP) it was also used to compare the proposed methods, and the obtained results (Figure 3) were in agreement with those presented by predictive ability comparison.

The high overfitting observed for net2 (with two layers first logistic and second identity activation function), net3 (with two layers and two identity activation function), net4 (with two layers first tangent hyperbolic and second identity activation function), net5 (with tree layers first logistic, second tangent hyperbolic, and the last identity activation function) and net6 (with tree layers first tangent hyperbolic, second logistic, and the last identity activation function) suggested that this model should not be used in its current form. It suggests that most efficient regularizations must be used into these models, like the combination of LASSO and elastic-net framework inside the BRNN structure. As complement, González-Recio*, et al.* (2014) indicated that using shrinkage or regularization in SNP regression models has delivered improvements in predictive ability in genome-based evaluations, thus avoiding serious overfitting arised in genomic selection.

In addition, these overparameterized BRNN models presented high computing requirement in relation to other tested methods (RR-BLUP, BLASSO e simplest BRNN), thus decreasing its availability in genome-enabled prediction studies. Ehret

(2014) reported that in several studies it was empirically found that higher number of hidden layers only increased computation time, while no significant improvement in predictive performance of the BRNN could be achieved.

González-Camacho*, et al.* (2012) found that BRNN had some similarities to BLASSO and seemed to be useful for predicting quantitative traits with different complex underlying gene action under varying types of genetic architecture. These authors suggested that it is possible to make further improvements in the accuracy of the BRNN models by exploiting differential weights for the considered markers. Under this view point, when estimating these weights we might think that some kind of biological explanation come also from artificial neural network, thus broking the paradigm of "black box" behavior described by Ehret (2014). In this context, these estimated weights could be related to QTL detection, so that the markers with higher weights are those most likely to be located into putative QTL regions. In addition, other interest on this biological interpretation of BRNN results could be the genetic parameter estimation, like heritability.

Following this idea, the heritability ($h^2$) values were calculated for both traits based on residual ($\sigma_e^2$) and additive genetic ($\sigma_\alpha^2$) variance estimates (Figure 4). The $\sigma_\alpha^2$ and $\sigma_e^2$ were estimated simply by calculating the variance of the genomic estimate breeding value and residual term, respectively, provided by each one of the considered methods. The true $h^2$ value assumed in the simulation was equal to 0.35, and the net1 found 0.33 for both traits, thus outperforming the RR-BLUP and BLASSO, that in average estimated $h^2$ as 0.215. In relation to this result, we have not found reports in the literature, since the all published studies have focused only in the prediction performance of the methods. Anyway, the results on $h^2$ presented here might enhance future studies using artificial neural networks by allowing exploiting also genetic

parameter estimations, as also reported by RR-BLUP, BLASSO and other Bayesian regression methods (A, B, Cpi, and others).

With respect to biological interpretation of BRNN results, the SNP importance (that could be interpreted as QTL detection) mentioned earlier, was also regarded in the present study (Figures 5 to 8). Many studies already proposed methods to identify and interpret covariate importance from learned neural networks, as example in the field of ecological and environment predictions (Dimopoulos*, et al.* 1995, Garson 1991, Gevrey*, et al.* 2003, Olden and Jackson 2002).

In the field of Genetics and Animal Breeding, up until now these methods have been scarcely used to complement the genomic prediction results from ANN widely reported in the literature. Thus, we think that the SNP importance identification from ANN could be an interesting way to aggregate relevance to the study by giving rise to discussion when exploiting the chromosome regions that contributes most effectively to the phenotype expression. In our concept, there is only one study (Okut*, et al.* 2011) that found out the SNP importance by using BRNN, which used an adaptation of Garson´s method considering only a pre-selected set of 798 SNPs. Even using this reduced number of markers, these authors reported that regarded method was able to detect QTLs and genomic regions most relevant to the studied phenotype, that in this case ws given by body mass index in mices.

Additionally, as presented in Figures 5 to 8, we have proposed in the present study one more method (Dimopoulos, et al., 1995) and exploited a larger number of markers (10,000 SNPs). Fortunately, the SNP importance identification methods based on BRNN proposed here found out effectively the true QTLs postulates in the simulation study, even showing correlation values higher than the traditional BLASSO

method, whose performance was not so positive due to the higher number of true QTLs shrunk to zero.

