Monthly Notices
of the
ROYAL ASTRONOMICAL SOCIETY

MNRAS **483,** 4828–4846 (2019)
Advance Access publication 2018 December 6

doi:10.1093/mnras/sty3307

Bayesian sparse reconstruction: a brute-force approach to astronomical imaging and machine learning

Edward Higson [1,2]★ Will Handley,[1,2] Michael Hobson[1] and Anthony Lasenby[1,2]

[1]*Astrophysics Group, Battcock Centre, Cavendish Laboratory, JJ Thomson Avenue, Cambridge CB3 0HE, UK*
[2]*Kavli Institute for Cosmology, Madingley Road, Cambridge CB3 0HA, UK*

Accepted 2018 December 3. Received 2018 November 25; in original form 2018 September 10

ABSTRACT

We present a principled Bayesian framework for signal reconstruction, in which the signal is modelled by basis functions whose number (and form, if required) is determined by the data themselves. This approach is based on a Bayesian interpretation of conventional sparse reconstruction and regularization techniques, in which sparsity is imposed through priors via Bayesian model selection. We demonstrate our method for noisy one- and two-dimensional signals, including astronomical images. Furthermore, by using a product-space approach, the number and type of basis functions can be treated as integer parameters and their posterior distributions sampled directly. We show that order-of-magnitude increases in computational efficiency are possible from this technique compared to calculating the Bayesian evidences separately, and that further computational gains are possible using it in combination with dynamic nested sampling. Our approach can also be readily applied to neural networks, where it allows the network architecture to be determined by the data in a principled Bayesian manner by treating the number of nodes and hidden layers as parameters.

Key words: methods: data analysis – methods: numerical – methods: statistical – techniques: image processing.

1 INTRODUCTION

Both sparse signal processing and Bayesian inference are well-established methods for data analysis and have a considerable amount in common. However, these two approaches are often considered somewhat distinct from one another, and this is often reflected in the relatively small overlap of the communities who develop and apply each technique. Nevertheless, Bayesian interpretations of sparse signal processing techniques have been pursued by a number of authors in the signal processing community – see for example Ji, Xue & Carin (2008), Tipping (2001), and Wipf & Rao (2004). In addition, Bayesian inference with imposed sparsity has been applied to astronomical problems (such as in Warren, Byers & Crump 2017; Jones & Heavens 2018; Sciacchitano, Lugaro & Sorrentino 2018).

In this paper we outline a principled Bayesian approach for simultaneously imposing sparsity and performing dictionary learning to determine the optimal basis set for representing the signal, and discuss how Bayesian inference provides a very natural framework for sparsity. In our method a signal is modelled as the superposition of a set of basis functions, whose number and form are determined by the data themselves. Sparsity can be imposed directly via the prior on the number of basis functions N, while simultaneous dictionary learning is performed through the estimation of parameters describing the location and shape of the basis functions.

The optimum number of basis functions N with which to model a signal can be determined using Bayesian model selection by calculating the Bayesian evidence for each value. However, it is equivalent (and often more computationally efficient and convenient) to treat N as an integer parameter, and sample directly from the joint posterior of N and the other parameters describing the N basis functions. The final inference may then be obtained by either by choosing the maximum *a posteriori* value of N or, better, by marginalizing over N to give a multimodel solution (Parkinson & Liddle 2013) with the fit for each number of basis functions weighted by its posterior probability. This method can be further generalized to select from a variety of types of basis functions T (such as Gaussians, Fourier modes, wavelet families, shapelets, etc.), with the full version involving inference over the joint space of T, N, and the basis functions' parameters.

While our principled approach is computationally expensive, we show that it is practical in the low data regime using current numerical methods at reasonable computational cost (see Table B1 in Appendix B for details of the number of core hours used to produce our results). In addition, this paper is intended as a proof of principle for applications where our method is not currently feasible but will be made so in the future by advances in numerical methods and increases in computational power.

★ E-mail: e.higson@mrao.cam.ac.uk

The paper proceeds as follows: Section 2 describes standard regression techniques, regularization, and sparsity. Section 3 then provides a Bayesian perspective on these topics, including introduction of our formulation of 'Bayesian sparse reconstruction' and a discussion of how it can be implemented numerically. Sections 4 and 5 demonstrate applying our approach to one- and two-dimensional signal processing, including of astronomical images from the *Hubble Space Telescope* eXtreme Deep Field (Illingworth et al. 2013). In addition, in Section 6, we apply the Bayesian sparse reconstruction framework to artificial neural networks, where we perform Bayesian inference over the joint space of network architectures and network parameters by treating the number of nodes and (if required) the number of hidden layers as parameters.

2 REGRESSION, REGULARIZATION, AND SPARSITY

We begin by presenting some background on standard approaches to regression, regularization, and sparsity. This provides context for the Bayesian framework presented in Section 3, in which all these methods may be reinterpreted. This section is intended to draw out the common themes in numerous popular signal reconstruction methods, and describe them in a unified manner.

2.1 Standard non-parametric regression

Regression involves using data points $\{\boldsymbol{x}_d, y_d\}$ (including random noise) to reconstruct some function $y = f(\boldsymbol{x};\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is some number of free parameters and the semicolon separates variables from parameters. Such inverse problems are common in science, and are typically ill-posed.[1] For example, in monochrome image reconstruction each data point is a pixel with two-dimensional (centre) position $\boldsymbol{x}$ and scalar intensity value y. In general, $\boldsymbol{x}$ and y can be vectors of any dimension; for simplicity, in this paper we consider only scalar outputs y, but the results easily generalize to vector outputs.

When a good model for the data is not available *a priori*, a traditional non-parametric approach is to use a *free-form* solution (Sivia & Skilling 2006) in which the function is pixelated and the value at each pixel is fitted. This is a standard way of performing 'brute-force' numerical calculations on computers, and is equivalent to fitting a delta function (or more accurately 'top-hat') basis function centred on each of the M pixels with their amplitudes as free parameters, giving M degrees of freedom. Ironically, such 'non-parametric' approaches thus contain many parameters – typically far more than 'parametric' approaches. The free form approach is illustrated for one-dimensional input x in Fig. 1 (which is based on Fig 6.1 of Sivia & Skilling 2006), but can be performed in arbitrary dimensions.

Smoothness can be encoded into the solution by replacing the delta functions with broader basis functions $\phi(\boldsymbol{x};\boldsymbol{x}_j,\sigma)$, with fixed centres $\boldsymbol{x}_j$ located on each of the M pixels and their width determined by a shared shape parameter σ. For Gaussian basis functions:

$$f(\boldsymbol{x};\boldsymbol{a},\sigma) = \sum_{j=1}^{M} a_j \phi(\boldsymbol{x};\boldsymbol{x}_j,\sigma) = \sum_{j=1}^{M} a_j \exp\left(-\frac{|\boldsymbol{x}-\boldsymbol{x}_j|^2}{2\sigma^2}\right). \quad (1)$$

[1] An 'ill-posed' problem does not satisfy all three of the conditions for a problem to be 'well-posed' outlined by Hadamard (1902). The conditions are: a solution exists, the solution is unique, and the behaviour of the solution changes continuously with changes in the parameters and data.

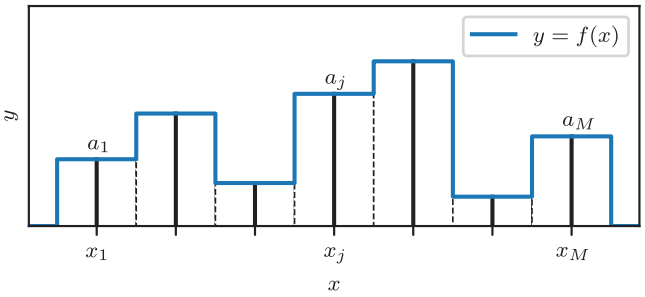


Figure 1. Free-form decomposition of a one-dimensional function $y = f(x)$ into M pixels with amplitudes (free parameters) $\boldsymbol{\theta} = (a_1, a_2, \ldots, a_M)$.

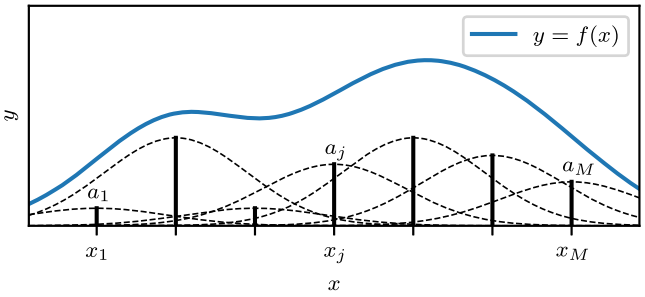


Figure 2. Free-form decomposition of a one-dimensional function $y = f(x)$ into M Gaussian basis functions with standard deviation σ, each centred on a pixel, with amplitudes (free parameters) $\boldsymbol{\theta} = (a_1, a_2, \ldots, a_M)$.

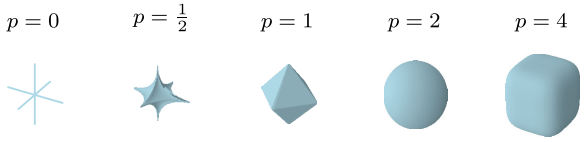


Figure 3. $L_p = 1$ surfaces in three dimensions for different values of p, with L_0 representing the number of non-zero components of the vector.

This is illustrated for a one-dimensional input x in Fig. 2 (based on fig. 6.7 of Sivia & Skilling 2006).

2.2 Optimization and regularization

The values of the parameters $\boldsymbol{\theta}$ are typically chosen by minimizing the squared L_2 norm of the differences between the model and the data (also referred to as the squared residuals or χ^2). Here the L_p norm of a vector is defined for $p > 0$ as $\|\boldsymbol{v}\|_p \equiv (\sum_i |v_i|^p)^{1/p}$ and the L_0 norm is the number of non-zero components; this is illustrated for different values of p in Fig. 3. The squared L_2 norm approach yields the maximum likelihood estimate (MLE) for $\boldsymbol{\theta}$ under certain restrictive conditions,[2] although it is commonly applied when these are not met; see Sivia & Skilling (2006, Chapter 8) for a more detailed discussion and recommended modifications to the least squares procedure for different types of data.

[2] These include that the residuals on each data point must be independently normally distributed, and that there are no errors in the independent variables $\boldsymbol{x}$.

When using the squared L_2 norm, the optimization is

$$\min_{\boldsymbol{\theta}} \sum_{d=1}^{D} (y_d - f(\boldsymbol{x}_d; \boldsymbol{\theta}))^2 = \min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2, \tag{2}$$

where $\boldsymbol{y} = \{y_1, \ldots, y_D\}$ are the data values and $\hat{\boldsymbol{y}} = \{f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_D)\}$ are the fit values. For simplicity we assume for the moment that the shape of the basis functions is fixed and only the amplitudes are free parameters, in which case

$$\min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2 = \min_{\boldsymbol{a} \in \mathcal{R}^M} \|\boldsymbol{y} - \Phi \boldsymbol{a}\|_2^2, \tag{3}$$

where the vector $\boldsymbol{a} = (a_1, a_2, \ldots, a_M)$ determines the basis functions' amplitudes and $\Phi = (\phi_1, \phi_2, \ldots, \phi_M)$ is a $D \times M$ basis matrix.

Typically a regularization term is added to penalize more complex models; this is to prevent the analysis fitting noise in the data set and producing a result that will not generalize to new data sets ('overfitting'). Some popular choices are:

(i) The (squared) L_2 norm – used in the Wiener filter (Wiener 1949) and ridge regression (Hoerl & Kennard 1970):

$$\min_{\boldsymbol{a} \in \mathcal{R}^M} \|\boldsymbol{y} - \Phi \boldsymbol{a}\|_2^2 + \lambda \|\boldsymbol{a}\|_2^2. \tag{4}$$

(ii) The L_1 norm – used in the Lasso (Tibshirani 1996), compressed sensing and for imposing sparsity:

$$\min_{\boldsymbol{a} \in \mathcal{R}^M} \|\boldsymbol{y} - \Phi \boldsymbol{a}\|_2^2 + \lambda \|\boldsymbol{a}\|_1. \tag{5}$$

(iii) The L_0 norm – used in matching pursuit (Mallat & Zhang 1993), iterative thresholding (Elad et al. 2007), compressed sensing and for imposing sparsity:

$$\min_{\boldsymbol{a} \in \mathcal{R}^M} \|\boldsymbol{y} - \Phi \boldsymbol{a}\|_2^2 + \lambda \|\boldsymbol{a}\|_0, \tag{6}$$

where $\|\boldsymbol{a}\|_0$ simply counts the number of non-zero elements in the amplitude vector $\boldsymbol{a}$.

(iv) The entropy – used in the maximum entropy method (MEM):

$$\min_{\boldsymbol{a} \in \mathcal{R}^M} \|\boldsymbol{y} - \Phi \boldsymbol{a}\|_2^2 - \lambda S(\boldsymbol{a}), \tag{7}$$

$$S(\boldsymbol{a}) = \sum_{i=1}^{M} a_i - m_i - a_i \ln\left(\frac{a_i}{m_i}\right), \tag{8}$$

where m_i is a (model) amplitude value assigned to each basis function (Ables 1974; Gull & Daniell 1978).

In principle, such optimizations define a 'solution curve' $\hat{\boldsymbol{a}}(\lambda)$. To obtain a particular solution, one must choose a value for the regularization parameter λ, which determines the relative importance of the accuracy of the fit to the data and the value of the regularizing function; it is often chosen *a priori* but can be determined using heuristics or cross-validation. For example, the regularization constant for MEM has historically been chosen so the residual statistic equals its expectation value – i.e. so $\chi^2 = D$, where D is the number of data points (Sivia & Skilling 2006). A more modern approach is to choose the value of λ, which maximizes the Bayesian evidence (see Section 3.1); this can also be used to select quantities such as the width σ of the basis functions shown in Fig. 2 (Sivia & Skilling 2006).

