Peering towards Cosmic Dawn
Proceedings IAU Symposium No. 333, 2018
V. Jelić & T. van der Hulst, eds.

doi:10.1017/S174392131700984X

Neural Network Emulation of Reionization Simulations

Claude J. Schmit[1,2] **and Jonathan R. Pritchard**[1,3]

[1]Blackett Laboratory, Imperial College,
London, SW7 2AZ, UK
[2]email: claude.schmit13@imperial.ac.uk
[3]email: j.pritchard@imperial.ac.uk

Abstract. Next generation radio experiments such as LOFAR, HERA and SKA are expected to probe the Epoch of Reionization and claim a first direct detection of the cosmic 21cm signal within the next decade. One of the major challenges for these experiments will be dealing with enormous incoming data volumes. Machine learning is key to increasing our data analysis efficiency. We consider the use of an artificial neural network to emulate 21cmFAST simulations and use it in a Bayesian parameter inference study. We then compare the network predictions to a direct evaluation of the EoR simulations and analyse the dependence of the results on the training set size. We find that the use of a training set of size 100 samples can recover the error contours of a full scale MCMC analysis which evaluates the model at each step.

Keywords. cosmology: reionization - methods: numerical - methods: statistical

1. Introduction

With the advent of the next generation of reionization experiments, such as LOFAR, HERA and SKA, a detection of the cosmic 21cm signal is close at hand. More and more stringent upper limits are becoming available from these experiments and their precursors, constraining and shaping our understanding of the Epoch of Reionization (EoR). As the systematic understanding of our experiments and foreground contaminations of the signal increase, imaging the EoR is becoming available as a tool to measure the 21cm power spectrum as a function of the observed frequency and to statistically characterize the signal. An important question is how to use these observations to infer astrophysical parameters and is most commonly approached as a Bayesian inference problem implemented using MCMC techniques as in Greig & Mesinger (2015). An MCMC algorithm walks through parameter space, evaluating the model in order to map out the posterior distribution for a given data set. This requires a potentially large number of model evaluations and can thus be prohibitively expensive depending on the execution time of the model. In order to increase the efficiency with which the posterior of any given observation may be analysed, instead of evaluating the model at each point in parameter space, we propose to emulate the model using an artificial neural network (ANN) (eg. Shimabukuro & Semelin 2017). Specifically, we use an ANN to emulate the output of 21cmFAST (Mesinger & Furlanetto 2007) simulations used in a Bayesian likelihood analysis and study the effect of training set size on the predictions of the network (Schmit & Pritchard 2017). We compare our emulated parameter inference results to those of 21CMMC from Greig & Mesinger (2015).

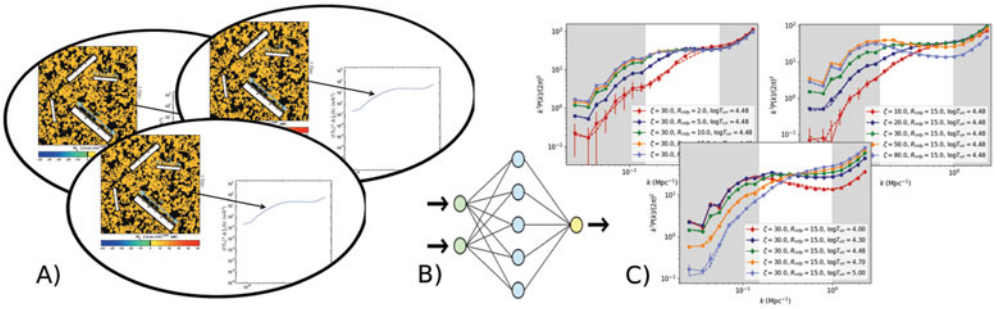

Figure 1. Flow chart describing our use of an ANN as an emulator. Neural Networks require a set of training data to determine the weights used to predict the ANN outputs. We illustrate a set of training data in A) consisting of input - output pairs of parameter values and 21cm power spectra, which is passed to the network in B). Once the ANN is sufficiently trained, we use it to predict the value of the power spectrum at any given parameter set, illustrated by C).

2. Methods

Let $\boldsymbol{x}$ be a set of input parameters for a simulation, and suppose the output is given by $y = f(\boldsymbol{x})$, where f is expensive to compute. Emulating the simulation is based on an approximate calculation of $\tilde{y} = \tilde{f}(\boldsymbol{x})$, where $\tilde{f}$ is a fast approximation of the true simulation. For our purposes, we seek to emulate the calculation of the 21cm power spectrum in a number of specified k bins i.e. $y = \{P(k_i|\boldsymbol{\theta})\}$, where the subscript specifies the bin, for a restricted set of astrophysics parameters, $\boldsymbol{\theta} = (\zeta, T_{\rm vir}, R_{\rm mfp})$. We can then use our emulation $\tilde{P}(k_i|\boldsymbol{\theta})$ to make rapid evaluations of the likelihood,

$$\ln \mathcal{L} = \sum_i \frac{\left[P_{\rm obs}(k_i) - P(k_i|\boldsymbol{\theta})\right]^2}{2P_N(k_i)}, \tag{2.1}$$

with $P \to \tilde{P}$, where $P_{\rm obs}$ is an observed or mock data set, and P_N is the noise power spectrum associated with a specified instrument.

