Genet. Res., Camb. (2011), **93**, *pp.* 189–201.

doi:10.1017/S0016672310000662

Prediction of body mass index in mice using dense molecular markers and a regularized neural network

HAYRETTIN OKUT[1,2]*, DANIEL GIANOLA[2,3,4], GUILHERME J. M. ROSA[3,4] AND KENT A. WEIGEL[2]

[1] *Department of Animal Sciences, University of Yuzuncy Yil, Van, 65080, Turkey*
[2] *Department of Dairy Science, University of Wisconsin, Madison, WI 53706, USA*
[3] *Department of Animal Sciences, University of Wisconsin, Madison, WI 53706, USA*
[4] *Department of Biostatistics and Medical Informatics, University of Wisconsin, Madison, WI 53706, USA*

(*Received 1 October 2010; revised 30 November 2010; accepted 6 December 2010; first published online 12 April 2011*)

Summary

Bayesian regularization of artificial neural networks (BRANNs) were used to predict body mass index (BMI) in mice using single nucleotide polymorphism (SNP) markers. Data from 1896 animals with both phenotypic and genotypic (12 320 loci) information were used for the analysis. Missing genotypes were imputed based on estimated allelic frequencies, with no attempt to reconstruct haplotypes based on family information or linkage disequilibrium between markers. A feed-forward multilayer perceptron network consisting of a single output layer and one hidden layer was used. Training of the neural network was done using the Bayesian regularized backpropagation algorithm. When the number of neurons in the hidden layer was increased, the number of effective parameters, γ, increased up to a point and stabilized thereafter. A model with five neurons in the hidden layer produced a value of γ that saturated the data. In terms of predictive ability, a network with five neurons in the hidden layer attained the smallest error and highest correlation in the test data although differences among networks were negligible. Using inherent weight information of BRANN with different number of neurons in the hidden layer, it was observed that 17 SNPs had a larger impact on the network, indicating their possible relevance in prediction of BMI. It is concluded that BRANN 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.

1. Introduction

Many genetic association studies aim at characterizing relationships among numerous single-nucleotide polymorphisms (SNPs) and a continuous or discrete trait (Curtis, 2007). Common diseases such as obesity or diabetes have been considered to be the result of a complex combination of genetic and environmental factors (Mutoh *et al*., 2005). Although genetic factors are known to be involved in the development of these diseases, their genetic determination remains largely unclear and SNPs are natural candidates for predictive models.

* Corresponding author: University of Wisconsin 1675 Observatory Drive, Madison, WI 53703, USA. Tel: +1 608 772 4922. e-mail: okut@wisc.edu

Several statistical and computational methods have been devised for the analysis of SNP data (Useche *et al*., 2001). An example is that of artificial neural networks (ANNs), which provide a powerful technique for learning about complex traits by predicting future outcomes based on training data (Shaneh & Butler, 2006). However, the application of ANNs on the prediction of complex phenotypes using genomic (SNPs) is new. Neural networks (NNs) are computer-based systems composed of many simple processing elements operating in parallel (http://www.scs.unr.edu/nevprop; Tu, 1997; Lampinen & Vehtari, 2001). An ANN is determined by the network structure (e.g. the number of neurons), connection strength and type of processing performed to accommodate elements or nodes. NNs have the ability of capturing

complex non-linear relationships between the response (e.g. phenotype) and input variables (e.g. SNPs), including all possible interactions between the latter, and many training algorithms are available (Tu, 1997; Lampinen & Vehtari, 2001). This makes ANN extremely interesting for the analysis of complex traits.

Backpropagation is a commonly used method of learning in multilayer feed-forward NNs. It is a supervised learning algorithm based on a suitable error function, whose values are determined by the target (phenotype) and mapped outputs of the network (fitted values); the error function is akin to a residual sum of squares and can be minimized via gradient-descent methods. The simplest implementation of backpropagation updates the network-regression coefficients (weights) and intercepts (biases) in the direction in which the performance function decreases most rapidly. For this, it is required that the activation functions of the neurons in the network should be differentiable, and it is customary to use some kind of sigmoid function (Aggarwal *et al*., 2005).

Like other parametric and non-parametric methods, such as kernel regression and smoothing splines, ANNs can produce overfitting (especially with highly dimensional data, such as SNPs) and predictions can be outside the range of the training data (Ping *et al*., 2003; Feng *et al*., 2006; Wang *et al*., 2009). Regularization (shrinkage) is a procedure that allows bias of parameter estimates towards what are considered to be plausible values, while reducing their variance; thus, there is a bias–variance trade-off. Two popular techniques for generalizing or predicting in ANN models are the Bayesian regularization (BR) and the cross-validated early-stopping (CVES) methods (Wang *et al*., 2009). Early stopping is used with a neural network trained via gradient descent methods. The data set, $D=\{\mathbf{p}_i,\ \mathbf{t},\ i=1,\ 2,\ \ldots,\ N\}$, where $\mathbf{p}_i$ is a vector of inputs (e.g. SNPs) for observation i and $\mathbf{t}$ is a vector of target variables (phenotype), is split into training and tuning (validation) sets. After each step in a set of iterations (epoch) through the training set, the network is evaluated on the tuning set. Once performance in the tuning set stops improving, the algorithm halts. Early stopping limits the effect of weights in the network and produces regularization. However, the measure of error in the tuning set may not provide a good estimate of the prediction error that would be achieved in practice. One method of producing an unbiased estimate of prediction error is to run the network on a third set of data, the testing set, which is not used at all during the training process (http://www.faqs.org/faqs/ai-faq/neural-nets/part3/section-5.html). Alternatively, BR gauges an objective function consisting of a residual sum of squares plus the sum of squared weights; the function is minimized with respect to the weights, and the aim is to produce a network that generalizes well (Bishop & Tipping, 1998; Titterington, 2004; Marwala, 2007; Ripley, 2007). In the Bayesian approach, the weights and intercepts of the network are assumed to be random variables following some specified prior distributions.

BR is a non-linear analog of ridge regression. This technique is more robust than standard backpropagation nets and can reduce or eliminate the need for lengthy cross-validation processes (Winkler & Burden, 2004). The Bayesian approach to neural network modelling consists of arriving at the posterior probability distribution of weights by updating a prior probability distribution using a training set in D ($D=\{\mathbf{p}_i,\ \mathbf{t},\ i=1,\ 2,\ \ldots,\ N\}$, where $\mathbf{p}_i$ is a vector of SNPs for observation i and $\mathbf{t}$ is a vector of body mass index (BMI) measurements). In the standard backpropagation algorithm, the cost function $E_D=(1/N)\sum_{i=1}^{N}(e_i)^2=(1/N)\sum_{i=1}^{N}[t_i-f(\mathbf{p}_i;\mathbf{w})]^2$, where N is the size of the training set, is minimized with respect to the vector of weights $\boldsymbol{w}$, which enters non-linearly into the ANN $f(\mathbf{p}_i;\boldsymbol{w})$. With BR, the cost function is modified into $F=\alpha E_w+\beta E_D$, where E_w is the sum of squares of the ANN weights and α and β are regularization parameters that need to be tuned. Hence, BR can be viewed as a penalized non-linear least-squares regression, where minimization with respect to w leads to a conditional (given α and β) posterior mode in a Bayesian model in which $p(t_i|\mathbf{w},\beta)\sim N[f(\mathbf{p}_j,\mathbf{w}),\ \beta^{-1}]$ and $w_{kj}\sim N\text{iid}(0,\ a^{-1})$, where, $j=1,\ 2,\ \ldots,\ R$ is the number of inputs and $k=1,\ 2,\ \ldots,\ S$ is the number of neurons in the hidden layer. Here $N(.,.)$ denotes normal distribution and iid stands for independent and identically distributed. The posterior predictive distribution of a new target for a new input data ($\mathbf{p}$) is obtained by averaging the predictions of the model over the posterior distribution of $\mathbf{w}$ (Kelemen & Liang, 2008).