It is worth noting that equations (4)–(7) refer to the *synthesis* formulation that optimizes over the parameters $\boldsymbol{a}$. An alternative is the *analysis* approach in which the optimization is performed directly with respect to the (vectorized) function $f(\boldsymbol{x})$, and $\boldsymbol{a}$ is replaced in equations (4)–(7) with $\Phi^{-1} f(\boldsymbol{x})$. This technique is commonly used in radio interferometry – see for example Maisinger, Hobson & Lasenby (2004), McEwen & Wiaux (2011), and Cai, Pereyra & McEwen (2018).

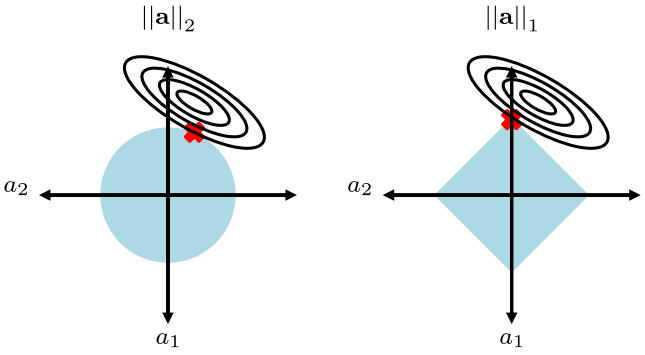


Figure 4. Illustration of L_p norms promoting sparsity when $p < 2$. The blue circle on the left-hand plot shows the region of the parameter space (a_1, a_2) with an L_2 norm less than some maximum value, and the blue diamond on the right-hand plot shows the region with an L_1 norm less than some maximum value. The black contours on each plot show an objective function to be optimized. Due to its angular shape, the region constrained by a maximum L_1 norm is more likely to have its maximum value of the objective function at a coordinate where one of the parameters is zero than the L_2 region.

2.3 Sparse representations

In many practical signal and image-processing applications we can use prior knowledge that the physical signals have 'sparse' representations in which they have very few non-zero components (a low L_0 norm). For example, astronomical images with many pixels can often be well represented by a relatively small number of point sources or wavelets. Sparse solutions are promoted by choosing a regularization term L_p with $p < 2$, in which case L_p surfaces have singular points at sparse solutions (Bach et al. 2012); this is illustrated graphically in Fig. 4.

Sparsity is key to *compressed sensing*: a popular signal processing technique for efficiently recovering high-dimensional vector signals under the assumption that they are sparse in some basis (see Eldar & Kutyniok 2012, for a detailed discussion). Sparse solutions can be found by L_0-optimization, but this is computationally challenging and is non-convex, meaning standard convex optimization cannot be used. The success of compressed sensing is based on approximating the L_0-norm with the L_1-norm (the smallest p for which the L_p norm is convex) – in certain circumstances the solutions can be shown to be identical and in others the error is bounded. Compressed sensing has been applied successfully to a variety of astronomical problems; see for example Bobin, Starck & Ottensamer (2008) and Wiaux et al. (2009).

2.4 Adaptive basis functions and dictionary learning

In order to find representations for data sets that are sparse (use relatively few basis functions), we now generalize the reconstructions described in (2.1) by allowing each basis function's location and shape to be determined by parameters $\boldsymbol{p}_i$ and fitted to the data. The

signal is reconstructed as

$$f(\boldsymbol{x};\boldsymbol{a},\boldsymbol{p}_1,\ldots,\boldsymbol{p}_N)=\sum_{i=1}^{N}a_i\phi(\boldsymbol{x};\boldsymbol{p}_i), \tag{9}$$

where now the number of basis functions N can easily be much smaller than the number of pixels M.

For a given data set, some types of basis function will provide more natural and sparse representation than others. We can further generalize (9) using parameterized dictionary learning, by fitting different families of standard basis functions (determined by a categorical variable T). The optimization then determines T, as well as each basis function's amplitude a_i and parameters $\boldsymbol{p}_i$ by reconstructing the signal as

$$f(\boldsymbol{x};T,\boldsymbol{a},\boldsymbol{p}_1,\ldots,\boldsymbol{p}_N)=\sum_{i=1}^{N}a_i\phi^{(T)}(\boldsymbol{x};\boldsymbol{p}_i). \tag{10}$$

Commonly used basis function families include Gaussians, wavelets, and shapelets.

3 A BAYESIAN APPROACH

We now consider the Bayesian interpretation of the regression and regularization discussed in Section 2, before outlining our approach to Bayesian sparse reconstruction and how it can be performed computationally.

3.1 Background: Bayesian inference

Bayesian inference (see Sivia & Skilling 2006 or MacKay 2003 for a detailed discussion) can be divided into *parameter estimation* and *model comparison*. Given a model $\mathcal{M}$, inferences about its parameters $\boldsymbol{\theta}$ from data $\mathcal{D}$ can be made by calculating the posterior distribution of $\boldsymbol{\theta}$ using Bayes' theorem (Bayes & Price 1763):

$$P(\boldsymbol{\theta}|\mathcal{M},\mathcal{D})=\frac{P(\mathcal{D}|\boldsymbol{\theta},\mathcal{M})P(\boldsymbol{\theta}|\mathcal{M})}{P(\mathcal{D}|\mathcal{M})}\equiv\frac{\mathcal{L}(\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\mathcal{Z}}, \tag{11}$$

where $\mathcal{L},\pi$, and $\mathcal{Z}$ are the likelihood, prior, and Bayesian evidence, respectively. The evidence $\mathcal{Z}$ is a normalization constant, and is computed by averaging the likelihood $\mathcal{L}$ over the prior π:

$$\mathcal{Z}\equiv P(\mathcal{D}|\mathcal{M})=\int\mathcal{L}(\boldsymbol{\theta})\pi(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}. \tag{12}$$

Bayes' theorem can also be used to compare different models $\mathcal{M}_1,\mathcal{M}_2,\ldots$ and assess which best describes the data. The posterior probability of a given model is

$$P(\mathcal{M}_j|\mathcal{D})=\frac{P(\mathcal{D}|\mathcal{M}_j)P(\mathcal{M}_j)}{P(\mathcal{D})}=\frac{\mathcal{Z}_j\Pi_j}{\sum_k\mathcal{Z}_k\Pi_k}, \tag{13}$$

where $\Pi_j\equiv P(\mathcal{M}_j)$ denotes the prior probability of each model and the denominator of the final term sums over all competing models. The evidence $\mathcal{Z}$ penalizes more complex models so this approach naturally includes Occam's razor and favours sparse solutions. Models may also be compared by computing log posterior odds ratios

$$\mathcal{P}_k^j\equiv\log\left(\frac{P(\mathcal{M}_j|\mathcal{D})}{P(\mathcal{M}_k|\mathcal{D})}\right)=\log\left(\frac{\mathcal{Z}_j}{\mathcal{Z}_k}\right)+\log\left(\frac{\Pi_j}{\Pi_k}\right), \tag{14}$$

where the ratio of evidences $\mathcal{B}_k^j=\mathcal{Z}_j/\mathcal{Z}_k$ is called a Bayes factor and is often used for comparison when the prior probability of the different models is not easily determined. The Bayes factors do, however, depend on the priors on the models' parameters $\pi(\boldsymbol{\theta}_{\mathcal{M}_j})$ through the calculation of $\mathcal{Z}_j$ from (12). If the prior on different models is uniform, the Bayes factors are equal to the posterior odds ratios.

3.2 Bayesian formulation of regression and regularization

Before introducing our full Bayesian sparse reconstruction framework in Section 3.4, we first give a Bayesian formulation of the regression and regularization problems discussed in Sections 2.1 and 2.2 as the comparison is very informative. In these cases the number, type, and shape of basis functions are fixed and the only parameters of the model are the amplitudes, i.e. $\boldsymbol{\theta}=\boldsymbol{a}$.

In general, defining the likelihood of the basis function fit given some data $\mathcal{D}$ requires knowledge of how measurement errors are distributed. For example, a common assumption in the literature is that there are independent Gaussian errors on the signal values $\{y_d\}$, and no errors on the data points' coordinates $\{\boldsymbol{x}_d\}$. In this case the likelihood of the data given the model is

$$\begin{aligned}\mathcal{L}(\boldsymbol{a})&=\prod_{d=1}^{D}\frac{1}{\sqrt{2\pi\sigma_y^2}}\exp\left(-\frac{(y_d-f(\boldsymbol{x}_d;\boldsymbol{a}))^2}{2\sigma_y^2}\right)\\&\propto\exp\left(-\frac{\|\boldsymbol{y}-\Phi\boldsymbol{a}\|_2^2}{2\sigma_y^2}\right),\end{aligned} \tag{15}$$

recovering the (exponentiated) least squares objective function from (3). Of course, this assumption may be inappropriate for some data sets. For example, for low-level photon-counting measurements a Poisson likelihood function (or similar) may be required. Although the Bayesian formulation can naturally accommodate other likelihood functions, for simplicity we will henceforth consider only independent Gaussian errors.

In order for the Bayesian approach to give the same maximum *a posteriori* parameter values as the optimization in Section 2.2, the prior $\pi(\boldsymbol{a})$ must correspond to the exponential of the regularization terms in (4–7). For a more formal derivation of this result in the context of the Wiener filter, see Hobson et al. (1998) and Lasenby, Barreiro & Hobson (2001).

In the Bayesian framework, the different regularization techniques in (4–7) are analogous to the following choices of priors:

(i) L_2 (squared) regularization (4) corresponds to a Gaussian prior:

$$\pi(\boldsymbol{a})\propto\exp(-\lambda\|\boldsymbol{a}\|_2^2). \tag{16}$$

(ii) L_1 regularization (5) corresponds to a Laplacian prior:

$$\pi(\boldsymbol{a})\propto\exp(-\lambda\|\boldsymbol{a}\|_1). \tag{17}$$

(iii) L_0 regularization (6) corresponds to an exponential prior on the number of non-zero components of $\boldsymbol{a}$:

$$\pi(\boldsymbol{a})\propto\exp(-\lambda\|\boldsymbol{a}\|_0)=\exp(-\lambda N), \tag{18}$$

where N is the number of basis functions used in the signal reconstruction.

(iv) Entropy regularization (7) corresponds to an entropic prior

$$\pi(\boldsymbol{a})\propto\exp(\lambda S(\boldsymbol{a})), \tag{19}$$

where $S(\boldsymbol{a})$ is defined in (7).

More generally other priors can be used. For example, one may promote sparsity by using any prior that has fatter tails than a Gaussian and is also more concentrated at zero – such priors prefer to

shrink amplitudes to zero while also being lenient in allowing larger amplitudes. Thus, as an alternative to the Laplacian distribution (17), one could use, for example, a Cauchy distribution

$$\pi(\boldsymbol{a}) = \prod_{j=1}^{M} \frac{\lambda}{\pi} \frac{1}{\lambda^2 + a_j^2}. \tag{20}$$

In addition, the L_2 regularization prior (16) can be generalized to include some covariance matrix $\boldsymbol{C}$, which may be a function of some further parameters $\boldsymbol{\theta}$,

$$\pi(\boldsymbol{a}) \propto \exp(-\lambda \boldsymbol{a}^{\intercal} \boldsymbol{C}^{-1} \boldsymbol{a}). \tag{21}$$

This form can be used to reconstruct a signal as a Gaussian process (see Rasmussen 2004, for an introduction), with $\boldsymbol{C}$ representing its correlation structure. When performing the optimization, the basis matrix Φ most naturally contains Fourier modes. The optimization is typically performed by selecting both $\boldsymbol{\theta}$ and λ to maximize the Bayesian evidence (sometimes the value of λ is chosen *a priori*). Indeed, Gaussian processes could be further generalized by using a different form for the prior term – for example 'entropic processes' with $\pi(\boldsymbol{a}) \propto \mathrm{e}^{-\lambda S(\boldsymbol{La})}$, where S is defined as in (7) and $\boldsymbol{C} = \boldsymbol{LL}^{\intercal}$ is the Cholesky decomposition of the signal correlation matrix (Hobson et al. 1998).

3.3 Sampling and model selection

Maximum *a posteriori* estimates of the parameters $\boldsymbol{a}$ can be found from the posterior distribution $\propto \mathcal{L}(\boldsymbol{a})\pi(\boldsymbol{a})$ in an analogous manner to the optimizations in (3–7). However, a major advantage of the Bayesian approach is that it provides a *generative* model and allows the full posterior distribution to be sampled. This provides additional information such as posterior distributions on the weights $\boldsymbol{a}$ and other quantities of interest.

Furthermore, the posterior distribution allows the appropriate number of basis functions to be chosen via Bayesian model selection by calculating posterior odds ratios (14). This naturally penalizes more complex models and, with an appropriate choice of priors, provides a principled Bayesian method for creating models with the level of complexity that is justified by the data. Finally one can either choose the maximum *a posteriori* number of basis functions or, better, marginalize over N so the fit with each number of basis functions is weighted in proportion to its posterior probability. Any *a priori* expectation of the degree of sparsity can be included in the priors, and there is no need for an additional regularization term.

3.4 Bayesian sparse reconstruction

Following the discussion in the previous sections, we propose reconstructing the relationship $y = f(\boldsymbol{x})$ as a sum of N basis functions $\phi^{(T)}$ of type T with weights a_i and shape and location parameters $\boldsymbol{p}_i$ as

$$f(\boldsymbol{x}; T, N, \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N) = \sum_{i=1}^{N} a_i \phi^{(T)}(\boldsymbol{x}, \boldsymbol{p}_i). \tag{22}$$

One can then perform Bayesian inference over the full parameter space of $\boldsymbol{\theta} = (T, N, \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N)$.

This approach has the desirable properties that:

(i) full posterior distributions on parameters can be recovered by sampling (rather than simply optimizing);

(ii) the Bayesian calculations naturally penalize overcomplex models. In addition, when there is an *a priori* justification, sparsity can be further enforced directly through priors on the total number of basis functions N;

(iii) there are a variable number of basis functions with variable positions;

(iv) there is no need to choose a regularization constant λ;

(v) families and/or shapes of basis functions are determined and can be marginalized over;

(vi) arbitrary constraints can be imposed on the reconstruction (not just positivity);

(vii) any type of noise can be included, e.g. Gaussian, Poisson, etc. If the size or nature of the noise is unknown, it can be expressed in terms of additional parameters that can be marginalized over;

(viii) missing and/or irregular data can be accommodated;

(ix) the model is generative and can easily be extended to deconvolution.

