Statistical Challenges in 21st Century Cosmology
Proceedings IAU Symposium No. 306, 2014
A. F. Heavens, J.-L. Starck & A. Krone-Martins, eds.

doi:10.1017/S1743921314013672

Machine-learning in astronomy

Michael Hobson[1], Philip Graff[2], Farhan Feroz[1] and Anthony Lasenby[1]

[1]Astrophysics Group, Cavendish Laboratory, J.J. Thomson Avenue, Cambridge, CB3 0HE, UK
email: mph@mrao.cam.ac.uk, ff235@mrao.cam.ac.uk, anthony@mrao.cam.ac.uk

[2]Gravitational Astrophysics Laboratory, NASA Goddard Space Flight Center, 8800 Greenbelt Rd., Greenbelt, MD 20771, USA
email: philip.b.graff@nasa.gov

Abstract. Machine-learning methods may be used to perform many tasks required in the analysis of astronomical data, including: data description and interpretation, pattern recognition, prediction, classification, compression, inference and many more. An intuitive and well-established approach to machine learning is the use of artificial neural networks (NNs), which consist of a group of interconnected nodes, each of which processes information that it receives and then passes this product on to other nodes via weighted connections. In particular, I discuss the first public release of the generic neural network training algorithm, called SkyNet, and demonstrate its application to astronomical problems focusing on its use in the BAMBI package for accelerated Bayesian inference in cosmology, and the identification of gamma-ray bursters. The SkyNet and BAMBI packages, which are fully parallelised using MPI, are available at http://www.mrao.cam.ac.uk/software/.

Keywords. methods: data analysis, methods: statistical, cosmological parameters, gamma rays: bursts

1. Introduction

In astrophysics and cosmology, one is faced with analysing large, complicated and multidimensional data sets. Such analyses typically include tasks such as: data description and interpretation, pattern recognition, prediction, classification, compression, inference and many more. One way of performing such tasks is through the use of machine-learning methods (see, e.g., MacKay (2003), Ball & Brunner (2010) and Way *et al.* (2012)).

In supervised learning, the goal is to infer a function from labeled training data, which consist of a set of training examples. Each example has known 'input' quantities whose values are to be used to predict the values of the 'outputs'. Thus, the function to be inferred is the mapping from input to outputs. Once learned, this mapping can be applied to datasets for which the values of the outputs are not known. Supervised learning is usually further subdivided into *classification* and *regression*.

An intuitive and well-established approach to machine learning is based on the use of artificial neural networks (NNs), which are loosely inspired by the structure and functional aspects of a brain. They consist of a group of interconnected nodes, each of which processes information that it receives and then passes this product on to other nodes via weighted connections. In this way, NNs constitute a non-linear statistical data modeling tool, which may be used to model complex relationships between a set of inputs and outputs. Many machine-learning applications can be performed using only feed-forward NNs: an input layer of nodes passes information to an output layer via zero, one, or many 'hidden' layers in between. Moreover, a universal approximation theorem Hornik *et al.* (1990) assures us that we can approximate any reasonable mapping with a NN of a given form. A useful introduction to NNs can be found in MacKay (2003).

In astronomy, feed-forward NNs have been applied to various machine-learning problems for over 20 years (see, e.g., Way *et al.* (2012), Tagliaferri *et al.* (2003)). Nonetheless, their more widespread use in astronomy has been limited by the difficulty associated with standard techniques, such as backpropagation, in training networks having many nodes and/or numerous hidden layers (i.e. 'large' and/or 'deep' networks), which are often necessary to model the complicated mappings between numerous inputs and outputs in modern astronomical applications. We therefore introduce SkyNet Graff *et al.* (2014), an efficient and robust neural network training algorithm that is capable of training large and/or deep feed-forward networks.