In Bayesian regularization of artificial neural networks (BRANNs), and particularly when the data sets are small, it is not necessary to split the data into training, testing and tuning sets, and all available information is devoted to model fitting and model comparison (Bishop & Tipping, 1998). This is important when training networks with small data sets, and there is evidence that BR has a better generalization performance than early stopping (http://www.faqs.org/faqs/ai-faq/neural-nets/part3/section-5.html). In contrast to conventional network training, where an optimal set of weights is chosen by minimizing an error function, the Bayesian approach involves a probability distribution of network weights. As a result, the predictions of the network are also realizations from probability distributions. Importantly, complex models are penalized in the Bayesian approach, reducing the problems of overfitting and overtraining (Sorich *et al*., 2003). It is important to note that a fully Bayesian solution requires Markov chain Monte

Carlo sampling. However, such techniques can be computationally expensive, and they also suffer from the difficulty of assessing convergence.

The objectives of this paper were: (1) to fit alternative BRANN using SNPs as input variables and BMI in mice as an output, considering different activation functions and varying number of neurons in the hidden layer, (2) to compare the predictive ability of different BRANN and (3) to detect relevant SNPs associated with BMI. The paper is organized as follows. The section Material and methods describes how the data were obtained and processed; it also gives an account of the architectures of the NNs used, of the tuning of α and β parameters, of the analyses carried out and of the computational strategy used. The section Results presents the performance of the regularized NNs in terms of predictive ability and the relevance of SNPs with respect to their association with BMI. The paper ends with a Discussion section and concluding remarks.

2. Material and methods

(i) *Data sets*

Records from a population of mice have been recently used for studying the predictive ability of genomic-based linear-regression models for quantitative traits using Bayesian methods (Legarra *et al*., 2008; de los Campos *et al*., 2009). The data sets are freely available at (http://gscan.well.ox.ac.uk), and details can be found in (Mott *et al*., 2000; Mott, 2006; Valdar *et al*., 2006*a*, *b*). Genotyping techniques and choice of SNPs are in Valdar *et al*. (2006*a*). Animals with both phenotypic and genotypic data were retained for analysis. Our data sets were composed of 1896 individuals genotyped at 12 320 SNP loci. Because the proportion of missing genotypes was low, to simplify the analysis, missing genotypes were imputed at random based on their allelic frequencies, with no attempt to reconstruct haplotypes based on the family information or the linkage disequilibrium between markers. The pedigree for the 1896 individuals included information on their parents, but not on their grandparents; parents of phenotyped and genotyped animals did not have phenotypic information.

Gender, cage density (number of animals per cage) and age were considered as potential factors affecting BMI (Valdar *et al*., 2006*b*). The following mixed linear model was fitted:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \mathbf{T}\mathbf{c} + \mathbf{e},$$

where $\mathbf{y}$ is the observed vector of BMI phenotypes; $\boldsymbol{\beta}$ is an unknown vector of fixed effects of gender, age and cage density; $\mathbf{u}$ is an unobserved vector of random additive genetic infinitesimal effects; $\mathbf{c}$ is an unobserved vector of random cage effects; $\mathbf{X}$, $\mathbf{Z}$ and $\mathbf{T}$ are the corresponding known incidence matrices, and $\mathbf{e}$ is a vector of random residuals assumed to follow a multivariate normal distribution, $\mathbf{e} \sim N(0, \mathbf{I}\sigma_e^2)$, where σ_e^2 is the residual variance and $\mathbf{I}$ is an 1896×1896 identity matrix. The random additive genetic and cage effects were assumed independent, with distributions $\mathbf{u} \sim N(0, \mathbf{A}\sigma_u^2)$ and $\mathbf{c} \sim N(0, \mathbf{I}_c\sigma_c^2)$, respectively. Here, σ_u^2 is the additive genetic variance, σ_c^2 is the variance among cages, $\mathbf{A}$ is the matrix of additive genetic relationships, and $\mathbf{I}_c$ is a 359×359 identity matrix, where 359 is the number of cages. Allocation of animals to cages was not at random, as most animals in a cage were full sibs. Within the 359 cages there were only 8 cages with offspring from more than one sire, and each full-sib group was allocated to an average of 2·84 cages. Therefore, there was some confounding in the least squares sense between family and cage effects (Legarra *et al*., 2008).

The BMI values were corrected to eliminate nuisance effects as $\mathbf{t} = \hat{\mathbf{y}} - \mathbf{X}\hat{\boldsymbol{\beta}} - \mathrm{T}\hat{\mathbf{c}}$, where $\hat{\boldsymbol{\beta}}$ and $\hat{\mathbf{c}}$ were conditional (given likelihood based estimates of σ_u^2, σ_c^2 and σ_e^2) generalized least squares estimates and best linear unbiased predictions of $\boldsymbol{\beta}$ and $\boldsymbol{c}$, respectively. The $\mathbf{t}$ values were used as target values in the NNs using 12 320 SNPs (aa, Aa and AA, genotypes were coded as 0, 1 and 2, respectively) as potential input variables. A total of 530 of these SNPs were discarded because loci were monomorphic. A prescreening of SNPs was performed using a simple linear regression (one SNP at a time) to obtain *P*-values under the null hypothesis of no marker effect. False discovery rate (FDR) was calculated using PROC MULTTEST in SAS (SAS, 2009); the FDR controls the expected proportion of incorrectly rejected null hypotheses (type I errors) among all rejected hypotheses. Using the FDR approach, 798 SNPs were called significant at $P = 0{\cdot}05$, and were then used as input variables for the NNs.

The 1896 cases $(\mathbf{t}_i, \mathbf{p}'_i)$ where $\mathbf{p}'_i = \{\mathbf{p}_{ij}\}$ is a row vector with genotypes on the 798 filtered SNPs considered, were randomly divided into three subsets: training, tuning and testing. The first subset ($n = 1138$; 60 % of the cases) was used for training the network and for updating the network weights and biases (intercepts) iteratively, in conjunction with the tuning set. The second or tuning subset ($n = 379$; 20 % of the cases) was used to monitor during the training process. Training and tuning errors normally decrease during the initial phases of training. When the network begins to overfit the training data, the error on the tuning set typically begins to increase. When the tuning error increased for a specified number of iterations, training was stopped, and the weights and biases that minimized the tuning error were returned. The third subset ($n = 379$; 20 % of the cases), was the testing set, which was not used at all during training, and was

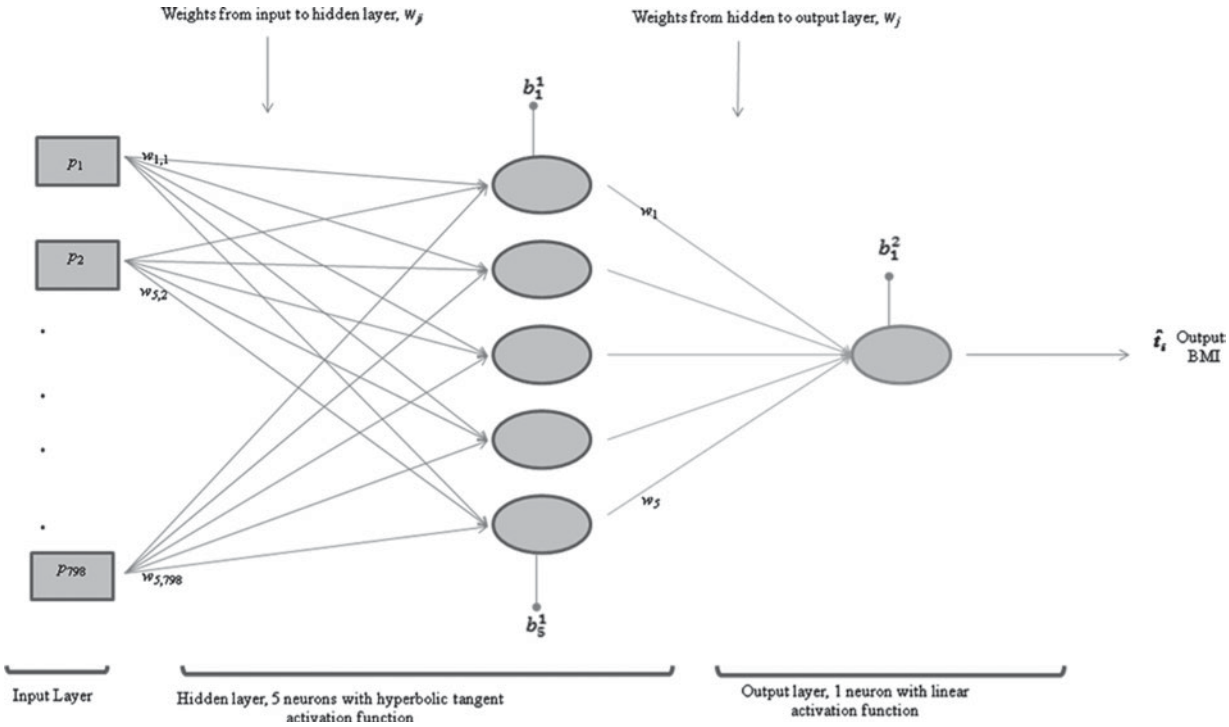

Fig. 1. ANN design used in this study. There were 798 SNP genotypes used as inputs (p_{ij}). Each SNP is connected to up to five neurons via coefficients w_{jk} (j denotes neuron, k denotes SNP). Each hidden and output neuron has a bias parameter $b_j^{(l)}$, j denotes neuron, l denotes layer).

used to evaluate the predictive ability of the different models (Chen *et al*., 2008).

