Monthly Notices
of the
ROYAL ASTRONOMICAL SOCIETY

MNRAS **455,** 2387–2401 (2016) doi:10.1093/mnras/stv2425

# A sparse Gaussian process framework for photometric redshift estimation

Ibrahim A. Almosallam,[1,2]⋆ Sam N. Lindsay,[3] Matt J. Jarvis[3,4] and Stephen J. Roberts[2]

[1]*King Abdulaziz City for Science and Technology, Riyadh 11442, Saudi Arabia*
[2]*Information Engineering, Parks Road, Oxford OX1 3PJ, UK*
[3]*Oxford Astrophysics, Department of Physics, Keble Road, Oxford OX1 3RH, UK*
[4]*Department of Physics, University of the Western Cape, Bellville 7535, South Africa*

Accepted 2015 October 19. Received 2015 October 16; in original form 2015 May 9

**ABSTRACT**

Accurate photometric redshifts are a lynchpin for many future experiments to pin down the cosmological model and for studies of galaxy evolution. In this study, a novel sparse regression framework for photometric redshift estimation is presented. Synthetic data set simulating the *Euclid* survey and real data from SDSS DR12 are used to train and test the proposed models. We show that approaches which include careful data preparation and model design offer a significant improvement in comparison with several competing machine learning algorithms. Standard implementations of most regression algorithms use the minimization of the sum of squared errors as the objective function. For redshift inference, this induces a bias in the posterior mean of the output distribution, which can be problematic. In this paper, we directly minimize the target metric $\Delta z = (z_s - z_p)/(1 + z_s)$ and address the bias problem via a distribution-based weighting scheme, incorporated as part of the optimization objective. The results are compared with other machine learning algorithms in the field such as artificial neural networks (ANN), Gaussian processes (GPs) and sparse GPs. The proposed framework reaches a mean absolute $\Delta z = 0.0026(1 + z_s)$, over the redshift range of $0 \leq z_s \leq 2$ on the simulated data, and $\Delta z = 0.0178(1 + z_s)$ over the entire redshift range on the SDSS DR12 survey, outperforming the standard ANNz used in the literature. We also investigate how the relative size of the training sample affects the photometric redshift accuracy. We find that a training sample of >30 per cent of total sample size, provides little additional constraint on the photometric redshifts, and note that our GP formalism strongly outperforms ANNz in the sparse data regime for the simulated data set.

**Key words:** methods: data analysis – galaxies: distances and redshifts.

## 1 INTRODUCTION

The radial component of the position of a distant object is inferred from its cosmological redshift, induced by the expansion of the Universe; the light observed from a distant galaxy appears to us at longer wavelengths than in the rest frame of that galaxy. The most accurate determination of the exact redshift, $z$, comes from directly observing the spectrum of an extragalactic source and measuring a consistent multiplicative shift, relative to the rest frame, of various emission (or absorption) features. The rest-frame wavelengths of these emission lines are known to a high degree of accuracy which can be conferred on to the measured spectroscopic redshifts, $z_s$. However, the demand on telescope time to obtain spectra for every source in deep, wide surveys is prohibitively high, and only relatively small area spectroscopic campaigns can reach faint magnitudes (e.g. Lilly et al. 2009; Le Fèvre et al. 2013, 2015), or at the other extreme, relatively bright magnitudes over larger areas (e.g. Colless et al. 2003; Driver et al. 2011; Alam et al. 2015). This forces us towards the use of photometric observations to infer the redshift by other means. Rather than individual spectra, the emission from a distant galaxy is observed in several broad filters, facilitating the characterization of the spectral energy distribution (SED) of fainter sources, at the expense of fine spectral resolution.

Photometric redshift methods largely fall into two categories, based on either SED template fitting or machine learning. Template fitting software such as HYPERZ; (Bolzonella, Miralles & Pelló 2000), ZEBRA; (Feldmann et al. 2006), EAZY; (Brammer, van Dokkum & Coppi 2008) and LE PHARE (Ilbert et al. 2006) rely on a library of SED templates for a variety of different galaxy types, which

⋆ E-mail: ialmosallam@kacst.edu.sa

(given the transmission curves for the photometric filters being used) can be redshifted to fit the photometry. This method can be refined in various ways, often with the use of simulated SEDs rather than only those observed at low redshift, composite SEDs, and through calibration using any available spectroscopic redshifts. Machine learning methods such as artificial neural networks (e.g. ANNz; Firth, Lahav & Somerville 2003; Collister & Lahav 2004), nearest-neighbour (NN; Ball et al. 2008), genetic algorithms (e.g. Hogan, Fairbairn & Seeburn 2015), self-organized maps (Geach 2012) and random forest (Kind & Brunner 2013), to name but a few, rely on a significant fraction of sources in a photometric catalogue having spectroscopic redshifts. These 'true' redshifts are used to train the algorithm. In addition to providing a point estimate, machine learning methods can provide the degree of uncertainty in their prediction (Kind & Brunner 2013; Bonnett et al. 2015; Rau et al. 2015).

Both methods have their strengths and weaknesses, with the best performance often depending on the available data and the intended science goals. As such, future surveys may well depend on contributions from both in tandem, but there has been extensive work on comparing the current state of the art in public software using a variety of techniques (Hildebrandt et al. 2010; Abdalla et al. 2011; Sánchez et al. 2014; Bonnett et al. 2015). Artificial neural networks motivate the most commonly used machine learning software (Firth et al. 2003; Vanzella et al. 2004; Brescia et al. 2014), however Gaussian processes (GPs; e.g. Way et al. 2009) have not yet become well established in this area, despite comparison by Bonfield et al. (2010) suggesting that they may outperform the popular ANNz code, using the rms error as a metric.

In this paper, we introduce a novel sparse kernel regression model that greatly reduces the number of basis (kernel) functions required to model the data considered in this paper. This is achieved by allowing each kernel to have its own hyperparameters, governing its shape. This is in contrast to the standard kernel-based models in which a set of global hyperparameters are optimized (such as is typical in GP methods). The complexity cost of such a kernel-based regression model is $O(n^3)$, where $n$ is the number of basis functions. This cubic time complexity arise from the cost of inverting an $n$ by $n$ covariance matrix. In a standard GP model (Rasmussen & Williams 2006), seen as a kernel regression algorithm, we may regard the basis functions, as located at the $n$ points in the training sample. This renders such an approach unusable for many large training data applications where scalability is a major concern. Much of the work done to make GPs more scalable is either to (a) make the inverse computation faster or (b) use a smaller representative set from the training sample to reduce the rank and ease the computation of the covariance matrix. Examples of (a) include methods such as structuring the covariance matrix such that it is much easier to invert, using Toeplitz (Zhang, Leithead & Leith 2005) or Kronecker decomposition (Tsiligkaridis & Hero 2013), or inverse approximation as an optimization problem (Gibbs & MacKay 1997). To reduce the number of representative points (b), an $m \ll n$ subset of the training sample can be selected which maximizes the accuracy or the numerical stability of the inversion (Foster et al. 2009). Alternatively, one may search for 'inducing' points not necessarily present in the training sample, and not necessarily even lying within the data range, to use as the basis set such that the probability of the data being generated from the model is maximized (Snelson & Ghahramani 2006). Approaches such as relevance vector machines (RVM; Tipping 2001) and support vector machines (SVM; Drucker et al. 1997) are basis-function models. However, unlike sparse GPs, they do not learn the basis functions' locations but rather apply shrinkage to a set of kernels in the form of weight-decay on the linear weights that couple the kernels, located at training data points, to the regression.