An important recent application of regression supervised learning in astrophysics and cosmology is the acceleration of the Bayesian analysis (both parametet estimation and model selection) of large data sets in the context of complicated models. At each point in parameter space, Bayesian methods require the evaluation of a 'likelihood' function describing the probability of obtaining the data for a given set of model parameters. For some problems each such function evaluation may take up to tens of seconds. Substantial gains in performance can thus be achieved if one is able to speed up the evaluation of the likelihood, and a NN is ideally suited for this task. The blind accelerated multimodal Bayesian inference (BAMBI) algorithm Graff *et al.* (2012) uses SkyNet for training such NNs, and combines them with a nested sampling Skilling (2004) approach that efficiently calculates the Bayesian evidence (also referred to as the marginal likelihood) for model selection and produces samples from the posterior distribution for parameter estimation. It particular, BAMBI employs the MultiNest algorithm Feroz & Hobson (2008), Feroz *et al.* (2009), Feroz *et al.* (2013), which is a generic implementation of nested sampling, extended to handle multimodal and degenerate distributions, and is fully parallelised.

This paper will discuss results from applying SkyNet within the BAMBI algorithm to accelerate Bayesian inference in cosmology, and also to the identification of gamma-ray bursters.

2. Network structure

A multilayer perceptron feed-forward neural network is the simplest type of network and consists of ordered layers of perceptron nodes that pass scalar values from one layer to the next. The perceptron is the simplest kind of node, and maps an input vector $\mathbf{x} \in \Re^n$ to a scalar output $f(\mathbf{x}; \mathbf{w}, \theta)$ via

$$f(\mathbf{x}; \mathbf{w}, \theta) = \theta + \sum_{i=1}^{n} w_i x_i, \tag{2.1}$$

where $\mathbf{w} = \{w_i\}$ and θ are the parameters of the perceptron, called the 'weights' and 'bias', respectively. In a feed-forward NN, the value at each node is computed by applying an 'activation function', g, to the scalar value calculated in Equation 2.1. The activation function used for nodes in hidden layers is $g(x) = 1/(1 + e^{-x}) = \mathrm{sig}(x)$, the sigmoid function. For output layer nodes, $g(x) = x$. The non-linearity of g for the hidden layers is essential to allowing the network to model non-linear functions. The weights and biases are the values we wish to determine in our training (described in Section 3). As they vary, a huge range of non-linear mappings from inputs to outputs is possible. By increasing the number of hidden nodes, we can achieve more accuracy at the risk of overfitting to our training data.

3. Network training

In training a NN, we wish to find the optimal set of network weights and biases that maximise the accuracy of predicted outputs. However, we must be careful to avoid over-

https://doi.org/10.1017/S1743921314013672 Published online by Cambridge University Press

fitting to our training data at the expense of making accurate predictions for input values on which the network has not been trained. The set of training data inputs and outputs, $\mathcal{D} = \{\mathbf{x}^{(k)}, \mathbf{t}^{(k)}\}$, is provided by the user. Approximately 75 per cent should be used for actual NN training and the remainder retained as a validation set that will be used to determine convergence and to avoid overfitting. This ratio of 3:1 gives plenty of information for training but still leaves a representative subset of the data for checks to be made.

3.1. *Network objective function*

Let us denote the network weights and biases collectively by the vector $\mathbf{a}$. SkyNet considers the parameters $\mathbf{a}$ to be random variables with a posterior distribution given by the likelihood multiplied by the prior. The likelihood, $\mathcal{L}(\mathbf{a}; \boldsymbol{\sigma})$, encodes how well the NN, characterised by a given set of parameters $\mathbf{a}$, is able to reproduce the known training data outputs. This is modulated by the prior $\mathcal{S}(\mathbf{a}; \alpha)$, which is assumed to have the form of a Gaussian centered at the origin with multiplicative factor α inside the exponential. α determines the relative importance of the prior and the likelihood. The prior here acts as a regularization function; in a full Bayesian treatment it may have a different functional form.