## Conclusion

The simplest Bayesian Regularized Neural Network (BRNN) model gave consistent predictions for both traits, which were similar to the results obtained from the traditional RR-BLUP and BLASSO methods. The SNP importance identification methods based on BRNN proposed here showed correlation values (0.61 and 0.81 for traits 1 and 2, respectively) between true and estimated marker effects higher than the traditional BLASSO (0.55 and 0.71, respectively for traits 1 and 2) method. With respect to $h^2$ estimates (assuming 0.35 as true value), the simplest BRNN found 0.33 for both traits, thus outperforming the RR-BLUP and BLASSO, that in average estimated $h^2$ equal to 0.215.

## References


Beale, M. H., Hagan, M. T. and Demuth, H. B. (2010) Neural Network Toolbox 7. User's Guide, MathWorks.

Bishop, C. M. (2006) *Pattern recognition and machine learning*. springer New York.

Crossa, J., Perez, P., Hickey, J., Burgueno, J., Ornella, L., Ceron-Rojas, J., Zhang, X., Dreisigacker, S., Babu, R., Li, Y., Bonnett, D. and Mathews, K. (2014) Genomic prediction in CIMMYT maize and wheat breeding programs. Heredity (Edinb),112, 48-60.

de Los Campos, G., Hickey, J. M., Pong-Wong, R., Daetwyler, H. D. and Calus, M. P. (2013) Whole-genome regression and prediction methods applied to plant and animal breeding. Genetics,193, 327-345.

Dimopoulos, Y., Bourret, P. and Lek, S. (1995) Use of Some Sensitivity Criteria for Choosing Networks with Good Generalization Ability. Neural Processing Letters,2, 1-4.

Ehret, A. (2014). Artificial Neural Networks for Genome-enabled Prediction in Cattle: Potential and Limitations (Doctoral dissertation, Selbstverl. des Inst. für Tierzucht und Tierhaltung der Christian-Albrechts-Univ. zu Kiel).

Endelman, J. B. (2011) Ridge Regression and Other Kernels for Genomic Selection with R Package rrBLUP. Plant Genome-Us,4, 250-255.

Felipe, V. P., Okut, H., Gianola, D., Silva, M. A. and Rosa, G. J. (2014) Effect of genotype imputation on genome-enabled prediction of complex traits: an empirical study with mice data. BMC Genetics,15, 149.

Fu, L. and Chen, T. (Year) Sensitivity analysis for input vector in multilayer feedforward neural networks. In: Proceedings of the Neural Networks, 1993., IEEE International Conference on. 1993. Copyright Publisher, Place Published.

Garson, D. G. (1991) Interpreting neural network connection weights.

Gevrey, M., Dimopoulos, L. and Lek, S. (2003) Review and comparison of methods to study the contribution of variables in artificial neural network models. Ecological Modelling,160, 249-264.

Gianola, D., Okut, H., Weigel, K. A. and Rosa, G. J. (2011) Predicting complex quantitative traits with Bayesian neural networks: a case study with Jersey cows and wheat. BMC Genet,12, 87.

Goh, A. (1995) Back-propagation neural networks for modeling complex systems. Artificial Intelligence in Engineering,9, 143-151.

González-Camacho, J., De Los Campos, G., Pérez, P., Gianola, D., Cairns, J., Mahuku, G., Babu, R. and Crossa, J. (2012) Genome-enabled prediction of genetic values using radial basis function neural networks. Theoretical and Applied Genetics,125, 759-771.

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

Hayes, B. and Goddard, M. E. (2001) The distribution of the effects of genes affecting quantitative traits in livestock. Genet Sel Evol,33, 209-229.

Hervás, C., Zurera, G., García, R. M., & Martinez, J. A. (2001). Optimization of computational neural network for its application in the prediction of microbial growth in foods. Food Science and Technology International-New York-, 7(2), 159-164.

Heslot, N., Yang, H. P., Sorrells, M. E. and Jannink, J. L. (2012) Genomic Selection in Plant Breeding: A Comparison of Models. Crop Science,52, 146-160.