In this paper, we propose a non-stationary sparse Gaussian model to target photometric redshift estimation. The key difference between the proposed approach and other basis function models, is that our model does not use shrinkage (automatic relevance determination) external to the kernel, but instead has a length-scale parameter in each kernel. This allows for parts of the input–output regression mapping to have different characteristic length-scales. We can see this as allowing for shrinkage and reducing the need for more basis functions, as well as allowing for non-stationary mappings. A regular GP, sparse GP or RVM does not do this, and we demonstrate that this is advantageous to photometric redshift estimation. Furthermore, the model is presented within a framework with components that address other challenges in photometric redshift estimation such as incorporating a weighting scheme as an integral part of the process to remove, or introduce, any systematic bias, and a prior mean function to enhance the extrapolation performance of the model. The results are demonstrated on photometric redshift estimation for a simulated *Euclid*-like survey (Laureijs et al. 2011) and on observational data from the 12th Data Release of the Sloan Digital Sky Survey (SDSS; Alam et al. 2015).[1] In particular, we use the weighting scheme to remove any distribution bias and introduce a linear bias to directly target the mission's requirement.

The paper is organized as follows, a brief introduction to GPs for regression is presented in Section 2 followed by an introduction to sparse GPs in Section 3. The proposed approach is described in Section 4 followed by an application to photometric redshift estimation in Section 5, where the details of the mock data set are described. The experiments and results are discussed in Section 6 on the simulated survey, and in Section 7 we demonstrate the performance of the proposed model and compare it to ANNz on the SDSS 12th Data Release. Finally, we summarize and conclude in Section 8.

## 2 GAUSSIAN PROCESSES

In many modelling problems, we have little prior knowledge of the explicit functional form of the function that maps our observables on to the variable of interest. Imposing, albeit sensible, parametric models, such as polynomials, makes a tacit bias. For this reason, much of modern function modelling is performed using *non-parametric* techniques. For regression, the most widely used approach is that of *GPs* (Rasmussen & Williams 2006). A GP is a supervised non-linear regression algorithm that makes few explicit *parametric* assumptions about the nature of the function fit. For this reason, GPs are seen as lying within the class of Bayesian non-parametric models. The underlying assumption in a GP is that, given a set of input $\mathbf{X} = \{\boldsymbol{x}_i\}_{i=1}^n \in \mathbb{R}^{n\times d}$ and a set of target outputs $\boldsymbol{y} = \{y_i\}_{i=1}^n \in \mathbb{R}^n$, where $n$ is the number of samples in the data set and $d$ is the dimensionality of the input, the observed target $y_i$ is generated by a function $\boldsymbol{f}$ of the input $\boldsymbol{x}_i$ plus additive noise $\epsilon_i$:

$$y_i = \boldsymbol{f}(\boldsymbol{x}_i) + \epsilon_i. \tag{1}$$

The noise $\epsilon$ is taken to be normally distributed with a mean of zero and variance $\sigma_n^2$, or $\epsilon \sim \mathcal{N}\left(0, \sigma_n^2\right)$. To simplify the notation, it is assumed that $\boldsymbol{y} \sim \mathcal{N}(0, 1)$ (this can readily be achieved without loss of generality, via a linear whitening process) and univariate,

[1] www.sdss.org

although the derivation can be readily extended to multivariable problems. The conditional probability of the observed variable given the function is hence distributed as follows:

$$p\left(\boldsymbol{y}|\boldsymbol{f}\right) \sim \mathcal{N}\left(\boldsymbol{f}, \sigma_n^2\right). \tag{2}$$

A GP then proceeds by applying a *Bayesian* treatment to the problem to infer a probability distribution over the space of possible functions $\boldsymbol{f}$ given the data:

$$p\left(\boldsymbol{f}|\boldsymbol{y}, \mathbf{X}\right) = \frac{p\left(\boldsymbol{y}|\boldsymbol{f}\right) p\left(\boldsymbol{f}|\mathbf{X}\right)}{p\left(\boldsymbol{y}|\mathbf{X}\right)}. \tag{3}$$

This requires us to define a prior, $p\left(\boldsymbol{f}|\mathbf{X}\right)$, over the function space. The function is normally distributed with a mean of zero, to match the mean of the normalized variable $\boldsymbol{y}$, with a covariance *function* $\mathbf{K}$, i.e. $p\left(\boldsymbol{f}|\mathbf{X}\right) \sim \mathcal{N}\left(0, \mathbf{K}\right)$. The covariance function captures prior knowledge about the relationships between the observables. Most widely used covariance functions assume that there is local similarity in the data, such that nearby inputs are mapped to similar outputs. The covariance $\mathbf{K}$ can therefore be modelled as a function of the input $\mathbf{X}$, $\mathbf{K} = k\left(\mathbf{X}, \mathbf{X}\right)$, where each element $\mathbf{K}_{ij} = k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ and $k$ is the covariance function. For $\mathbf{K}$ to be a valid covariance it has to be symmetric and positive semidefinite matrix; arbitrary functions for $k$ cannot guarantee these constraints. A class of functions that guarantees these structural constraints are referred to as *Mercer kernels* (Mercer 1909). A commonly used kernel function, which is the focus of this work, is the squared exponential kernel, defined as follows:

$$k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \sigma^2 \exp\left(-\frac{1}{2\lambda^2}\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2\right), \tag{4}$$

where $\sigma$ and $\lambda$ are referred to as the height (output, or variance) and characteristic length (input) scale respectively, which correspond to tunable *hyperparameters* of the model. The similarity between two input vectors, under the squared exponential kernel, is a non-linear function of the Euclidean distance between them. We note that this choice of kernel function guarantees continuity and smoothness in the function and all its derivatives. For a more extensive discussion of covariances, the reader is referred to Rasmussen & Williams (2006) or Roberts et al. (2013). With the likelihood $p\left(\boldsymbol{y}|\boldsymbol{f}\right)$ and prior $p\left(\boldsymbol{f}|\mathbf{X}\right)$, the marginal likelihood $p\left(\boldsymbol{y}|\mathbf{X}\right)$ can be computed as follows:

$$p\left(\boldsymbol{y}|\mathbf{X}\right) = \int p\left(\boldsymbol{y}|\boldsymbol{f}\right) p\left(\boldsymbol{f}|\mathbf{X}\right) \mathrm{d}\boldsymbol{f}, \tag{5}$$

By multiplying the likelihood and the prior and completing the square over $\boldsymbol{f}$, we can express the integration as a normal distribution independent of $\boldsymbol{f}$ multiplied by a another normal distribution over $\boldsymbol{f}$. The distribution independent of $\boldsymbol{f}$ can then be taken out of the integral, and the integration of the second normal distribution with respect to $\boldsymbol{f}$ will be equal to one. The resulting distribution of the marginal likelihood is distributed as follows (Rasmussen & Williams 2006):

$$p\left(\boldsymbol{y}|\mathbf{X}\right) \sim \mathcal{N}\left(0, \mathbf{K} + \sigma_n^2\boldsymbol{I}\right). \tag{6}$$

The marginal likelihood of the full data set can hence be computed as follows:

$$p\left(\boldsymbol{y}|\mathbf{X}\right) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\left|\mathbf{K} + \sigma_n^2\boldsymbol{I}\right|}} \exp\left(-\frac{1}{2} y_i^T \left(\mathbf{K} + \sigma_n^2\boldsymbol{I}\right)^{-1} y_i\right). \tag{7}$$

