Neural network within a Bayesian inference framework

Neural network within a Bayesian inference framework

Isidro Gómez Vargas[1,2], Ricardo Medel Esquivel[1,2], Ricardo García Salcedo[1] and J Alberto Vázquez[2]

[1]CICATA-Legaria, Instituto Politécnico Nacional, 11500, Ciudad de México, México.
[2] ICF, Universidad Nacional Autónoma de México, 62210, Cuernavaca, Morelos, México.

E-mail: igomezv0701@alumno.ipn.mx

Abstract. In Bayesian inference, the likelihood functions are evaluated thousands of times. In this paper we explore the use of an Artificial Neural Network to learn how to calculate the likelihood function and thus speed up the Bayesian inference process. We test the performance of the neural network on a parameter estimation of the standard cosmological model and show that this method can reduce the computational time.

1. Introduction

Bayesian inference is a technique for obtaining statistical information on unknown probability density functions using Bayes' theorem, which to test a theoretical model from known data, has the following form:

$$P(\theta|D,H) = \frac{P(\theta|D,H)P(\theta|H)}{P(\theta|D)} \tag{1}$$

where D represents the observational data set, H is the hypothesis –or model– under test and θ is the set of its free parameters. The prior probability $P\ (\theta|H)$ represents our knowledge of the parameters θ before considering the observable data. The likelihood function is the probability of the data given the model parameters $P\ (D|\theta,\ H)$. Finally, $P\ (D|H)$ is a normalization constant, that is the likelihood marginalization and is called Bayesian evidence:

$$P(D|H) = \int d^N\, P(\theta|D,H)P(\theta|H) \tag{2}$$

where N is then umber of free parameters of the theoretical model.

The ultimate goal of Bayesian inference is to obtain the posterior probability $P(\theta|D,H)$, which represents the state of knowledge of the model parameters once the information provided by the data has been taken into account. By calculating the posterior probability, we can obtain predictions based on the theoretical model considered and on the intrinsic information of the data [1, 2].

Likelihood functions link the data to theory and are constructed by assuming some particular statistical distribution for the D data, usually a Gaussian distribution. In the calculation of likelihood function, at a given point, it is also necessary to evaluate the theoretical model several times. On the other hand, if several types of observations are involved, the probability density function proposed as likelihood function should be a multiplication of several likelihoods (one for each type of data). The nature of Bayesian inference requires multiple evaluations of the likelihood function to generate a new sample with a higher likelihood value than its predecessor, and if these functions are complex, the computational time spent on these evaluations can be considerable.

In this paper we evaluate the performance of an Artificial Neural Network (ANN) in the calculation of likelihood functions within a Bayesian inference process [3, 4]. In the following section we describe, in general terms, the procedure of our study and how we use the ANN. In Section 3, we show an application on the standard cosmological model and make the comparison of results of Bayesian inference using and not using the neural network to calculate the values of the likelihood function. Finally, the Section 4 contains our conclusions.

2. Methodology

We based this work in the Ref. [3] that proposes a feed forward ANN to learn the likelihood function within a nested sampling algorithm [5]. In this direction, we use a nested sampling algorithm [6] available in Dynesty [7] and we adapt it to use with the pyBambi package [8] (a development based on [3]). The ANN was implemented with the tensor flow Python library.

2.1. Nested Sampling

Nested sampling [5] is an algorithm that allows the calculation of Bayesian evidence through a Riemann sum by mapping the parameter space in an interval between 0 and 1 (for more details see Ref. [9]). This method generates an initial number of random points (called live points) within the parameter space, sorts them by their likelihood value, and in each iteration deletes the worst one, generates a new sample in a reduced parameter space and improves the calculation of the Bayesian evidence. Since a point is generated with respect to its predecessor, it also allows the sampling of the posterior probability function, like any other Markov Chain Monte Carlo algorithm.

2.2. Artificial Neural Network

The *Universal Approximation Theorem* [10] allows the use of an ANN to learn how to calculate the likelihood function. It states that an Artificial Neural Network with at least one hidden layer with a finite number of neurons can approach any continuous function if the activation function is continuous and non-linear.

In our case, we use a feed forward ANN with three hidden layers and the Rectified Linear Unit (RELU) as activation function. The loss function is the Mean Square Error (MSE):

$$MSE = \frac{1}{N}\sum_{i}^{n}(yi - \hat{y}i)^2 \tag{3}$$

we apply the Adam gradient descent method to minimize it, initially with learning rate of 0.1 and reducing it by a factor of 0.1, until 0.0001, if through 5 epochs the value of the loss function does not improve.

