MULTILAYER PERCEPTRON ARTIFICIAL NEURAL NETWORKS: AN APPROACH FOR LEARNING THROUGH THE BAYESIAN FRAMEWORK

Suellen Teixeira Zavadzki DE PAULI[1]
Mariana KLEINA[2]
Wagner Hugo BONAT[3]

▪ ABSTRACT: The machine learning area has recently gained prominence and artificial neural networks are among the most popular techniques in this field. Such techniques have the learning capacity that occurs during an iterative process of model fitting. Multilayer perceptron (MLP) is one of the first networks that emerged and, for this architecture, backpropagation and its modifications are widely used learning algorithms. In this article, the learning of the MLP neural network was approached from the Bayesian perspective by using Monte Carlo via Markov Chains (MCMC) simulations. The MLP architecture consists of the input, hidden and output layers. In the structure, there are several weights that connect each neuron in each layer. The input layer is composed of the covariates of the model. In the hidden layer there are activation functions. In the output layer, there are the result which is compared with the observed value and the loss function is calculated. We analyzed the network learning through simulated data of known weights in order to understand the estimation by the Bayesian method. Subsequently, we predicted the price of WTI oil and obtained a credibility interval for the forecasts. We provide an `R` implementation and the datasets as supplementary materials.

▪ KEYWORDS: Machine Learning; Bayesian Inference; Prediction; Credibility interval

[1]Universidade Federal do Paraná - UFPR, CEP: 81530-000, Curitiba, PR, Brasil. E-mail: *est.suellen@gmail.com*
[2]Universidade Federal do Paraná - UFPR, Departamento de Engenharia de Produção, CEP: 81530-000, Curitiba, PR, Brasil. E-mail: *marianakleina11@gmail.com*
[3]Universidade Federal do Paraná - UFPR, Departamento de Estatística, CEP: 81530-000, Curitiba, PR, Brasil. E-mail: *wbonat@gmail.com*

1 Introduction

Artificial Neural Networks (ANNs) are popular machine learning techniques that simulate the learning mechanism of biological organisms. The ANN calculates the input functions and propagates the values obtained towards the output neurons. The neural network consists of the input, hidden and output layers, each contains neurons. In the first layer, neurons correspond to covariates, which are weighted and summarized in each neuron in the hidden layer, where there is an activation function. This result is weighted and summarized in the output layer.

The resulting value is compared with the observed value, then we have the loss function which is used by the optimization algorithms in order to search for the smallest error. The learning process occurs by changing the weights that connect neurons, thus the model error is reduced.

The perceptron is the simplest neural network architecture and consists of input variables directly related to the output layer. When there are intermediate layers, we have the so called multilayer perceptron neural network, which is the main focus of this article.

After learning the network weights, it is possible to generalize the knowledge obtained in the training data to unobserved data (AGGARWAL, 2018). Several ANN models have been developed with variations in the learning process and network architecture, each has its own training algorithm (MÓDULO, 2016).

There are many learning methods for neural networks. Any method of non-linear optimization, whether local or global, can be applied to weight optimization of feed-forward neural networks. Training performance varies depending on the objective function and the underlying error surface for a given problem and network configuration (ILONEN *et al.*, 2003).

Gradient descent is widely used, in this method the weights and biases are updated towards the negative gradient of the function, it is based on a linear approximation of the error function (MOLLER, 1997). The backpropagation algorithm (RUMELHART and MCCLELLEND, 1986) is the most popular in this category. Plaut *et al.* (1986) update the descending gradient with the addition of a momentum term, which incorporated a second order information to the method. In this context, some heuristics have been proposed to adapt learning rates and momentum.

Adaptive algorithms based on gradient have been proposed in order to overcome the difficulty in choosing the best learning rates for each region in the search space. In this context, the resilient Retropropagation algorithm (RIEDMILLER and BRAUN, 1993) gained prominence and motivated the development of several variants (ANASTASIADIS *et al.*, 2005).

Other methods using second derivatives have been proposed and, some of the most relevant examples of this type of methods are the quasi-Newton Levenberg-Marquardt (LEVENBERG, 1944; MARQUARDT, 1963) and the conjugate gradient algorithms (BEALE, 1972).

The Newton method is not applicable for training neural networks in their original form, because it involves an inversion of the Hessian matrix, which is an expensive computational task. Several modifications of Newton's algorithm exist to incorporate techniques in order to guarantee a definite non-singular positive hessian matrix. This class of algorithms is called quasi-newton, which is based on the idea of accumulating the curvature information as the iteration process (PLAUT *et al.*, 1986).