The aim of a GP, is to maximize the probability of observing the target $\boldsymbol{y}$ given the input $\mathbf{X}$, equation (7). Note that the only free parameters to optimize in the marginal likelihood are the parameters of the kernel and the noise variance, collectively referred to as the *hyperparameters* of the model. It is more convenient however to maximize the log of the marginal likelihood, equation (8), since the log function is a monotonically increasing function, maximizing the log of a function is equivalent to maximizing the original function. The log likelihood is given as

$$\log p(\boldsymbol{y}|\mathbf{X}) = -\frac{1}{2}\boldsymbol{y}^T \left(\mathbf{K} + \boldsymbol{I}\sigma_n^2\right)^{-1} \boldsymbol{y} - \frac{1}{2}\log\left|\mathbf{K} + \boldsymbol{I}\sigma_n^2\right| - \frac{n}{2}\log(2\pi). \tag{8}$$

We search for the optimal set of hyperparameters using a gradient-based optimization, hence we require the derivatives of the log marginal likelihood with respect to each hyperparameter. In this paper, the L-BFGS algorithm was used to optimize the objective which uses a quasi-Newton method to compute the search direction in each step by approximating the inverse of the Hessian matrix from the history of gradients in previous steps (Nocedal 1980; Schmidt 2005). It is worth mentioning that non-parametric models require the optimization of few hyperparameters that do not grow with the size of the data and are less prone to overfitting. The distinction between parameters and hyperparameters of a model is that the former directly influence the input–output mapping, for example the linear coupling weights in a basis function model, whereas the latter affect properties of distributions in the probabilistic model, for example the widths of kernels. Although this distinction is somewhat semantic, we keep to this nomenclature as it is standard in the statistical machine learning literature.

Once the hyperparameters have been inferred, the conditional distribution of future predictions $\boldsymbol{f}_*$ for test cases $\mathbf{X}_*$ given the training sample can be inferred from the joint distribution of $\boldsymbol{f}_*$ and the observed targets $\boldsymbol{y}$. If we assume that the joint distribution is a multivariate Gaussian, then the joint probability is distributed as follows:

$$p\left(\boldsymbol{y}, \boldsymbol{f}_*|\mathbf{X}, \mathbf{X}_*\right) \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathrm{K}_{\mathrm{xx}} + \sigma_{\mathrm{n}}^2\boldsymbol{I} & \mathrm{K}_{\mathrm{x}*} \\ \mathrm{K}_{*\mathrm{x}} & \mathrm{K}_{**} \end{bmatrix}\right), \tag{9}$$

where we introduce the shorthand notations $\mathrm{K}_{\mathrm{xx}} = k\left(\mathbf{X}, \mathbf{X}\right)$, $\mathrm{K}_{\mathrm{x}*} = k\left(\mathbf{X}, \mathbf{X}_*\right)$, $\mathrm{K}_{*\mathrm{x}} = k\left(\mathbf{X}_*, \mathbf{X}\right)$ and $\mathrm{K}_{**} = k\left(\mathbf{X}_*, \mathbf{X}_*\right)$. The conditional distribution $p\left(\boldsymbol{f}_*|\boldsymbol{y}, \mathbf{X}, \mathbf{X}_*\right)$ is therefore distributed as follows:

$$\begin{aligned} p\left(\boldsymbol{f}_*|\boldsymbol{y}, \mathbf{X}, \mathbf{X}_*\right) &\sim \mathcal{N}\left(\mu, \Sigma\right), \\ \mu &= \mathrm{K}_{*\mathrm{x}}\left(\mathrm{K}_{\mathrm{xx}} + \sigma_n^2\boldsymbol{I}\right)^{-1}\boldsymbol{y}, \\ \Sigma &= \mathrm{K}_{**} - \mathrm{K}_{*\mathrm{x}}\left(\mathrm{K}_{\mathrm{xx}} + \sigma_n^2\boldsymbol{I}\right)^{-1}\mathrm{K}_{\mathrm{x}*}. \end{aligned} \tag{10}$$

If we assume a non-zero prior mean $\mu_f$ over the function, $p(\boldsymbol{f}) \sim \mathcal{N}\left(\mu_f, \mathbf{K}\right)$, and an un-normalized $\boldsymbol{y}$ with mean $\mu_y$, the mean of the posterior distribution will be equal to

$$\mu = \mu_f + \mathrm{K}_{*\mathrm{x}}\left(\mathrm{K}_{\mathrm{xx}} + \sigma_n^2\boldsymbol{I}\right)^{-1}\left(\boldsymbol{y} - \mu_y\right), \tag{11}$$

The main drawback of GPs is the $O(n^3)$ computational cost required to invert the $n \times n$ matrix $\mathrm{K}_{\mathrm{xx}} + \sigma_n^2\boldsymbol{I}$. The *sparse GP* allows us to reduce this computational cost and is detailed in the following section.

## 3 SPARSE GPs

GPs are often described as non-parametric regression models due to the lack of an explicit parametric form. Indeed GP regression can also be viewed as a functional mapping $\mathbf{X} \in \mathbb{R}^{n\times d} :\rightarrow \mathbf{K} \in \mathbb{R}^{n\times n}$ parametrized by the data and the kernel function, followed by linear

regression via optimization of the following objective:

$$\min_{\boldsymbol{w}} \tfrac{1}{2}(\mathbf{K}\boldsymbol{w}-\boldsymbol{y})^T(\mathbf{K}\boldsymbol{w}-\boldsymbol{y}) + \tfrac{1}{2}\sigma_n^2 \boldsymbol{w}^T\boldsymbol{w}, \tag{12}$$

where $\boldsymbol{w}$ are the set of coefficients for the linear regression model that maps the transformed features $\mathbf{K}$ to the desired output $\boldsymbol{y}$. The feature transformation $\mathbf{K}$ evaluate how 'similar' a datum is to every point in the training sample, where the similarity measure is defined by the kernel function. If two points have a high kernel response via equation (4), this will result in very correlated features, adding extra computational cost for very little or no added information. Selecting a subset of the training sample that maximizes the preserved information is a research question addressed in Foster et al. (2009), whereas in Snelson & Ghahramani (2006) the basis functions are treated as a search problem rather than a selection problem and their locations are treated as hyperparameters which are optimized. These approaches result in a transformation $\mathbf{X} \in \mathbb{R}^{n\times d} :\rightarrow \mathbf{K} \in \mathbb{R}^{n\times m}$, in which $m \ll n$ is the number of basis functions used. The transformation matrix $\mathbf{K}$ will therefore be a rectangular $n$ by $m$ matrix and the solution for $\boldsymbol{w}$ in equation (12) is calculated via standard linear algebra as

$$\boldsymbol{w} = \left(\mathbf{K}^T\mathbf{K} + \boldsymbol{I}\sigma_n^2\right)^{-1}\mathbf{K}^T\boldsymbol{y}. \tag{13}$$

Even though these models improve upon the computational cost of a standard GP, very little is done to compensate for the reduction in modelling power caused by the 'loss' of basis functions. The selection method is always bounded by the full GP's accuracy, on the *training* sample, since the basis set is a subset of the full GP basis set. On the other hand, the sparse GP's ability to place the basis set freely across the input space does go some way to compensate for this reduction, as the kernels can be optimized to describe the distribution of the data. In other words, instead of training a GP model with all data points as basis functions, or restricting it to a subset of the training sample which require some cost to select them, a set of inducing points is used in which their locations are treated as hyperparameters of the model to be optimized. In both a full and a low rank approximation GP, a global set of hyperparameters is used for all basis functions, therefore limiting the algorithm's local modelling capability. Moreover, the objective in equation (12) minimizes the sum of squared errors, therefore for any non-uniformly distributed output, the optimization routine will bias the model towards the mean of the output distribution and will seek to fit preferentially the region of space where there are more data. Hence, the model might allow for very poor predictions for few points in poorly represented regions, e.g. the high redshift range, in order to produce good predictions for well represented regions. Therefore, the error distribution as a function of redshift is not uniform unless the training sample is well balanced, producing a model that is sensitive to how the target output is distributed.