The remainder of this section discusses how Bayesian sparse reconstructions can be computed, with numerical tests presented in the following section.

3.5 Vanilla and adaptive methods

Given some noisy signal to be reconstructed, Bayesian model selection can be used to determine an appropriate type T and number N of basis functions to use by calculating the Bayesian evidence $\mathcal{Z}_{T,N}$ for the fit using each combination (model) T, N. Using (13), the posterior probability of each model is proportional to $\mathcal{Z}_{T,N}\Pi_{T,N}$, where $\Pi_{T,N}$ is the prior probability of the model and overcomplex models are penalized by lower evidences. We term this the *vanilla method*. One can then either select the model with the highest posterior probability or, better, use a combination of all models weighted by their posterior probability ('multimodel analysis').

The *adaptive method* is an alternative product-space approach, which analyses a 'meta-model' containing one or more discrete parameters with values corresponding to each individual model. The likelihood of a sample is found by selecting the model indicated by the discrete parameters, and then working out the likelihood for this model using the remaining parameters. A fixed dimensionality that is sufficient for the individual model with the most parameters is used; for models with fewer parameters, the likelihood is independent of the remaining unneeded parameters. This is an alternative to transdimensional sampling methods such as reversible-jump MCMC (Green 1995). Hee et al. (2016, 2017) used the adaptive method in reconstructing one-dimensional signals by linearly interpolating between N points ('nodes'), with their coordinates as free parameters. We generalize this approach by letting the integer parameter N represent the number of basis functions (of any dimension) to be used, and when needed also including a second integer parameter T to determine the form of the basis functions. Posterior distributions of T and N are found using parameter estimation.

3.6 Practical considerations for sampling the posterior

The posterior is typically of moderate to large dimensionality, and will be non-convex and multimodal with pronounced degeneracies. Furthermore, due to the integer parameters T and N, methods requiring gradients cannot be used. We explore the posterior using nested sampling (Skilling 2006), which is well suited to such problems and can be performed using software packages such as MULTINEST (Feroz & Hobson 2008; Feroz, Hobson & Bridges 2008; Feroz et al. 2013) or POLYCHORD (Handley, Hobson & Lasenby 2015a,b). The adaptive method calculates posterior odds ratios indirectly by

sampling the integer parameters T and N, and as a result its sampling errors have different properties to those of the vanilla method (which uses evidence calculations). For a detailed discussion of sampling errors in nested sampling parameter estimation, see Higson et al. (2018).

Nested sampling calculations can be made significantly more computationally efficient (or alternatively more accurate for the same amount of computation) using dynamic nested sampling (Higson et al. 2017). In particular, dynamic nested sampling gives large efficiency gains for parameter estimation, meaning it works well with the adaptive method. In contrast, the efficiency gains for evidence calculations are relatively modest (except in low dimensions), so dynamic nested sampling only produces small speedups for model selection via the vanilla method and we do not use it in this case. Results in this paper were calculated using DYPOLYCHORD (Higson 2018b) – a dynamic nested sampling package based on POLYCHORD. Due to the challenging multimodal posteriors produced by the integer parameter in the adaptive method, we use a large fraction (50 per cent) of the total computational budget for each calculation on DYPOLYCHORD's initial exploratory run. This reduces the possible efficiency gain, but DYPOLYCHORD is still able to produce significant speedups compared to standard nested sampling.

4 FITTING ONE-DIMENSIONAL DATA

We first demonstrate Bayesian sparse reconstruction by finding the dependence of some scalar quantity y on another scalar variable x, and to make the example more challenging we allow errors on both the data values y_d and positions x_d.

If each measurement has an independent error distribution $P(x_d, y_d|X_d, Y_d)$ about its true value X_d, Y_d, then the probability of the observed data given some set of true values is

$$P(\mathcal{D}|\{X_d, Y_d\}) = \prod_{d=1}^{D} P(x_d, y_d|X_d, Y_d). \tag{23}$$

The unknown true data values X_d, Y_d are then marginalized out using the basis fitting model by taking $Y_d = f(X_d; T, N, \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N)$ and integrating over the distribution of the x coordinates at which data points were sampled $P(X_d)$. Hence, each likelihood call involves an integral for every data point and

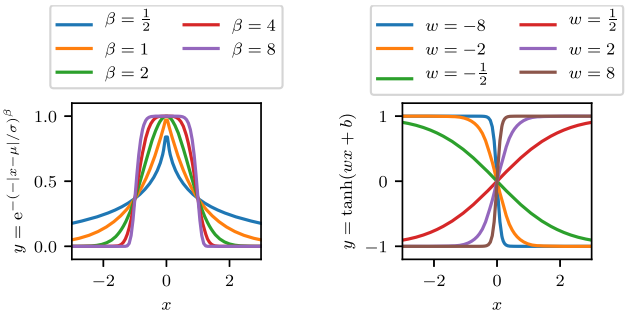
$$P(\mathcal{D}|P(X_d), T, N, \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N) = \prod_{d=1}^{D} \int P(x_d, y_d|X_d, f(X_d))\, P(X_d)\, \mathrm{d}X_d, \tag{24}$$

where for brevity we have omitted the dependence of $f(X_d; T, N, \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N)$ on the parameters $T, N, \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N$.

We first consider samples to be taken uniformly in the range $X_- < X_d < X_+$ with independent Gaussian x and y errors of size σ_x and σ_y. In this case, (24) gives the likelihood (Hee et al. 2016)

$$\mathcal{L}(T, N, \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N) = P(\mathcal{D}|T, N, \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N) = \prod_{d=1}^{D} \int_{X_-}^{X_+} \frac{\exp\left[-\frac{(x_d - X_d)^2}{2\sigma_x^2} - \frac{(y_d - f(X_d))^2}{2\sigma_y^2}\right]}{2\pi\sigma_x\sigma_y(X_+ - X_-)}\, \mathrm{d}X_d. \tag{25}$$

The priors on the parameters and models can be specified as required, and together with the likelihood can be used to sample numerically from the posterior and calculate evidences. As $f(x; T, N, \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N)$ is typically invariant under interchange of basis function index the prior space can be shrunk by a factor of $N!$ by enforcing ordering using 'forced identifiability' (sorted) priors; see Handley et al. (2015a, Appendix A2) for a more detailed discussion.



(a) Generalised Gaussians (26) for different values of β; each has $\mu = 0$ and $\sigma = 1$.

(b) tanh functions (27) with different values of w. All the lines shown use $b = 0$.

Figure 5. Illustrations of one-dimensional basis functions.

4.1 Basis functions

The likelihood (25) applies for mixture models with any one-dimensional basis function; we demonstrate it using one-dimensional generalized Gaussians

$$\phi^{(\mathrm{g1d})}(x; \boldsymbol{p}) = \phi^{(\mathrm{g1d})}(x, \mu, \sigma, \beta) = \mathrm{e}^{-(|x-\mu|/\sigma)^\beta} \tag{26}$$

and one-dimensional tanh functions

$$\phi^{(\mathrm{t1d})}(x, \boldsymbol{p}) = \phi^{(\mathrm{t1d})}(x, w, b) = \tanh(wx + b). \tag{27}$$

Their shape and location are determined by parameters $\boldsymbol{p} = (\mu, \sigma, \beta)$ and $\boldsymbol{p} = (w, b)$, respectively; the effects of different parameters are illustrated in Fig. 5. The magnitude of each basis function in the fit is controlled by an amplitude parameter a.

When $\beta = 2$, (26) is proportional to a normal distribution with variance $\sigma^2/2$, and when $\beta = 1$ it is proportional to a Laplace distribution. For large values of β, (26) is approximately uniform $\in [\mu - \sigma, \mu + \sigma]$ and zero elsewhere. The normalization constant $\beta/(\Gamma(\frac{1}{\beta})2\sigma)$ is omitted from (26) as it causes pronounced degeneracies in the joint posterior distributions of a, β and σ due to all three parameters affecting the height of the basis function at its centre.

The priors used for the basis functions are shown in Table 1. We use a uniform prior on the number and type of basis functions, but other priors can be used when they are justified for the problem considered – for example a prior favouring small values of N will result in more sparse solutions. The exponential prior on the amplitudes $\boldsymbol{a}$ of the generalized Gaussians has the desirable property that it is a function of only the sum of the amplitudes and does not vary based on how the total is split between basis functions. We use a uniform prior on the generalized Gaussians' σ, rather than a scale prior favouring smaller values, as we find the latter causes overfitting by encouraging the addition of narrow generalized Gaussians to fit noise in the data. The priors on the tanh basis functions are chosen for consistency with the neural networks discussed in Section 6.

4.2 Numerical results

We first illustrate Bayesian sparse reconstruction using simulated one-dimensional data points sampled from basis function mixture

Table 1. Priors on basis function parameters used in this paper. Sorted priors have ordering enforced; see Handley et al. (2015a, Appendix A2) for more details. The half Gaussian prior on the amplitudes of the tanh basis functions is truncated at zero and permits only positive values.

Parameter	Prior type	Prior parameters
One-dimensional generalized Gaussian (26)		
N	Uniform (integer)	$\in \mathbb{Z} \cap [1, 5]$
a	Sorted exponential	$\lambda = 1$
μ	Uniform	$\in [0, 1]$
σ	Uniform	$\in [0.03, 1.0]$
β	Exponential	$\lambda = 0.5$
One-dimensional tanh (27)		
N	Uniform (integer)	$\in \mathbb{Z} \cap [1, 5]$
a	Sorted Half Gaussian	$\mu = 0, \sigma = 5$
w	Gaussian	$\mu = 0, \sigma = 5$
b	Gaussian	$\mu = 0, \sigma = 5$
Adaptive basis function family selection		
T	Uniform (integer)	$\in \mathbb{Z} \cap [1, 2]$
Two-dimensional generalized Gaussian (29)		
N	Uniform (integer)	$\in \mathbb{Z} \cap [1, 5]$
a	Sorted exponential	$\lambda = 1$
μ_1	Uniform	$\in [0, 1]$
μ_2	Uniform	$\in [0, 1]$
σ_1	Uniform	$\in [0.03, 0.5]$
σ_2	Uniform	$\in [0.03, 0.5]$
β_1	Exponential	$\lambda = 0.5$
β_2	Exponential	$\lambda = 0.5$
Ω	Uniform	$\in [-\pi/4, \pi/4]$

models. Independent Gaussian x- and y-errors of size $\sigma_x = \sigma_y = 0.07$ are added to each data point.

Fig. 6 shows the results from fitting different Gaussian mixture models; plots of the posterior distribution of y were created using the `fgivenx` package (Handley 2018). Despite the large measurement noise and visually similar data in Figs 6(a)–(c), our approach is able to correctly reconstruct the different true signals and identify the increasing number of basis functions required to model each signal.

Fig. 7 shows examples of signal reconstructions conditioned on specific numbers of basis functions, and illustrates the effect of increasing N on the fit produced. Such plots can be calculated from adaptive method nested sampling runs by marginalizing over different values for the integer parameter N. However, we use the vanilla method runs to make these plots as the adaptive method dedicates relatively few samples to exploring the highly disfavoured N values which make negligible contribution to the overall fit. This is a desirable feature that makes fitting with the adaptive method more efficient, but as a consequence the vanilla method can produce more accurate plots conditioned on disfavoured values of N.

Figs 8 and 9 show examples of fitting tanh basis functions to one-dimensional data, and are similar to Figs 6 and 7. As for the generalized Gaussian basis functions, our approach is able to accurately reconstruct the true signal from the noisy data and identify the increasing complexity of the successive signals in Figs 8(a)–(c). However, it is not necessarily the case that the most probable *a posteriori* value of N, given the noisy data and priors, is the same as the number of basis functions from which the data were sampled; in Fig. 8(c), $P(N = 4|\mathcal{L}, \pi)$ and $P(N = 5|\mathcal{L}, \pi)$ are greater than $P(N = 3|\mathcal{L}, \pi)$.

4.3 Adaptive basis function families

We now illustrate including a second integer parameter T, which selects the basis function family, in addition to N, which selects the number of basis functions given the family. Fig. 10 shows fits using both generalized Gaussians ($T = 1$) and tanh functions ($T = 2$), with a uniform prior on $T \in \mathbb{Z} \cap [1, 2]$.

The generalized Gaussians are a much better fit for the data in Fig. 10(a), with posterior probability from the adaptive method with dynamic nested sampling of $P(T = 2|\mathcal{L}, \pi) = (1 \pm 1) \times 10^{-7}$. In contrast, the two families are competitive for the data in Fig. 10(b), with $P(T = 2|\mathcal{L}, \pi) = 0.6 \pm 0.1$ indicating only a weak favouring of the tanh basis function. These results are, however, highly dependent on the priors used for the basis functions' parameters.

A possible application of this adaptive selection of basis function families T would be to compare different parametric models for sources in astronomical images in which the true number of sources is unknown. In this case computing a posterior distribution on T would not only marginalize over the distributions of the sources' parameters, but also over the unknown number of sources N.

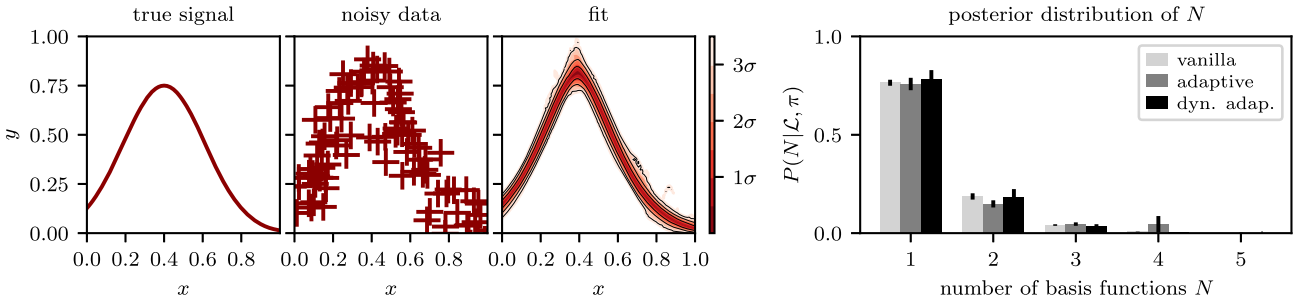
4.4 Comparison of vanilla and adaptive results

The adaptive method allows significant improvements in accuracy of the overall fit for a given computational cost by allocating fewer samples into disfavoured models that make a small or negligible contribution to the output. In addition, by transforming the model selection from evidence calculations (as in the vanilla method) to a parameter estimation problem on N, the adaptive method changes the nature of the sampling errors. Uncertainty in the rate of shrinkage at each step before any significant posterior mass is reached – the dominant source of error in nested sampling evidence calculations – has a negligible effect on parameter estimation of the posterior distribution of N. This can allow order-of-magnitude gains in computational efficiency of posterior odds ratios from the adaptive method compared to the vanilla method, as observed by Chua et al. (2018). However, a downside of the method is that including all the models and the integer parameter makes the posterior distribution highly multimodal and more challenging for the sampler to explore.