Overfitting results from an excessive number of parameters and can occur when the number of neurons in the hidden layer increases. While a neural network can learn patterns in the training set perfectly, it may not be able to make reasonable predictions when presented with independent cases (Alados *et al*., 2004). A brief description of ANN is presented next.

(ii) *Feed-forward neural networks*

In the feed-forward NN used herein, the input vector of SNP genotypes $\mathbf{p}_i$ was related to the target t_i (adjusted BMI) using the architecture depicted in Fig. 1. The network architecture implemented in this paper was such that, $\mathbf{p}_i' = (\mathbf{p}_{i1}, \mathbf{p}_{i2}, \ldots, \mathbf{p}_{i798})$ contained genotype codes for 798 SNPs in mouse i. The SNPs are connected to each of S neurons in a single hidden layer via weights (w_{kj}, $k = 1, 2, \ldots, S$), which are specific to each SNP(j)–neuron(k) connection, and there is a bias (intercept) specific to each neuron. For example, if there are S neurons in the architecture, the biases are $b_1^{(1)}, b_2^{(1)}, \ldots, b_S^{(1)}$. The input into neuron k, prior to activation is $b_k^{(1)} + \sum_{j=1}^{798} w_{kj} p_j$. Subsequently, this input is transformed (*activated*) using some linear or non-linear activation function $f(.)$ (Fig. 1) as $f_k(b_k^{(1)} + \sum_{j=1}^{798} w_{kj} p_j)$, $k = 1, 2, \ldots, S$. This activated emission is then sent to the output layer and collected as $\sum_{k=1}^{S} w_k f_k(b_k^{(1)} + \sum_{j=1}^{798} w_{kj} p_j) + b^{(2)}$, where w_k ($k = 1, 2, \ldots, S$) are weights specific to each neuron and $b^{(1)}$ and $b^{(2)}$ are bias parameters in the hidden and output layers, respectively. Finally, this quantity is activated again with function $g(.)$ as $g[\sum_{k=1}^{S} w_k f_k(.) + b^{(2)}]$, which then becomes the predicted BMI value of t_i in the training set, or $\hat{t}_i$. Typically, g(.) is a linear activation function when $\mathbf{t}_i$ is a continuous outcome.

With 798 SNPs and five neurons, there are close to 4000 weights and biases to estimate from only 1138 cases in the training set. Training is the process by which the weights are modified in the light of the data while the network attempts to produce the desired output. Before training, weights do not have any meaning (Forshed *et al*., 2002; Alados *et al*., 2004) beyond that conveyed by the prior distribution in a Bayesian setting. After training, the fitted value for corrected BMI is calculated as:

$$\hat{t}_i = g\left\{ \sum_{k=1}^{s} w_k f\left(\sum_{j=1}^{R} w_{kj} p_j + b_k^{(1)} \right) + b^{(2)} \right\}; \quad (1)$$
$$j = 1, 2, \ldots, R, \quad k = 1, 2, \ldots, S.$$

Linear or tangent sigmoid activation functions were used in this study for all neurons.

(iii) *BR*

The Bayesian framework for NNs involves a probability distribution of network weights, so that

predictions from the network can also be cast in a probabilistic framework (Sorich *et al.*, 2003). Regularization is a technique that can improve predictive ability (generalization) of ANNs. Once a set of values weights, **w**, is assigned to the connections in the networks, this defines a mapping from the input vector p to the output $\hat{t}$. If M denotes a specific network architecture, the typical performance function used for training a neural network is the sum of squared prediction errors

$$E_D(D|\mathbf{w}, M) = \sum_{i=1}^{N} (\hat{t}_i - t_i)^2 \tag{2}$$

for N input–target pairs resulting in data D. The network architecture consists of a specification of the number of layers, the number of neurons in each layer, and the type of activation functions used. Early stopping as a default in MATLAB (Demuth *et al.*, 2009) uses as default values of **w** chosen such that E_D is minimized, but this procedure fails for large models, especially when the number of coefficients exceeds N. Different algorithms are used; the Levenberg–Marquardt algorithm is the fastest and can be used in moderate-sized ANN for obtaining parameter estimates (MacKay, 1996; Gencay & Qi, 2001).

In BRANN, on the other hand, an additional term that penalizes large weights in the hope of achieving a smoother mapping is added to the objective function. Gradient-based optimization is then used to minimize the function:

$$F = \beta E_D(\mathrm{D}|\mathbf{w}, \mathrm{M}) + \alpha E_W(\mathbf{w}|\mathrm{M}), \tag{3}$$

where E_W (**w**|M), is the sum of squares of network weights. Here, n is the number of weights in the ANN, and α and β are regularization parameters that need to be estimated. If $\alpha >> \beta$, emphasis is on reducing the magnitude of weights at the expense of goodness of fit, while producing a smoother network response (Foresee & Hagan, 1997). Training involves a trade-off between model complexity and goodness of fit. If Bayes estimates of α are large, the posterior densities of the weights are highly concentrated around zero, so that the weights effectively disappear and the model discounts connections in the network (Titterington, 2004; MacKay, 2008). Therefore, complex models are automatically self-penalized. The second term in eqn (3), known as weight decay, favors small values of **w** and decreases the tendency of a model to overfit (MacKay, 2008).

The empirical Bayes approach (MacKay, 2008) is as follows. The posterior distribution of **w** given α, β, D and M is

$$P(\mathbf{w}|D, \alpha, \beta, M) = \frac{P(D|\mathbf{w}, \beta, M)P(\mathbf{w}|\alpha, M)}{P(D|\alpha, \beta, M)}, \tag{4}$$

where D is the training data set. In eqn (4), $P(\boldsymbol{w}|\alpha, M)$ is the prior distribution of weights under M, $P(D|\boldsymbol{w}, \beta, M)$ is the likelihood function, which is the probability of observing the data given **w**, and $P(D|\ \alpha, \beta, M)$ is a normalization factor, which does not depend on **w** (Nguyen & Widrow, 1990; Thodberg, 1996; Kumar *et al.*, 2004)

$$P(D|\alpha, \beta, M) = \int P(D|\mathbf{w}, \beta, M)P(\mathbf{w}|\alpha, M)\mathrm{d}\mathbf{w}.$$

The weights **w** were assumed to be identically distributed, *a priori*, each following the normal distribution $(\mathbf{w}|\alpha,\ M) \sim N(0,\ \alpha^{-1})$, so that the joint prior density of **w** is

$$p(\mathbf{w}|a, M) \propto \prod_{l=1}^{m} \exp\left(-\frac{\alpha w_{kj}^2}{2}\right) = \exp\left[-\frac{\alpha E_W(\mathbf{w}|M)}{2}\right].$$

