2013 IEEE 13th International Conference on Data Mining Workshops

Neural Networks for Astronomical Data Analysis and Bayesian Inference

Philip Graff
Gravitational Astrophysics Laboratory
NASA Goddard Space Flight Center
8800 Greenbelt Rd.
Greenbelt, MD 20771, USA

Farhan Feroz
Astrophysics Group
Cavendish Laboratory
JJ Thomson Avenue
Cambridge CB3 0HE, UK

Michael P. Hobson
Astrophysics Group
Cavendish Laboratory
JJ Thomson Avenue
Cambridge CB3 0HE, UK

Anthony Lasenby
Kavli Institute for Cosmology
Madingley Road
Cambridge CB3 0HA, UK

***Abstract*—We present our generic neural network training algorithm, called SkyNet and the accelerated Bayesian inference algorithm, BAMBI. SkyNet combines multiple techniques already developed individually in the literature to create an efficient and robust machine-learning tool that is able to train large and deep feed-forward neural networks for use in a wide range of learning applications, such as regression, classification, density estimation, clustering and dimensionality reduction. SkyNet uses a powerful 'pre-training' method, to obtain a set of network parameters close to the true global maximum of the training objective function, followed by further optimisation using an automatically-regularised variant of Newton's method that uses second-order derivative information to improve convergence, but without the need to evaluate or store the full Hessian matrix, by using a fast approximate method to calculate Hessian-vector products. This combination of methods allows for the training of complicated networks that are difficult to optimise using standard backpropagation techniques. The blind accelerated multimodal Bayesian inference (BAMBI) algorithm implements the MultiNest package for nested sampling as well as the training of an artificial neural network by SkyNet to learn the likelihood function. In the case of computationally expensive likelihoods, this allows the substitution of a much more rapid approximation in order to increase significantly the speed of the analysis. Astrophysical examples are provided for both SkyNet and BAMBI.**

I. Introduction

In modern astrophysics and cosmology, one is increasingly faced with the problem of analysing large, complicated and multidimensional data sets. Such analyses typically include tasks such as: data description and interpretation, inference, pattern recognition, prediction, classification, compression, and many more. One way of performing such tasks is through the use of machine-learning methods. For accessible accounts of machine-learning and its use in astronomy, see, for example, [1], [2] and [3]. Moreover, machine-learning software for astronomy, such as the Python-based ASTROML package[1], has recently started to become available.

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 both 'properties' and 'labels'. The properties are known 'input' quantities whose values are to be used to predict the values of the labels, which may be considered as 'output' quantities. Thus, the function to be inferred is the mapping from properties to labels. Once learned, this mapping can be applied to datasets for which the values of the labels are not known. Supervised learning is usually further subdivided into *classification* and *regression*. In classification, the labels take discrete values, whereas in regression the labels are continuous. One atronomical example uses multifrequency observations of a supernova lightcurve to determine its type (e.g. Ia, Ib, II, etc.); this is a classification problem since the label (supernova type) is discrete (see, e.g., [4]).

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, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables. In general, the structure of a NN can be arbitrary, but many machine-learning applications can be performed using only feed-forward NNs, for which the structure is directed: an input layer of nodes passes information to an output layer via zero, one, or many 'hidden' layers in between. Such a network is able 'learn' a mapping between inputs and outputs, given a set of training data, and can then make predictions of the outputs for new input data. Moreover, a universal approximation theorem assures us that we can accurately and precisely approximate the mapping with a NN of a given form. A useful introduction to NNs can be found in [1].

In astronomy, feed-forward NNs have been applied to various machine-learning problems for over 20 years (see, e.g., [3], [5]). 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 the numerous inputs and outputs in modern astronomical applications. In this

[1]`http://astroml.github.com/`

presentation, we therefore introduce SKYNET [6], an efficient and robust neural network training algorithm that is capable of training large and/or deep feed-forward networks.