Conjugate gradient, inicially developed by Hestenes and Stiefel in 1952, can be considered as intermediate method between gradient descent and Newton's method. Conjugate gradient methods produce non-interfering directions of search, with the assumption that the error function is quadratic (PLAUT *et al.*, 1986).

The main goal of this article is to approach MLP learning through the Bayesian framework, with Monte Carlo via Markov chain method. Therefore, we assume a *priori* distribution for the model weights. In order to understand the learning process and the estimation, we simulated a set of data and observed the results of the *posterioris* ones and the convergence of the chains. We also assessed the estimates and convergence for a time series of WTI oil prices with which it was possible to make a forecast with a credibility interval.

Section 2 describes the multilayer perceptron neural network and presents the configuration of the network used in the simulation study. Section 3 describes how the simulation was performed. Section 4 presents the Bayesian approach for weights estimation. In Section 5, we apply the model to historical oil series WTI and some final remarks are presented in Section 6. The `R` implementation and the datasets are available in the supplementary materials.

2 Multilayer Perceptron Artificial Neural Network

The multilayer perceptron (MLP) model, whose learning is supervised by error correction, has an architecture with more than one layer, it is acyclic (when the output of a neuron cannot serve as an input to some previous neuron) and it is connected (each input is processed by all neurons). The propagation rule is given by the internal product of the weighted inputs with the addition of the bias term and the output from the previous layer is the input from the current layer (HAYKIN, 2003).

MLP training occurs in the forward and backward phases. In the first phase, the entries pass through the entire network calculation structure with a certain set of weights and its output is compared with the observed values, in order to compute some loss function. In the second phase, a training algorithm is applied to reduce the loss function by fitting the weights (AGGARWAL, 2018).

Figure 1 presents an illustration of a multilayer perceptron network structure. For this example, the input layer contains 5 covariates; the single hidden layer contains two neurons, and the output layer contains one neuron. This example structure was used to approach the applications.

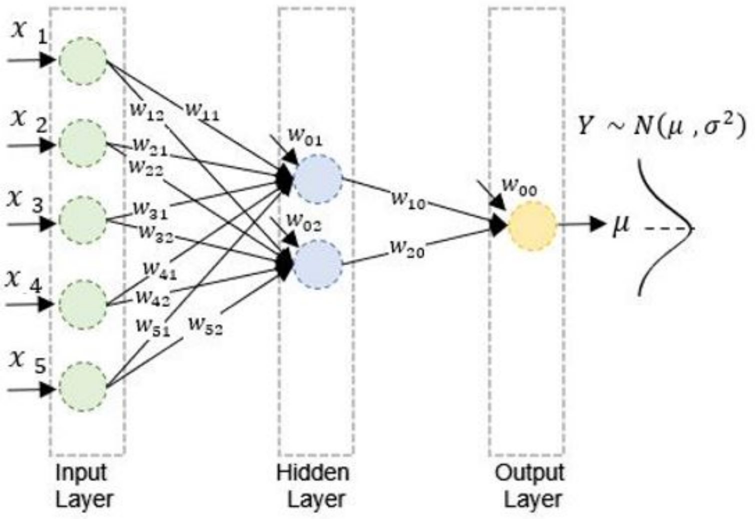

Figure 1 - Structure of a multilayer perceptron model with five inputs and two hidden layers.

Each of the n covariate x_i, for $i = 1, \ldots, n$, is weighted by w_{ij} which links neuron i from the input layer to hidden neuron j, with $j = 1, \ldots, k$, where k is the number of neurons in the hidden layer.

Thus, in each hidden neuron j, there is the sum of the input signals weighted by the neuron synaptic weights with the addition of the term bias b_j, according to Equation 1 (HAYKIN, 2003)

$$v_j = \sum_{i=1}^{n} x_l w_{lj} + b_j. \tag{1}$$

The output v_j goes through an activation function, which for this article is the logistical sigmoid, according to the Equation 2

$$\varphi(v_j) = \frac{1}{1 + e^{-v_j}}. \tag{2}$$

Each j neuron in the hidden layer is then weighted by w_{jh}, where h corresponds to the output layers, that for this case are unique. Then, these weights are added with the bias, according to the Equation 3. Finally, the result goes through an activation function, which in this case is the identity function

$$\hat{\mu}_h = \sum_{j=1}^{k} w_{jh}\varphi(v_j) + b_h. \tag{3}$$

3 Data simulation

We simulated a data set using a historical oil price data set so that the simulated data would be close to a real data. We chose 5 daily historical for the covariates and the response variable was the next day. The weights were estimated using a multilayer perceptron neural network with 2 neurons in the hidden layer and a sigmoid logistic function. In the output layer, the function used was the identity. The 15 weights were estimated using a function from the `neuralnet` (FRITSCH *et al.*, 2016) package of the `R` (R Core Team, 2019) software and these estimates were saved, which resulted in the model of the Equation 4.