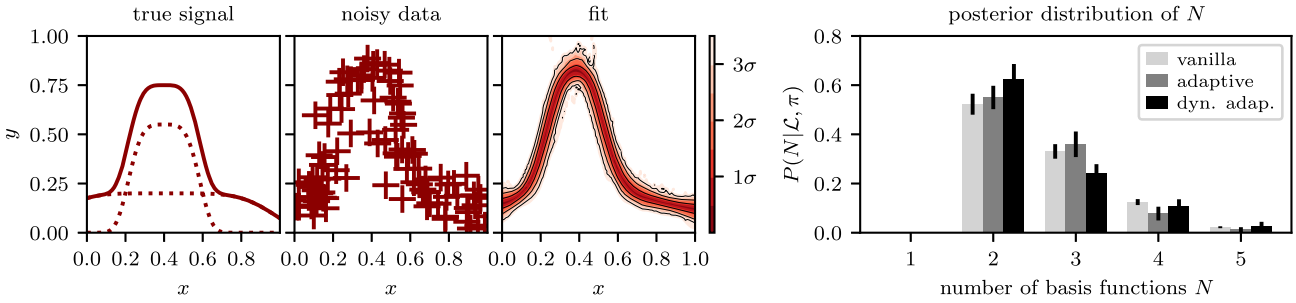
The errors on nested sampling calculations scale in inverse proportion to the square root of the computational effort used, and for a given likelihood and prior the number of samples produced is roughly proportional to the computational effort. Following Higson et al. (2017), we therefore measure the increase in computational efficiency (adjusted for any differences in the number of samples taken) from alternative methods compared to the vanilla method with standard nested sampling as

$$\text{Efficiency gain} = \frac{\text{Var[vanilla NS results]}}{\text{Var[method NS results]}} \times \frac{\overline{N_{\text{samp,van}}}}{\overline{N_{\text{samp,meth}}}}. \tag{28}$$

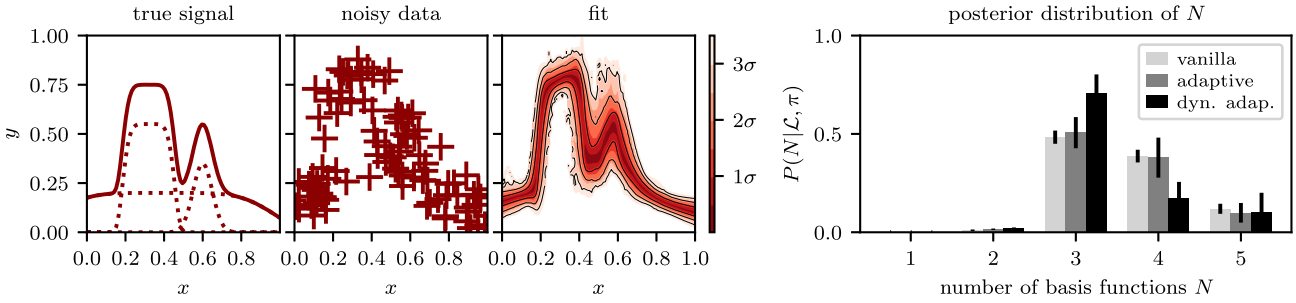
Here the first term is the ratio of the estimated variance of the results of repeated calculations using the vanilla method and the alternative method; the second term is the ratio of the mean number of samples from the nested sampling runs using each method. Numerical results for the efficiency gains from the different methods are shown in Table C4 in Appendix C. These use estimates of the variance of results calculated using the bootstrap resampling method described in Higson et al. (2018), which avoids the need to compute large numbers of nested sampling runs but also does not include additional errors from the sampler failing to explore the parameter space fully (see Higson et al. 2019, for a detailed discussion of such errors).



(a) Data from a single generalised Gaussian.



(b) Data from the sum of two generalised Gaussians.



(c) Data from the sum of three generalised Gaussians.

Figure 6. Fitting generalized Gaussian basis functions to 100 data points sampled from different combinations of basis functions. In each row the first plot shows the true signal (a sum of basis functions); where this contains more than one basis function, the individual components are shown with dashed lines. The data, which include added normally distributed x- and y-errors with $\sigma_x = \sigma_y = 0.07$, are shown in the second plot. The third plot shows the fit calculated using the adaptive method with dynamic nested sampling; coloured contours represent posterior iso-probability credible intervals on $y(x)$. The bar plots on the right-hand panel display the posterior distribution for different numbers of basis functions N; values calculated using the vanilla method and using the adaptive method with standard nested sampling are also included for comparison. Results shown for the adaptive method use a combined inference from five runs, each of which computes a full posterior on N and uses 1000 live points; adaptive runs using dynamic nested sampling have DYPOLYCHORD settings $n_{\text{init}} = 500$ and `dynamic_goal` $= 1$. Results for the vanilla method use five separate runs, each with 200 live points, to compute the evidence for each value of N. All runs use the setting `num_repeats` $= 100$. The parameters of the basis functions in the true signal and numerical results for the computational efficiency of the different methods are shown in Tables C1 and C4, respectively, in Appendix C.

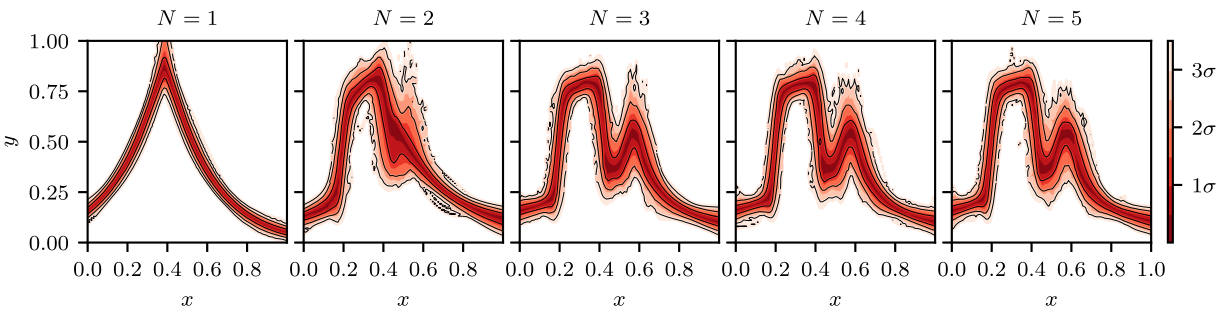
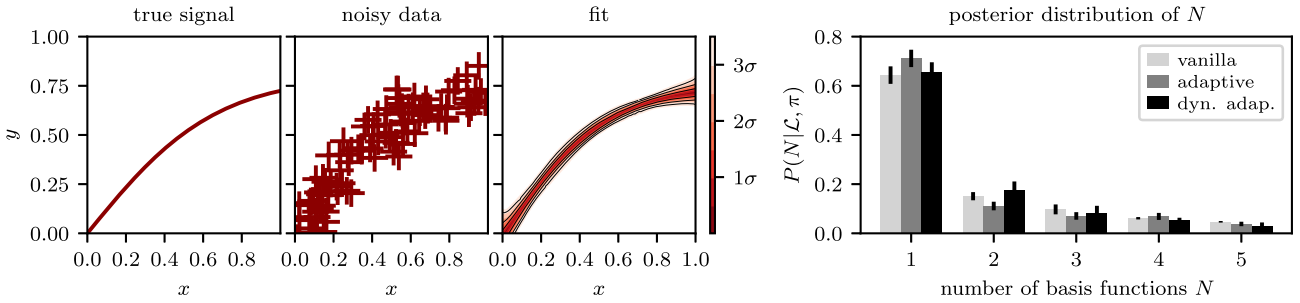
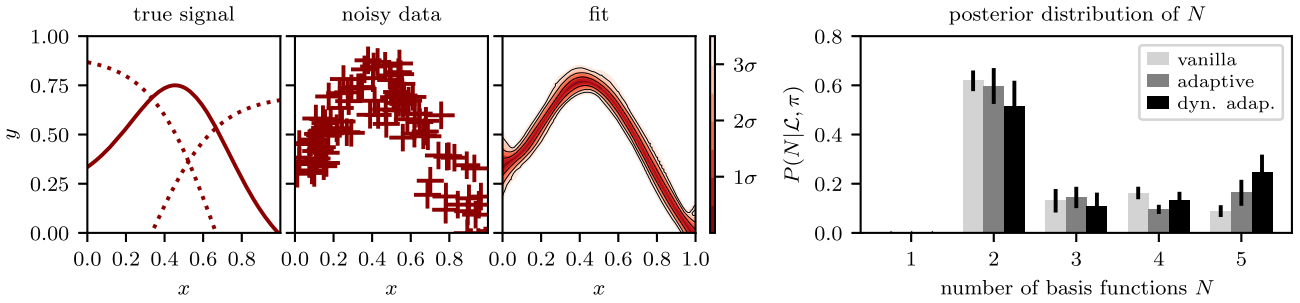


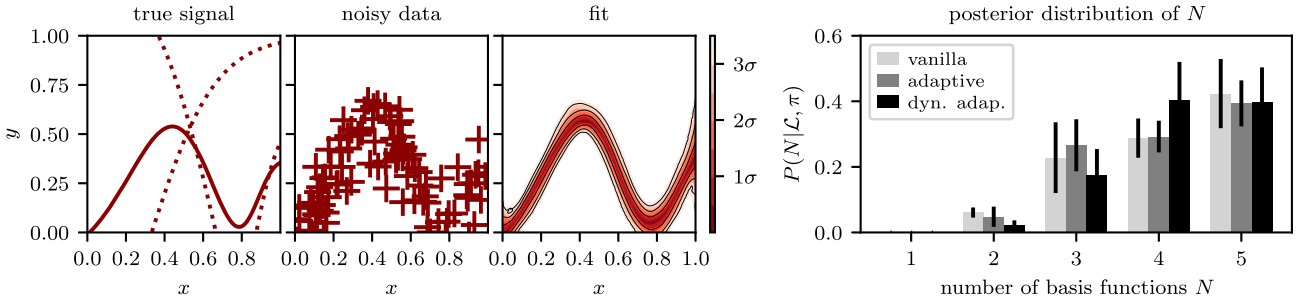
Figure 7. Fits of the data shown in Fig. 6(c) conditioned on different numbers N of basis functions. These plots are made using the vanilla method nested sampling runs, as the adaptive method runs contain relatively few samples from the heavily disfavoured values of N.



(a) Data from a single tanh basis function.



(b) Data from the sum of two tanh basis functions.



(c) Data from the sum of three tanh basis functions.

Figure 8. As for Fig. 6 but using tanh basis functions instead of generalized Gaussians. The parameters of the tanh basis functions in true signal are shown in Table C2 in Appendix C.

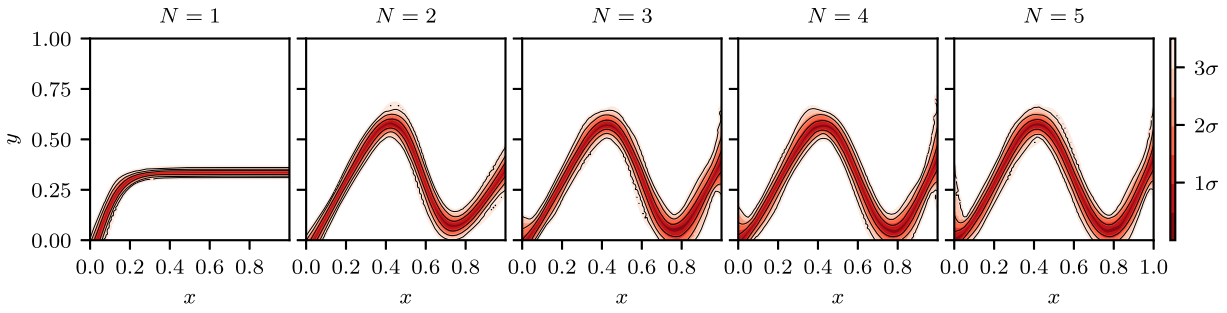
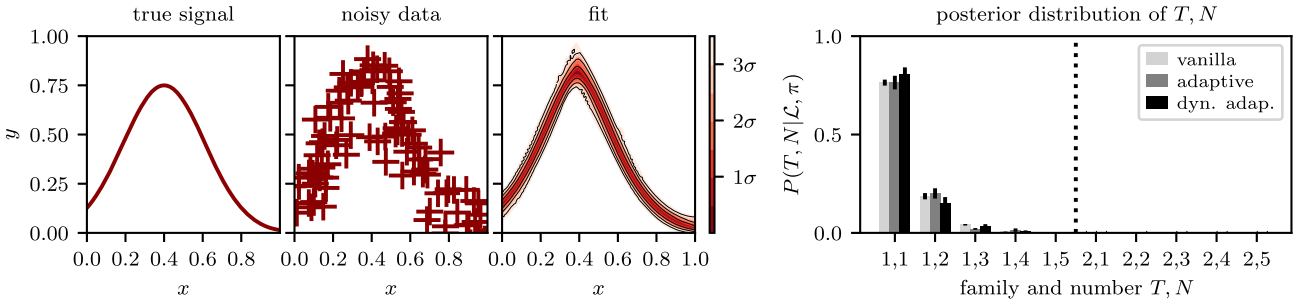


Figure 9. Fits of the data sets shown in Fig. 8(c) conditioned on different numbers N of basis functions.