Bayesian methods of inference are widely used in astronomy and cosmology and are gaining popularity in other fields, such as particle physics. They can generally be divided into the performance of two main tasks: parameter estimation and model selection. Nested sampling [7] is a method of Monte Carlo sampling designed for efficient calculation of the Bayesian evidence (also referred to as the marginal likelihood) which also provides samples from the posterior distribution as a by-product, thus allowing parameter estimation at no additional cost. The MULTINEST algorithm [8], [9], [10] is a generic implementation of nested sampling, extended to handle multimodal and degenerate distributions, and is fully parallelised.

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 cosmological and particle physics problems each such function evaluation takes up to tens of seconds. MULTINEST is able to efficiently sample, but further gains can be achieved if we are able to speed up the evaluation of the likelihood itself. A NN is ideally suited for this task. The blind accelerated multimodal Bayesian inference (BAMBI) algorithm [11] combines these two elements. After a specified number of new samples from MULTINEST have been obtained, BAMBI uses these to train a network on the likelihood function with the SKYNET algorithm. After convergence to the optimal 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 likelihood until enough new samples have been made for training to be done again. Once a network is trained that is sufficiently accurate, its predictions are used in place of the original likelihood function for future samples for MULTINEST. Using the network reduces the likelihood evaluation time from seconds to milliseconds, allowing MULTINEST to complete the analysis much more rapidly. As a bonus, the user also obtains a network that is trained to easily and quickly provide more likelihood evaluations near the peak if needed, or in subsequent analyses.

This presentation will discuss results from applying SKYNET within the BAMBI algorithm to the accelerate solving the problem of cosmological parameter estimation.

II. 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

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

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 (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 III). As they vary, a huge range of non-linear mappings from inputs to outputs is possible. In fact, a universal approximation theorem [12] states that a NN with three or more layers can approximate any continuous function to a given accuracy, as long as the activation function is locally bounded, piecewise continuous, and not a polynomial (hence our use of sigmoid g, although other functions would work just as well, such as $\tanh$). By increasing the number of hidden nodes, we can achieve more accuracy at the risk of overfitting to our training data.

III. 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 overfitting 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.

A. 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.

1) Regression likelihood: For regression problems, SKYNET assumes a log-likelihood function for the network parameters $\mathbf{a}$ given by the standard χ^2 misfit function

$$\log \mathcal{L}(\boldsymbol{a};\boldsymbol{\sigma}) = -\frac{K\log(2\pi)}{2} - \sum_{i=1}^{N} \log(\sigma_i) - \frac{1}{2}\sum_{k=1}^{K}\sum_{i=1}^{N}\left[\frac{t_i^{(k)} - y_i(\boldsymbol{x}^{(k)};\boldsymbol{a})}{\sigma_i}\right]^2, \quad (2)$$

where N is the number of outputs, K is the number of training data examples and $\boldsymbol{y}(\boldsymbol{x}^{(k)};\boldsymbol{a})$ is the NN's predicted output vector for the input vector $\boldsymbol{x}^{(k)}$ and network parameters $\boldsymbol{a}$. The hyperparameters $\boldsymbol{\sigma} = \{\sigma_i\}$ describe the standard deviation (error size) of each of the outputs.

2) Classification likelihood: 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, namely

$$y_i(\mathbf{x}^{(k)};\mathbf{a}) \to \frac{\exp[y_i(\mathbf{x}^{(k)};\mathbf{a})]}{\sum_{j=1}^{N}\exp[y_j(\mathbf{x}^{(k)};\mathbf{a})]}. \quad (3)$$

The classification likelihood is then given by the *cross-entropy* of the targets and softmaxed output values [1],

$$\log \mathcal{L}(\boldsymbol{a};\boldsymbol{\sigma}) = -\sum_{k=1}^{K}\sum_{i=1}^{N} t_i^{(k)} \log y_i(\boldsymbol{x}^{(k)};\boldsymbol{a}). \quad (4)$$

In this scenario, the true and predicted output values are both probabilities (which lie in $[0,1]$). For the true outputs, all are zero except for the correct output which has a value of unity. For classification networks, the $\boldsymbol{\sigma}$ hyper-parameters do not appear in the log-likelihood.