For the simulation study, we passed a single time with the 5 covariates by the Equation 4. It is worth mentioning that there was no training in this part, the intention was only to generate data with a known network structure.

$$\begin{aligned}\mu = -3,4 + 2,5\frac{1}{(1+\exp-(2{,}40+1{,}2x_1+0{,}004x_2-0{,}002x_3+0{,}003x_4-0{,}02x_5))}+\\ 4,1\frac{1}{(1+\exp-(-1{,}10+0{,}89x_1+0{,}006x_2-0{,}005x_3+0{,}12x_4-0{,}08x_5))}\end{aligned} \tag{4}$$

In general, we specified a neural network model to explain the expected value of a random variable. Thus, the model predicts the average of the response variable given a set of covariates, as given in Equation 4.

For continuous output, it is reasonable to expect that the distribution of the errors is symetric. Therefore, we assume that the response variable follows a Gaussian distribution with an expected value μ and a variance $\sigma^2 = 1$, i.e. $Y \sim N(\mu, 1)$. Thus, we have the weights to simulate the data set and, based on this knowledge, we estimate the weights using Bayesian inference.

4 Bayesian Inference

In the statistical context, making inferences about new phenomena or new observations is a common objective and, there are different strategies for achieving this objective. The classical frequentist approach whose founders are Karl Pearson, Ronald Aylmer Fisher and Jerzy Neyman, has been prevalent for many years. Basically, this approach is based on the idea of inference as a generalization about the population based on a sample. An important aspect is to recognize the variability that exists from sample to sample and to analyze the observed data as one of many sets that could have been obtained in the same circumstances. The interpretation depends not only on the observed sample but also on the hypotheses adopted about the possible sample sets (TURKMAN *et al.*, 2018).

In addition to the classic frequentist approach, there are others, such as Bayesian, structuralist, likelihoodist and others. In the Bayesian inference context, the adjustment of a data set to a probabilistic model is summarized by the result of the probability distribution of the model parameters and the sample observations. The statistical conclusions about a given parameter θ or about unobserved data $\tilde{y}$ are based on probability statements, which are conditional on the observed value y and can be written as $p(\theta|y)$ or $p(\tilde{y}|y)$. Statements of θ given y start with a joint probability distribution of θ and y. The probability density can be written as the product of two densities that are called *priori* $p(\theta)$ distribution and sample distribution $p(y|\theta)$. Thus, we have the Equation 5 (GELMAN *et al.*, 2014).

$$p(\theta, y) = p(\theta)p(y|\theta). \tag{5}$$

Conditioning the known value of y and using the conditional probability property, the *a posteriori* density is described in the Equation 6. An equivalent form omits $p(y)$, this being fixed, it produces the *a posteriori* density, according to the Equation 7.

$$p(\theta|y) = \frac{p(\theta)p(y|\theta)}{p(y)}. \tag{6}$$

$$p(\theta|y) \propto p(\theta)p(y|\theta). \tag{7}$$

Therefore, in the Bayesian inference, after additional information about an event that occurred, such as the observation of a data set, the *priori* probabilities are reviewed using the Bayes formula, as in the Equation 7 and the *posteriori* probabilities are then designated (TURKMAN *et al.*, 2018).

Several methods were created for sampling from the *posteriori* distribution in the Bayesian context. Monte Carlo via Markov chain (MCMC) is a general method based on the values obtained for θ to approximate distributions which are corrected to better approximate the *posteriori* distribution $p(\theta|y)$ (GELMAN *et al.*, 2014).

This method operates by sequentially sampling parameter values from a Markov chain, whose stationary distribution is exactly the interest *posteriori* distribution (CARLIN and LOUIS, 2008).

The chain is a sequence of random variables $\theta^1, \theta^2, \ldots$ where for any t, the distribution of θ^t, considering all previous θ, depends only on the most recent value, that is, $\theta^{(t-1)}$. Then, it starts from a point θ^0 and for each t we get θ^t from a transition distribution $T_t(\theta^t|\theta^{(t-1)})$, which is dependent on $\theta^{(t-1)}$. Transition distributions must be constructed so that the Markov chain converges to a single stationary distribution, which is the *posteriori* $p(\theta|y)$ distribution. The success of the method is because the approximate distributions are improved at each step of the simulation, which means that it is converging to the target distribution (GELMAN *et al.*, 2014).