The form of the likelihood depends on the type of network being trained. For regression problems, SkyNet assumes a log-likelihood function for the network parameters $\mathbf{a}$ given by the standard χ^2 misfit function, but including hyperparameters $\boldsymbol{\sigma}$ that describe the standard deviation (error size) of each of the outputs. For classification problems, SkyNet again uses continuous outputs (rather than discrete ones), which are interpreted as the probabilities that a set of inputs belongs to a particular output class. This is achieved by applying the *softmax* transformation to the output values, so that they are all non-negative and sum to unity. The classification likelihood is then given by the *cross-entropy* of the targets and softmaxed output values MacKay (2003).

3.2. *Initialisation and pre-training*

The training of the NN can be started from some random initial state, or from a state determined from a 'pre-training' procedure. In the former case, the network training begins by setting random values for the network parameters, sampled from a normal distribution with zero mean and variance of 0.01. SkyNet can also make use of the pre-training approach developed by Hinton *et al.* (2006), Hinton & Salakhutdinov (2006), which obtains a set of network weights and biases close to a good solution of the network objective function.

3.3. *Optimisation of the objective function*

Once the initial set of network parameters have been obtained, either by assigning them randomly or through pre-training, the network is then trained by iterative optimisation of the objective function. NN training proceeds using an adapted form of a truncated Newton (or 'Hessian-free' Martens (2010)) optimisation algorithm, to calculate the step, $\delta\mathbf{a}$, that should be taken at each iteration. The required step is obtained using a stable and efficient procedure applicable to NNs that avoids explicit calculation and calculation of the Hessian Schraudolph (2002), Pearlmutter (1994). Following each such step, adjustments to α and $\boldsymbol{\sigma}$ may be made before another step is calculated (using formulas derived from the MemSys software package Gull & Skilling (1999)). The combination of the above methods makes practical the use of second-order derivative information even for large networks.

3.4. *Convergence*

Following each iteration of the optmisation algorithm, the likelihood, posterior, correlation, and error squared values are calculated both for the training data and for the

https://doi.org/10.1017/S1743921314013672 Published online by Cambridge University Press

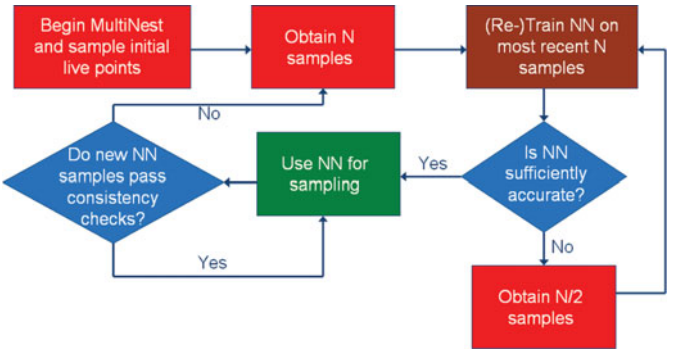

Figure 1. A flowchart depicting the transitions between sampling and NN training within BAMBI. N is given by `updInt` from MultiNest.

validation data. When these values begin to diverge and the predictions on the validation data no longer improve, it is deemed that the training has converged and optimization is terminated. This helps to prevent excessive over-fitting to the training data.

4. SkyNet Regression and BAMBI: learning likelihoods

The BAMBI algorithm combines neural networks with nested sampling. After a specified number of new samples from MultiNest have been obtained, BAMBI uses these to train a regression network on the log-likelihood function using the SkyNet algorithm in order to perform likelihood interpolation. Using the network reduces the log-likelihood evaluation time from seconds to microseconds, allowing MultiNest to complete analysis much more rapidly. The user also obtains a network or set of networks that are trained to easily and quickly provide more log-likelihood evaluations near the peak if needed, or in subsequent analyses. Other methods exist for likelihood interpolation and extrapolation, as well as the fast estimation of the Bayesian evidence, best-fit parameters and errors. See Higden *et al.* (2012), Rasmussen (2003), Bliznyuk *et al.* (2008) and references therein for more information.

4.1. *The structure of BAMBI*

The flow of sampling and training within BAMBI is demonstrated by the flowchart given in Figure 1. After convergence to the optimal NN weights, we test that the network is able to predict likelihood values to within a specified tolerance level. If not, sampling continues using the original log-likelihood until enough new samples have been made for training to be resumed. Once a network is trained that is sufficiently accurate, its predictions are used in place of the original log-likelihood function for future samples in MultiNest. Consistency checks are made to ensure the NN is not making predictions outside the range of data on which it was trained.