B. 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 [13], which obtains a set of network weights and biases close to the global optimum of the network objective function. This method was originally devised with unsupervised learning (training data without labels) in mind but can be applied to general feed-forward networks as well. More details can be found in [13] and [14] has useful diagrams and explanations.

C. 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' [15]) optimisation algorithm, to calculate the step, $\delta\mathbf{a}$, that should be taken at each iteration. 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 [16]).

To obtain the step $\delta\mathbf{a}$ at each iteration, we first note that one may approximate a general function f up to second-order in its Taylor expansion by

$$f(\mathbf{a}+\delta\mathbf{a}) \approx f(\mathbf{a}) + \mathbf{g}^{\mathrm{T}}\delta\mathbf{a} + \tfrac{1}{2}(\delta\mathbf{a})^{\mathrm{T}}\mathsf{B}\,\delta\mathbf{a}, \quad (5)$$

where $\mathbf{g} = \nabla f(\mathbf{a})$ is the gradient and $\mathsf{B} = \nabla\nabla f(\mathbf{a})$ is the Hessian matrix of second derivatives, both evaluated at $\mathbf{a}$. For our purposes, the function f is the log-posterior distribution of the NN parameters and hence (5) represents a Gaussian approximation to the posterior. The Hessian of the log-posterior is the regularised Hessian of the log-likelihood function, where the prior provides the regularisation. If we define the Hessian matrix of the log-likelihood as H, then $\mathsf{B} = \mathsf{H} + \alpha\mathsf{I}$, where I is the identity matrix.

Ideally, we seek a step $\delta\mathbf{a}$, such that $\nabla f(\mathbf{a}+\delta\mathbf{a}) = 0$. Using the approximation (5), one thus requires

$$\mathsf{B}\,\delta\mathbf{a} = -\mathbf{g}. \quad (6)$$

In the standard Newton's method of optimisation one simply solves this equation directly for $\delta\mathbf{a}$ to obtain

$$\delta\mathbf{a} = -\mathsf{B}^{-1}\mathbf{g}. \quad (7)$$

In principle, iterating this stepping procedure will eventually bring us to a local maximum value of f.

We replace the Hessian H with a form of Gauss–Newton approximation G, which guarantees invertibility and can be defined both for the regression likelihood and the classification likelihood, respectively [17]. In particular, the approximation used differs from the classical Gauss–Newton matrix in that it retains some second derivative information. Second, to avoid the prohibitive expense of calculating the inverse in (7), we instead solve (6) (with H replaced by G in B) for $\delta\mathbf{a}$ iteratively using a conjugate-gradient algorithm, which requires only matrix-vector products $\mathsf{B}\mathbf{v}$ for some vector $\mathbf{v}$. We calculate $\mathsf{B}\mathbf{v}$ products using a stable and efficient procedure applicable to NNs [17], [18] that avoids explicit calculation and calculation of B.

The combination of the above methods makes practical the use of second-order derivative information even for large networks.

D. 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 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.

IV. SkyNet Regression: Mapping Dark Matter challenge

The Mapping Dark Matter (MDM) Challenge was presented on `www.kaggle.com` as a simplified version of the GREAT10 Challenge [19]. In this problem, each data item consists of two 48×48 greyscale images of a galaxy and a star. Each pixel value in each image is drawn from a Poisson distribution with mean equal to the (unknown) underlying intensity value in that pixel. Moreover, both images have been convolved with the same, but unknown, point spread function. The learning task is to predict the ellipticities e_1 and e_2 (defined below) of the underlying true galaxy image, which constitutes a regression problem. The training data set contains $40,000$ image pairs and the challenge data set contains $60,000$ image pairs. Further details about the challenge and descriptions of the top results can be found in [20].

A. Galaxy image model

The true underlying galaxy image is assumed to be elliptical with semi-major axis a, semi-minor axis b, and position angle θ. The ellipticities e_1 and e_2 are related to these parameters by

$$e_1 = \frac{a-b}{a+b}\cos(2\theta), \;\; e_2 = \frac{a-b}{a+b}\sin(2\theta), \tag{8}$$

and may vary in the range $[-1, 1]$. Further details about the data set can be found at the challenge's webpage [2].