In order to estimate the parameters via Bayesian inference, JAGS (Just Another Gibbs Sampler) was used, which is a program for analyzing Bayesian models using MCMC. It is written in `C++`, however, it has an interface with software `R` by the `rjags` (PLUMMER, 2017) package, which was used for the application. JAGS generates the sample from the *posteriori* distribution given the observed sample. The number of Markov chains can be defined in the code and produces a sequence of independent samples from the *posteriori* distributions selected for each model parameter. The MCMC output is divided into two parts. The first part is called burn in, which is the interval between the initial value and the beginning of the period to be considered. The burn in is an adaptive period that can be selected for disposal. The second part occurs from the burn in point until the end of the iterations and, the output is considered convergent (PLUMMER, 2017).

The basic methods of Markov chain simulation are Gibbs sampler (GEMAN *et al.*, 1984) and Metropolis-Hastings. The last one is the main method and the others are originated from it. Initially proposed by Metropolis *et al.* (METROPOLIS *et al.*, 1953) and, posteriorly generalized by Hastings (Hastings, 1970). The Gibbs sampler algorithm, also called full conditional sampling, generates a sample sequence from the initial joint probability distribution $\theta_1, \theta_2, ..., \theta_k$. In each Gibbs cycles iteration, a conditional subset is extracted from the others sampled so far, using the initial subvector of θ (GELMAN *et al.*, 2014).

Models with 1 to 5 neurons in the hidden layer were considered for the simulated data from the multilayer perceptron neural network. For one neuron, for instance, we have the model according to the Equation 8. In the Equations 9 and 10, there are response and *priori* distributions of weights. In order to have have a large seacrh space, the variance of the *priori* distribution was considered as 10.

$$\mu = w_{00} + w_{10}\frac{1}{(1+\exp\left(-(w_{01}+w_{11}x_1+w_{21}x_2+w_{31}x_3+w_{41}x_4+w_{51}x_5)\right))} \tag{8}$$

$$Y \sim N(\mu, 1) \tag{9}$$

$$w_{ij} \sim N(0, 10), i = 1, 2, ..., 5, j = 1, 2 \tag{10}$$

The neural network model is a overparametrized nonlinear structure, in addition there is multiplication between weights, which makes it difficult to find a global optimum. The distribution considered a priori to the weights was Gaussian so that the search was possible in an $\mathbb{R}$ space with symmetry. After the simulations it is possible to realize that the assumption of Gaussianity may be very strong given the structure of the model.

Considering 2 neurons in the hidden layer, it is possible to evaluate whether the mean of the *posteriori* distribution is similar to that which generated the data set. In Figure 2, we have the *posteriori* distribution.

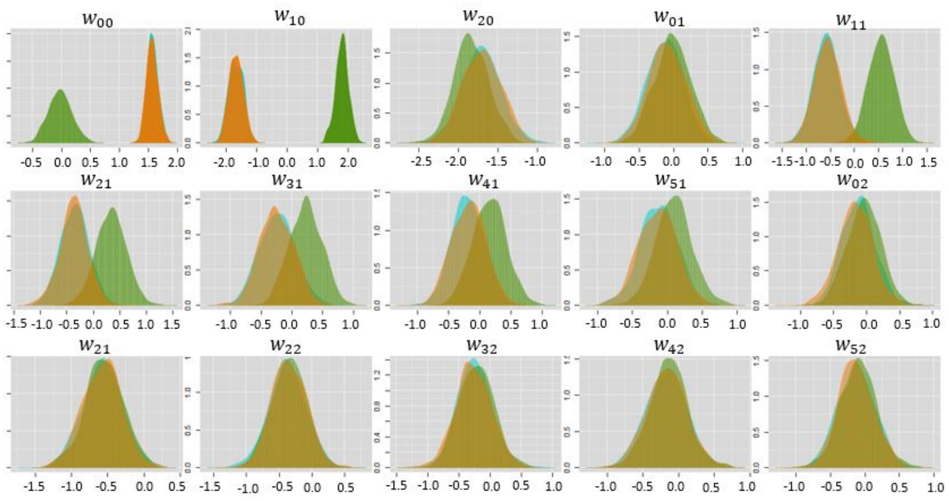

Figure 2 - *Posteriori* distribution for the MLP weights with 2 hidden neurons.

Markov chains are presented in Figure 3 that, for this case, did not show the same direction in all weights, which indicates convergence difficulty.