In the next section, a method is proposed which addresses the above issues by parametrizing each basis function with bespoke hyperparameters which account for variable density and/or patterns across the input space. This is particularly pertinent to determining photometric redshifts, where complete spectroscopic information may be restricted or biased to certain redshifts or galaxy types, depending on the target selection for spectroscopy of the training sample. This allows the algorithm to learn more complex models with fewer basis functions. In addition, a weighting mechanism to remove any distribution bias from the model is directly incorporated into the objective.

## 4 PROPOSED APPROACH

In this paper, we extend the sparse GP approach by modelling each basis (kernel) function with its own set of hyperparameters. The kernel function in equation (4) is hence redefined as follows:

$$k(\boldsymbol{x}_i, \boldsymbol{p}_j) = \exp\left(-\frac{1}{2\lambda_j^2}\left\|\boldsymbol{x}_i - \boldsymbol{p}_j\right\|^2\right), \tag{14}$$

where $\boldsymbol{P} = \{\boldsymbol{p}_j\}_{j=1}^m \in \mathbb{R}^{m\times d}$ are the set of basis coordinates and $\lambda_j$ is the corresponding length scale for basis $j$. The multivariate input is denoted as $\mathbf{X} = \{\boldsymbol{x}_i\}_{i=1}^n \in \mathbb{R}^{n\times d}$. Throughout the rest of the paper, $\mathbf{X}_{i,*}$ denotes the $i$th row of matrix $\mathbf{X}$, or $\boldsymbol{x}_i$ for short, whereas $\mathbf{X}_{*,j}$ denotes the $j$th column and $\mathbf{X}_{i,j}$ refers to the element at row $i$ and column $j$ in matrix $\mathbf{X}$, and similarly for other matrices. Note that the hyperparameter $\sigma$ has been dropped, as it interferes with the regularization objective. This can be seen from the final prediction equation $\hat{y}_i = \sum_{j=1}^m \boldsymbol{w}_j\sigma_j^2 \exp\left(-\left\|\boldsymbol{x}_i - \boldsymbol{p}_j\right\|^2/2\lambda_j^2\right)$, the weights are always multiplied by their associated $\sigma$. Therefore, the optimization process will always compensate for decreasing $\boldsymbol{w}_j^2$ by increasing $\sigma_j^2$. Dropping the height variance ensures that the kernel functions do not grow beyond control and delegates learning the linear coefficients and regularization to the weights in $\boldsymbol{w}$. The derivatives with respect to each length-scale and position are provided in equations (15d) and (15e), respectively:

$$\mathbf{E} = \left((\mathbf{K}\boldsymbol{w}-\boldsymbol{y})\,\boldsymbol{w}^T\right)\circ\mathbf{K}, \tag{15a}$$

$$\Delta_j = \mathbf{X} - \mathbf{1}_n\boldsymbol{p}_j, \tag{15b}$$

$$\mathbf{D}_{i,j} = \left\|\boldsymbol{x}_i - \boldsymbol{p}_j\right\|^2, \tag{15c}$$

$$\frac{\partial f(\mathbf{X},\boldsymbol{y},\boldsymbol{w})}{\partial\lambda_j} = \mathbf{E}_{*,j}^T\mathbf{D}_{*,j}\lambda_j^{-3}, \tag{15d}$$

$$\frac{\partial f(\mathbf{X},\boldsymbol{y},\boldsymbol{w})}{\partial\boldsymbol{p}_j} = \mathbf{E}_{*,j}^T\Delta_j\lambda_j^{-2}. \tag{15e}$$

The symbol $\bigcirc$ denotes the Hadamard product, i.e. element-wise matrix multiplication and $\mathbf{1}_n$ denotes a column vector of length $n$ with all elements set to 1. Finding the set of hyperparameters that optimizes the solution, is in effect finding the set of radial basis functions defined by their positions $\boldsymbol{p}$ and radii $\lambda$ that jointly describe the patterns across the input space. By parametrizing them differently, the model is more capable to accommodate different regions of the space more specifically. A global variance model assumes that the relationship between the input and output is global or equal across the input space, whereas a variable variance model, or non-stationary GP, makes no assumptions and learns the variable variances for each basis function which reduces the need for more basis functions to model the data. The kernel in equation (14) can be further extended to, not only model each basis function with its own radius $\lambda_j$, but also model each one with its own covariance $\mathbf{C}_j \in \mathbb{R}^{d\times d}$. This enables the basis function to have any arbitrary shaped ellipses giving it more flexibility. The kernel in equation (14) can be extended as follows:

$$k(\boldsymbol{x}_i, \boldsymbol{p}_j) = \exp\left(-\frac{1}{2}\left(\boldsymbol{x}_i - \boldsymbol{p}_j\right)\mathbf{C}_j^{-1}\left(\boldsymbol{x}_i - \boldsymbol{p}_j\right)^T\right). \tag{16}$$

Furthermore, to make the optimization process faster and simpler, we define the additional variables:

$$\mathbf{C}_j^{-1} = \Lambda_j \Lambda_j^T, \tag{17a}$$

$$\mathbf{V}_j = \Delta_j \Lambda_j, \tag{17b}$$

where $\Lambda_j \in \mathbb{R}^{d \times d}$ is a local affine transformation matrix for basis function $j$ and $\mathbf{V}_j$ is the application of the local transformation to the data. Optimizing with respect to $\Lambda_j$ directly ensures that the covariance matrix is positive definite. This makes it faster from a computational perspective as the kernel functions for all the points with respect to a particular basis can be computed more efficiently as follows:

$$k(\mathbf{X}, \boldsymbol{p}_j) = \exp\left(-\frac{1}{2}\left(\mathbf{V}_j \circ \mathbf{V}_j\right)\mathbf{1}_d\right). \tag{18}$$

The exponent in equation (18) basically computes the sum of squares in each row of $\mathbf{V}_j$. This allows for a more efficient computation of the kernel functions for all the points in a single matrix operation. The derivatives with respect to each $\Lambda_j$ and $\boldsymbol{p}_j$ are shown in equations (19a) and (19b), respectively.

$$\frac{\partial f(\mathbf{X}, \boldsymbol{y}, \boldsymbol{w})}{\partial \Lambda_j} = -\left(\Delta_j^T \circ \left(\mathbf{1}_d \mathbf{E}_{*,j}^T\right)\right)\mathbf{V}_j, \tag{19a}$$

$$\frac{\partial f(\mathbf{X}, \boldsymbol{y}, \boldsymbol{w})}{\partial \boldsymbol{p}_j} = \mathbf{E}_{*,j}^T \mathbf{V}_j \Lambda_j^T. \tag{19b}$$

Setting up the problem in this manner allows the setting of matrix $\Lambda_j$ to be of any size $d$ by $q$, where $q < d$ which can be considered as a low rank approximation to $\mathbf{C}_j^{-1}$ without affecting the gradient calculations. In addition, the inverse of the covariance can be set to $\mathbf{C}_j^{-1} = \Lambda_j \Lambda_j^T + \mathrm{diag}(\lambda_j)^{-2}$ in the low rank approximation case to ensure that the final covariance can model a diagonal covariance. This is referred to as *factor analysis distance* (Rasmussen & Williams 2006, p. 107) but previously used to model a global covariance as opposed to variable covariances as is the case here.