B. Results

We use SkyNet to train several regression networks, each of which takes the galaxy and star images as inputs and produces estimates of the true galaxy ellipticities as outputs. Following the approach used in the original challenge, the quality of a network's predictions are measured by the root mean squared error (RMSE) of its predicted ellipticities over the challenge data set of $60,000$ pairs of images.

For this demonstration, we train only relatively small networks, but used two different data sets: (i) the full galaxy and star images and (ii) the full galaxy image and a centrally cropped star image. The RMSE values for trained networks of different architecture, evaluated on the challenge dataset, are given in Table I.

The RMSE values obtained even for the naive first approach of using the full dataset (i) as inputs are quite good; for comparison, the standard software package SExtractor [21] produced an RMSE score of 0.0862191 on this test data and the un-weighted quadrupole moments (UWQM) method scores 0.1855600. We see from Table I that increasing the number of hidden nodes beyond two does improve the network accuracy, but only very slowly. The NN results can, however, be improved more easily simply by reducing the number of inputs without affecting the information content, for example by cropping the star images to the central 24×24 pixels to yield dataset (ii). This example demonstrates good regression training for an immediate 'out-of-the-box' application of SkyNet, that involves no specialised data processing.

[2]`http://www.kaggle.com/c/mdm/`

TABLE I
RMSE on ellipticity predictions evaluated on the $60,000$ challenge image pairs.

Data set	Hidden layers	RMSE
Full galaxy and star images	0	0.0224146
($48 \times 48 \times 2 = 4608$ inputs)	2	0.0186944
	5	0.0184237
	10	0.0182661
Full galaxy and cropped star images	0	0.0175578
($48 \times 48 + 24 \times 24 = 2880$ inputs)	2	0.0176236
	5	0.0175945
	10	0.0174997
	50	0.0172977
	50+10	0.0171719

V. 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 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 [22]. 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 further, a recent study by [23] performed a Monte Carlo analysis that generated a mock sample of GRBs, using the GRB rate and luminosity function of [24], and processed them through an entire simulated Swift detection pipeline.

A. 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 [23]. 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; we will focus on the former for our analysis.

B. 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 [25], 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 classifications.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 that have been correctly classified as detected).

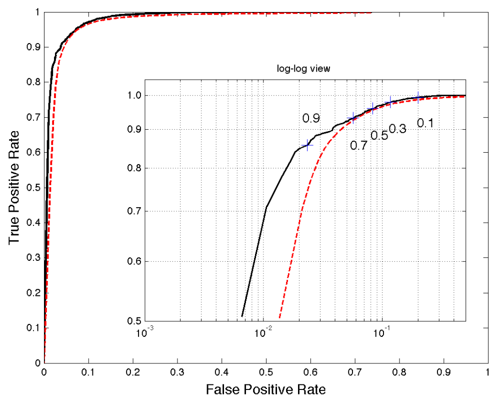

Fig. 1. Actual (black solid) and expected (dashed red) ROC curves for a NN classifier that predicts whether a GRB will be detected by Swift.

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_{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., [26]). The ROC curve plots the true positive rate (identical to completeness)against the false positive rate (also known as contamination). In general, the larger the area underneath a ROC curve, the more powerful the classifier.

From Figure 1, 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; it should be noted that here the actual ROC curve is better than the expected one. The curves also indicate that this test is quite powerful at predicting which GRBs will be detected by Swift.

Using the completeness, purity, and ROC curves, one can make a decision as to appropriate value of p_{th} to use. One may require a certain level of completeness, regardless of false positives, or we may require a minimal level of contamination in the final sample. Alternatively , one can use the ROC curve to derive an optimal value for p_{th}. We conclude that $p_{\text{th}} = 0.5$, the original naive choice, is a near-optimal threshold value.

Using $p_{\text{th}} = 0.5$, 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.

We note that networks used to obtain these results were trained in a few CPU hours and can make an accurate determination of whether a GRB will be detected by Swift in just a few microseconds, instead of the minutes of computation time required by the full simulated Swift trigger pipeline.