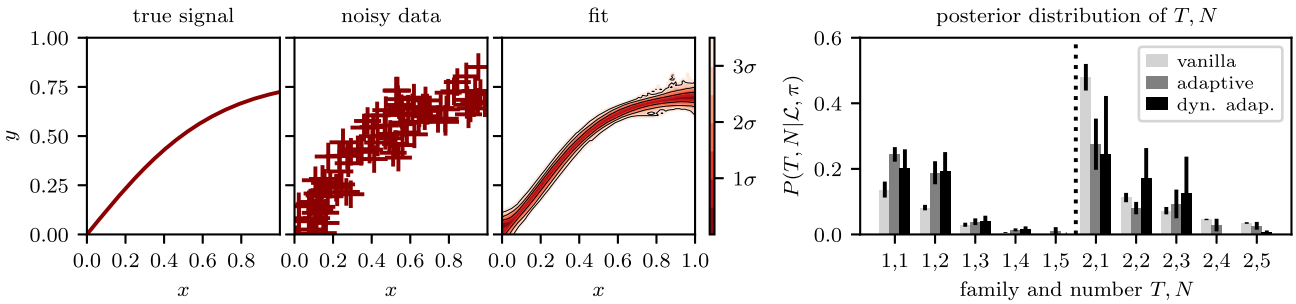
As described in Appendix C, we find that the sampler is not able to explore the parameter space perfectly with the settings used, meaning the true variance of results is higher than the bootstrap estimates. As a result, given the adaptive method's more complex posterior distribution, the efficiency gains of factors of up to 14 ± 3 for the adaptive method and 46 ± 9 for the adaptive method using dynamic nested sampling are likely to be overestimates and are best viewed as an indication of what is possible using the method with more computational power (such as using a higher value for DYPOLYCHORD's `num_repeats` setting).

5 TWO-DIMENSIONAL IMAGE FITTING

We now demonstrate Bayesian sparse reconstruction for monochrome images. Here, for each data point (pixel) d, $\boldsymbol{x}_d = (x_1, x_2)_d$ is the pixel location and $y_d \in [0, 1]$ is the scalar signal. For



(a) Data used in Figure 6a from a single generalised Gaussian.



(b) Data used in Figure 8a from a single tanh function.

Figure 10. Fitting generalized Gaussian and tanh basis functions to data in a fully adaptive manner with the family determined by an integer parameter T. In each row the first plot shows the true signal (a sum of basis functions). The data, shown in the second plot, contains normally distributed x- and y-errors with $\sigma_x = \sigma_y = 0.07$. The third plot shows the fit calculated using the adaptive method with dynamic nested sampling; coloured contours represent posterior iso-probability credible intervals on $y(x)$. The bar plots on the right-hand side display the posterior distribution on different families T and numbers of basis functions N; values calculated using the vanilla method and using the adaptive method with standard nested sampling are also included for comparison. Results for the adaptive method use a combined inference from 5 runs, each of which computes a full posterior on T, N and uses 1000 live points; adaptive runs using dynamic nested sampling have DYPOLYCHORD settings $n_{\text{init}} = 500$ and `dynamic_goal` $= 1$. Results for the vanilla method use separate runs, each with 200 live points, to compute the evidence for each combination of T and N. All runs use the setting `num_repeats` $= 100$.

simplicity we assume that the errors in the pixel positions $\boldsymbol{x}_d$ are negligible, and consider the case that the signal y_d for each pixel contains independent Gaussian noise with size $\sigma_y = 0.2$ – in this case the likelihood is given by (15).

We define two-dimensional generalized Gaussians as the product of two one-dimensional generalized Gaussians (26) rotated by angle Ω around their mean $\boldsymbol{\mu}$:

$$\begin{aligned}\phi^{(\mathrm{g2d})}(\boldsymbol{a}, \boldsymbol{p}) &= \phi^{(\mathrm{g2d})}(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\beta}, \Omega) \\ &= \phi^{(\mathrm{g1d})}(x_1', \mu_1, \sigma_1, \beta_1) \times \phi^{(\mathrm{g1d})}(x_2', \mu_2, \sigma_2, \beta_2) \qquad (29)\end{aligned}$$

$$\text{where} \quad \boldsymbol{x}' = \boldsymbol{\mu} + (\boldsymbol{x} - \boldsymbol{\mu}) \begin{pmatrix} \cos(\Omega) & -\sin(\Omega) \\ \sin(\Omega) & \cos(\Omega) \end{pmatrix}.$$

The priors used are shown in Table 1.

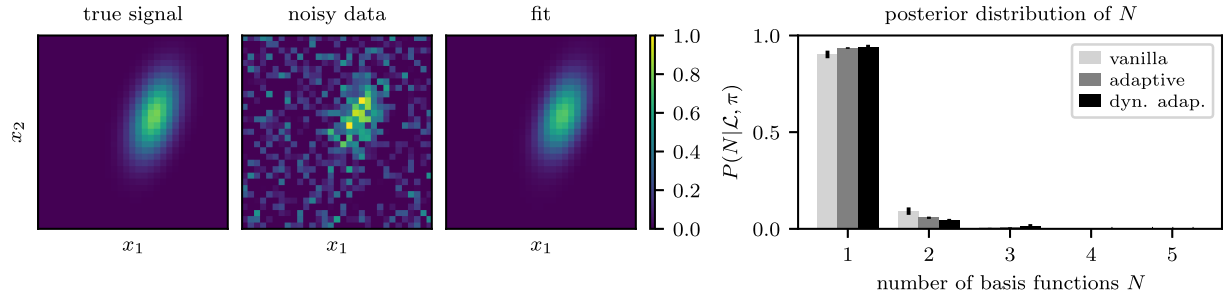
Fig. 11 shows examples of Bayesian sparse reconstruction fitting two-dimensional images. The fits show the mean values predicted for each pixel, averaged over all the samples produced in proportion to their posterior weight. Using the mean value avoids overfitting – which would occur if, for example, the fit was simply calculated from the sample with the highest likelihood (the maximum likelihood estimate). The samples provide a full posterior distribution on the parameters and output signal, so other quantities such as the uncertainty on each pixel can also be easily calculated. Fig. 12 shows fits conditioned on specific values of N, and illustrates how increasing the number of basis functions allows increasingly complex structure to be included in the recovered image.

As in the one-dimensional case, our approach is able to faithfully reconstruct the signal from the noisy data and the numbers of basis functions with the highest posterior probability (shown in the bar charts on the right-hand side of each subfigure) match the number of components in the mixture model used for the signal. Furthermore, Table C5 in Appendix C shows efficiency gains (28) from the adaptive method of up to 10 ± 2 and from the adaptive method with dynamic nested sampling of up to 16 ± 3 in these cases. However, as discussed in Section 4.4, these numbers may overestimate the efficiency gains observed in practice with the settings used.

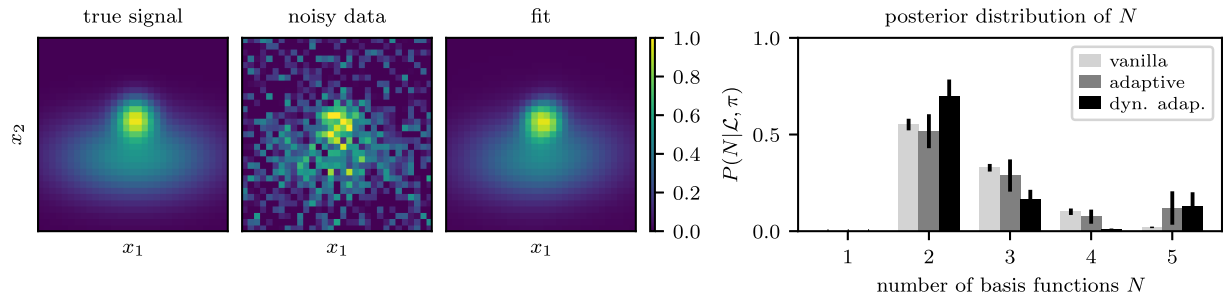
5.1 Application to astronomical images

We now apply the two-dimensional fitting techniques from the previous section to astronomical images from the *Hubble Space Telescope* eXtreme Deep Field (Illingworth et al. 2013). These are not 'true signals' as in the previous examples because the images contain some measurement uncertainty, but this is relatively small compared to our added Gaussian errors of $\sigma_y = 0.2$. We therefore use them as an approximation of a realistic physical signal for testing our method. Furthermore, for this first trial application of our method, we provide only a visual demonstration of the accuracy of our image reconstructions (to be assessed qualitatively). A more quantitative evaluation can be performed in the future using simulations where the noise-free signal values are available.

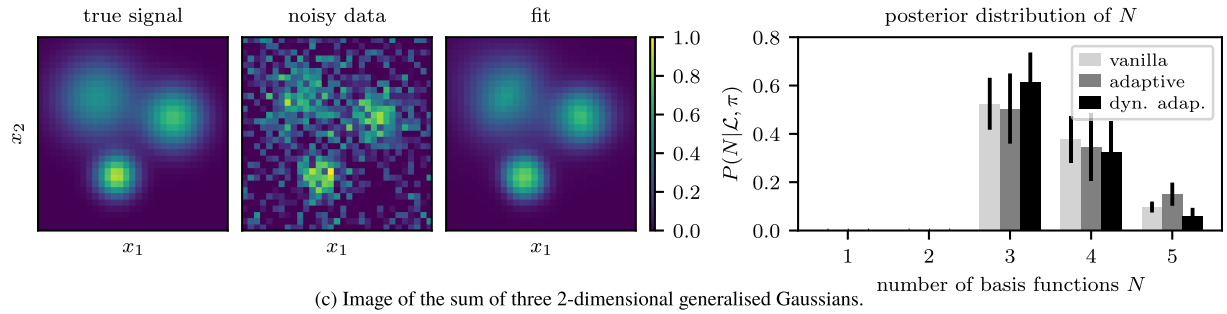
Fig. 13 shows fitting images of galaxies from the Hubble deep field using two-dimensional generalized Gaussians (29), and Fig. 14 shows fits of specific numbers of basis functions (marginalized for



(a) Image of a single 2-dimensional generalised Gaussian.



(b) Image of the sum of two 2-dimensional generalised Gaussians.



(c) Image of the sum of three 2-dimensional generalised Gaussians.

Figure 11. Fitting two-dimensional generalized Gaussian basis functions to 32 × 32 images of mixtures of generalized Gaussians. In each row the 2 plots on the left show the true signal and the data, which has added normally distributed y-errors with $\sigma_y = 0.2$. The third column shows the mean value of $y(\boldsymbol{x})$ from the posterior samples produced using the adaptive method with dynamic nested sampling. The bar plots on the right display the posterior distribution for different numbers of basis functions N; values calculated using the vanilla method and adaptive method without dynamic nested sampling are also included for comparison. Results for the adaptive method show a combined inference from five runs, each of which computes a full posterior on N and uses 2000 live points; adaptive runs using dynamic nested sampling have DYPOLYCHORD settings $n_{\text{init}} = 1000$ and `dynamic_goal` $= 1$. Results for the vanilla method use five separate runs, each with 400 live points, to compute the evidence for each value of N. All runs use the setting `num_repeats` $= 250$. The parameters of the basis functions in the true signal and numerical results for the computational efficiency of the different methods are shown in Tables C3 and C5, respectively, in Appendix C.

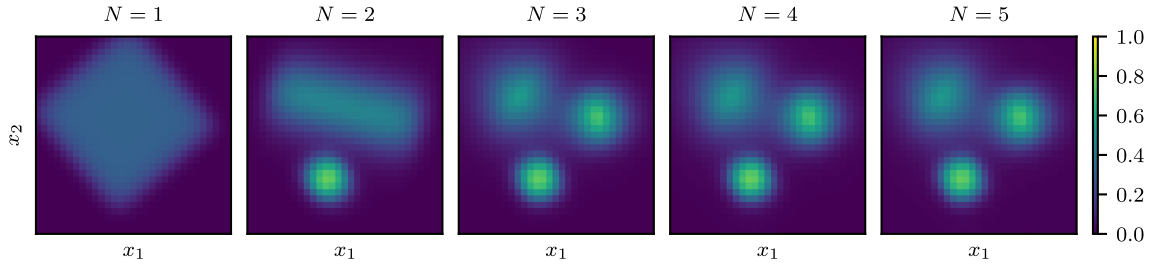
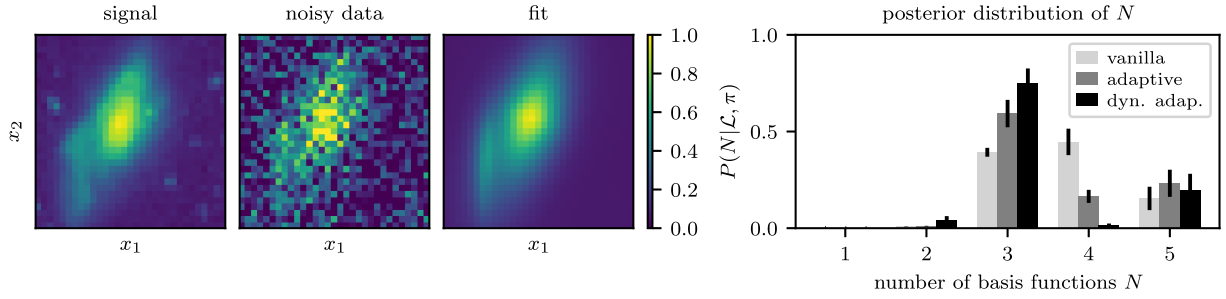
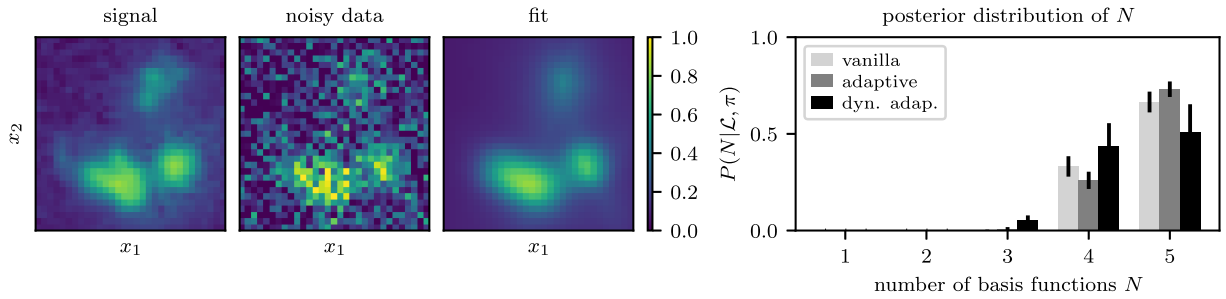


Figure 12. Fits of the data in Fig. 6(c) conditioned on different numbers N of basis functions; these plots use results from the vanilla method.