Here, we use a neural network as our emulation technique (see Figure 1 B). Using a set of training data, a neural network finds a mapping between input and output data, which is sensitive to the key features of the training set via a supervised learning algorithm. This mapping can then be used on unknown data where the neural network uses its acquired knowledge of the system to infer an output. A neural network consists of three types of layers each consisting of a set of nodes or neurons. The input layer takes N_i data points into N_i input nodes from which we want to predict some output. Each node in the input layer is connected to all of N_j nodes in the first of L hidden layers via some weight $w_{ij}^{(1)}$. The input to the nodes in the hidden layer is a linear combination of the input data and the weights,

$$s_j^{(1)} = \sum_{i=1}^{N_i} x_i w_{ij}^{(1)}. \tag{2.2}$$

A neuron is then activated by some activation function $g : \mathbb{R} \to \mathbb{R}$. We use a sigmoid activation function, $g(s) = 1/(1 + e^{-s})$. This activation step can be interpreted as each neuron having specialised on a certain feature in the system (Gal 2016) and when the data reflects this feature the neuron will be activated. The output from the neuron activation is then fed into the next hidden layer as input, such that the $j^{\rm th}$ neuron in the $l^{\rm th}$ hidden

https://doi.org/10.1017/S174392131700984X Published online by Cambridge University Press

layer, for $1 < l \leqslant L$, computes,

$$t_j^{(l)} = g\left(s_j^{(l)}\right), \text{ where } s_j^{(l)} = \sum_{i=1}^{N_j} t_i^{(l-1)} w_{ij}^{(l)}. \tag{2.3}$$

Finally, the output layer combines the outputs from the final hidden layer into N_k desired output values,

$$y_k = \sum_{i=1}^{N_j} t_i^{(L)} w_{ik}^{(L+1)}. \tag{2.4}$$

The weights between neurons $w_{ij}^{(l)}$ are obtained during the training of the network, where we apply the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) algorithm (Mcloone *et al.* 2002) to minimize the mean-square error between the true value provided by the training data and the value predicted by the network.

3. Analysis

We train our network on a variety of training sets obtained from evaluating 21cmFAST on a latin hypercube sample, as this guarantees homogeneous sampling of parameter space and allows for sensible scaling to higher dimensions. Using a mock observation, assuming a HERA331 design, we combine the predictions at redshifts 8, 9, and 10, to sample the likelihood and determine the error contours using both our ANN emulator and the brute force method of Greig & Mesinger (2015).

Similar to Kern *et al.* (2017), we find a significant speed-up for the parameter estimation. For our fiducial chain size, we observe a speed up by 3 orders of magnitude for the sampling of the likelihood by emulation compared to the brute-force method. In addition to the sampling, the Neural Network training requires on the order of ~ 1 minute for 100 training samples to ~ 1 hour for 10^4 training samples, using a commercial CPU, which is not needed when evaluating the model at each point. However, compared to the total runtime of 21CMMC the training time presents a minor factor.

Figure 2 shows the contraints obtained when combining observations in three redshift bins at $z = 8$, 9 and 10 for training sets of 100 and 1000 samples per redshift in panels A and B respectively. We find that our fiducial parameter values are well recovered by the 1σ confidence intervals. Although the contours for 100 training samples present a visibly worse fit, they too recover the fiducial values. As our recovered confidence regions are not gaussian, we compute the median and 68% confidence intervals around them, and compare them to those obtained from 21CMMC in table 1. We observe that errors retrieved by our network can be smaller than those obtained by 21CMMC, this is due to systematics. During the training period, our ANN constructs a model which approximates 21cmFAST and we proceed to sample the likelihood of the approximation. Therefore, any difference between the recovered 68% confidence intervals is most likely due to systematic difference between the two models that are sampled. We estimate that we are subject to these systematic effects on the 1% - 10% level for large to small training sets.