The optimal network possible with a given set of training data may not be able to predict log-likelihood values accurately enough, so an additional criterion is placed on when to use the trained network. This requirement is that the RMSE of log-likelihoods predictions is less than a user-specified tolerance, `tol`. When the trained network does not pass this test, then BAMBI will continue using the original log-likelihood function to replace the older half of the samples to generate a new training data set. Network training will then resume, beginning with the weights that it had found as optimal for the previous data set. Since samples are generated from nested contours and each new data set contains half of the previous one, the saved network will already be able to produce reasonable predictions on this new data.

https://doi.org/10.1017/S1743921314013672 Published online by Cambridge University Press

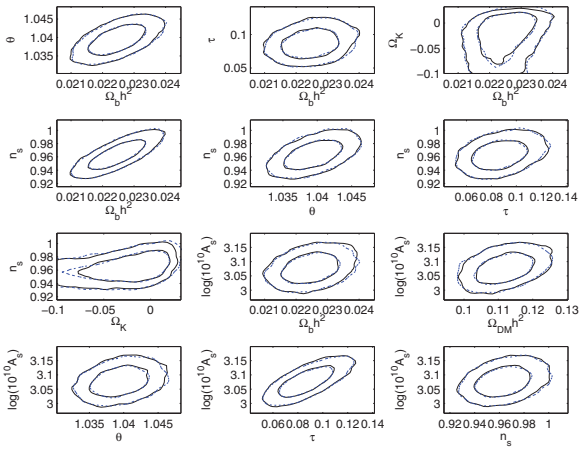

Figure 2. Marginalised 2D posteriors for the non-flat model (ΛCDM+Ω_K) using only CMB data. The 12 most correlated pairs are shown. MultiNest is in solid black, BAMBI in dashed blue. Inner and outer contours represent 68% and 95% confidence levels, respectively.

Once a NN is in use in place of the original log-likelihood function its evaluations are taken to be the true log-likelihoods. Checks are made to ensure that the network is maintaining its accuracy and will re-train when these fail. To re-train the network, BAMBI first substitutes the original log-likelihood function back in and gathers the required number of new samples from MultiNest. Training then commences, resuming from the previously saved network. These criteria ensure that the network is not trusted too much when making predictions beyond the limits of the data it was trained on.

4.2. *Accelerated cosmology inference using BAMBI*

We implement BAMBI within the standard CosmoMC code Lewis & Bridle (2002). Bayesian parameter estimation in cosmology requires evaluation of theoretical temperature and polarisation CMB power spectra (C_l values) using codes such as CAMB Lewis *et al.* (2000). These evaluations can take on the order of tens of seconds depending on the cosmological model. The C_l spectra are then compared to observations. Considering that thousands of these evaluations will be required, this is a computationally expensive step and a limiting factor in the speed of any Bayesian analysis. BAMBI has the benefit of not requiring a pre-computed sample of points as in CosmoNet Auld *et al.* (2008a), Auld *et al.* (2008b) InterpMC Bouland *et al.* (2011), PICO Fendt & Wandelt (2007), and others, which is particularly important when including new parameters or new physics in the model.

We use a standard set of eight cosmological parameters, each with a uniform prior distribution. A non-flat cosmological model incorporates all of these parameters, while we set $\Omega_K = 0$ for a flat model. We use $w = -1$ in both cases. The flat model thus represents the standard ΛCDM cosmology. We use two different data sets for analysis: (1) CMB observations alone and (2) CMB observations plus Hubble Space Telescope constraints on H_0, large-scale structure constraints from the luminous red galaxy subset of the SDSS and the 2dF survey, and supernovae Ia data.