### 4.1 Prior mean functions

In the absence of observations, all Bayesian models, GPs included, rely on their priors to provide function estimation. For the case of GPs this requires us to consider the prior over the function, especially the prior mean. For example, the first term in the mean prediction in equation (11), $\mu_f$, is our prior mean in which we learn the deviation from using a GP. Similarly, we may consider a mean *function* that is itself a simple linear regression from the independent to dependent variable. The parameters of this function are then inferred and the GP infers non-linear deviations. In the absence of data, e.g. in extrapolative regions, the GP will fall back to the linear regression prediction (Roberts et al. 2013). We can incorporate this directly into the optimization objective instead of having it as a separate preprocessing step by redefining $\mathbf{K}$ as a concatenation of the linear and non-linear features, or setting $\hat{\mathbf{K}} = [\mathbf{K}|\mathbf{X}|\mathbf{1}_n]$ and $\hat{\mathbf{w}} = [\boldsymbol{w}|\boldsymbol{w}_\mathrm{L}|b]$, where $\boldsymbol{w}_\mathrm{L}$ is the linear regression's coefficients and $b$ is the bias. The prediction can then be formulated as follows:

$$\hat{\mathbf{K}}\hat{\mathbf{w}} = \mathbf{K}\boldsymbol{w} + \mathbf{X}\boldsymbol{w}_\mathrm{L} + b. \tag{20}$$

Furthermore, the regularization matrix, $\boldsymbol{I}\sigma_n^2$, in equation (13) can be modified so that it penalises the learning of high coefficients for the non-linear terms, $\boldsymbol{w}$, but small or no cost for learning high linear terms, $\boldsymbol{w}_\mathrm{L}$ and $b$, by setting the corresponding elements in the diagonal of $\boldsymbol{I}$ to 0 instead of $\sigma_n^2$, or the last $d + 1$ elements. Therefore, as $\sigma_n^2$ goes to infinity, the model will approach a simple linear regression model instead of fallen back to zero.

### 4.2 Cost-sensitive learning

Thus far in the discussion, we make the tacit assumption that the objective of the inference process is to minimize the sum of squared errors between the model and target function values. Although this is a suitable objective for many applications, it is intrinsically biased by uneven distributions of training data, sacrificing accuracy in less represented regions of the space. Ideally we would like to train a model with a balanced data distribution to avoid such bias. This however, is a luxury that we often do not have. For example, the lack of strong emission lines that are detectable with visible-wavelength spectrographs in the 'redshift-desert' at $1.2 < z < 1.8$ means that this redshift range is often under-represented in spectroscopic samples. A common technique is to either oversample or under-sample the data to achieve balance (Weiss, McCarthy & Zabar 2007). In undersampling, samples are removed from highly represented regions to achieve balance, oversampling on the other hand duplicates under represented samples. Both approaches come with a cost; in the former good data are wasted and in the latter more computation is introduced due to the data size increase. In this paper, we perform cost-sensitive learning, which increases the intrinsic error function in under-represented regions. In regression tasks, such as we consider here, the output can be either discretized and treated as classes for the purpose of cost assignment, or a specific bias is used such as $1/(1 + z_\mathrm{s})$. To mimic a balanced data set in our setup, the galaxies were grouped by their spectroscopic redshift using non-overlapping bins of width 0.1. The weights are then assigned as follows for balanced training:

$$\omega_i = \frac{\max\left(\{f_1, \ldots, f_B\}\right)}{\{f_\mathrm{b} : i \in S_\mathrm{b}\}}, \tag{21}$$

where $\omega_i$ is the error cost for sample $i$, $f_\mathrm{b}$ is the frequency of samples in bin number number $i$, $B$ is the number of bins and $S_\mathrm{b}$ is the set of samples in set number $b$. equation (21) assigns a weight to each training point which is the maximum bin frequency over the frequency of the bin in which the source belongs. This ensures that the error cost of source $i$ is inversely proportional to its spectroscopic redshift frequency in the training sample. The normalized weights are assigned as follows:

$$\omega_i = \left(\frac{1}{1 + z_\mathrm{s}^{(i)}}\right)^2. \tag{22}$$

After the weights have been assigned, they can be incorporated directly into the objective as follows:

$$\min_{\boldsymbol{w}} \tfrac{1}{2}\left(\mathbf{K}\boldsymbol{w} - \boldsymbol{y}\right)^T \Omega \left(\mathbf{K}\boldsymbol{w} - \boldsymbol{y}\right) + \tfrac{1}{2}\sigma_n^2 \boldsymbol{w}^T \boldsymbol{w}. \tag{23}$$

The difference between the objectives in equations (12) and (23) is the introduction of the diagonal matrix $\Omega$, where each element $\Omega_{ii}$ is the corresponding cost $\omega_i$ for sample $i$. The first term in equation (23) is a matrix operation form for a weighted sum of squares $\sum_{i=1}^n \omega_i \left(\mathbf{K}_{i,*}\boldsymbol{w} - \boldsymbol{y}_i\right)^2$, where the solution can be found analytically as follows:

$$\boldsymbol{w} = \left(\mathbf{K}^T \Omega \mathbf{K} + \boldsymbol{I}\sigma_n^2\right)^{-1} \mathbf{K}^T \Omega \boldsymbol{y}. \tag{24}$$

The only modification to the gradient calculation is to set the matrix $\mathbf{E} = \Omega\left(\left(\mathbf{K}\boldsymbol{w} - \boldsymbol{y}\right)\boldsymbol{w}^T\right) \circ \mathbf{K}$. In standard sum of squared errors, $\Omega = \boldsymbol{I}$ or the identity matrix. It is worth emphasizing that this component of the framework does not attempt to weight the training sample in

order to match the distribution of the test sample, or matching the spectroscopic distribution to the photometric distribution as proposed in Lima et al. (2008), Cunha et al. (2009) and applied to photometric redshift in Sánchez et al. (2014), but rather gives the user of the framework the ability to control the cost per sample to serve different science goals depending on the application. In this paper, the weighting scheme was used for two different purposes, the first was to virtually balance the data to mimic training on a uniform distribution, and the second was to directly target the weighted error of $1/(1+z_s)$.

## 5 APPLICATION TO PHOTOMETRIC REDSHIFT ESTIMATION

In this section, we specifically target the photometric bands and depths planned for *Euclid*. *Euclid* aims to provide imaging data in a broad *RIZ* band and the more standard near-infrared *Y*, *J* and *H* bands, while ground-based ancillary data are expected in the optical *g*, *r*, *i* and *z* bands (Laureijs et al. 2011).

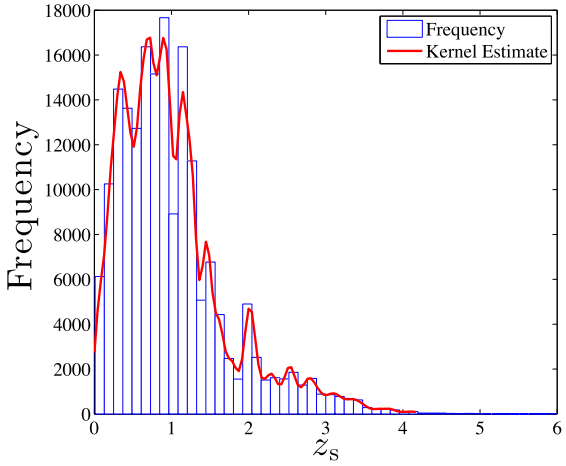
### 5.1 Simulated data set

We use a mock data set from Jouvel et al. (2009), consisting of the *g*, *r*, *i*, *z*, *RIZ*, *Y*, *J* and *H* magnitudes (to $10\sigma$ depths of 24.6, 24.2, 24.4, 23.8, 25.0 for the former, and $5\sigma$ depth of 24.0 for each of the latter three near-infrared filters) for 185 253 simulated sources. We remove all sources with $RIZ > 25$ to simulate the target magnitudes set for *Euclid*. In addition, we remove any sources with missing measurements in any of their bands prior to training (only 15 sources). No additional limits on any of the bands were used, however in Section 6.2 we do explicitly impose limits on the *RIZ* band to test the extrapolation performance of the models. The distribution of the spectroscopic redshift is provided in Fig. 1. For all experiments on the simulated data, we ignore the uncertainties on the photometry in each band and train only on the magnitudes since in the simulated data set, unlike real data sets, the log of the associated errors are linearly correlated with the magnitudes, especially in the targeted range, therefore adding no information. However, they were fed as input to ANNz to satisfy the input format of the code.