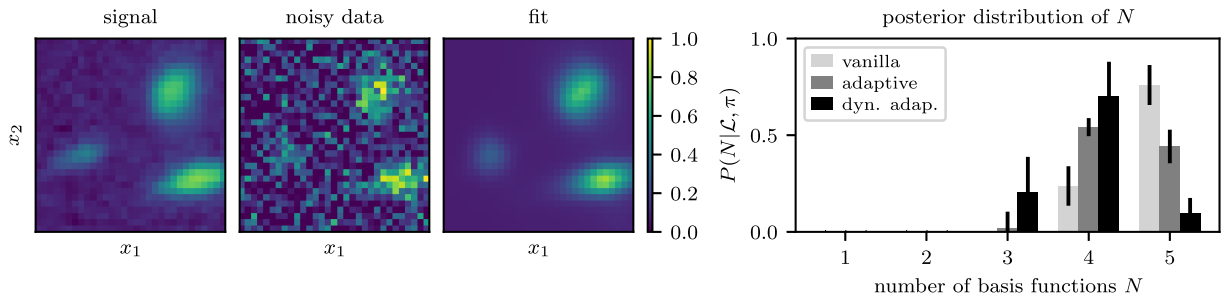
different values of N). Our method is able to faithfully reconstruct the signal from the noisy data, as can be seen from a visual comparison of the fit and the signal. In this case, with the settings used, the posterior distributions of N show some inconsistencies between the different methods. These occur as in order to explore the challenging posterior consistently, POLYCHORD and DYPOLYCHORD require higher live points and/or `num_repeats` setting than those used; this leads to additional random errors. However, this lack of



(a) Image of an irregularly shaped galaxy.



(b) Image containing several galaxies.



(c) Another image containing several galaxies.

Figure 13. As for Fig. 11 but fitting 32 × 32 images from the *Hubble Space Telescope* eXtreme Deep Field (Illingworth et al. 2013); each pixel has added normally distributed y-errors with $\sigma_y = 0.2$.

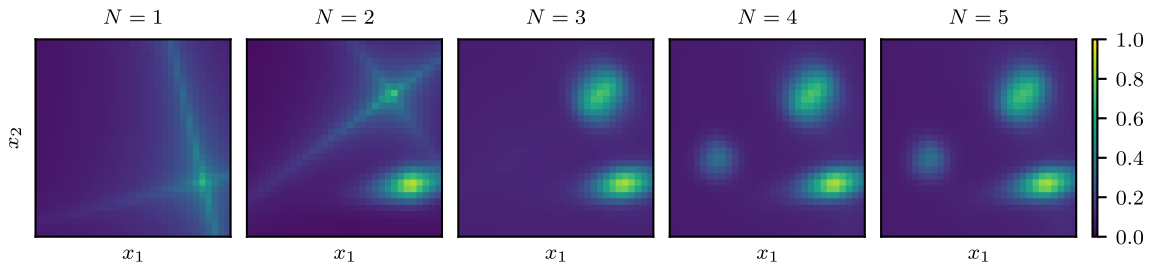


Figure 14. Fits of the data in Fig. 6(c) conditioned on different numbers of basis functions N; these plots use results from the vanilla method.

precision in the posterior probabilities of N has little negative impact on the overall fit, as in each case all the posterior mass is allocated to values of N, which provide good representations of the data.

The posterior probabilities of different values of N (shown on the right of each row of Fig. 13) provide a measure of the complexity of the model justified by the data. However, unless each basis function represents a justified physical model for the sources in the image, N cannot necessarily be interpreted as the number of sources; for example, a single source with a non-Gaussian structure may be represented by several Gaussian basis functions.

6 NEURAL NETWORKS AS ADAPTIVE BASIS REGRESSION

We now apply our Bayesian sparse reconstruction framework to artificial neural networks, where it allows a dynamic selection of the optimum network architecture.

6.1 Background: feed forward neural networks

Artificial neural networks (hereafter neural networks) are a popular machine learning technique loosely inspired by biological brains.

MacKay (2003, Section V) provides a good introduction; for a detailed Bayesian reference, see Neal (2012). Neural networks have been successfully applied to many areas of astronomical data analysis, including image processing (see for example Graff et al. 2014; Ball & Brunner 2010).

Neural networks are made up of nodes ('neurons') that receive input signals and map them to a scalar signal ('activation'), which is then passed to other nodes. We restrict our analysis to 'fully connected' 'feed-forward' networks, in which nodes are arranged in layers and each node receives inputs from every node in the previous layer and passes its output to every node in the following layer (in this case the network is a directed acyclic graph). Layers of nodes between the network's input and output are termed 'hidden layers' as their outputs are not directly specified by the signal.

Following the neural network literature, we denote the activation of the j^{th} node in the l^{th} layer as $a_j^{[l]}$; this is differentiated from the basis function amplitudes used earlier in the paper by the superscript label in square brackets and by the context. The activation of each node is computed as

$$a_j^{[l]} = \phi^{[l]} \left(\sum_{i=1}^{N^{[l-1]}} a_i^{[l-1]} w_{ji}^{[l]} + b_j^{[l]} \right), \tag{30}$$

where $w_{j1}^{[l]}, \ldots, w_{jN^{[l-1]}}^{[l]}$ are the weights assigned to the activations of the $N^{[l-1]}$ nodes in the previous layer and conventionally an additional parameter $b_j^{[l]}$ (referred to as the 'bias') is included. The activation function $\phi^{[l]}$ is typically non-linear function of the inputs such as tanh or rectifier functions.[3] For a feed-forward neural network with a d-dimension input $\boldsymbol{x}$ and one hidden layer containing N nodes, the activations are

$$a_j^{[1]} = \phi^{[1]} \left(\sum_{i=1}^{d} x_i w_{ji}^{[1]} + b_j^{[1]} \right), \tag{31}$$

$$y_j = a_j^{[2]} = \phi^{[2]} \left(\sum_{i=1}^{N} a_i^{[1]} w_{ji}^{[2]} + b_j^{[2]} \right). \tag{32}$$

Such a network with a single output y is illustrated in Fig. 15.

Values for the network parameters can be selected using gradient-based optimization and regularization – this is useful for 'deep learning', in which networks have large numbers of hidden layers (are 'deep') and sampling the posterior distribution over the full parameter space is not computationally feasible. Often some part of the data set is held back and used for selecting the regularization parameter.

Bayesian methods also provide a natural framework for neural networks, and simplified Bayesian computation can be performed in the space of neural network parameter values using techniques such as Bayes by backprop (Blundell et al. 2015) and Gaussian approximations (Mackay 1995). In addition, many regularization techniques commonly applied to neural networks can be interpreted from a Bayesian perspective (see for example Gal 2016).

[3]Rectifier functions such as $\phi(x) = \max(0, x)$ are now popular for deep neural networks, as they make it easier to optimize the network's weights with gradient-based methods because their gradient does not become small when x is large (Lecun, Bengio & Hinton 2015). This paper uses the hyperbolic tangent function as we do not rely on gradient-based optimization and our networks only have one or two hidden layers.

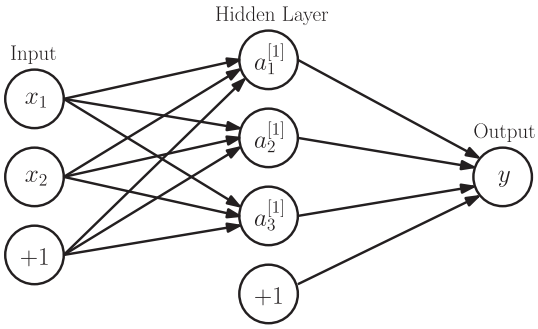


Figure 15. A feed-forward neural network with two inputs, a single hidden layer with three nodes and a scalar output y. Circles labelled $+1$ and the arrows leading from them represent the bias parameters $b_j^{[l]}$.

Table 2. Priors on neural network parameters. Sorted priors have ordering enforced; see Handley et al. (2015a, Appendix A2) for more details.

Parameter	Prior type	Prior parameters
L	Uniform (integer)	$\in \mathbb{Z} \cap [1, 2]$
N	Uniform (integer)	$\in \mathbb{Z} \cap [1, 10]$
σ_w	Uniform in σ_w^{-2} (34)	$\in [0.1, 10]$
Output weights $w_{ij}^{[L+1]}$	Sorted Gaussian[4]	$\mu = 0, \sigma = \sigma_w$
Other weights & biases	Gaussian	$\mu = 0, \sigma = \sigma_w$

6.2 Applying Bayesian sparse reconstruction to neural networks

To illustrate the connection between neural networks and the basis function fitting in previous sections of the paper, consider fitting a scalar signal y using a signal hidden layer neural network in which the output layer has an identity activation function $\phi^{[2]}(x) = x$ and its bias parameter $b^{[2]}$ set to zero. In this case the output (32) is simply a sum of basis functions. If a tanh activation function is used for the nodes in the hidden layer and there is a scalar input x, such a network is equivalent to fitting one-dimensional tanh basis functions as shown in Figs 8 and 9. In this case, determining the value of N using the framework introduced earlier in the paper represents Bayesian inference on the optimum number of nodes in the hidden layer given the data.

When there is more than one hidden layer, the output is no longer a direct sum of the inputs but our Bayesian sparse reconstruction framework can still be readily applied. Furthermore the number of hidden layers L can be determined by treating it as an integer parameter, in the same way the basis function family was represented by the integer parameter T in Section 4.3. We use the same number of nodes N in each hidden layer, but if required one could allow the hidden layers to have different numbers of nodes governed by multiple integer parameters $N^{[1]}, \ldots, N^{[L]}$. We consider only a single output for simplicity, but our results easily generalize to neural networks with multiple outputs $(y_1, y_2, \ldots)$ and to classification problems in which the output takes only discrete values.

We now apply our Bayesian sparse reconstruction framework to neural networks, and show that this approach for principled adaptive Bayesian selection of network architecture without Gaussian approximations works well for 'shallow' neural networks with a small number of hidden layers. We use tanh activation functions for the nodes in the L hidden layers, and a sigmoid activation function

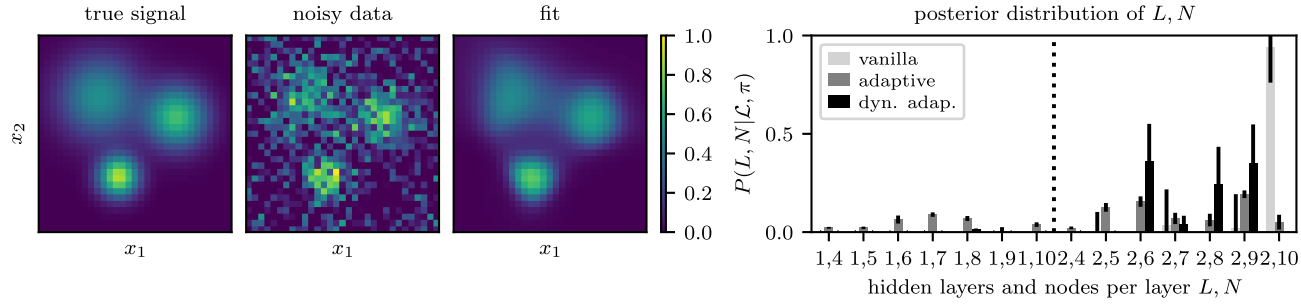


Figure 16. Fitting neural networks with the number of hidden layers L and nodes per hidden layer N determined through Bayesian inference. The first two colour plots show the true signal and the data, which includes added normally distributed y-errors with $\sigma_y = 0.2$; these are the same as in Fig. 11(c). The third colour plot shows the mean value of $y(\boldsymbol{x})$ from the posterior samples produced using the adaptive method with dynamic nested sampling. The bar plot displays the posterior distribution on L, N; values calculated using the vanilla method and the adaptive method without nested sampling are also included for comparison. Bars showing posterior probabilities for $N = 1$, $N = 2$, and $N = 3$ are omitted for brevity as they contain negligible posterior mass for both $L = 1$ and $L = 2$. Adaptive results use a combined inference from five runs, each of which computes a full posterior on L, N and uses 2000 live points; adaptive runs using dynamic nested sampling have DYPOLYCHORD settings $n_{\rm init} = 1000$ and `dynamic_goal` $= 1$. Results for the vanilla method use five separate runs, each with 400 live points, to compute the evidence for each combination L, N. All runs use the setting `num_repeats` $= 250$.