Analyses with MultiNest and BAMBI were run on all four combinations of models and data sets. In Figure 2 we show the recovered two-dimensional marginalised posterior probability distributions for the non-flat model using the CMB-only data set. We see very close agreement between MultiNest (in solid black) and BAMBI (in dashed blue) across all parameters. Using the full data set one finds similar correspondence in the

https://doi.org/10.1017/S1743921314013672 Published online by Cambridge University Press

Table 1. Time per likelihood evaluation in a follow-up analysis and speed-up factors with respect to MultiNest.

Method	Model	Data	$t_{\mathcal{L}}$ (ms)	factor
MultiNest	flat	CMB	2775	—
		all	3813	—
	non-flat	CMB	12830	—
		all	10980	—
BAMBI	flat	CMB	0.2077	13360
		all	0.2146	17770
	non-flat	CMB	0.08449	151900
		all	0.2032	54040

posterior probability contours. The posterior probability distributions for the flat model with either data set are extremely similar to those of the non-flat flodel with setting $\Omega_K = 0$, as expected. Moreover, the estimated log-evidences obtained by MultiNest and BAMBI for each case are consistent to within the (chosen) statistical error of 0.1 units.

An important comparison is the running time required. The analyses were run using MPI parallelisation on 48 processors. We recorded the time required for the complete analysis, not including any data initialisation prior to initial sampling. We then divide this time by the number of likelihood evaluations performed to obtain an average time per likelihood. Therefore, time required to train the NN is still counted as a penalty factor. If a NN takes more time to train, this will hurt the average time, but obtaining a usable NN sooner and with fewer training calls will give a better time since more likelihoods will be evaluated by the NN. We are able to obtain a significant decrease in running time of order 40%, while adding in the bonus of having a NN trained on the likelihood function.

A major benefit of BAMBI is that following an initial run the user is provided with a trained NN, or multiple ones, that model the log-likelihood function. These can be used in any subsequent analysis to obtain much faster results. This is a comparable analysis to that of CosmoNet Auld *et al.* (2008a), Auld *et al.* (2008b), except that the NNs here are a product of an initial Bayesian analysis where the peak of the distribution was *not* previously known. No prior knowledge of the structure of the likelihood surface was used to generate the networks that are now able to be re-used. When multiple NNs are trained and used in the initial BAMBI analysis, we must determine which network's prediction to use in the follow-up. We decide this by using a computationally cheap method of estimating the error on predictions suggested by MacKay (1995).

To demonstrate the speed-up potential of using the NNs in follow-up analyses, we repeated each of the four analyses above, but set the prior ranges to be uniform over the region defined by $\mathbf{x}_{\max(\log(\mathcal{L}))} \pm 4\boldsymbol{\sigma}$, where $\boldsymbol{\sigma}$ is the vector of standard deviations of the marginalised one-dimensional posterior probabilities. Table 1 shows the average time taken per log-likelihood function evaluation when this follow-up was performed with MultiNest and with BAMBI. In all cases, the BAMBI evidences matched the MultiNest values to within statistical error and the posteriors agreed closely. We can thus obtain accurate posterior distributions and evidence calculations orders of magnitude faster than originally possible.

5. SkyNet Classification: identifying gamma-ray bursters

Long gamma-ray bursts (GRBs) are almost all indicators of core-collapse supernovae from the deaths of massive stars. The ability determine the intrinsic rate of these events

https://doi.org/10.1017/S1743921314013672 Published online by Cambridge University Press

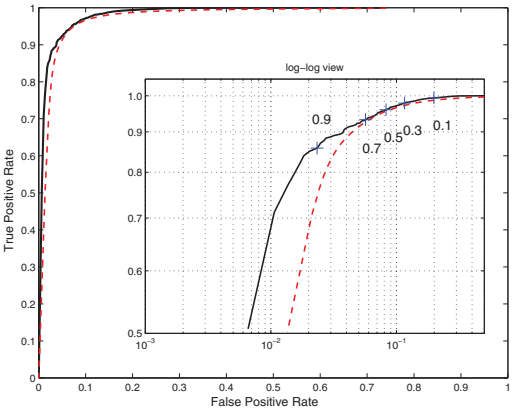

Figure 3. Actual (black solid) and expected (dashed red) ROC curves for a NN classifier that predicts whether a GRB will be detected by Swift.

as a function of redshift is essential for studying numerous aspects of stellar evolution and cosmology. The Swift space telescope is a multi-wavelength detector that is currently observing hundreds of GRBs Gehrels *et al.* (2004). However, Swift uses a complicated combination of over 500 triggering criteria for identifying GRBs, which makes it difficult to infer the intrinsic GRB rate.