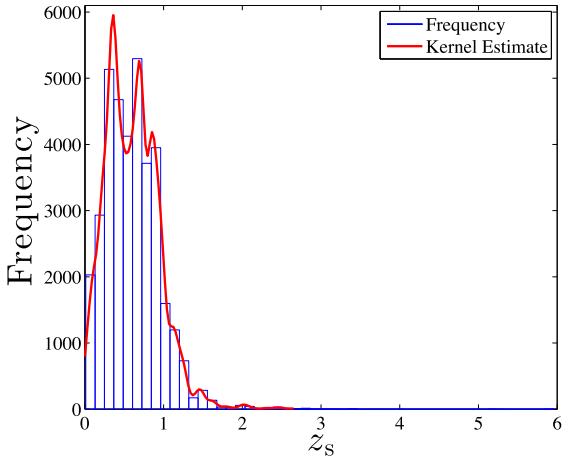
We preprocess the data using principle component analysis (PCA; Jolliffe 1986) to de-correlate the features prior to learning, but retain all features with no dimensionality reduction. Decorrelation accelerates the convergence rate of the optimization routine especially when using a logistic-type kernel machines such as neural networks (LeCun et al. 1998). To understand this, consider a simple linear regression example where we would like to solve for $\boldsymbol{w}$ in $\mathbf{A}\boldsymbol{w} = \boldsymbol{b}$, the solution for this is $\boldsymbol{w} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\boldsymbol{b}$. Note that if $\mathbf{A}$ is decorrelated $\mathbf{A}^T\mathbf{A} = \boldsymbol{I}$, therefore learning $\boldsymbol{w}_i$ depends only on the $i$th column of $\mathbf{A}$ and it is independent from learning $\boldsymbol{w}_j$, where $i \neq j$. In an optimization approach, the convergence rate is a function of the condition number of the $\mathbf{A}^T\mathbf{A}$ matrix, which is minimized in the case of decorrelated data. This represents a quadratic error surface which helps accelerate the search. This is particularly important in the application addressed in this paper because the magnitudes measured in each filter are strongly correlated with each other.

## 6 RESULTS ON THE SIMULATED DATA

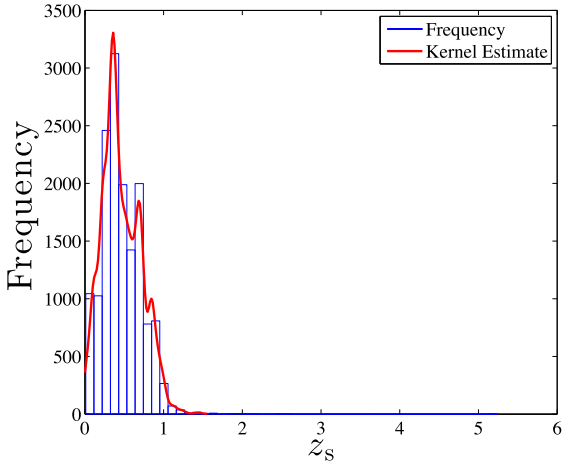
Five algorithms are considered to model the data; ANN (ANNz; Collister & Lahav 2004), a GP with low rank approximation




(a) Full




(b) $RIZ$ <23




(c) $RIZ$ <22

**Figure 1.** The spectroscopic redshift distribution of the (a) full data set, (b) sources with *RIZ* magnitude<23 and (c) sources with *RIZ* magnitude<22 from the simulated data.

**Table 1.** The time complexity of each approach.

| Method | Time complexity |
|---|---|
| ANNz ($l$-layers) | $O(nmd + (l-1)(nm^2))$ |
| STABLEGP | $O(nm^2)$ |
| GP-GL | $O(nmd + nm^2)$ |
| GP-VL | $O(nmd + nm^2)$ |
| GP-VC | $O(nmd^2 + nm^2)$ |

(STABLEGP; Foster et al. 2009), a sparse GP with global length-scale (GP-GL), a GP with variable length-scale (GP-VL) and a GP with variable covariances (GP-VC). For ANNz, a single layer network is used, and to satisfy the input format for the code, the data were not decorrelated and the uncertainties on photometry for each band were used as part of the training input. For STABLEGP, we use the SR-VP method proposed in Foster et al. (2009). In subsequent tests, the variable $m$ refers to the number of hidden units in ANNz, the rank in STABLEGP, and the number of basis functions in GP-GL, GP-VL and GP-VC. The time complexities for each algorithm are shown in Table 1. The data were split at random into 80 per cent for training, 10 per cent for validation and 10 per cent for testing. We note that we investigate the accuracy for various training sample sizes in Section 6.4. All models were trained using the entire redshift range available, but we only report the performance on the redshift range of $0 \leq z_s \leq 2$ to target the parameter space set out in Laureijs et al. (2011). We train each model for 500 iterations in each run and the validation sample was used for model selection and parameter tuning, but all the results here are reported on the test sample, which is not used in any way during the training process. Table 2 shows the metrics used to report the performance of each algorithm.

### 6.1 Modelling performance

In the first test, all models were trained using a fixed $m = 10$ to cross-compare the performance of the methods using the same number of basis functions. The number of basis functions was set deliberately low to highlight the sparse-limit modelling capabilities of each algorithm, as for large values of $m$ the performance gap between the algorithms reduces making it harder to compare the strengths and weaknesses of each algorithm. The standard sum of squares objective was used, without cost-sensitive learning or a prior mean function to keep the comparison as simple as possible. The $z_s$ versus $z_p$ density scatter plots are shown in Fig. 2 and their performance scores of each algorithm are reported in Table 3. Fig. 2 clearly shows the advantage of using inducing points over an active set when we compare GP-GL's performance in Fig. 2(b) with STABLEGP's performance in Fig. 2(c). The problem formulation of the two are identical, except that GP-GL's basis set is learned as part of the optimization objective whereas STABLEGP's basis set is pre-selected in an unsupervised fashion.

### 6.2 Prior mean

We also test the extrapolation performance of the GP-VC model using different prior means, namely a zero mean, a linear regression prior and a joint optimization approach that learns the linear and non-linear features simultaneously by regularizing the non-linear features more aggressively than linear features and compares them with ANNz. The difference between the linear regression prior and the joint optimization approach, is that the former first fits a linear model to the data then subtracts the predictions from the ground truth before training a GP model, whereas the latter learns both the linear model and the non-linear deviations from it jointly. To test this more effectively, the models were trained using sources, with $RIZ < 23$ (29 024 objects from the training sample) and tested on the unseen samples with $RIZ < 23$, $RIZ \geq 23$, and the entire test sample. A similar test was also conducted using a split of $RIZ < 22$ (12 056 objects from the training sample). This also demonstrates the effectiveness of the algorithms in the scenario where the brightest sources dominate the training sample, as may be true in practice. The results are reported in Table 4 and the density scatter plots are shown for comparison in Fig. 3. The results show that the 'Joint' method consistently outperformed the other methods in extrapolation as well as in interpolation, especially when trained with a small sample size as in the $RIZ < 22$ case. Moreover, upon examining the density scatter plots in Fig. 3, it has fewer systematic and catastrophic errors than the other methods with a factor of $\sim$2 improvement over ANNz where the training data are limited in magnitude/flux-density.