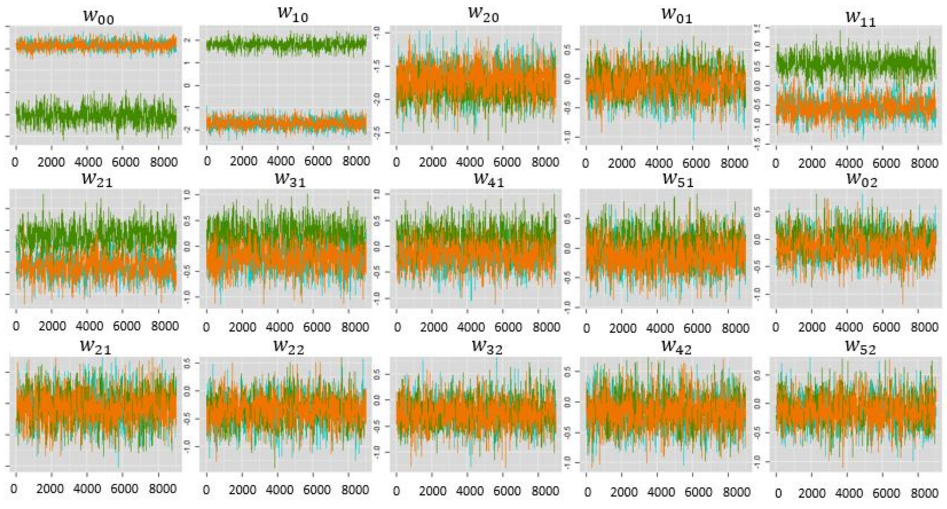

Figure 3 - Markov chains for an MLP trained with 2 hidden neuron.

We also tested models with 3, 4 and 5 neurons in the hidden layer. Figure 4 shows the *posteriori* distribution and Figure 5 the Markov chains, both for 5 neurons. We can verify bimodal distributions in the chains for certain weights, which shows the difficulty in finding a single optimal value.

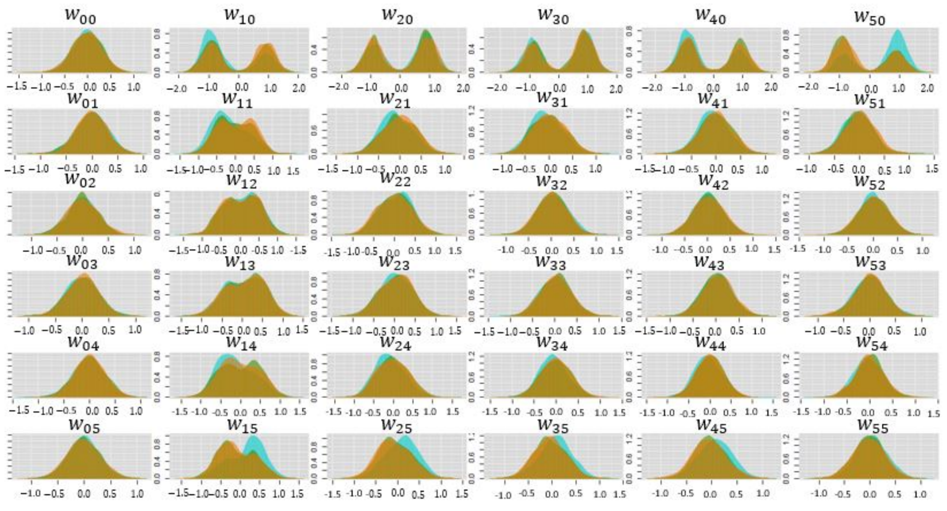

Figure 4 - *Posteriori* distribution for the MLP weights with 5 hidden neurons.