After normalization, the prior distribution is then

$$p(\mathbf{w}|\alpha, M) = \frac{\exp[-(\alpha E_W(\mathbf{w}|M)/2)]}{\int \exp[-(\alpha E_W(\mathbf{w}|M)/2)]\mathrm{d}\mathbf{w}} = \frac{\exp[-(\alpha E_W(\mathbf{w}|M)/2)]}{Z_w(\alpha)}, \tag{5}$$

where

$$Z_W(\alpha) = \left(\frac{2\pi}{\alpha}\right)^{n/2}.$$

The target variable, t, expressed as a function of inputs, **p**, is modeled as $t_i = f(\mathbf{p_i}) + e$, where $e \sim \mathrm{N}(0, \beta^{-1})$ and $f(\mathbf{p_i})$ is the neural network approximation to $E(t|\mathbf{p})$. Under Gaussian assumptions, the joint density of the target variables, given the input variables, β and M is:

$$P(\mathbf{t}|\mathbf{p}, \mathbf{w}, \beta, M) = \left(\frac{\beta}{2\pi}\right)^{N/2} \exp\left[-\frac{\beta}{2}\sum_{i=1}^{N}(t_i - f(\mathbf{p_i}))^2\right] = \left(\frac{\beta}{2\pi}\right)^{N/2} \exp\left[-\frac{\beta}{2}E_D(D|\mathbf{w}, M)\right], \tag{6}$$

where $E_D(D|\mathbf{w},M)$ is as given in eqn (2). Letting

$$Z_D(\beta) = \int \exp\left[-\frac{\beta}{2}E_D(D|\mathbf{w}, M)\right] = \left(\frac{2\pi}{\beta}\right)^{N/2},$$

the posterior density of **w** in eqn (4) can be expressed as

$$P(\mathbf{w}|D, \alpha, \beta, M) = \frac{[1/Z_W(\alpha)Z_D(\beta)]\exp\left[-\frac{1}{2}(\beta E_D + \alpha E_W)\right]}{P(D|\alpha\beta M)} = \frac{1}{Z_F(\alpha, \beta)}\exp\left[-\frac{F(w)}{2}\right], \tag{7}$$

where $Z_F(\alpha,\beta) = [Z_w(\alpha)Z_D(\beta)P(D|\alpha,\beta,M]$ and $F = \beta E_D + \alpha E_W$. In an empirical Bayesian framework, the 'optimal' weights are those that maximize the posterior density $P(\mathbf{w}|D, \alpha, \beta, M)$, which is equivalent to minimizing the regularized objective function F given in equation (3); this implies that some values of

α and β must be estimated. While minimization of F is identical to finding the (locally) *maximum a posteriori* estimates $\mathbf{w}^{MP}$, minimization of E_D by back-propagation is identical to finding the maximum likelihood estimates $\mathbf{w}^{ML}$ (MacKay, 2008).

(iv) *Optimizing regularization parameters α and β*

Regard now α and β as unknown, and consider the joint posterior density

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

If the prior density $P(\alpha, \beta|M)$ is uniform, maximization of $P(\alpha\,\beta \mid D, M)$ with respect to α is equivalent to maximization of $P(D \mid \alpha, \beta, M)$. From equations (4), (5), (6) and (7)

$$\begin{aligned}P(D|\alpha,\beta,M)&=\frac{P(D|\mathbf{w},\beta,M).P(\mathbf{w}|\alpha,M)}{P(\mathbf{w}|D,\alpha,\beta,M)}\\&=\frac{[(1/Z_D(\beta))\exp(-\beta E_D/2)][(1/Z_W(\alpha))\exp(-\alpha E_W/2)]}{(1/Z_F(\alpha,\beta))\exp(-F(w)/2)}\\&=\frac{Z_F(\alpha,\beta)}{Z_D(\beta)Z_W(a)}\\&=\frac{Z_F(\alpha,\beta)}{(2\pi/\beta)^{N/2}(2\pi/\alpha)^{m/2}}\propto\beta^{N/2}\alpha^{m/2}Z_F(\alpha,\beta). \quad (9)\end{aligned}$$

A Laplacian approximation to $Z_F(\alpha, \beta)$ yields (Kumar *et al*., 2004):

$$Z_F(\alpha,\beta)\propto\left|\mathbf{H}^{MP}\right|^{-1/2}\exp\left(-\frac{F(w^{MP})}{2}\right), \quad (10)$$

where $\mathbf{H}^{MP}$ is the Hessian matrix of the objective function evaluated at $\mathbf{w}^{MP}$, which in turn depends on current values of α and β and m the number of weights. Optimal values of α and β can be solved by optimizing (9) while using (10) as an auxiliary function. The expression $\gamma=n-2\alpha^{MP}tr(\mathbf{H}^{MP})^{-1}$ is called the number of effective parameters in the neural network, where n is the total number of parameters; $0\leqslant\gamma\leqslant n$. It can be shown (MacKay, 1992; Xu *et al*., 2006) that

$$\alpha^{MP}=\frac{\gamma}{2E_w(\mathbf{w}^{MP})}\quad\text{and}\quad\beta^{MP}=\frac{n-\gamma}{2E_D(\mathbf{w}^{MP})}. \quad (11)$$

Bayesian optimization of the regularization parameters requires computation of the Hessian matrix of the objective function F at the optimum point $\mathbf{w}^{MP}$ (Xu *et al*., 2006). As proposed by (MacKay, 1992), the Gauss–Newton approximation to the Hessian matrix, is readily available if the Levenberg–Marquardt optimization algorithm is used to locate the minimum of F (Tu, 1997; Lampinen & Vehtari, 2001; Shaneh & Butler, 2006; www.scs.unr.edu/nevprop).

The flow chart in Fig. 2 summarizes the training steps of a BRANN. This gives all steps required for Bayesian optimization of the regularization parameters.

(v) *Analyses and computing environment*

MATLAB (Demuth *et al*., 2009) was used for fitting the BRANN. The NNs considered had two layers (hidden and output layers) and were fully connected feed-forward networks, as shown in Fig. 1. Determination of an appropriate number of neurons (hidden nodes) is a critical task in neural network design. A network with too few neurons may be incapable of capturing complex patterns. In contrast, if the network has too many neurons it will follow the noise in the data due to overparameterization, leading to poor predictive ability of yet to be observed data. In the present study the number of neurons in a single hidden layer was varied from one to seven. Therefore, there were 798 inputs (SNPs), one to seven neurons and one node in the output layer.

To avoid overtraining and to improve predictive ability, as well as to eliminate spurious effects caused by the starting values, 20 independent BRANN were trained for each architecture. Results were recorded as the average of these 20 runs, for each architecture. Two combinations of activation functions were used: (1) hyperbolic tangent sigmoidal activation functions from the input layer to the hidden layer plus a linear activation function from the hidden layer to the output layer and (2) linear activation functions both from the input layer to the hidden layer and from the hidden layer to the output layer.

MATLAB (Demuth *et al*., 2009) used the Levenberg–Marquardt algorithm. BR took place within this algorithm with backpropagation to minimize F. Each iteration (epoch) in backpropagation has two sweeps: a forward activation to produce a solution, and a backward propagation of the computed error to modify the weights. The sweeps are performed repeatedly until a pre-specified tolerance is met (Hajmeer *et al*., 2006; Haykin, 2008). The number of epochs used was 1000. Training was stopped if: (1) the maximum number of epochs was reached; (2) performance had met a suitable level; (3) the gradient was below a suitable target; or (4) the Levenberg–Marquardt μ parameter exceeded a suitable maximum (training stopped when it became larger than 10^{10}). Each of these targets and goals were set at the default values set by the MATLAB implementation.

3. Results

(i) *Performance of BRANN*

Table 1 summarizes estimated features of the network architectures used in the current study. The highest and lowest effective number of parameters were obtained for the one-neuron linear (linear-activation function in hidden and output layers) and one-neuron non-linear (tangent sigmoid activation function in

Table 1. *Parameter estimates and their standard deviations for different network architectures* (*results are averages of 20 independent runs*)

S	F	SSE	SSE_{test}	r_{train}	r_{test}	$r_{t-\hat{t}}$	n	γ
Linear	1·751 ± 0·1	3·08 ± 0·06	0·66 ± 0·04	0·44 ± 0·02	0·15 ± 0·05	0·30 ± 0·02	801	101·1 ± 30·3
1	1·883 ± 0·11	3·17 ± 0·22	0·64 ± 0·04	0·46 ± 0·02	0·14 ± 0·04	0·25 ± 0·04	801	117·3 ± 12·5
2	1·85 ± 0·12	3·17 ± 0·09	0·65 ± 0·05	0·44 ± 0·02	0·14 ± 0·05	0·25 ± 0·06	1601	112·3 ± 16·0
3	1·83 ± 0·11	3·14 ± 0·11	0·67 ± 0·04	0·45 ± 0·03	0·16 ± 0·05	0·27 ± 0·05	2401	114·0 ± 14·0
4	1·79 ± 0·11	3·11 ± 0·09	0·65 ± 0·04	0·44 ± 0·03	0·15 ± 0·05	0·28 ± 0·05	3201	109·9 ± 15·7
5	1·80 ± 0·09	3·11 ± 0·06	0·64 ± 0·05	0·43 ± 0·02	0·18 ± 0·04	0·27 ± 0·03	4001	106·8 ± 11·7
6	1·78 ± 0·12	3·15 ± 0·16	0·69 ± 0·05	0·43 ± 0·03	0·14 ± 0·04	0·28 ± 0·05	4801	105·6 ± 14·1
7	1·795 ± 0·06	3·13 ± 0·03	0·65 ± 0·05	0·45 ± 0·03	0·15 ± 0·04	0·27 ± 0·01	5601	112·7 ± 15·6

S, number of neurons; F, objective function; SSE, sum of squares error in the training set; SSE_{test}, sum of squares error in the testing set; $r_{train(test)}$, correlation between predictions and observations in the training(testing) set; $r_{t-\hat{t}}$, overall correlation between observed and predicted data; n, number of parameters; γ, effective number of parameters.