**Table 2.** Performance metrics used to evaluate the models. The number of samples is denoted by $n$.

| Metric | Equation | Description |
|---|---|---|
| $\delta^{(i)}$ | $z_s^{(i)} - z_p^{(i)}$ | Error for the $i$th object |
| $\delta_{\rm norm}^{(i)}$ | $\delta^{(i)} / \left(1 + z_s^{(i)}\right)$ | Normalized error for the $i$th object |
| $\Delta z$ | $\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\delta^{(i)}\right)^2}$ | Root mean squared error |
| $\Delta z_{\rm norm}$ | $\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\delta_{\rm norm}^{(i)}\right)^2}$ | Normalized root mean squared error |
| $\max_z$ | $\max(\{\lvert\delta^{(1)}\rvert, \ldots, \lvert\delta^{(n)}\rvert\})$ | Maximum error |
| $\max_{\rm norm}$ | $\max\left(\left\{\left\lvert\delta_{\rm norm}^{(1)}\right\rvert, \ldots, \left\lvert\delta_{\rm norm}^{(n)}\right\rvert\right\}\right)$ | Maximum normalized error |
| $\mu_z$ | $\frac{1}{n}\sum_{i=1}^{n}\delta^{(i)}$ | Bias |
| $\mu_{\rm norm}$ | $\frac{1}{n}\sum_{i=1}^{n}\delta_{\rm norm}^{(i)}$ | Normalized bias |
| $\sigma_z$ | $\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\delta^{(i)} - \mu_z\right)^2}$ | Standard deviation of the errors |
| $\sigma_{\rm norm}$ | $\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\delta_{\rm norm}^{(i)} - \mu_{\rm norm}\right)^2}$ | Standard deviation of the normalized errors |
| ${\rm out}_z$ | $\frac{1}{n}\left\lvert\left\{i : \delta^{(i)} > 2\sigma_z\right\}\right\rvert$ | Fraction of errors above two standard deviations from the mean |
| ${\rm out}_{\rm norm}$ | $\frac{1}{n}\left\lvert\left\{i : \delta_{\rm norm}^{(i)} > 2\sigma_{\rm norm}\right\}\right\rvert$ | Fraction of normalized errors above two standard deviations from the mean |

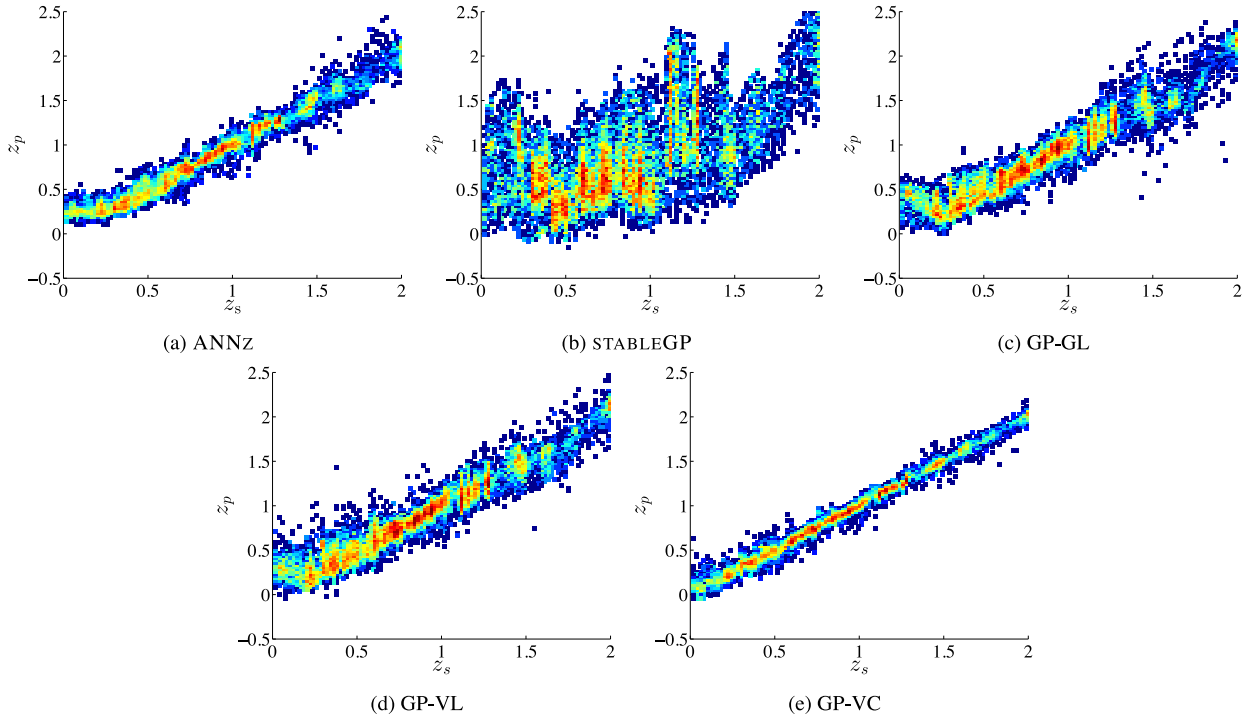



**Figure 2.** Density scatter plots of the true $z_s$ versus the predicted $z_p$ for (a) ANNz, (b) STABLEGP, (c) GP-GL, (d) GP-VL and (e) GP-VC using $m = 10$ basis functions trained on the simulated data set. The plots shows the performance on the same test sample, the colours however are scaled differently according to the density range in each plot to avoid colour saturation.

**Table 3.** Performance measures for each algorithm trained on the simulated survey using $m = 10$ basis functions. The best-performing algorithm is highlighted in bold font.

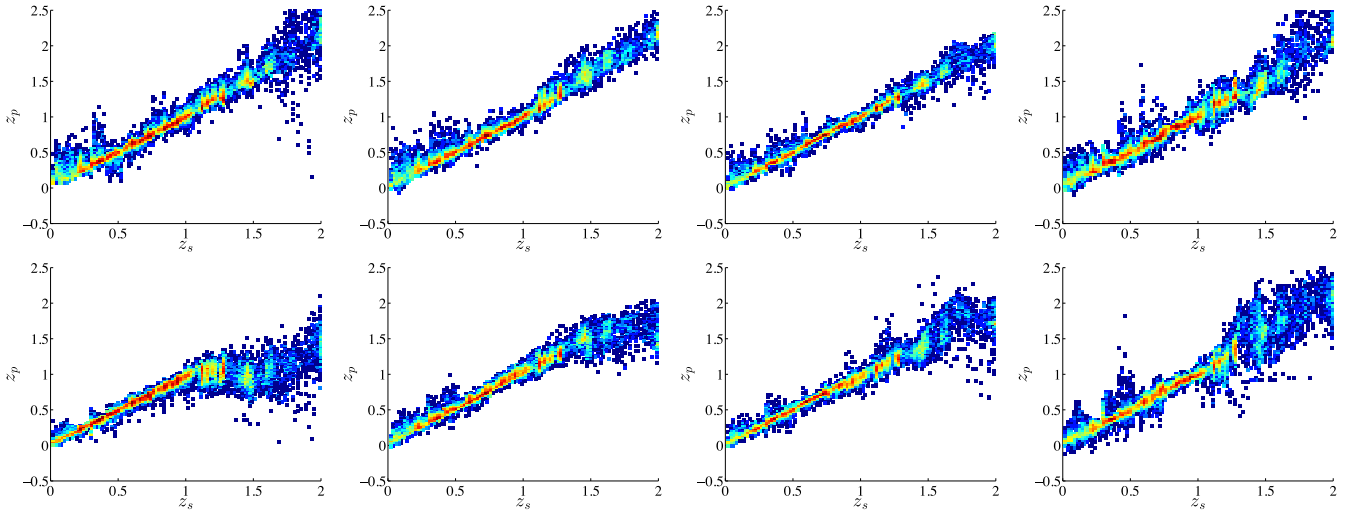
| | $\Delta z$ | $\Delta z_{\rm norm}$ | $\max_z$ | $\max_{\rm norm}$ | $\mu_z$ | $\mu_{\rm norm}$ | $\sigma_z$ | $\sigma_{\rm norm}$ | $\text{out}_z$ | $\text{out}_{\rm norm}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| ANNz | 0.0848 | 0.0568 | 1.0126 | **0.4199** | −0.0025 | −0.0048 | 0.0847 | 0.0566 | 0.0505 | 0.0532 |
| STABLEGP | 0.4399 | 0.2836 | 1.5906 | 1.4441 | −0.0085 | −0.0365 | 0.4399 | 0.2812 | 0.0509 | 0.0539 |
| GP-GL | 0.1420 | 0.0952 | 1.1183 | 0.5953 | −0.0064 | −0.0106 | 0.1418 | 0.0946 | 0.0548 | 0.0530 |
| GP-VL | 0.1251 | 0.0833 | 1.0349 | 0.7953 | −0.0074 | −0.0094 | 0.1249 | 0.0828 | 0.0549 | 0.0552 |
| GP-VC | **0.0435** | **0.0294** | **0.5488** | 0.5380 | **−0.0005** | **−0.0007** | **0.0435** | **0.0294** | **0.0487** | **0.0473** |