Hornik, K., Stinchcombe, M. and White, H. (1989) Multilayer Feedforward Networks Are Universal Approximators. Neural Networks,2, 359-366.

Matsuoka, K. (1990) An approach to generalization problem in back-propagation learning. In: Proceedings of the International Neural Network Conference.

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

Moser, G., Tier, B., Crump, R. E., Khatkar, M. S. and Raadsma, H. W. (2009) A comparison of five methods to predict genomic breeding values of dairy bulls from genome-wide SNP markers. Genet Sel Evol,41, 56.

Okut, H., Gianola, D., Rosa, G. J. M. and Weigel, K. A. (2011) Prediction of body mass index in mice using dense molecular markers and a regularized neural network. Genetics Research,93, 189-201.

Olden, J. D. and Jackson, D. A. (2002) Illuminating the “black box”: a randomization approach for understanding variable contributions in artificial neural networks. Ecological modelling,154, 135-150.

Park, T. and Casella, G. (2008) The Bayesian Lasso. Journal of the American Statistical Association,103, 681-686.

Perez-Rodriguez, P., Gianola, D., Weigel, K. A., Rosa, G. J. and Crossa, J. (2013) Technical note: An R package for fitting Bayesian regularized neural networks with applications in animal breeding. Journal of Animal Science,91, 3522-3531.

Shahinfar, S., Mehrabani-Yeganeh, H., Lucas, C., Kalhor, A., Kazemian, M. and Weigel, K. A. (2012) Prediction of breeding values for dairy cattle using artificial neural networks and neuro-fuzzy systems. Comput Math Methods Med,2012, 127130.

Tusell, L., Pérez-Rodríguez, P., Forni, S., Wu, X.-L. and Gianola, D. (2013) Genome-enabled methods for predicting litter size in pigs: a comparison. animal,7, 1739-1749.

Wang, H., Misztal, I., Aguilar, I., Legarra, A. and Muir, W. M. (2012) Genome-wide association mapping including phenotypes from relatives without genotypes. Genetics Research,94, 73-83.

## Apêndix

### Commands using the Garson method for neural network with one layer (*software R*)

```
iw=as.matrix(t(read.table("iw.txt"))) %weights of the layer 1



  library(MASS)

  pm=abs(iw)  %weights of the layer 1

  tpm=t(pm)

  pmatrix=matrix(ncol=ncol(tpm),nrow=nrow(tpm))

  for(i in 1:nrow(pmatrix)){

    px=tpm[i,]

    pmatrix[i,]=px

  }



  qm=matrix(ncol=ncol(tpm),nrow=nrow(tpm))

  for(i in 1:nrow(pmatrix)){

    for(j in 1:ncol(pmatrix)){

      qm[i,j]=pmatrix[i,j]/sum(pmatrix[i,])

    }}
```

```
  smatrix=c()
  for(i in 1:ncol(pmatrix)){
    smatrix[i]=sum(qm[,i])
  }
  svector=as.matrix(smatrix)# SNPs effects using Garson method
```

**Commands using the Garson method for neural network with two layer (*software R*)**

```
iw=as.matrix(t(read.table("iw.txt"))) %weights of the layer 1
lw1=as.matrix(t(read.table("lw1.txt")))%weights of the layer 2

  library(MASS)
  pm=abs(iw)%*%abs(lw1)  %weights  of  the  layer  1  multiple  with
%weights of the layer 2

  tpm=t(pm)
  pmatrix=matrix(ncol=ncol(tpm),nrow=nrow(tpm))
  for(i in 1:nrow(pmatrix)){
```

```
  px=tpm[i,]
  pmatrix[i,]=px
}

qm=matrix(ncol=ncol(tpm),nrow=nrow(tpm))
for(i in 1:nrow(pmatrix)){
  for(j in 1:ncol(pmatrix)){
    qm[i,j]=pmatrix[i,j]/sum(pmatrix[i,])
  }}

smatrix=c()
for(i in 1:ncol(pmatrix)){
  smatrix[i]=sum(qm[,i])
}
svector=as.matrix(smatrix) # SNPs effects using Garson method
```