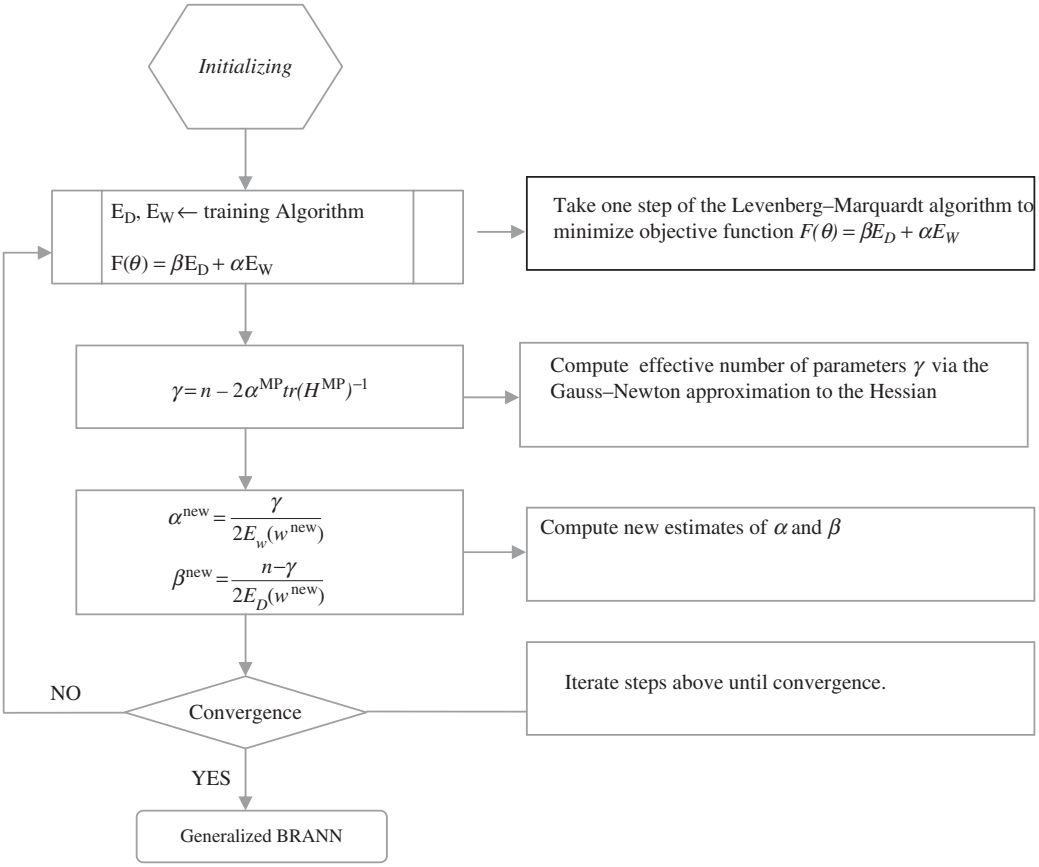

Fig. 2. Flow chart for Bayesian optimization of regularization parameters α and β in NNs; MP, maximum *a posterio* (adapted from Shaneh & Butler, 2006).

hidden layer and linear activation in output layer) networks, with $\gamma = 101{\cdot}1 \pm 30{\cdot}3$ and $\gamma = 117{\cdot}3 \pm 12{\cdot}5$, respectively. This linear network is akin to Bayesian ridge regression, but with α and β tuned similar to the other NNs. When the number of neurons in the hidden layer was increased for non-linear networks from one to seven, γ varied between 105·6 and 117·3, but without clear differences between networks (Fig. 3*c*). Even though the nominal number of parameters (n) increased from 801 to 5601, γ changed slightly only, indicating the impact of regularization.

As shown in Table 1 and Fig. 3*a*, values of the objective function (F) and mean squared error of prediction in the testing set were very similar across networks, with the distribution of values overruns showing considerable overlap. Networks with one and five neurons with non-linear activation functions had the smallest mean squared of error prediction,

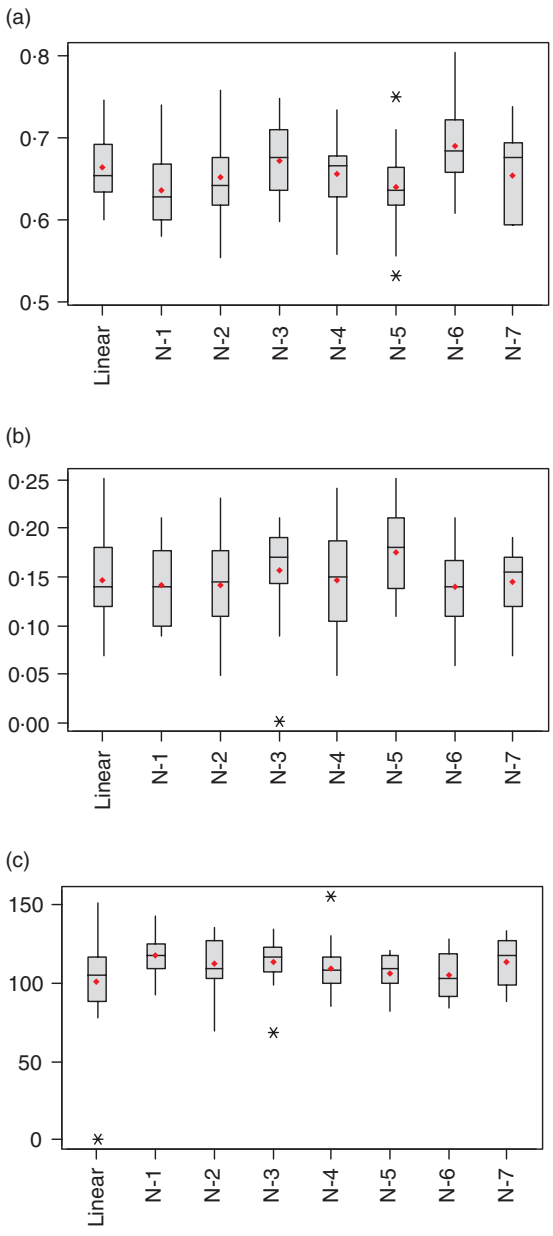

Fig. 3. Box-plots for (*a*) mean-squared error in the testing set, (*b*) correlation between predictions and observations in the testing set and (*c*) effective number of parameters, γ, after 20 independent runs (*indicates extreme values).

but there was no evidence that more complex networks provided better predictions than those attained by the linear model.

The generalization ability of the networks was also assessed by means of the correlation between predicted and observed values in the training, testing and tuning sets. As shown in Figs. 3*b* and 4 the correlations were much larger in the training than in the testing and tuning sets, as expected. Predictive ability was low, and some networks did not improve over the predictive ability attained by a linear model.

The distribution of weights in a network can also provide an indication of predictive ability; small values lead to better generalization and large weights tend to produce a more local representation. The average sum of squares of weights ranged between 0·156 and 0·196 and was smallest for the architecture with 5 neurons; this was consistent with its slightly better predictive ability. Figure 5 depicts the distributions of weights for the linear and 5-neuron architectures for the run with the largest correlation ($r_{\text{test}} = 0{\cdot}25$ for the two networks). The weights for the linear model were larger and more variable than for the 5-neuron model.