This MSE has a direct relationship with the variance of the training set [11], therefore nested sampling allows the neural network loss function achieve lower values for samples near to the end of the inference process.

2.3. Method
The likelihood function evaluates points in the parameter space that have as many coordinates as free parameters have the theoretical model considered to make the Bayesian inference. As will be described later, the model used in this paper has three free parameters. Therefore, the number of nodes in the input layer of the neural network must match this value. On the other hand, the output layer have a single node that is the prediction of the likelihood function.

We tune the hyper-parameters of the neural network running 35 combinations of them and choosing the one that achieve a lower value for the loss function. For this test we use 50, 100, 150, 200, 250 and 300 nodes for the three hidden layers and 4, 8, 16, 32 and 64 for the batch size value. The better combinations was the shown in figure 1 with 8 for the batch size.

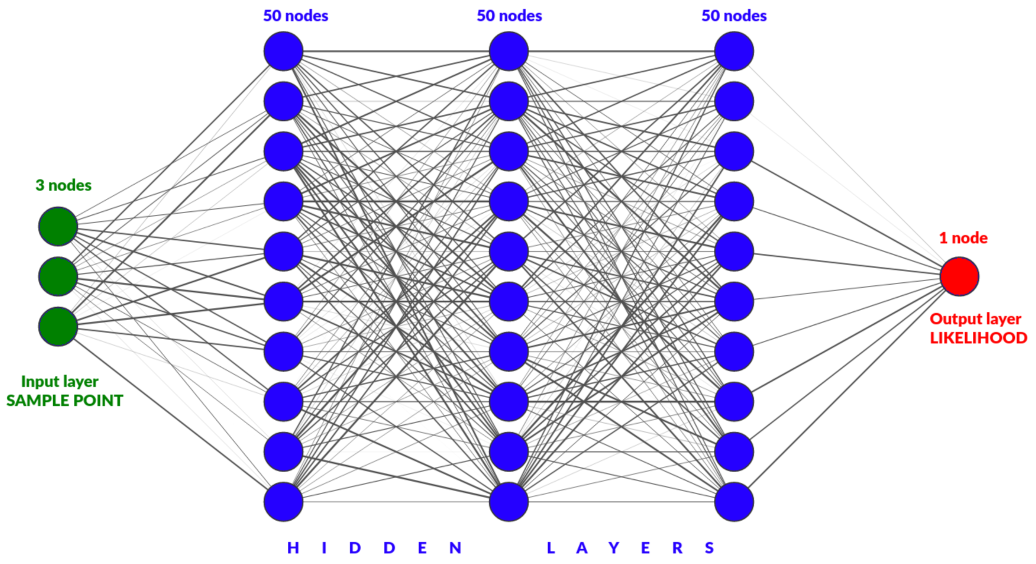

Figure 1. Feed forward ANN with three hidden layers used in this work

In the generation of every 500 new samples, within the Bayesian inference framework, the neural network was trained. Therefore the 80% of this 500 samples with their respective likelihoods are used as training set, the remaining samples conform the test set.

When the MSE is below a predefined value (in our case, we use 0.1), the trained neural network replaces the analytical calculations of the likelihood function, otherwise the ANN is retrained after another 500 samples are generated and in the meantime the analytical likelihood continues to be used.

If, using the neural network, its predictions are outside the range of existing likelihoods (with a small deviation as tolerance, 0.1 in our case), the neural network is no longer used and the analytical calculation is returned. Subsequently, if the neural network is retrained correctly in the way described above, it can come back into action.

3. Example: Bayesian inference on the ΛCDM model
The standard cosmological model, also known as ΛCDM , represents a flat universe with a cosmological constant that provides accelerated expansion. We use the Friedmann's equation, with a constriction for the cosmological constant energy density $\Omega_\Lambda = 1-\Omega_m$ (Ω_m is the matter energy density), to reduce the number of free parameters:

$$\frac{H^2(a)}{H_0^2} = \frac{1\,\Omega_{0b}h^2 + \Omega_{0c}h^2}{a^3} + (1 - \Omega_{0m}), \tag{4}$$

where H is the Hubble factor, H_0 the Hubble constant, a is the scale factor (function of time representing the relative expansion of the universe), Ω_{0b} is the current energy density of baryons, Ω_{0c} the current energy density of cold dark matter, $\Omega_{0m} = \Omega_{0b} + \Omega_{0c}$ and $h = \frac{H}{100}$ Therefore, we use three free parameters: h^2 , Ω_{0m} and $\Omega_{0b}h^2$. There are wll know in cosmology [12] and allow us to evaluate our results of the Bayesian inference with and without neural network.