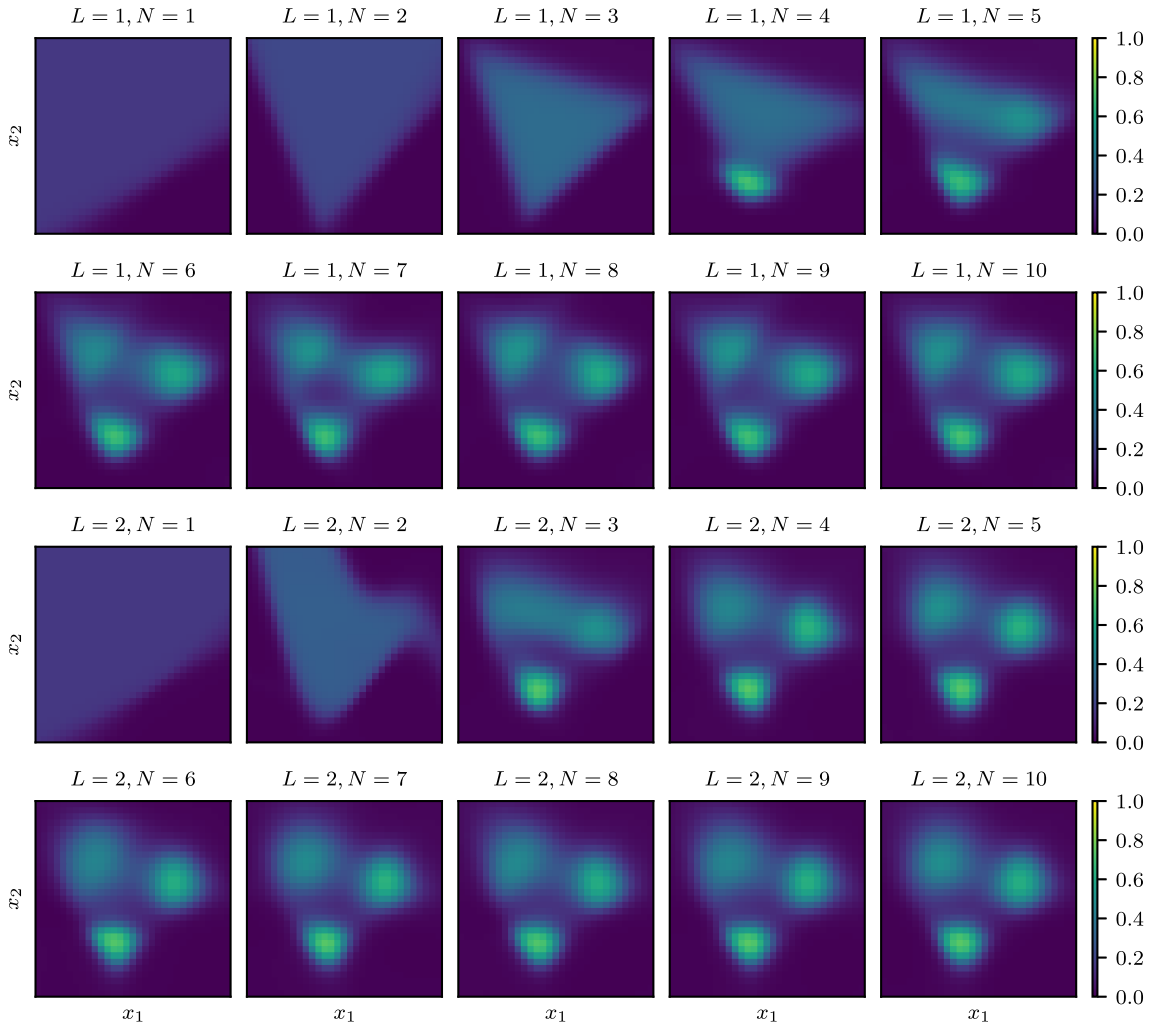


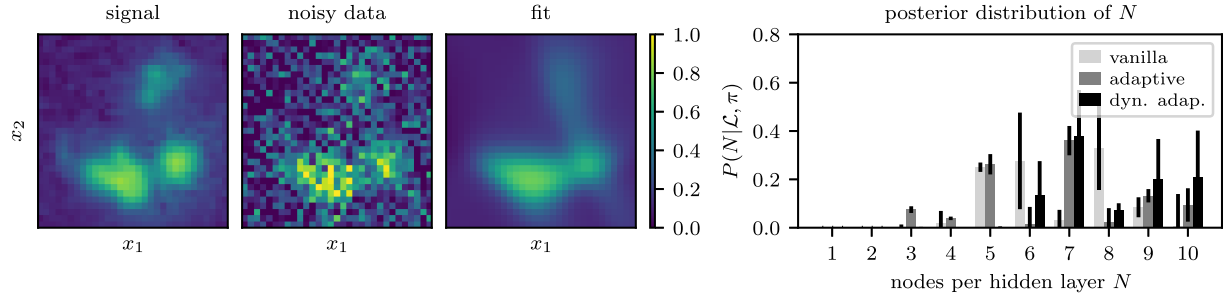
Figure 17. Fits from Fig. 16 conditioned on different numbers of hidden layers L and nodes per hidden layer N. The plots use results from the vanilla method.

for the output

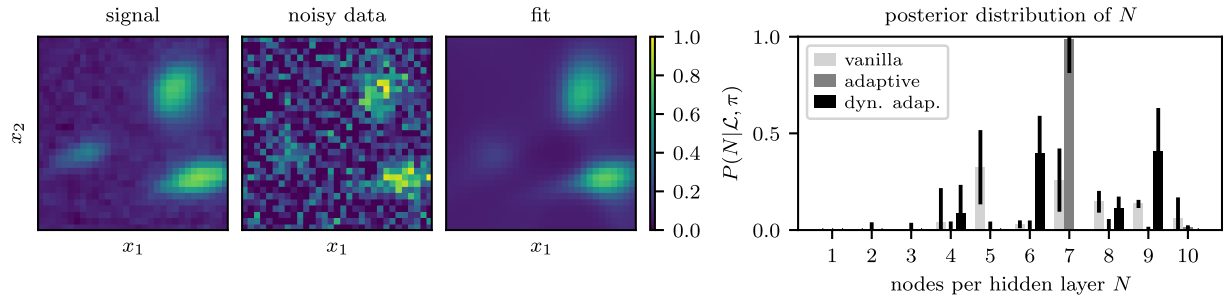
$$\phi^{[L+1]}(x) = \text{sigmoid}(x) = \frac{1}{1+\mathrm{e}^{-x}} = \frac{\mathrm{e}^{x}}{1+\mathrm{e}^{x}}. \tag{33}$$

This conveniently maps the output y into [0,1], which is the range of the target signal in the numerical examples.

We use Gaussian priors on the neural network's weight parameters, as summarized in Table 2. Due to the difficulty in selecting the



(a) Image containing several galaxies.



(b) Another image containing several galaxies.

Figure 18. Fitting 32 × 32 images from the *Hubble Space Telescope* eXtreme Deep Field (Illingworth et al. 2013) using neural networks with two hidden layers. In each row the two plots on the left show the true signal and the data, which includes added normally distributed y-errors with $\sigma_y = 0.2$. The third column shows the mean value of y from the posterior samples produced using the adaptive method with dynamic nested sampling. The bar plots display the posterior distribution for different numbers of nodes per hidden layer N; values calculated using the vanilla method and the adaptive method without dynamic nested sampling are also included for comparison. Adaptive results show a combined inference form five runs, each of which computes a full posterior on N and uses 2000 live points; adaptive runs using dynamic nested sampling have DYPOLYCHORD settings $n_{\text{init}} = 1000$ and `dynamic_goal` $= 1$. Results for the vanilla method use separate runs, each with 400 live points, to compute the evidence for each value of N. All runs use the setting `num_repeats` $= 250$.

priors' scale *a priori*, we use a hyperparameter σ_w for the width of the Gaussian priors on the weights – this can be marginalized out when calculating posterior inferences. Following Mackay (1995), we use a uniform prior on σ_w^{-2}, meaning

$$\pi(\sigma_w) = \frac{3\sigma_w^{-3}}{\sigma_{w,\text{min}}^{-2} - \sigma_{w,\text{max}}^{-2}} \tag{34}$$

where $\sigma_w > 0$.

6.3 Fitting two-dimensional images with neural networks

Fig. 16 shows signal reconstruction with neural networks, including Bayesian inference on the number of hidden layers L and nodes per hidden layer N, using our Bayesian sparse reconstruction framework. Readers who are less familiar with neural networks might expect them to struggle to fit the challenging data set (the same one used in Fig. 11c) using their tanh activation functions. However, we see that our approach yields good results, and the network is able to reconstruct the generalized Gaussians in the signal by overlaying two-dimensional tanh functions from different nodes. How it does this is illustrated in Fig. 17, which shows fits conditioned on different values of L and N. The first two rows with $L = 1$ represent a network with a single hidden layer, and show how increasing N allows the tanh functions to first create a triangle around the three maxima and then to represent the maxima themselves. The second two rows use $L = 2$; a comparison with the $L = 1$ plots shows how the two hidden layer architecture allows more complex signal structure to be represented using a given value of N. The posterior distribution of L heavily favours two hidden layers, with the adaptive method using dynamic nested sampling giving $P(L = 2|\mathcal{L}, \pi) = 0.984 \pm 0.007$.

The network with $L = 2$ hidden layers and $N = 10$ nodes per hidden layer has 151 weight parameters plus the hyperparameter σ_w and the integer parameters L and N; the resulting parameter space is 154-dimensional, as well as highly multimodal and degenerate. The default POLYCHORD and DYPOLYCHORD settings for this dimensionality are $25 \times d = 3,850$ live points and $5 \times d = 770$, so it is not surprising that with the settings used our results show large inconsistencies in the calculated posterior distribution of L and N due to imperfect exploration of the parameter space (see Higson et al. 2019, for a detailed discussion). However our approach is still able to allocate almost all the posterior mass to L, N combinations that are good fits for the data, leading to good results and demonstrating the robustness of the method.

Furthermore, neural network and basis function fits can be compared using the adaptive method. For example, one could include an additional integer parameter T, with values $T = 1$ and $T = 2$ representing fitting with two-dimensional generalized Gaussians and with neural networks, respectively.

6.4 Application to astronomical images

We now apply neural networks to the *Hubble Space Telescope* eXtreme Deep Field images used in Section 5.1. We find the adaptive selection of L for these data sets strongly favours $L = 2$ over $L = 1$, so for brevity we show only results using 2 hidden layers.

Fig. 18 shows results from fitting neural networks with two hidden layers to the data used in Fig. 13, with fits conditioned on specific

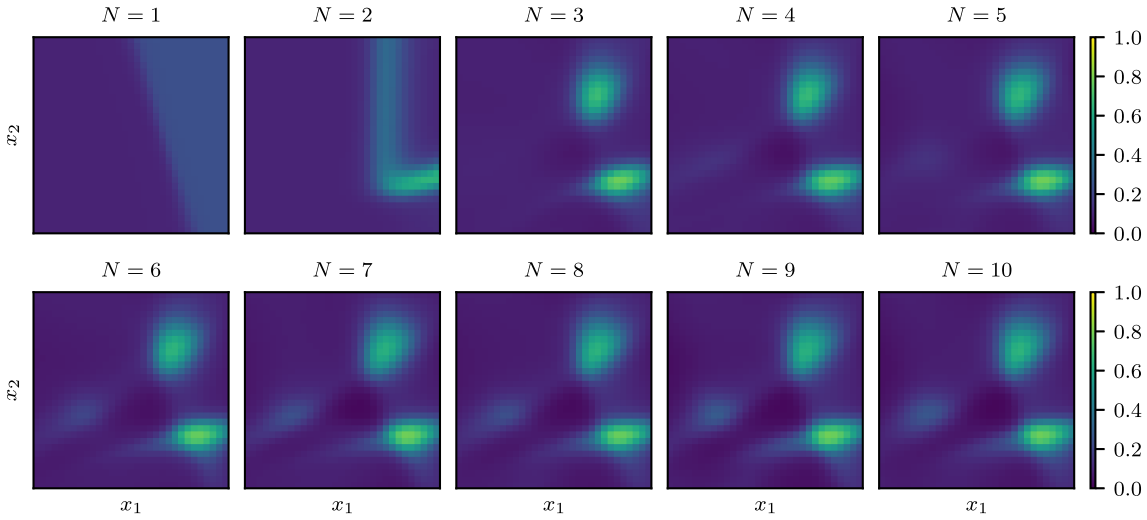


Figure 19. Fits of the data sets shown in Fig. 18(b) conditioned on different numbers of nodes per hidden layer N. The plots use results from the vanilla method.

values of N shown in Fig. 19. As for the two-dimensional Gaussian basis functions, a visual assessment shows the neural networks are able to faithfully reconstruct the true image from the noisy data with good accuracy. However the neural networks (with tanh activation functions) do not provide as natural a representation of the blob-shaped sources as the two-dimensional generalized Gaussians, so the fits are not as good as those shown in Figs 13 and 14. Nevertheless the example provides a proof of principle, and the versatility of neural networks means that they can be applied to a wide range of data sets using this technique.

The posterior distributions of the number of nodes in each hidden layer N, shown on the right-hand side of each row of plots in Fig. 18, illustrate the number of nodes (degree of complexity of the model), which is justified by the astronomical image. However, unlike the basis functions, the contributions of each individual node to the output fit are not readily interpretable. As in the previous section, we find that the network's fits of the images are good – despite inconsistencies in the posterior probabilities of different values of N between the different methods due to the sampler imperfectly exploring the parameter space. The posterior distribution of N can be calculated more precisely using more computational resources (for example by increasing PolyChord and dyPolyChord's `num_repeats` settings).

7 CONCLUSION

We have introduced Bayesian sparse reconstruction, a principled framework for signal reconstruction that allows the model's complexity to be determined by the data. The Bayesian calculations naturally penalize overcomplex models, and the priors can be used to specify the degree to which sparse solutions should be favoured. Our approach performs well at fitting noisy one- and two-dimensional test data from mixture models, as well as reconstructing astronomical images. We further showed the framework naturally applies to neural networks, where it allows Bayesian inference to be performed over the space of possible network architectures by treating the number of nodes and hidden layers as parameters.

While the techniques described in this paper are computationally expensive (see Appendix B for details of the compute used to produce our results), we show that they are now feasible in the low data regime with current software and are capable of producing excellent results. Furthermore, we intend this paper to provide a proof of principle for future application of our approach to larger data sets, when advances in numerical techniques and increases in computational power make this feasible.

ACKNOWLEDGEMENTS

This work was performed using the Darwin Supercomputer of the University of Cambridge High Performance Computing Service (http://www.hpc.cam.ac.uk/), provided by Dell Inc. using Strategic Research Infrastructure Funding from the Higher Education Funding Council for England and funding from the Science and Technology Facilities Council.