To investigate this issue, a recent study by Lien *et al.* (2012) performed a Monte Carlo analysis that generated a mock sample of GRBs, using the GRB rate and luminosity function of Wanderman & Piran (2010), and processed them through an entire simulated Swift detection pipeline.

5.1. *Form of the classification problem*

Our goal here is to replace the simulated Swift trigger pipeline with a classification NN, which (as we will see) can determine in just a few microseconds whether a given GRB is detected. We use as training data a pre-computed mock sample of 10, 000 GRBs from Lien *et al.* (2012). In particular, we divide this sample randomly into ∼4000 for training, ∼1000 for validation, and the final ∼5000 as a blind test set on which to perform our final evaluations. For each GRB we have 13 inputs that describe the GRB and how it is seen by the detector. The two outputs correspond to the probabilities that the GRB is or is not detected.

5.2. *Results*

We can investigate the quality of the classification as a function of a threshold probability, $p_{\rm th}$, required to claim a detection. As discussed in Feroz *et al.* (2008), we can compute the *expected* number of total GRB detections, correct detections, and false detections, as well as other derived statistics as a function of $p_{\rm th}$, *without* knowing the true classification. From these, we can compute the completeness ϵ (fraction of detected GRBs that have been correctly classified; also referred to as the efficiency) and purity τ (fraction of all GRBs correctly classified as detected).

Values of the actual and expected completeness and purity are nearly identical. Thus, without knowing the true classifications of the GRBs as detected or not, we can set $p_{\rm th}$ to obtain the desired completeness and purity levels for the final sample.

With this information, we can also plot the actual and expected receiver operating characteristic (ROC) curves (see, e.g., Fawcett (2006)). The ROC curve plots the true positive rate (identical to completeness) against the false positive rate (also known as

https://doi.org/10.1017/S1743921314013672 Published online by Cambridge University Press

contamination). In general, the larger the area underneath a ROC curve, the more powerful the classifier. From Figure 3, we can see that the expected and actual ROC curves for a NN classifier are very close, with deviations occuring only at very low false positive rates. We conclude that $p_{\rm th} = 0.5$, the original naive choice, is a near-optimal threshold value. The curves also indicate that this test is quite powerful at predicting which GRBs will be detected by Swift.

Using $p_{\rm th} = 0.5$, we now wish to determine how well the GRB detection rate with respect to redshift is reproduced, since this relationship is key to deriving scientific results. The detected GRB event counts as a function of redshift for both our NN classifier and the original Swift pipeline agree very well. When histogrammed, the error between the counts differs by less than the Poissonian counting uncertainty on the true values inherent in this type of process.

6. Conclusion

We present an efficient and robust neural network training algorithm, called SkyNet. This generic tool is capable of training large and deep feed-forward networks and may be applied to machine-learning tasks in astronomy. We describe the application of SkyNet to the regression problem of learning likelihoods, in combination with MultiNest to create the BAMBI algorithm for accelerated Bayesian inference. This is a generic tool and requires no pre-processing. We apply BAMBI to the problem of cosmological parameter estimation and model selection. By calculating a significant fraction of the likelihood values with the NN instead of the full function, we are able to reduce the running time by $\sim 40\%$ on an initial run, and subsequent analyses using the resulting trained networks enjoy speed-up factors of $O(10^{4-5})$. We also demonstrate the application of SkyNet to the classification problem of identifying gamma-ray bursters.