(ii) *Ranking SNPs for BMI*

When the output layer involves a single neuron, the influence of input variables on the output is directly reflected in the weights assigned to each input, SNPs in this case. With many neurons, the influence of each SNP is more difficult to evaluate. Two different approaches for assessing the relative importance of individual SNPs in predicting BMI were considered. The relative importance of a given SNP on BMI (Joseph *et al.*, 2003) was assessed as:

$$I_{\text{SNP}j} = \frac{\sum_{j=1}^{S} |w_{kj}^{(1,1)}|}{\sum_{j=1}^{S}\sum_{k=1}^{R} |w_{kj}^{(1,1)}|} 100,$$

where $w_{kj}^{(1,1)}$ is the connection weight from SNP j to neuron k, $|.|$ is the absolute value function, and R and S are the number of SNPs and of hidden neurons, respectively. For indexing and ranking SNPs, data were not split into training, tuning and testing sets; rather, the whole data set was used to estimate the impact of SNPs on BMI with 1000 epochs of the algorithms run for each network. Results are shown in Table 2 and in Fig. 6 for the seven networks. Results were similar across the networks. For example, SNPs 3978, 12 132, 1096 and 2770 had the greatest impact on the networks, indicating that these SNPs may be more relevant for prediction of BMI than other SNPs. In Table 2, it is shown that SNP 3978 contributed by far the most (close to 1%) to the sum (over neurons and SNPs) of absolute values of network weights. These SNPs could be could be indicative of genomic regions associated with BMI.

Another measure used was the contribution of a given neuron, say k, to the entire network (Guha *et al.*, 2005). The bias term was taken into account, because when assessing the contribution of a given hidden neuron, all effects are relevant (Guha *et al.*, 2005). The contribution of the kth neuron to the entire network was calculated as:

$$CN_k = \frac{1}{R+1}\left(\sum_{j=1}^{R} w_{kj}^{(1,1)} w_k^{(2,1)} + b_k^{(1)} w_k^{(2,1)}\right).$$

Here, $b_k^{(1)}$ are biases in the hidden layer, $w_{kj}^{(1,1)}$ are weights from the input layer to the hidden layer, and

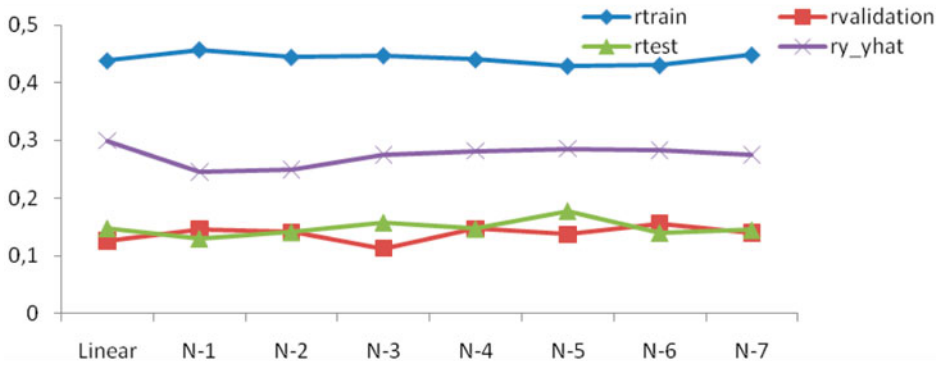

Fig. 4. Correlations between predictions and observations for training (r_{train}), testing (r_{test}), tuning ($r_{\text{validation}}$) and overall ($r_{y-\hat{y}}$) for linear and seven (N-1–N-7) network architectures.

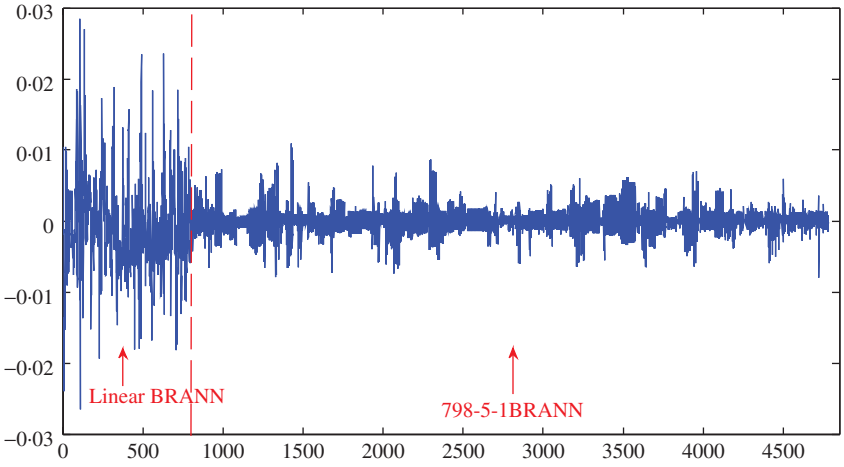

Fig. 5. Distribution of weights (w_{kj}) for the linear model and for neural network with five neurons.

the $w_k^{(2,1)}$ are weights from the hidden layer to the output (BMI). The relative contribution of neuron k to the network was obtained as:

$$\text{SCN}_k = \frac{\text{CN}_k^2}{\sum_{k=1}^{S} \text{CN}_k^2}.$$

As shown in Table 3 the SCN_k values indicate that neurons contributed about equally in networks with 2, 4, 5, 6 and 7 neurons, whereas in the 3-neuron network, the first and third neurons were more influential than the second one. It is difficult to assign an interpretation to these results.

4. Discussion

This study explored the association between 12 320 SNPs and BMI by exploiting properties of BRANN. Data were fitted using a linear activation function in both the hidden and output layers of the networks to obtain a benchmark equivalent to Bayesian ridge regression on markers. Subsequently, different architectures were explored, including a varying number of neurons, a tangent sigmoid activation function in the hidden layer and a linear activation function in the output layer (Fig. 1). Using a sigmoidal-type function in the hidden layer and a linear transfer function in the output layer may be advantageous when it is necessary to extrapolate beyond the range of the training data (Maier & Dandy, 2000).

The Levenberg–Marquardt algorithm, as implemented in MATLAB, was adopted to optimize weights and biases, because previous evaluations with networks containing a smaller number of weights indicated that it was a suitable method (Demuth *et al.*, 2009). In the training process, overfitting often occurred, leading to a loss of generalization of the predictive model. Hence, BR was adopted to avoid over-fitting and improve generalization. Bayesian methods can simultaneously optimize regularization parameters in ANNs, a process that is very laborious using cross-validation (Fernandez & Caballero, 2006).

For the networks trained with BR, we examined how the effective number of parameters γ varied with architecture. As shown in Table 1, although the total number of parameters ranged from 801 to 5601, the

Table 2. *Relative importance of SNPs with I_{SNPj} values larger than 0·45% for the each of the non-linear networks*

7 neurons		6 neurons		5 neurons		4 neurons		3 neurons		2 neurons		1 neuron	
SNP ID	I_{SNP} (%)	SNP ID	I_{SNP} (%)	SNP ID	I_{SNP} (%)	SNP ID	I_{SNP} (%)	SNP ID	I_{SNP} (%)	SNP ID	I_{SNP} (%)	SNP ID	I_{SNP} (%)
420	0·45	7985	0·46	420	0·45	1513	0·45	5010	0·46	4319	0·46	1513	0·45
7985	0·47	5012	0·48	7985	0·46	7985	0·45	4319	0·46	8590	0·48	7985	0·45
5012	0·47	8590	0·48	8590	0·48	348	0·48	10 136	0·46	348	0·49	348	0·48
8590	0·48	4319	0·48	5012	0·48	8590	0·48	348	0·47	5012	0·50	8590	0·49
384	0·48	384	0·48	4319	0·48	3891	0·53	10 141	0·47	384	0·50	3891	0·53
4319	0·48	5010	0·49	384	0·48	5012	0·53	472	0·48	5010	0·51	5012	0·53
5010	0·49	3891	0·49	5010	0·49	2487	0·53	3891	0·49	3891	0·51	2487	0·53
3891	0·49	472	0·50	3891	0·49	384	0·54	2487	0·51	472	0·53	384	0·54
472	0·50	10 136	0·52	472	0·50	10 136	0·54	2770	0·54	10 136	0·53	10 136	0·54
10 136	0·52	10 141	0·52	10 136	0·52	5010	0·54	10 961	0·55	2487	0·53	5010	0·54
10 141	0·52	348	0·52	348	0·52	472	0·54	12 132	0·59	10 141	0·53	472	0·54
348	0·53	2487	0·55	10 141	0·53	10 141	0·55	3978	0·92	10 961	0·58	10 141	0·55
2487	0·55	2770	0·58	2487	0·55	10 961	0·58			2770	0·60	10 961	0·58
2770	0·58	10 961	0·59	2770	0·58	2770	0·63			12 132	0·64	2770	0·63
10 961	0·59	12 132	0·64	10 961	0·59	12 132	0·64			3978	0·94	12 132	0·64
12 132	0·64	3978	0·93	12 132	0·64	3978	0·96					3978	0·96
3978	0·93			3978	0·94								