REFERENCES

Ables J. G., 1974, A&AS, 15, 383
Bach F., Jenatton R., Mairal J., Obozinski G., 2012, Found. Trends Mach. Learn., 4, 1
Ball N. M., Brunner R. J., 2010, Int. J. Mod. Phys. D, 19, 1049
Bayes T., Price R., 1763, Phil. Trans. R. Soc. Lond., 53, 370
Blundell C., Cornebise J., Kavukcuoglu K., Wierstra D., 2015, in Proceedings of the 32nd International Conference on Machine Learning. PMLR, preprint (arXiv:1505.05424)
Bobin J., Starck J. L., Ottensamer R., 2008, IEEE J. Sel. Top. Signal Process., 2, 718
Cai X., Pereyra M., McEwen J. D., 2018, MNRAS, 480, 4154
Chua A. J. K., Hee S., Handley W. J., Higson E., Moore C. J., Gair J. R., Hobson M. P., Lasenby A. N., 2018, MNRAS, 478, 28
Elad M., Matalon B., Shtok J., Zibulevsky M., 2007, Proc. SPIE, Wavelets XII, vol. 6701, SPIE, Bellingham, p. 1
Eldar Y., Kutyniok G., 2012, Compressed Sensing: Theory and Applications. Cambridge Univ. Press, Cambridge
Feroz F., Hobson M. P., 2008, MNRAS, 384, 449
Feroz F., Hobson M. P., Bridges M., 2008, MNRAS, 398, 1601
Feroz F., Hobson M. P., Cameron E., Pettitt A. N., 2013, preprint (arXiv:1306.2144)
Gal Y., 2016, in International Conference on Machine Learning, Available at: http://www.jmlr.org/proceedings/papers/v48/gal16.pdf

Graff P., Feroz F., Hobson M. P., Lasenby A., 2014, MNRAS, 441, 1741
Green P. J., 1995, Biometrika, 82, 711
Gull S. F., Daniell G. J., 1978, Nature, 272, 686
Hadamard J., 1902, Princeton Univ. Bull., 13, 49
Handley W., 2018, J. Open Sour. Softw., 3, 849
Handley W., Hobson M., Lasenby A., 2015a, MNRAS, 15, 1
Handley W., Hobson M., Lasenby A., 2015b, MNRAS, 450, L61
Hee S., Handley W., Hobson M., Lasenby A., 2016, MNRAS, 455, 2461
Hee S., Vázquez J., Handley W., Hobson M., Lasenby A., 2017, MNRAS, 466, 369
Higson E., 2018a, J. Open Sour. Softw., 3, 916
Higson E., 2018b, J. Open Sour. Softw., 3, 965
Higson E., 2018c, J. Open Sour. Softw., 3, 985
Higson E., Handley W., Hobson M., Lasenby A., 2017, preprint (arXiv:1704.03459)
Higson E., Handley W., Hobson M., Lasenby A., 2018, Bayesian Anal., 13, 873
Higson E., Handley W., Hobson M., Lasenby A., 2019, MNRAS, 483, 2044
Hobson M., Jones A., Lasenby A., Bouchet F., 1998, MNRAS, 300, 1
Hoerl A. E., Kennard R. W., 1970, Technometrics, 12, 55
Illingworth G. D. et al., 2013, ApJS, 209, 6
Ji S., Xue Y., Carin L., 2008, IEEE Trans. Signal Process., 56, 2346
Jones D. M., Heavens A. F., 2019, MNRAS, 483, 2487
Lasenby A. N., Barreiro R. B., Hobson M. P., 2001, Mining the Sky. Springer, Berlin, p. 15
Lecun Y., Bengio Y., Hinton G., 2015, Nature, 521, 436
Mackay D. J. C., 1995, Netw. Comput. Neural Syst., 6, 469
MacKay D. J. C., 2003, Information Theory, Inference, and Learning Algorithms. Cambridge Univ. Press, Cambridge
Maisinger K., Hobson M. P., Lasenby A. N., 2004, MNRAS, 347, 339
Mallat S. G., Zhang Z., 1993, IEEE Trans. Signal Process., 41, 3397
McEwen J. D., Wiaux Y., 2011, MNRAS, 413, 1318
Neal R., 2012, Bayesian Learning for Neural Networks. Springer Science & Business Media
Parkinson D., Liddle A. R., 2013, Stat. Anal. Data Min. ASA Data Sci. J., 6, 3
Rasmussen C., 2004, Advanced Lectures on Machine Learning. Springer, Berlin, p. 63
Sciacchitano F., Lugaro S., Sorrentino A., 2018, preprint (arXiv:1807.11287)
Sivia D., Skilling J., 2006, Data Analysis: A Bayesian Tutorial. Oxford University Press, Oxford
Skilling J., 2006, Bayesian Anal., 1, 833
Tibshirani R., 1996, J. R. Stat. Soc. B, 58, 267
Tipping M., 2001, J. Mach. Learn. Res., 1, 211
Warren H. P., Byers J. M., Crump N. A., 2017, ApJ, 836, 215
Wiaux Y., Jacques L., Puy G., Scaife A. M. M., Vandergheynst P., 2009, MNRAS, 395, 1733
Wiener N., 1949, The Interpolation, Extrapolation and Smoothing of Stationary Time Series. MIT Press , Cambridge, MA
Wipf D. P., Rao B. D., 2004, IEEE Trans. Signal Process., 52, 2153

APPENDIX A: CODE

The code used to make the results and plots in this paper can be downloaded at https://github.com/ejhigson/bsr.

APPENDIX B: COMPUTATIONAL RESOURCES USED

Table B1 shows the approximate number of core hours used for each calculation in this paper, and is intended to provide a rough guide to the computational cost of our method. We used the CDS3 Peta4 cluster, which has 2.6GHz 16-core Intel Xeon Skylake 6142 processors (2 processors and 32 cores per node). Note that the number of core hours used can vary significantly when the same calculation is repeated.

Table B1. Approximate numbers of core hours used per calculation for results shown in the paper; these were run on the CDS3 Peta4 cluster, which has 2.6GHz 16-core Intel Xeon Skylake 6142 processors (2 processors and 32 cores per node). For adaptive results, each calculation is a single nested sampling run. For vanilla runs a calculation involves a separate nested sampling run for each value of N – the values of the table show the total core hours used by these. For calculations fitting basis functions to one-dimensional signals (Figs 6, 8, and 10) `num_repeats` = 100, vanilla runs use 200 live points and adaptive runs use 1000. For calculations fitting two-dimensional images (Figs 11, 13, 16, and 18) `num_repeats` = 250, vanilla runs use 400 live points and adaptive runs use 2000. Note that plots of results all use combined inferences from five calculations.

	Vanilla	Adaptive	Dynamic adaptive
Fitting 1*d* generalized Gaussians (Fig. 6)			
Fig. 6(a)	1	1	3
Fig. 6(b)	1.5	1.5	4
Fig. 6(c)	2	2	4
Fitting 1*d* tanhs (Fig. 8)			
Fig. 8(a)	1	1	3
Fig. 8(b)	1.5	1.5	4
Fig. 8(c)	3	3	5
Fitting 1*d* basis functions with adaptive T (Fig. 10)			
Fig. 10(b)	2	20	20
Fig. 10(a)	2	20	20
Fitting 2*d* generalized Gaussians (Figs 11 and 13)			
Fig. 11(a)	5	7	50
Fig. 11(b)	10	14	70
Fig. 11(c)	12	20	80
Fig. 13(a)	8	26	70
Fig. 13(b)	11	60	100
Fig. 13(c)	10	40	80
Fitting neural networks with adaptive L (Fig. 16)			
Fig. 16	150	200	300
Fitting neural networks with 2 hidden layers (Fig. 18)			
Fig. 18(a)	100	100	200
Fig. 18(b)	100	100	200

Vanilla and adaptive calculations use PolyChord and dynamic adaptive calculations use dyPolyChord. dyPolyChord performs dynamic nested sampling by saving and resuming PolyChord runs; this is not yet parallelized in the current version of PolyChord and can become a bottleneck when running with large numbers of processes, increasing the amount of core hours required for this method. We intend this process to be more computationally efficient in future dynamic nested sampling software. All calculations use C++ likelihoods except the adaptive and dynamic adaptive selection of T in Fig. 8, which were run using a python likelihood and consequently required more computation time. The code used can be downloaded from the link in Appendix A.

When fitting the same basis functions to different data sets, reconstructing more complex signals requires more computation – this can be seen in Table B1 for Figs 6(a)–(c) and Figs 8(a)–(c).

APPENDIX C: ADDITIONAL NUMERICAL RESULTS

This Appendix contains details of the parameters of the mixture models used to generate true signals in the numerical examples,

Table C1. Parameters for the sum of one-dimensional generalized Gaussian basis functions (26) from which the data shown in Fig. 6 were sampled.

# functions	a	μ	σ	β
1	0.75	0.4	0.3	2
2	0.2	0.4	0.6	5
	0.55	0.4	0.2	4
3	0.2	0.4	0.6	5
	0.35	0.6	0.07	2
	0.55	0.32	0.14	6

Table C2. Parameters for the sum of one-dimensional tanh basis functions (27) from which the data shown in Fig. 8 were sampled.

# functions	a	b	w
1	0.8	0	1.5
2	0.7	−1	3
	0.9	2	−3
3	0.6	−7	8
	1	−1	3
	1.4	2	−3

Table C3. Parameters for the sum of two-dimensional generalized Gaussian basis functions (29) from which the data shown in Fig. 11 were sampled.

# functions	a	μ_1	μ_2	σ_1	σ_2	β_1	β_2	Ω
1	0.8	0.6	0.6	0.1	0.2	2	2	$\pi/10$
2	0.5	0.5	0.4	0.4	0.2	2	2	0
	0.8	0.5	0.6	0.1	0.1	2	2	0
3	0.5	0.3	0.7	0.2	0.2	2	2	0
	0.7	0.7	0.6	0.15	0.15	2	2	0
	0.9	0.4	0.3	0.1	0.1	2	2	0

as well as tables comparing the computational efficiency of results calculated through the adaptive and vanilla methods.

C1 Parameters for test signals

Tables C1–C3 show the parameters of the mixture models used for the signals in Figs 6, 8, and 11, respectively.

C2 Efficiency gain results

Tables C4 and C5 show numerical values for the mean fit at the centre of the signal's domain for Figs 6 and 11, as well as estimates of the efficiency gain (28) from the adaptive method (with and without dynamic nested sampling) compared to the vanilla method. Efficiency gains reported using results' estimated variation, calculated from bootstrap resampling using then NESTCHECK package (Higson 2018a).

Bootstrap resampling allows the variation of results due to the stochasticity of the nested sampling algorithm to be determined accurately without the need to repeat the calculation many times. However it assumes that the nested sampling algorithm was performed perfectly; for some special cases this is possible (see for example Higson 2018c), but in practice for challenging posterior distributions there may be additional errors – for example due to software producing correlated samples, or missing a mode in a multimodal posterior. These additional errors are discussed in detail in Higson et al. (2018), and can be reduced by changing the software settings; for POLYCHORD and DYPOLYCHORD this entails increasing the `num_repeats` setting and/or the number of live points. Diagnostics provided by NESTCHECK indicate the presence of such

Table C4. Numerical values for the accuracy of the mean fit at $x = 0.5$ using the data and nested sampling runs shown in Fig. 6. The columns show results for the vanilla and adaptive methods using standard nested sampling and the adaptive method using dynamic nested sampling. For each data set, the first two rows show the total number of samples used by the nested sampling runs, and the mean value of $y(0.5;\boldsymbol{\theta})$. The next two rows show the efficiency gain (28) of the adaptive method with and without dynamic nested sampling; these are calculated using estimates of the standard deviation of results from bootstrap resampling 100 bootstrap replications. The numbers in brackets show 1σ errors on the final digit.

	Vanilla	Adaptive	Dynamic adaptive
Data from 1 generalized Gaussian (shown in Fig. 6a)			
# samples	128 719	114 507	116 867
$y(0.5;\boldsymbol{\theta})$	0.6526(4)	0.6517(3)	0.6527(1)
Efficiency gain		1.7(4)	46(9)
Data from 2 generalized Gaussians (shown in Fig. 6b)			
# samples	128 052	133 061	131 637
$y(0.5;\boldsymbol{\theta})$	0.6340(3)	0.6351(1)	0.6338(1)
Efficiency gain		14(3)	4.3(9)
Data from 3 generalized Gaussians (shown in Fig. 6(c))			
# samples	152 516	171 853	173 959
$y(0.5;\boldsymbol{\theta})$	0.4040(3)	0.4029(2)	0.4072(2)
Efficiency gain		2.5(5)	2.7(6)

Table C5. Numerical values for the accuracy of the mean fit at $\boldsymbol{x} = (0.5, 0.5)$ using the data and nested sampling runs shown in Fig. 11. The columns show results for the vanilla and adaptive methods using standard nested sampling and the adaptive method using dynamic nested sampling. For each data set, the first two rows show the total number of samples used by the nested sampling runs, and the mean value of $y(0.5, 0.5;\boldsymbol{\theta})$ produced. The next two rows show the efficiency gain (28) of the adaptive method with and without dynamic nested sampling; these are calculated using estimates of the standard deviation of results from bootstrap resampling. The numbers in brackets show 1σ errors on the final digit.

	Vanilla	Adaptive	Dynamic adaptive
Data from 1 $2d$ generalized Gaussian (shown in Fig. 11a)			
# samples	377 360	324 294	322 799
$y(0.5, 0.5;\boldsymbol{\theta})$	0.3353(3)	0.3360(2)	0.3359(1)
Efficiency gain		3.7(7)	16(3)
Data from 2 $2d$ generalized Gaussians (shown in Fig. 11b)			
# samples	476 932	520 691	503 380
$y(0.5, 0.5;\boldsymbol{\theta})$	0.6850(5)	0.6853(2)	0.6856(1)
Efficiency gain		10(2)	12(2)
Data from 3 $2d$ generalized Gaussians (shown in Fig. 11c)			
# samples	562 830	652 359	656 852
$y(0.5, 0.5;\boldsymbol{\theta})$	0.1435(1)	0.1438(1)	0.1437(1)
Efficiency gain		1.2(2)	4.3(9)

additional variation in our results; it also explains how estimates of the fit $y(0.5; \boldsymbol{\theta})$ and $y(0.5, 05; \boldsymbol{\theta})$ using different methods (shown in Table C4 and C5, respectively) sometimes differ by slightly more than would be expected from their bootstrap uncertainties. As the posterior is more challenging and complex in the adaptive method than the vanilla method, this is likely to mean that the efficiency gain observed in practice is lower than the estimates using the bootstrap estimates of variation with the settings we use. However we include it as a rough estimate and an indication of the efficiency gain, which could be achieved with more live points and/or a higher `num_repeats` setting.

This paper has been typeset from a TEX/LATEX file prepared by the author.