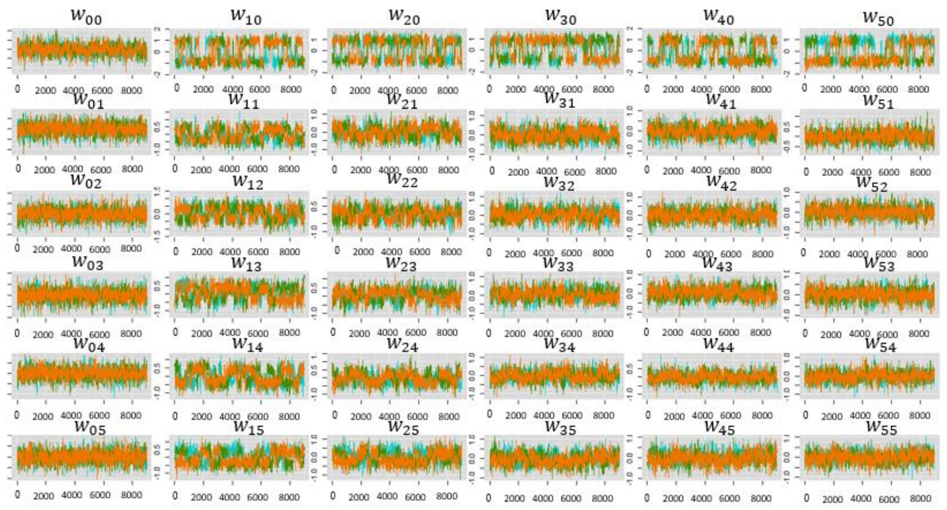

Figure 5 - Markov chains for an MLP trained with 5 hidden neuron.

5 Application

In addition to the simulation study, we did an application with a historical daily series of WTI oil price corresponding to the period from January 1, 2015 to December 16, 2019. The first 1000 days were used for training, which corresponds to approximately 80% of the total, and 262 for the forecast.

In Figure 6 it is possible to check the times series and the cutoff point that separates training and validation sets, which corresponds to November 30, 2018. The data were obtained directly from the `ipeadatar` (GOMES, 2019) package.

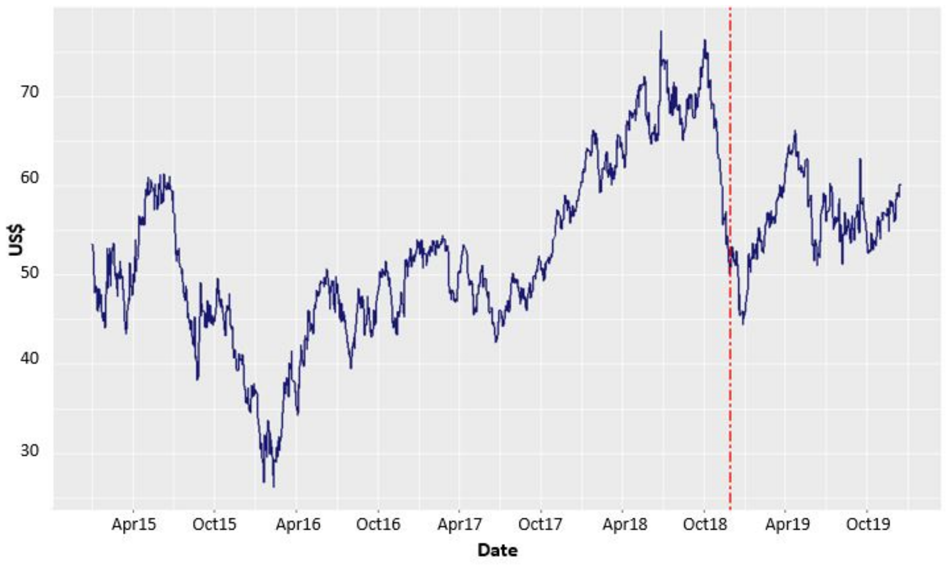

Figure 6 - WTI oil times series.

The correlation between the variables is represented in Figure 7 and, as expected, it is possible to see a high correlation and that it gradually decreases as the days move away from the predicted day. The covariates used are $t-1$, $t-2$, $t-3$, $t-4$ and $t-5$. The first considers the previous day value, the second two previous days of the forecast and so on up to five days before the forecast.

The weights were estimated by Bayesian inference with the attempts of 1 to 5 neurons in the hidden layer of the MLP. The activation function used for all cases was the logistical sigmoid. The variance of the *priori* distribution was considered as 1.

In Figure 8 there is the 95 % credibility interval for the validation period with two neuron in the hidden layer. Frequentist statistics uses confidence intervals while Bayesian statistics uses credibility intervals. In general, the Bayesian credibility intervals do not coincide with the standard confidence intervals, since the first one incorporates contextual information specific to the *priori* distribution (CARLIN, 2008; TURKMAN *et al.*, 2018).

Despite the following historical behavior, the range is quite wide. It is also possible to observe the average represented in white dotted lines.

For models with 1 to 5 neurons in the hidden layer, the credibility intervals of 95 % are similar and quite wide, this shows the difficulty of convergence of the model. Other versions of the MCMC with the Hamiltonian Monte Carlo could be better for the problem.