Table 3. *Relative contribution of the* k*th neuron for several neural network architectures for BMI in mice*

	2 neurons	3 neurons	4 neurons	5 neurons	6 neurons	7 neurons
First neuron	0·50	0·40	0·28	0·22	0·16	0·14
Second neuron	0·50	0·25	0·22	0·23	0·23	0·16
Third neuron		0·35	0·23	0·14	0·10	0·15
Fourth neuron			0·28	0·21	0·16	0·11
Fifth neuron				0·20	0·15	0·15
Sixth neuron					0·19	0·14
Seventh neuron						0·15

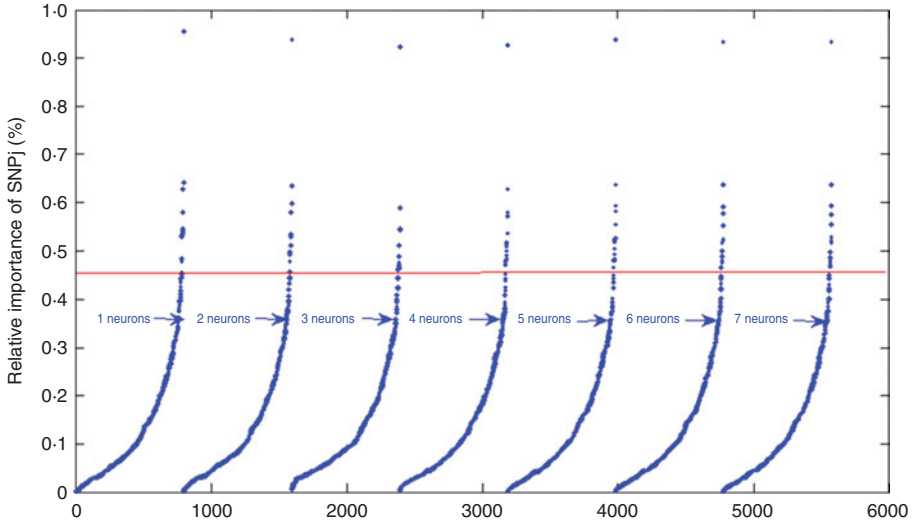

Fig. 6. Plots for the index values of 798 SNPs as prediction of BMI. The solid line gives the cutoff point separating SNPs with index values larger than 0·45%.

effective number of parameters varied only from 101 to 117, on average, illustrating the extent of regularization attained. There were minor differences in predictive ability, and the network with five neurons in the hidden layer had slightly better performance, but not significantly. Although differences were minor, this agrees with several studies (Gencay & Qi, 2001; Joseph *et al*., 2003; Kumar *et al*., 2004; Xu *et al*., 2006), suggesting that a model with a single neuron may not provide a proper representation of the true unknown function to be predicted. Since complex models are penalized in a Bayesian approach, we were able to explore complex architectures (Sorich *et al*., 2003). While more free parameters in a model can lead to smaller data error E_D (MacKay, 1992), monitoring training error to choose among networks is not representative of prediction error.

A useful measure of generalization is the correlation coefficient between predictions and realizations in the test data set. Here, the network with five neurons had the highest correlation in the test data set ($r_{\text{test}} = 0{\cdot}18$), but the lowest in the training data set ($r_{\text{train}} = 0{\cdot}43$). NNs had a performance that was similar to that attained with other procedures reported in literature. Data used in our study have already been analysed in two independent studies, one comparing genome-assisted genetic evaluation methods using BR models (Legarra *et al*., 2008), and another one that compared the Bayesian LASSO with other marker-based regression models (de los Campos *et al*., 2009). The across-family predictive ability of markers in Legarra *et al*. (2008) was 0·17 which is close to $r_{\text{test}} = 0{\cdot}18$ obtained here with the five-neuron architecture. Further, de los Campos *et al*. (2009) reported a rank correlation of 0·30 between phenotypic values and genomic predictions in cross-validation. The overall correlation ($r_{t-\hat{t}}$) estimated between observed and predicted data in our study varied between 0·25 and 0·30 for different network architectures (Table 1), in agreement with de los Campos *et al*. (2009). It is worth noting; however, that the models in Legarra *et al*. (2008) and de los Campos *et al*. (2009) used all 10 946 SNP markers available in their predictions, whereas here we used only 798 pre-selected SNP markers. Moreover, the 798 SNPs used to implement the BRANN were selected in a simplistic way using single marker analyses coupled with an FDR approach. Potentially more efficient approaches for preselection of SNPs (e.g. Long *et al*., 2007; Vazquez *et al*., 2010) could be used and tested to improve the final prediction with BRANN.

Our testing set consisted of 20 % ($n_{\text{test}} = 379$) of the available data, and this was deemed to be large enough for assessing generalization. We observed that the variability in correlations for testing data was greater than for training data. These results are in agreement with several other studies (Aggarwal *et al*., 2005; Marwala, 2007; Kelemen & Liang, 2008; Wang *et al*., 2009). At any rate, the testing set must be a representative sample of the cases to which one wants to generalize.

ANNs are often considered to be more accurate predictors than other classes of models. However, they do not provide clear information regarding how input values correlate with output values. Therefore, it is challenging to obtain effective information from a neural network. Here, exploration of an extensive pool of SNPs allowed detection of relevant SNPs in connection with BMI, as pointed out in other contexts (Fernandez & Caballero, 2006; Xu *et al*., 2006). In our study, almost the same SNPs were found to have an impact on BMI by different networks. This is suggestive of stability of the neural network approach. The importance of neurons in the hidden layer was evaluated in this study as well. An interpretation is that hidden layer neurons are analogous to latent variables in a partial least squares model. The contribution of each hidden layer neuron to the output value of the network indicates which such neurons are most relevant and which can be neglected (Guha *et al*., 2005).

5. Conclusions

The ability of predicting BMI in this mouse data set was low irrespective of the architecture of the NNs considered. This result is consistent with other studies of the same data set, but employing different methods. Among the networks examined, there was a slight superiority of a network with five neurons in the hidden layer. BR allowed estimating all weights, and the effective number of parameters was much lower than the nominal number. Further, several networks were consistent in flagging SNPs that were associated with BMI. It is concluded that BRANN may be at least as useful as other methods for high-dimensional genome enabled prediction, where the number of inputs (e.g. SNPs) is typically much larger than the number of cases in the sample. Finally, NNs have the potential ability of capturing non-linear relationships, which may also be useful in the study of quantitative traits under complex gene action. The data set analysed here did not allow us to corroborate this possibility.

We extend our thanks to The Wellcome Trust Centre for Human Genetics, Oxford, for making the heterogeneous stock data available at http://gscan.well.ox.ac.uk.

References

Aggarwal, K. K., Singh, Y., Chandra, P. & Puri, M. (2005). Bayesian regularization in a neural network model to estimate lines of code using function points. *Journal of Computer Sciences* **1**, 505–509.

Alados, I., Mellado, J. A., Ramos, F. & Alados-Arboledas, L. (2004). Estimating UV erythema1 irradiance by means of neural networks. *Photochemistry and Photobiology* **80**, 351–358.

Bishop, C. M. & Tipping, M. E. (1998). A hierarchical latent variable model for data visualization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20**(3), 281–293.