If we assume a Gaussian distribution for the data, we can construct the log-likelihood function as a chi-square test involving the theoretical model of the Equation 4 and the observational data. In our test, the likelihood function considers data from Type-Ia Supernovae [13], Cosmic Chronometers [14], Baryon Acoustic Oscillations [15] and a compressed information of Planck-15 [16].

The figure 2 shows the behavior of the loss function (MSE) for the ANN, described in the previous section, in the training and validation sets, using 500 samples of the Bayesian inference process. If the value for MSE is high, it is very likely that the predictions made by the ANN will be wrong, so it is necessary to wait until the final stage of sampling in order to properly use the neural network predictions. Figure 3 shows the 1D and 2D plots of the resulting posterior distribution for Bayesian inference with and without ANN.

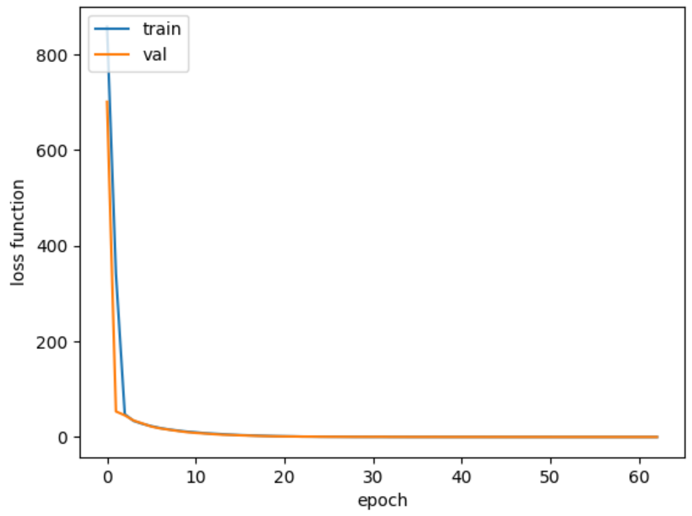

Figure 2. Loss function of the neural network in the training and validation set

The parameter estimation process for ΛCDM was performed five times, both with and without ANN. The averages of these five procedures for each case are reported in table 3. We can note that the parameter estimation by nested sampling with and without the neural network are very close to each other and can be statistically interpreted in the same way. To get closer to the reference values, it would be necessary to add even more data to our Bayesian inference and that is not the purpose of this work.

We found that, on average, the neural network only calculates about 6% of the total likeli- hood calls and generates 4% of the total samples. However, in our example, this reduces the computational time by about 9.1 percent.