**Commands using the Garson method for neural network with tree layer (*software R*)**

```
iw=as.matrix(t(read.table("iw.txt"))) %weights of the layer 1

lw1=as.matrix(t(read.table("lw1.txt")))%weights of the layer 2

lw2=as.matrix(scan("lw2.txt")) %weights of the layer 3




  library(MASS)

  pm=abs(iw)%*%abs(lw1)  %weights  of  the  layer  1  multiple  with
%weights of the layer 2



  lw2=abs(lw2) %weights of the layer 3

  tpm=t(pm)

  pmatrix=matrix(ncol=ncol(tpm),nrow=nrow(tpm))

  for(i in 1:nrow(pmatrix)){

    px=tpm[i,]*lw2[i]

    pmatrix[i,]=px

  }

  dim(pmatrix)

  qm=matrix(ncol=ncol(tpm),nrow=nrow(tpm))

  for(i in 1:nrow(pmatrix)){
```

```
  for(j in 1:ncol(pmatrix)){
    qm[i,j]=pmatrix[i,j]/sum(pmatrix[i,])
  }}

smatrix=c()
for(i in 1:ncol(pmatrix)){
  smatrix[i]=sum(qm[,i])
}
svector=as.matrix(smatrix) # SNPs effects using Garson method
```

**Commands using the Dimopoulus method for neural network with one layer (*software* MATLAB)**

```
wb = formwb(net,net.b,net.iw,net.lw);
[b,IW,LW] = separatewb(net,wb) %extract the net.weights

b1=b{1,1}; %bias of the layer 1
b11=repmat(b1,1,3000);
```

```
a1=iw*x1;

oj=purelin(a1+b11); %output of the layer 1

d1=1;

[a b]=size(d1); %norm of the layer 1

      for i=1:a

md1(i)=norm(d1(i,:));

      end

dmd1=diag(md1);

dj=dmd1*iw;

contri=dj/sum(dj); % SNPs effects using Dimopoulus method
```

**Commands using the Dimopoulus method for neural network with two layer (*software* MATLAB)**

```
wb = formwb(net,net.b,net.iw,net.lw);

[b,IW,LW] = separatewb(net,wb) %extract the net.weights
```

```
b1=b{1,1}; %bias of the layer 1

b11=repmat(b1,1,3000);

b2=b{2,1}; %bias of the layer 2

b21=repmat(b2,1,3000);



a1=iw*x1;

oj=tansig(a1+b11); %output of the layer 1

ok=purelin(lw1*tansig(a1+b11)+b21); %output of the layer 2




d1=dtansig(x1,oj); %calculate the derivative of a hyperbolic

tangent function for the layer 1

d2=1;



[a b]=size(d1); %norm of the layer 1

for i=1:a

md1(i)=norm(d1(i,:));
```

```
end



[c d]=size(d2); %norm of the layer 2

for j=1:c

md2(j)=norm(d2(j,:));

end



dmd1=diag(md1);

dmd2=diag(md2(1,:));




dj=dmd2*lw1*dmd1*iw;



contri=dj/sum(dj); % SNPs effects using Dimopoulus method
```

**Commands using the Dimopoulus method for neural network with tree layer (software MATLAB)**

```
wb = formwb(net,net.b,net.iw,net.lw);

[b,IW,LW] = separatewb(net,wb) %extract the net.weights



b1=b{1,1}; %bias of the layer 1

b11=repmat(b1,1,3000);

b2=b{2,1}; %bias of the layer 2

b21=repmat(b2,1,3000);

b3=b{3,1}; %bias of the layer 3

b31=repmat(b3,1,3000);

a1=iw*x1;

oj=tansig(a1+b11); %output of the layer 1

ok=logsig(lw1*tansig(a1+b11)+b21); %output of the layer 2

ol=purelin(lw2*logsig(lw1*tansig(a1+b11)+b21)+b31); %output of
the layer 3



d1=dtansig(x1,oj); %calculate the derivative of a hyperbolic
tangent function for the layer 1
```

```
d2=dlogsig(oj,ok); % calculate the derivative of a logistic
function for the layer 2

d3=dpurelin(ok,ol); % calculate the derivative of the identity
function for the layer 1



[a b]=size(d1); %norm of the layer 1

for i=1:a

md1(i)=norm(d1(i,:));

end



[c d]=size(d2); %norm of the layer 2

for j=1:c

md2(j)=norm(d2(j,:));

end



[e f]=size(d3); %norm of the layer 3

for j=1:c

md3(j)=norm(d3(j,:));
```

```
end


dmd1=diag(md1);

dmd2=diag(md2(1,1:2));

dmd3=diag(md3);


dj=dmd3*lw2*dmd2*lw1*dmd1*iw;


contri=dj/sum(dj); % SNPs effects using Dimopoulus method
```