Chen, L. J., Cui, L. Y., Xing, L. & Han, L. J. (2008). Prediction of the nutrient content in dairy manure using artificial neural network modeling. *Journal of Dairy Science* **91**, 4822–4829.

Curtis, D. (2007). Comparison of artificial neural network analysis with other multimarker methods for detecting genetic association. *BMC Genetics* **8**, 49.

de los Campos, G., Hugo, N., Gianola, G., Crossa, J., Legarra, A., Manfredi, E., Weigel, K. & Miguel, C. J. (2009). Predicting quantitative traits with regression models for dense molecular markers and pedigree. *Genetics* **182**, 375–385.

Demuth, H., Beale, M. & Hagan, M. (2009). *Neural Network Toolbox™ 6 User's Guide*. The MathWorks Inc. Natick, MA, USA.

Feng, N., Wang, F. & Qiu, Y. (2006). Novel approach for promoting the generalization ability of neural networks. *International Journal of Signal Processing* **2**, 131–135.

Fernandez, M. & Caballero, J. (2006). Ensembles of Bayesian-regularized genetic neural networks for modeling of acetylcholinesterase inhibition by huprines. *Chemistry and Biology Drug Design* **68**, 201–212.

Foresee, F. D. &. Hagan, M. T. (1997). Gauss-Newton approximation to Bayesian learning. In *Proceedings of IEEE International Conference on Neural Networks 1997* (ed. M. T. Hagan), pp. 1930–1935.

Forshed, J., Anderson, O. F. & Jacobsson, P. S. (2002). NMR and Bayesian regularized neural network regression for impurity determination of 4-aminophenol. *Journal of Pharmaceutical and Biomedical Analysis* **29**, 495–505.

Gencay, R. & Qi, M. (2001). Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging. *IEEE Transactions on Neural Networks* **12**, 726–734.

Guha, R., Stanton, T. D. & Jurs, C. P. (2005). Interpreting computational neural network quantitative structure-activity relationship models: a detailed interpretation of the weights and biases. *Journal of Chemical Information and Modelling* **45**, 109–1121.

Hajmeer, M., Basheer, I. & Cliver, D. O. (2006). Survival curves of Listeria monocytogenes in chorizos modeled with artificial neural networks. *Food Microbiology* **23**, 561–70.

Haykin, S. (2008). *Neural Networks: Comprehensive Foundation*. 2nd edn. Upper Saddle River, NJ: Prentice-Hall. *A Comprehensive Foundation 3rd edit*: Prentice-Hall.

Joseph, H., Huang, W. L. & Dickman, M. (2003). Neural network modelling of coastal algal blooms. *Ecology Modelling* **159**, 179–201.

Kelemen, A. & Liang, Y. (2008). Statistical advances and challenges for analyzing correlated high dimensional SNP data in genomic study for complex. *Diseases Statistics Surveys* **2**, 43–60.

Kumar, P., Merchant, S. N. & Desai, U. B. (2004). Improving performance in pulse radar detection using Bayesian regularization for neural network training. *Digital Signal Processing* **14**, 438–448.

Lampinen, J. & Vehtari, A. (2001). Bayesian approach for neural networks review and case studies. *Neural Networks* **14**, 257–274.

Legarra, A., Robert-Granie, C., Manfredi, E. & Elsen, J. M. (2008). Performance of genomic selection in mice. *Genetics* **180**, 611–618.

Long, N., Gianola, D., Rosa, G. J. M., Weigel, K. A. & Avendan, S. (2007). Machine learning classification procedure for selecting SNPs in genomic selection: application to early mortality in broilers. *Journal of Animal Breeding and Genetics* **124**, 377–389.

MacKay, D. J. C. (1992). Bayesian interpolation. *Neural Computation* **4**, 415–447.

MacKay, J. C. D. (1996). Comparison of approximate methods for handling hyperparameters. *Neural Computation* **8**, 1–35.

MacKay, J. C. D. (2008). *Information theory, inference and learning algorithms*. Cambridge: Cambridge University Press.

Maier, H. R. & Dandy, C. G. (2000). Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications. *Environmental Modelling and Software* **15**, 101–124.

Marwala, T. (2007). Bayesian training of neural networks using genetic programming. *Pattern Recognition Letters* **28**, 1452–1458.

Mott, R. (2006). Finding the molecular basis of complex genetic variation in humans and mice. *Philosophical Transactions* **361**, 393–401.

Mott, R., Talbot, C. J., Turri, M. G., Collins, A. C. & Flint, J. (2000). A method for fine mapping quantitative trait loci in outbred animal stocks. *Proceedings of the National Academy of Sciences of the USA* **97**, 12649–12654.

Mutoh, H., Hamajima, N., Tajima, K., Kobayahsi, T. & Honda, H. (2005). Exhaustive exploring using artificial neural network for identification of SNPs combination to *Heliobacter pylori* infection susceptibility. *Chem-Bio Informatics* **5**, 15–26.

Nguyen, D. & Widrow, B. (1990). Improving the learning speed of two-layer neural networks by choosing initial values of the adaptive weights. *Proceedings of International Joint Conference on Neural Networks* **3**, 21–26.

Ping, G., Michael, R. L. & Chen, C. L. P. (2003). Regularization Parameter Estimation for Feedforward Neural Networks. *IEEE Transactions of Systems, Man, and Cybernetics—Part B: Cybernetics* **33**, 35–44.

Ripley, B. D. (2007). *Pattern Recognition and Neural Networks*. New York: Cambridge University Press.

SAS/STAT® (2009). *Version 9.13*. Cary, NC: SAS Institute Inc.

Shaneh, A. & Butler, G. (2006). Bayesian learning for feed-forward neural network with application to proteomic data: the glycosylation sites detection of the epidermal growth factor-like proteins associated with cancer as a case study. In *Canadian AI* LNAI 4013, 2006 (ed. L. Lamontagne & M. Marchand), pp. 110–121. Berlin-Heiddelberg: Springer-Verleg.

Sorich, M. J., Miners, J. O., Ross, A. M., Winker, D. A., Burden, F. R. & Smith, P. A. (2003). Comparison of linear and nonlinear classification algorithms for the prediction of drug and chemical metabolism by human UDP-Glucuronosyltransferase isoforms. *Journal of Chemical Information and Computer Sciences* **43**, 2019–2024.

Thodberg, H. H. (1996). A review of Bayesian neural networks with an application to near infrared spectroscopy. *IEEE Transactions on Neural Networks* **7**, 56–72.

Titterington, D. M. (2004). Bayesian methods for neural networks and related models. *Statistical Science* **19**, 128–139.

Tu, J. V. (1997). Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of Clinical Epidemiology* **49**, 1225–1231.

Useche, F., Hanafey, G. G. & Rafalski, A. (2001). High-throughput identification, database storage and analysis of SNPs in EST sequences. *Genome Informatics* **12**, 194–203.

Valdar, W., Solberg, L. C., Gauguier, D., Burnett, S. & Klenerman, P. (2006*a*). Genome-wide genetic association of complex traits in heterogeneous stock mice. *Nature Genetics* **38**, 879–887.

Valdar, W., Solberg, L. C., Gauguier, D., Cookson, W. O. & Rawlins, J. N. P. (2006*b*). Genetic and environmental effects on complex traits in mice. *Genetics* **174**, 959–984.

Vazquez, A. I., Rosa, G. J. M., Weigel, K. A., de los Campos, G., Gianola, G. & Allison, D. B. (2010). Predictive ability of subsets of single nucleotide polymorphisms with and without parent average in US Holsteins. *Journal of Dairy Sciences* **93**(12), 5942–5949.

Wang, H. J., Ji, F., Leung, C. S. & Sum, P. F. (2009). Regularization parameter selection for faulty neural networks. *International Journal of Intelligent Systems and Technologies* **4**, 45–48.

Winkler, D. A. & Burden, F. R. (2004). Modelling blood–brain barrier partitioning using Bayesian neural nets. *Journal of Molecular Graphics and Modelling* **22**, 499–505.

Xu, M., Zengi, G., Xu, X., Huang, G., Jiang, R. & Sun, W. (2006). Application of Bayesian regularized BP neural network model for trend analysis, acidity and chemical composition of precipitation in North. *Water, Air, and Soil Pollution* **172**, 167–184.