VI. The Structure of BAMBI

The Blind Accelerated Multimodal Bayesian Inference (BAMBI) algorithm combines nested sampling and neural networks. 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.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. 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. This is particularly useful when the model to be evaluated is computationally expensive or the data sets being used are large enough that comparison requires significant computation time. In both cases, substituting the much faster likelihood evaluation from the NN will more than make up for time spent in training.

Other research groups are actively working on methods of likelihood interpolation and extrapolation as well as the fast estimation of the Bayesian evidence as well as best-fit parameters and errors. See [27], [28], [29] and references therein for more information.

A. *When to Use the Trained Network*

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.

Once a NN is in use in place of the original log-likelihood function its evaluations are taken to be the actual log-likelihoods. Checks are made to ensure that the network is maintaining its accuracy and will re-train when these fail. To

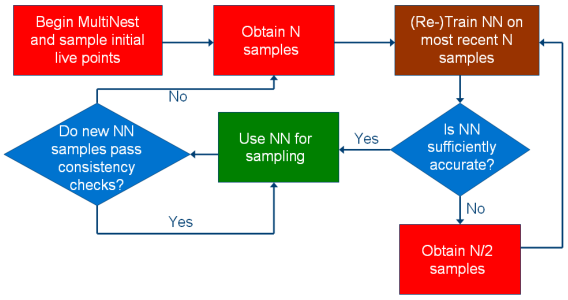
Begin MultiNest and sample initial live points
Obtain N samples
(Re-)Train NN on most recent N samples
Is NN sufficiently accurate?
Yes
No
Use NN for sampling
Do new NN samples pass consistency checks?
No
Yes
Obtain N/2 samples

Fig. 2. A flowchart depicting the transitions between sampling and NN training within BAMBI. N is given by `updInt` from MULTINEST.

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. The flow of sampling and training within BAMBI is demonstrated by the flowchart given in Figure 2.

VII. COSMOLOGICAL PARAMETER ESTIMATION WITH BAMBI

We implement BAMBI within the standard COSMOMC code [30], which by default uses MCMC sampling. This allows us to compare the performance of BAMBI with other methods, such as MULTINEST [9], COSMONET [31], [32], INTERPMC [33], PICO [34], and others. In this paper, we will only report performances of BAMBI and MULTINEST, but these can be compared with reported performance from the other methods.

Bayesian parameter estimation in cosmology requires evaluation of theoretical temperature and polarisation CMB power spectra (C_l values) using codes such as CAMB [35]. These evaluations can take on the order of tens of seconds depending on the cosmological model. The C_l spectra are then compared to WMAP and other observations for the likelihood function. 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 and PICO, 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. The COSMOMC website [30] provides full references for the most recent sources of these data.

Algorithm	Model	Data Set	$\log(\mathcal{Z})$
MULTINEST	ΛCDM	CMB only	-3754.58 ± 0.12
BAMBI	ΛCDM	CMB only	-3754.57 ± 0.12
MULTINEST	ΛCDM	all	-4124.40 ± 0.12
BAMBI	ΛCDM	all	-4124.11 ± 0.12
MULTINEST	ΛCDM+Ω_K	CMB only	-3755.26 ± 0.12
BAMBI	ΛCDM+Ω_K	CMB only	-3755.57 ± 0.12
MULTINEST	ΛCDM+Ω_K	all	-4126.54 ± 0.13
BAMBI	ΛCDM+Ω_K	all	-4126.35 ± 0.13

TABLE II
EVIDENCES CALCULATED BY MULTINEST AND BAMBI FOR THE FLAT (ΛCDM) AND NON-FLAT (ΛCDM+Ω_K) MODELS USING THE CMB-ONLY AND COMPLETE DATA SETS.

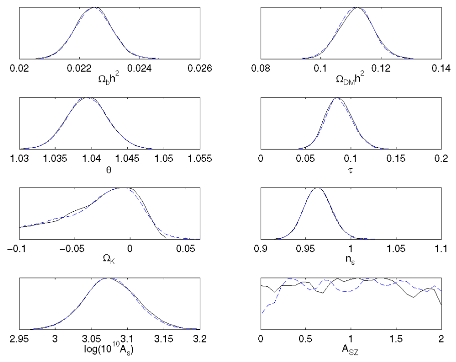

Fig. 3. Marginalised 1D posteriors for the non-flat model (ΛCDM+Ω_K) using only CMB data. MULTINEST is in solid black, BAMBI in dashed blue.