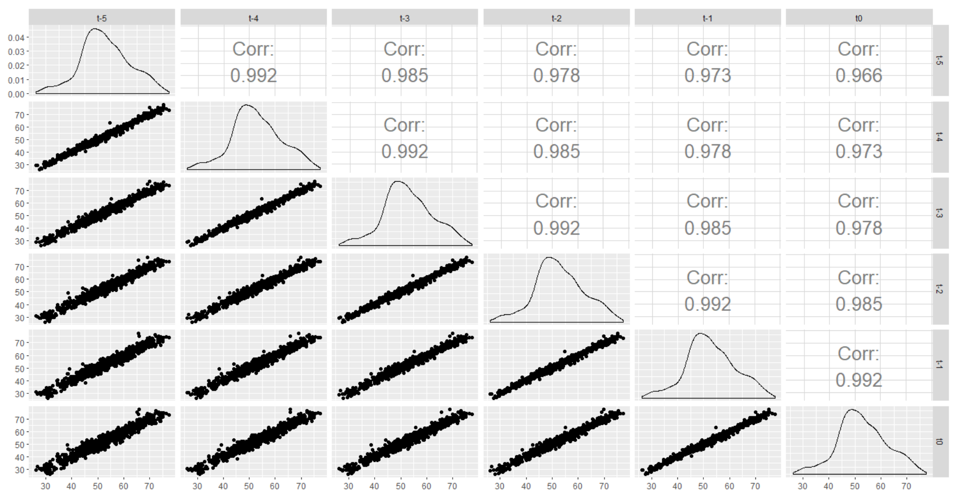

Figure 7 - Variables Correlation.

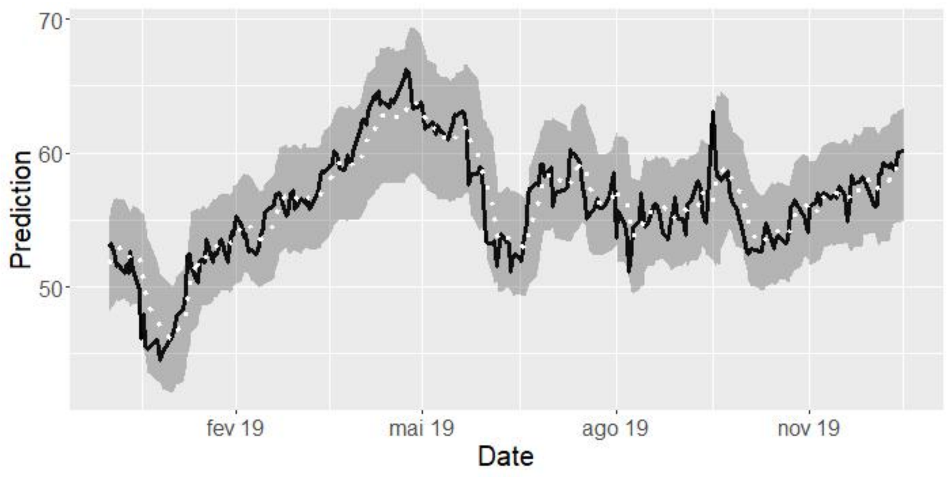

Figure 8 - Credibility interval for an MLP trained with 2 hidden neuron.

6 Discussion

The aim of this article was to understand the multilayer perceptron neural network structure with a learning approach via Bayesian inference in order to better interpret the network. First, we applied it in simulated data and after in real data.

The chains, in general, did not obtain a stationary behavior. Especially with the *posteriori* distribution for the weights with two neurons in the hidden layer, the mean did not coincide with the one that generated the simulated data. In addition, for some structures, a bimodal *posteriori* distribution was obtained, which indicates that more than one mean was found.

There is also evidence that as the structure of neurons in the hidden layer increases, there is also greater difficulty in convergence. The network structure allows more than one possible configuration for the combination of weights, which means that a different combination can find the same model error and consequently it is difficult to find a single model structure for the data.

For future studies, the Bayesian approach could be tested for other configurations besides the multilayer perceptron, such as Recurrent Neural Networks (RNN) that have gained prominence in time series applications. Applications could also be made with other activation functions besides the logistical sigmoid. Regularization could also be carried out to alleviate identifiability problems.

Supplementary material

http://www.leg.ufpr.br/doku.php/publications:papercompanions

DE PAULI, S. T. Z.; KLEYNA, M.; BONAT, W. H. Redes neurais artificiais multicamadas perceptron: uma abordagem para aprendizagem através da estrutura bayesiana. *Rev. Bras. Biom.,* Lavras, v.39, n.1, p.45-59, 2020.