**Commands used to fit Bayesian Regularized Neural Network (BRNN) (*software* MATLAB)**

```
ftreing='fenotipo.txt'; % Phenotype Data

ftreing=load(ftreing)';


gtreing='tbv.txt'; % true breeding value for the train dataset
```

```
gtreing=load(gtreing)';

Mtreing='Mptrein.txt'; % Markers matrix for the phenotype

Mtreing=load(Mtreing)';

Mvalidg='Mpvalid.txt'; % Markers matrix for the validation data

Mvalidg=load(Mvalidg)';

gvalidg='genvalid.txt'; % validation data

gvalidg=load(gvalidg)';

net = newff(minmax(Mtreing),[1], {'purelin'}, 'trainbr'); % 1
layer with one neuron and an identity activation function
```

net = newff(minmax(Mtreing),[2 1], {'logsig' 'purelin'}, 'trainbr'); % neural network with two layers first logistic and second identity activation function with two and one neuron respectively

net = newff(minmax(Mtreing),[2 1], {'purelin' 'purelin'}, 'trainbr'); % neural network with two layers and two identity activation function with two and one neuron respectively

net = newff(minmax(Mtreing),[2 1], {'tansig' 'purelin'}, 'trainbr'); %

neural network with two layers first tangent hyperbolic and second identity activation function with two and one neuron respectively

net = newff(minmax(Mtreing),[2 2 1], {'logsig' 'tansig' 'purelin'}, 'trainbr'); % neural network with tree layers first logistic, second tangent hyperbolic, and the last identity activation function with two, two and one neuron respectively

```
net = newff(minmax(Mtreing),[2 2 1], {'logsig' 'tansig'
'purelin'}, 'trainbr'); % neural network with tree layers first
tangent hyperbolic, second logistic, and the last identity
activation function with two, two and one neuron respectively


net = init(net); %initiation the BRNN


[net,tr] = train(net, Mtreing, ftreing);
```

**Commands used for the RR-BLUP method (*software R*)**

```
#Allelic frequencies
p=NULL
for(i in 1:ncol(markers)) {

p[i]=(0.5*length(which(markers[,i]==1))+length(which(markers[,i]==2)))/nrow(markers
)
}
q=1-p
```

```
pq=p*q



##############################RR-BLUP##############

library(rrBLUP)

rrb=mixed.solve(fentrein,Z=Mtrein)



gvrrbluph=markers%*%rrb$u

gvrrblup=Mvalid%*%rrb$u

vadd=2*sum(pq*(rrb$u^2))

h2rr1=var(gvrrbluph)/(var(gvrrbluph)+rrb$Ve)

h2rr2=vadd/(vadd+ rrb$Ve)



gen=Mvalid%*%rrb$u # predict using RR-BLUP methods
```

**Commands used for the Bayesian LASSO method (*software R*)**

```
library(BLR)

#especification  MCMC
```

```
nIter=50000;burnIn=20000; thin=3;

fit_bl=BLR(y=fentrein,XL=Mtrein,nIter=nIter,burnIn=burnIn,thin=thin)

gebvLh=markers%*%fit_bl$bL

gebvL=Mvalid%*%fit_bl$bL

vgL=2*sum(pq*(fit_bl$bL^2))

veL=fit_bl$varE

h2blr1=var(gebvLh)/(var(gebvLh)+veL)

h2blr2=vgL/(vgL+veL)
```

**Commands used to calculate the Standard Error of Prediction (*software R*)**

```
for i= 1:len

SEP1(i)=(dados_validfenNor(i)-y_hat(i))^2;

end

SEP=sqrt(sum(SEP1)/len)% SEP Prediction
```