Analyses with MULTINEST and BAMBI were run on all four combinations of models and data sets. Table II reports the recovered evidences from the two algorithms for both models and both data sets. It can be seen that the two algorithms report equivalent values to within statistical error for all four combinations. In Figures 3 and 4 we show the recovered one- and 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. The only exception is A_{SZ} since it is unconstrained by these models and data and is thus subject to large amounts of variation in sampling. Using the full data set displays similar correspondence in the 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.

The most 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 ($t_{\text{wall clock, sec}} \times N_{\text{CPUs}}/N_{\log(\mathcal{L})\text{ evals}}$). Therefore, time required to train the NN is still counted as a penalty factor. If a NN

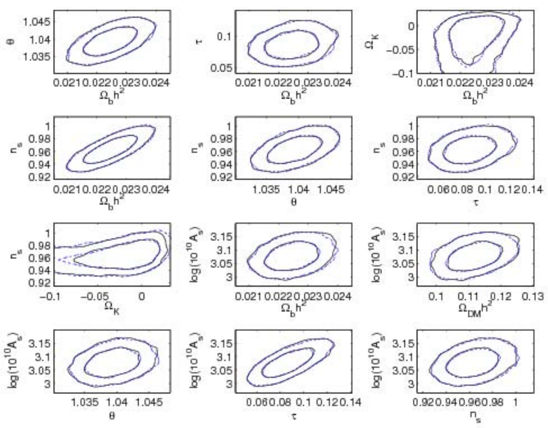

Fig. 4. Marginalised 2D posteriors for the non-flat model (ΛCDM+Ω_K) using only CMB data. The 12 most correlated pairs are shown. MULTINEST is in solid black, BAMBI in dashed blue. Inner and outer contours represent 68% and 95% confidence levels, respectively.

Model	Data set	MULTINEST $t_{\mathcal{L}}$ (s)	BAMBI $t_{\mathcal{L}}$ (s)	Speed factor
ΛCDM	CMB only	2.394	1.902	1.26
ΛCDM	all	3.323	2.472	1.34
ΛCDM+Ω_K	CMB only	12.744	9.006	1.42
ΛCDM+Ω_K	all	12.629	10.651	1.19

TABLE III
TIME PER LIKELIHOOD EVALUATION, FACTOR OF SPEED INCREASE FROM MULTINEST TO BAMBI ($t_{\rm MN}/t_{\rm BAMBI}$).

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. The resulting average times per likelihood and speed increases are given in Table III. We are able to obtain a significant decrease in running time while adding in the bonus of having a NN trained on the likelihood function.

VIII. USING TRAINED NETWORKS FOR FOLLOW-UP IN BAMBI

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 a subsequent analysis with different priors to obtain much faster results. This is a comparable analysis to that of COSMONET [31], [32], 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. A computationally cheap method of estimating the error on predictions was suggested by [36]. To find the error, one adds Gaussian noise to the true outputs of the training data and trains a new network on this noisy data. After performing many realisations, the networks' predictions will average to the predictions in the absence of the added noise. Moreover, the standard deviation of their predictions will provide a good estimate of the error in the original network's predictions.

BAMBI includes the option for the user to add random Gaussian offsets to the parameters before training is performed on the new data set. This offset will aid the training in moving the optimisation from a potential local maximum in the posterior distribution of the network parameters, but the size of the offset must be chosen for each problem; for this, we recommend using a value $s \lesssim 1/\alpha$.

A log-likelihood is calculated by first making a prediction with the final NN to be trained and saved and then calculating the error for this prediction. If the error, $\sigma_{\rm pred}$, is greater than that NN's threshold (set as 20% lager than largest error of the training data for the NN), then we consider the previous trained NN. We again calculate the predicted log-likelihood value and error to compare with its threshold. This continues to the first NN saved until a NN makes a sufficiently confident prediction. If no NNs can make a confident enough prediction, then we set $\log(\mathcal{L}) = -\infty$ ('discarding') or evaluate with the original log-likelihood function. Discarding is justified because the NNs are trained to predict values in the highest likelihood regions of parameter space and if a set of parameters lies outside their collective region of validity, then it must not be within the region of highest likelihoods.