References

Auld, T., Bridges, M., Hobson, M. P., & Gull, S. F. 2008, *MNRAS*, 376, L11
Auld, T., Bridges, M., & Hobson, M. P. 2008, *MNRAS*, 387, 1575
Ball, N. M. & Brunner, R. J. 2010, *Int. J. Mod. Phys.*, 19, 1049
Bliznyuk, N., *et al.* 2008, *J. Comput. Graph. Statist.*, 17, 270
Bouland, A., Easther, R., & Rosenfeld, K. 2011, *J. Cosmol. Astropart. Phys.*, 5, 016
Fawcett, T. 2006, *Pattern Recogn. Lett.*, 27, 861
Fendt, W. A. & Wandelt B. D. 2007, *ApJ*, 654, 2
Feroz, F. & Hobson, M. P. 2008, *MNRAS*, 384, 449
Feroz, F., Marshall, P. J., & Hobson M. P. 2008, arXiv:0810.0781 [astro-ph]
Feroz, F., Hobson, M. P., & Bridges, M. 2009, *MNRAS*, 398, 1601
Feroz, F., Hobson, M. P., Cameron, E., & Pettitt A. N. 2013, arXiv:1306.2144 [astro-ph.IM]
Gehrels, N., *et al.* 2004, *ApJ*, 611, 1005
Graff, P. Feroz, F., Hobson, M. P., & Lasenby, A. N. 2014, *MNRAS*, 441, 1741
Graff, P., Feroz, F., Hobson, M. P., & Lasenby, A. N. 2012, *MNRAS*, 421, 169
Gull, S. F. & Skilling, J. 1999, *Quantified Maximum Entropy: MemSys5 Users' Manual* (Maximum Entropy Data Consultants Ltd. Bury St. Edmunds, Suffolk, UK. `http://www.maxent.co.uk/`)
Higdon, D., Lawrence, E., Heitmann, K., & Habib S. 2012, in: E.D. Feigelson & G.J. Babu (eds.), *Statistical Chanllenges in Modern Astronomy V* (New York: Springer), p. 41
Hinton, G. E., Osindero, S., & Teh, Y.-W. 2006, *Neural Comput.*, 18, 1527
Hinton, G. E. & Salakhutdinov, R. R. 2006, *Science*, 313, 504
Hornik, K., Stinchcombe, M., & White, H. 1990, *Neural Networks*, 3, 359
Lewis, A. & Bridle, S. 2002, *Phys. Rev. D*, 66, 103511,

https://doi.org/10.1017/S1743921314013672 Published online by Cambridge University Press

Lewis, A., Challinor, A., & Lasenby, A. N. 2000, *ApJ*, 538, 473
Lien, A., Sakamoto, T., Gehrels, N., Palmer, D., & Graziani, C. 2012, *Proceedings of the International Astronomical Union*, 279, 347
MacKay, D. J. C. 1995, *Network: Computation in Neural Systems*, 6, 469
&MacKay, D. J. C., 2003, *Information Theory, Inference, and Learning Algorithms* (Cambridge: CUP)
Martens, J. 2010, in: J. Fürnkranz & T. Joachims (eds.), *Proc. 27th Int. Conf. Machine Learning* (Haifa: Omnipress), p. 735
Pearlmutter, B. A. 1994, *Neural Comput.*, 6, 147
Rasmussen, C. E. 2003, in: J.M. Bernardo, M.J. Bayarri, J.O. Berger, A.P. Dawid, D. Heckerman, A.F.M. Smith, & M. West (eds.), *Bayesian statistics 7* (New York: OUP), p. 651
Schraudolph, N. N. 2002, *Neural Comput.*, 14, 1723
Skilling, J. 2004, *AIP Conference Series*, 735, 395
Tagliaferri, R. *et al.* 2003 *Neural Networks*, 16, 297
Wanderman, D. & Piran, T. 2010, *MNRAS*, 406, 1944
Way, M. J., Scargle, J. D., Ali, K. M., & Srivastava, A. N. 2012, *Advances in Machine Learning and Data Mining for Astronomy* (CRC Press)

https://doi.org/10.1017/S1743921314013672 Published online by Cambridge University Press