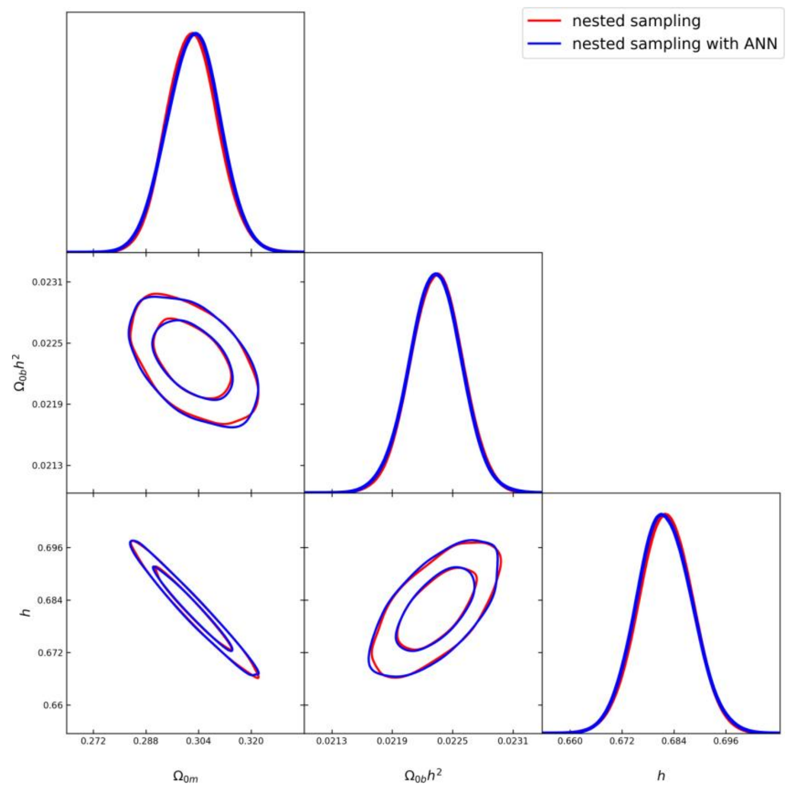

Figure 3. Posterior samples for the ΛCDM model obtained by nested sampling with and without neural network. We implemented this in the SimpleMC repository [17].

Table 1. Results of the nested sampling algorithm applied to Λ*CDM* model

	Reference value [12]	without ANN	with ANN
Ω_m	0.3166±0.0084	0.2978±0.0680	0.2982±0.0660
$\Omega_b h^2$	0.02242±0.00014	0.0224±0.0009	0.0224±0.0010
h	0.6727±0.006	0.6918±0.0734	0.6906±0.0723
log(*Bayesian evidence*)		-41.890±0.196	-41.849±0.195
Samples generated with dynesty		7742	7700
Samples generated with ANN predictions			282
Likelihood predicted with ANN			2202
Total likelihood calls		33007	33323
time (minutes)		73.2	66.8

4. Conclusions

According to our results, for the standard cosmological model and the data sets mentioned above, we have noticed that if the neural network is well calibrated and achieves a low MSE, it can substitute the analytical calculation of the likelihood function in the final part of a Bayesian inference process without significant alterations in the statistical analysis. Although the samples generated with the neural network likelihood predictions make up a small percentage, the acceleration in the Bayesian inference process is noticeable.

As future work we want to test this technique with more complex models, both in the field of cosmology and in any other branch of science in which a Bayesian inference process can be applied. We are also interested in testing with larger data sets and implementing it in parallel.

Acknowledgments

This work was partially supported by CONACyT-Mexico scholarship, BEIFI-IPN, the SIP20200666 grant by IPN and CONACYT 2558591. JAV acknowledges the support provided by FOSEC SEP-CONACYT Investigación Básica A1-S-21925, and UNAM DGAPA-PAPIIT IA102219.

References

[1] Medel Esquivel R, Gómez Vargas I, Vázquez J A and García Salcedo R 2021 Accepted in *Boletín de Estadística e Investigación Operativa* 37
[2] Padilla L E, Tellez L O, Escamilla L and Vázquez J A 2020 *Preprint abs*/1903.11127
[3] Graff P, Feroz F, Hobson M P and Lasenby A 2012 *Monthly Notices of the Royal Astronomical Society* **421**(1) 169-180
[4] Gómez-Vargas I, Medel-Esquivel R, García-Salcedo R and Vázquez J A 2019 *Komputer Sapiens* **2**(11) 12-17
[5] Skilling J 2004 *AIP Conference Proceedings* **735**(1) 395-405
[6] Feroz F, Hobson M P and Bridges M 2009 *Monthly Notices of the Royal Astronomical Society* **398**(4) 1601- 1614
[7] Speagle J S 2020 *Monthly Notices of the Royal Astronomical Society* **493**(3) 3132-3158
[8] Handley W et al 2019 Zenodo http://doi.org/10.5281/zenodo.2560069
[9] Skilling J 2006 *Bayesian analysis* **1**(4) 833-859
[10] Hornik K, Stinchcombe M and White H 1989 *Neural networks* **2**(5) 359-366
[11] Geman S, Bienenstock E and Doursat R 1992 *Neural computation* **4**(1) 1-58
[12] Aghanim N et al 2020 *Astronomy & Astrophysics* 641 A6
[13] Suzuki, N et al 2012 *The Astrophysical Journal* **746**(1) 85
[14] Riess A G et al 2018 *The Astrophysical Journal* **855**(2) 136
[15] Ata M et al 2018 *Monthly Notices of the Royal Astronomical Society* **473**(4) 4773-4794
[16] Aubourg E et al 2015 *Physical Review D* **92**(12) 123516
[17] Vazquez J A 2020 Github repository https://github.com/ja-vazquez/SimpleMC