To demonstrate the speed-up potential of using the NNs, we first ran an analysis of the cosmological parameter estimation using both models and both data sets. The auxiliary networks were then trained with a supplemental program, whose run-time depends only on the size of the network and data set. We then repeated each of the four analyses, 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 IV shows the average time taken per log-likelihood function evaluation when this follow-up was performed with MULTINEST and with BAMBI using NNs for the likelihood function. In all cases where points were not discarded, the BAMBI evidences matched the MULTINEST values to within statistical error. When discarding points, only the non-flat model with CMB-only data calculated an evidence that did not agree; this was due to this being the only case with only a single trained NN. In the other three cases, two trained networks were available to use.

Overall, the massive gains in speed from using the already trained NNs along with discarding points (that the networks cannot predict precisely enough) of $O(10^4)$ to $O(10^5)$ make it worthwhile to use this as an initial follow-up procedure. Should there be too many discarded points, the analysis may be performed again without discarding to compute a more reliable evidence value but still obtaining a speed increase of $O(10)$ to $O(100)$. We can thus obtain accurate posterior distributions and evidence calculations orders of magnitude faster than originally possible. Follow-up without discarding

Method	Model	Data	$t_{\mathcal{L}}$ (ms)	$t_{\mathcal{L}}$ (ms) (discard)	factor	factor (discard)
MN	flat	CMB	2775	—	—	—
		all	3813	—	—	—
	non-flat	CMB	12830	—	—	—
		all	10980	—	—	—
BAMBI	flat	CMB	3.883	0.2077	714.7	13360
		all	28.01	0.2146	136.1	17770
	non-flat	CMB	1105	0.08449	11.61	151900
		all	402.9	0.2032	27.25	54040

TABLE IV

TIME PER LIKELIHOOD EVALUATION IN A FOLLOW-UP ANALYSIS AND SPEED FACTORS WITH RESPECT TO MULTINEST. AVERAGE TIMES WITHOUT DISCARDING BAD PREDICTIONS ARE LIMITED BY THE NUMBER OF CALLS TO THE ORIGINAL LIKELIHOOD FUNCTION.

is limited by the number of calls to the original likelihood function while follow-up with discarding is limited by the number of stored networks being used.

IX. CONCLUSION

We have described 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. SKYNET employs a combination of (optional) pre-training followed by iterative refinement of the network parameters using an automatically-regularised variant of Newton's optimisation algorithm that incorporates second-order derivative information without the need to compute or store the Hessian matrix. SKYNET adopts convergence criteria that naturally prevent overfitting.

In an astronomical context, SKYNET is applied to the regression problem of measuring the ellipticity of noisy and convolved galaxy images in the Mapping Dark Matter Challenge and the classification problem of identifying gamma-ray bursters that are detectable by the Swift satellite. In each case, the use of SKYNET produces networks that perform the desired task quickly and accurately.

Finally, we describe the application of SKYNET in combination with MULTINEST to create the BAMBI algorithm for accelerated Bayesian inference. This is not tailored to any one application and requires no pre-processing. We apply BAMBI to the problem of cosmological parameter estimation and model selection. We performed this using flat and non-flat cosmological models and incorporating only CMB data and using a more extensive data set. In all cases, the NN is able to learn the likelihood function to sufficient accuracy after training on early nested samples and then predict values thereafter. 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 a factor of 1.19 to 1.42.

As larger data sets and more complicated models are used in cosmology, particle physics, and other fields, the computational cost of data analysis and Bayesian inference will increase. The SKYNET and BAMBI algorithms can, without any pre-processing, significantly reduce the required running time for these analysis and inference problems. The work and examples shown here are just the first step and future work aims to improve both the capabilities and performance of the software.