4. Conclusions

With a speed-up of ~ 3 orders of magnitude, 21cm power spectrum emulation can be used for a variety of new or existing analyses. By varying the experimental layout or survey strategy, we effectively vary the noise power spectrum $P_N(k)$ in Equation 2.1, and can thus fit the optimal layout or survey strategy, making 21cm power spectrum

https://doi.org/10.1017/S174392131700984X Published online by Cambridge University Press

Table 1. Median values and 68% confidence interval found in the parameter search via the brute-force method (21CMMC) and our ANN emulation. The fiducial parameter values are given by $(\zeta, R_{\rm mfp}, \log T_{\rm vir}) = (30, 15, 4.48)$.

Code - Training Set	z	ζ	$R_{\rm mfp}$	$\log T_{\rm vir}$
21CMMC	8,9,10	$31.08^{+8.70}_{-6.04}$	$15.15^{+2.86}_{-3.21}$	$4.51^{+0.17}_{-0.17}$
ANN - 100 LHS	8,9,10	$31.51^{+8.57}_{-6.32}$	$15.86^{+2.47}_{-3.62}$	$4.49^{+0.16}_{-0.19}$
ANN - 1000 LHS	8,9,10	$31.18^{+8.47}_{-6.08}$	$14.97^{+2.91}_{-3.78}$	$4.51^{+0.16}_{-0.17}$

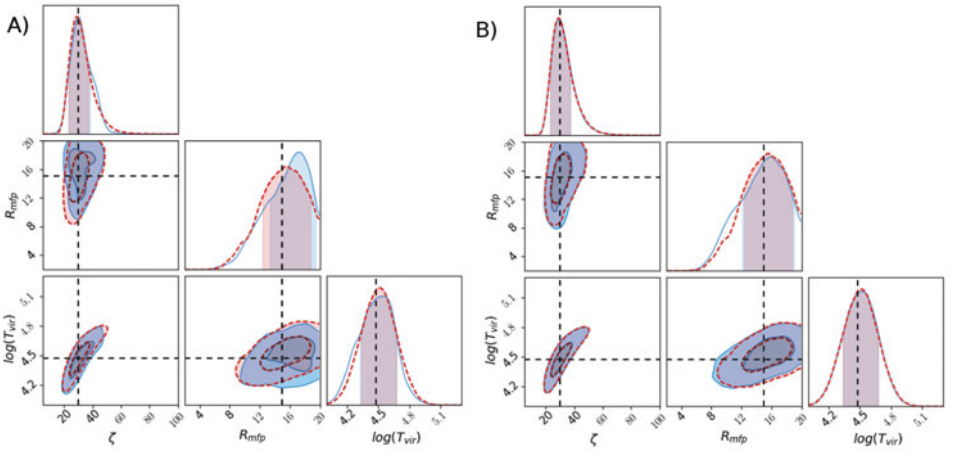

Figure 2. Comparisons between the recovered 1σ and 2σ confidence regions of 21CMMC (red dashed lines) and our ANN emulator (blue solid lines) combining redshifts $z = 8$, $z = 9$, and $z = 10$. Panel A shows the recovered contours for an ANN trained on 100 training sets sampled from a latin hypercube. Panel B shows the results when training the emulator on 1000 latin hypercube samples. The dashed vertical and horizontal lines indicate the true parameter values of the mock observation, $(\zeta, R_{\rm mfp}, \log T_{\rm vir}) = (30, 15, 4.48)$.

emulators useful for experimental design studies. Further, finding accurate results for as few as 100 training samples may open up the possibility to move away from semi-numerical models such as 21cmFAST and for the first time use radiative transfer codes in EoR parameter searches (eg. Iliev *et al.* 2006). Semelin *et al.* (2017) have produced a data base of 45 evaluations of their fully numerical model which could be used as a training set for an emulator analysis. Finally, Bayesian model comparison analyses could benefit greatly from fast model evaluations via ANN emulation.

References

Greig, B. & Mesinger, A. 2015, *MNRAS*, 449, 4246
Shimabukuro, H. & Semelin, B. 2017, *MNRAS*, 468, 3869
Mesinger, A. & Furlanetto, S. 2007, *ApJ*, 669, 663
Schmit, C. J. & Pritchard, J. R. 2017, *preprint*, arxiv:1708.00011
Gal, Y. 2016, *PhD Thesis*, University of Cambridge
Mcloone, S. F., Asirvadam, V. S., & Irwin, G. W. 2002, *IEEE Int. Conf. Neural Networks*, 2, 513
Kern, N. S., Liu, A., Parsons, A. R., Mesinger, A., & Greig, B. 2017, *ApJ*, 848, 23
Iliev, I. T., Mellema, G., Pen, U. L., Merz, H., Shapiro, P. R., & Alvarez, M. A. 2006, *MNRAS*, 369, 1625
Semelin, B., Eames, E., Bolgar F., & Caillat M. 2017, *MNRAS*, 472, 4508

https://doi.org/10.1017/S174392131700984X Published online by Cambridge University Press