▪ *RESUMO: A área de aprendizado de máquina ganhou destaque recentemente e as redes neurais artificiais estão entre as técnicas mais populares nessa área. Tais técnicas têm a capacidade de aprendizagem, que ocorre durante um processo iterativo de ajuste do modelo. A Multilayer Perceptron (MLP) é uma das primeiras redes a surgir e, para essa arquitetura, backpropagation e suas modificações são algoritmos de aprendizagem amplamente utilizados. Neste artigo, o aprendizado da rede neural MLP foi abordado a partir da perspectiva Bayesiana por meio de simulação de Monte Carlo via cadeias de Markov (MCMC). A arquitetura de uma MLP consiste nas camadas de entrada, ocultas e de saída. Na estrutura, existem vários pesos que conectam cada neurônio em cada camada. A camada de entrada é composta pelas covariáveis do modelo. Na camada oculta, existem funções de ativação. Na camada de saída, está o resultado, o que é comparado com o valor observado e a função de perda é avaliada. Analisamos o aprendizado da rede por meio de dados simulados de pesos conhecidos a fim de compreender a estimação pelo método Bayesiano. Posteriormente, previmos o preço do óleo WTI e obtivemos um intervalo de credibilidade para as projeções. Nós fornecemos uma implementação* `R` *e os conjuntos de dados como materiais complementares.*

▪ *PALAVRAS-CHAVE: Aprendizado de máquina; Inferência Bayesiana; Predição; Intervalo de credibilidade*

References

AGGARWAL, C. C. *Neural networks and deep learning.* Springer, 2018. 512p.

BEALE, E. M. L. *A derivation of conjugate gradients. In LOOSTMA, F. A. Numerical methods for nonlinear optimization*, London: Academic Press, London, 1972. p.39-43.

CARLIN, B. P.; LOUIS, T. A. *Bayesian methods for data analysis.* 3.ed. Boca Raton:Chapman and Hall/CRC, 2008. 552p.

FRITSCH, S.; GUENTHER, F.; GUENTHER, M. F. *Package neuralnet. The Comprehensive R Archive Network*, 2016.

GELMAN, A.; CARLIN, J. B.; STERN, H. S.; RUBIN, D. B. *Bayesian data analysis.* 2.ed. Boca Raton: Chapman and Hall, 2014. 690p.

GEMAN, S.; GEMAN, D. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, v.6, n.6 p.721-741, 1984.

GOMES, L. E. S. *ipeadatar: API Wrapper for Ipeadata.* R package version 0.1.0, 2019.

HASTINGS, W. K. Monte carlo sampling methods using markov chains and their applications. *Biometrika* , v.57, n.1, p.97-109, 1970.

HAYKIN, S. *Redes neurais: princípios e prática.* 2.ed., Bookman Editora, 2003. 898p.

HESTENES, M. R.; STIEFEL, E. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, v.49, n.6, p.409-436, 1952.

ILONEN, J.; KAMARAINEN, J. K.; LAMPINEN, J. Differential evolution training algorithm for feed-forward neural networks. *Neural Processing Letters*, v.17, n.1, p.93-105, 2003.

LEVENBERG, K. A method for the solution of certain non-linear problems in least squares. *Quaterly Journal of Applied Mathematics*, v.2, n.2, p.164-168, 1944.

MARQUARDT, D. W. An algorithm for least-squares estimation of non-linear parameters. *Journal of the Society of Industrial and Applied Mathematics*, v.11, n.2, p.431-441, 1963.

METROPOLIS, N.; ROSENBLUTH, A. W.; ROSENBLUTH, M. N.; TELLER, A. H.; TELLER, E. Equation of state calculations by fast computing machines. *The journal of chemical physics*, v.21 , p.1087-1091, 1953.

MÓDULO, M. *Classificação Automática de Supernovas Usando Redes Neurais Artificiais.* 2016. p. Tese (Doutorado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais (INPE), Sao José dos Campos, 2016.

MOLLER,M. *Efficient training of feed-forward neural networks*, 1997. p. Ph.D. Thesis, Computer Science Department, Aarhus University, Arhus, Denmark, 1997.

PLAUT, D.; NOWLAN, S. and HINTON, G. Experiments on Learning by Back-Propagation, *Technical Report CMU-CS-86-126*, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1986.

PLUMMER, M. *Jags version 4.3. 0 user manual* [software manual], 2017.

R Core Team: *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2019. URL https://www.R-project.org/

TURKMAN, M.; PAULINO, C.; MURTEIRA, B.; SILVA, G. *Estatística bayesiana.* Lisboa: Fundação Calouste Gulbenkian. 2018. 616p.

Received on 26.08.2020.

Approved after revised on 28.01.2021.