REFERENCES

[1] MacKay D.J.C, 2003, Information Theory, Inference, and LearningAlgorithms. Cambridge Univ. Press. www.inference.phy.cam.ac.uk/mackay/itila/

[2] Ball N.M., Brunner R.J., 2010, Int. J. Mod. Phys., 19, 1049

[3] Way M.J., Scargle J.D., Ali K.M., Srivastava A.N., 2012, Advances in Machine Learning and Data Mining for Astronomy. CRC Press.

[4] Karpenka N.V., Feroz F., Hobson M.P., 2013, MNRAS, in press

[5] Tagliaferri R. et al., 2003, Neural Networks, 16, 297

[6] Graff P. Feroz F., Hobson M.P., Lasenby A.N., 2013, arXiv:1309.0790 [astro-ph.IM]

[7] Skilling J., 2004, AIP Conference Series, 735, 395

[8] Feroz F., Hobson M.P., 2008, MNRAS, 384, 449

[9] Feroz F., Hobson M.P., Bridges M., 2009, MNRAS, 398, 1601

[10] Feroz F., Hobson M. P., Cameron E., & Pettitt A. N., 2013, arXiv:1306.2144 [astro-ph.IM]

[11] Graff P., Feroz F., Hobson M.P., Lasenby A.N., 2012, MNRAS, 421, 169

[12] Hornik K., Stinchcombe M. & White H., 1990, Neural Networks, 3, 359

[13] Hinton G.E., Osindero S., & Teh Y.-W., 2006, Neural Comput., 18, 1527–1554

[14] Hinton G.E. & Salakhutdinov R.R., 2006, Science, 313, 504–507

[15] Martens J., 2010, in Fürnkranz J., Joachims T., eds, Proc. 27th Int. Conf. Machine Learning. Omnipress. Haifa, Israel. p. 735

[16] 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/

[17] Schraudolph N.N., 2002, Neural Comput., 14, 1723

[18] Pearlmutter B.A., 1994, Neural Comput., 6, 147

[19] Kitching T. et al., 2011, Annals of Applied Statistics, 5, 2231–2263

[20] Kitching T. et al., 2012, New Astronomy Reviews, submitted

[21] Bertin E., Arnouts S., 1996, A&AS Supplement, 117, 393

[22] Gehrels N. et al., 2004, ApJ, 611, 1005

[23] Lien A., Sakamoto T., Gehrels N., Palmer D., Graziani C., 2012, Proceedings of the International Astronomical Union, 279, 347

[24] Wanderman D., Piran T., 2010, MNRAS, 406, 1944

[25] Feroz F., Marshall P. J., Hobson M. P., 2008, arXiv:0810.0781 [astro-ph]

[26] Fawcett T., 2006, Pattern Recogn. Lett., 27, 861

[27] Higdon D., Lawrence E., Heitmann K., & Habib S., 2012, in Feigelson E. D. & Babu G. J., eds, Statistical Chanllenges in Modern Astronomy V. Springer. New York, NY. p. 41–57.

[28] Rasmussen C. E., 2003, in Bayesian statistics, 7 (Tenerife, 2002). Oxford Univ. Press. New York, NY. p. 651–660.

[29] Bliznyuk N., et al., 2008, J. Comput. Graph. Statist., 17, 270–294.

[30] Lewis A & Bridle S, 2002, Phys. Rev. D, 66, 103511, arXiv:astro-ph/0205436. http://cosmologist.info/cosmomc/

[31] Auld T., Bridges M., Hobson M.P., Gull S.F., 2008, MNRAS, 376, L11

[32] Auld T., Bridges M., Hobson M.P., 2008, MNRAS, 387, 1575

[33] Bouland A, Easther R & Rosenfeld K, 2011, J. Cosmol. Astropart. Phys., 5, 016, arXiv:1012.5299 [astro-ph.CO]

[34] Fendt W.A., Wandelt B.D., 2007, ApJ, 654, 2

[35] Lewis A, Challinor A & Lasenby A, 2000, ApJ, 538, 473, arXiv:astro-ph/9911177

[36] MacKay D.J.C., 1995, Network: Computation in Neural Systems, 6, 469