**Table 4.** The value of $\Delta z$ for the GP-VC model when trained on the simulated survey using $m = 10$ basis functions with different prior mean functions and *RIZ* splits. The results for ANNz are shown for comparison.

| Trained on | *RIZ* < 22 | | | *RIZ* < 23 | | | Full |
|---|---|---|---|---|---|---|---|
| Tested on | <22 | ≥22 | Full | <23 | ≥23 | Full | Full |
| ANNz | 0.0385 | 0.1383 | 0.1325 | 0.0537 | 0.1458 | 0.1315 | 0.0848 |
| Zero | 0.0233 | 0.2539 | 0.2424 | 0.0362 | 0.1261 | 0.1129 | 0.0435 |
| Linear | 0.0199 | 0.1043 | 0.0997 | 0.0321 | 0.1097 | 0.0983 | 0.0412 |
| Joint | **0.0192** | **0.0982** | **0.0939** | **0.0277** | **0.0653** | **0.0593** | **0.0298** |

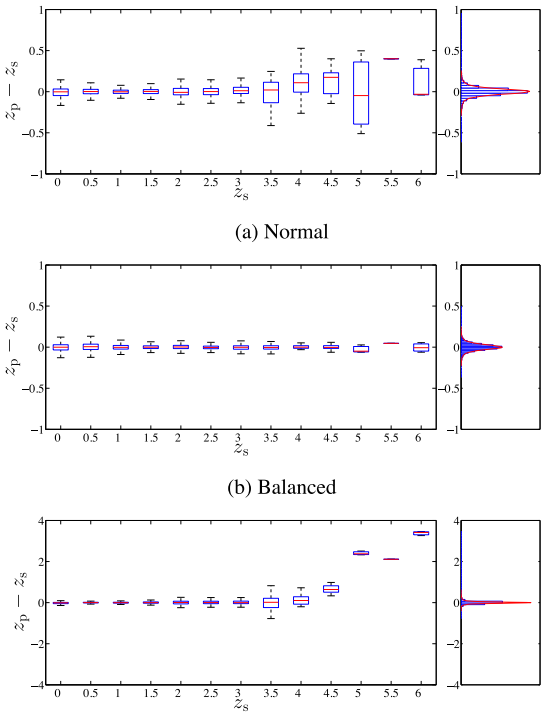
### 6.3 Cost-sensitive learning

We now perform a comparison between cost-sensitive learning and the normal sum of squared errors for the GP-VC model. Two different weight configurations are tested, the first is to assign an error cost to each sample as in equation (22) (Normalized), and the second experiment is to weight each sample according to the frequency of their true redshift to ensure balanced learning (Balanced) as in equation (21), in addition to the (Normal) sum of squared errors. The algorithms were trained such that they have equal $\Delta z$ score, to examine the differences between the other metrics and the resulting error distributions. The box plots for the 'Normal', 'Balanced' and 'Normalized' are shown in Fig. 4. The figures show the performance on the held out test sample for the entire range to demonstrate the effect more clearly. Cost-sensitive learning is more consistent across the redshift range as opposed to the normal sum of squares, especially in the high redshift regions where there are less data. The confidence intervals are also considerably smaller for the 'Balanced' case. The 'Normalized' training on the other hand results in a systematic bias, as expected, towards the lower part of the redshift range. The performance comparison for the 'Normal', 'Balanced' and 'Normalized' training are summarized in Table 5, but the metrics are reported on the desired range of $0 < z_s < 2$. Balanced training shows a better generalization performance, as it




(a) Zero (b) Linear (c) Joint (d) ANNz

**Figure 3.** Density scatter plots of the true $z_s$ versus the predicted $z_p$ after training the GP-VC model on samples with $RIZ < 23$ (top) and $RIZ < 22$ (bottom) from the simulated data using $m = 10$ basis functions with (a) a zero mean prior, (b) linear regression prior and (c) a joint prior optimization and (d) ANNz. The plots shows the performance on the same test sample, the colours however are scaled differently according to the density range in each plot to avoid colour saturation.

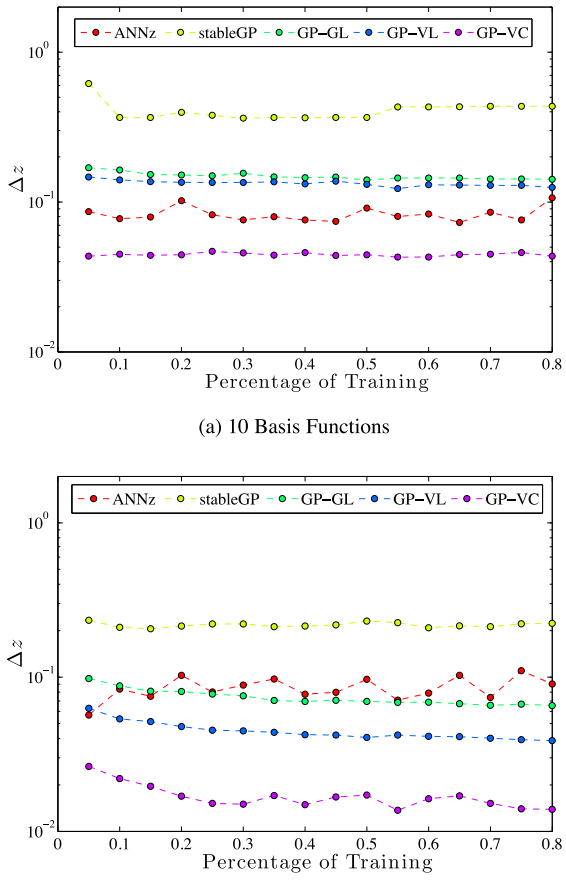



(a) Normal

(b) Balanced

(c) Normalized

**Figure 4.** Box plots of residual errors on the hold-out test sample, showing median (bar), interquartile range (box) and range (whiskers) for (a) the direct sum of squared errors, (b) the balanced cost-sensitive learning and (c) the normalized cost learning for the GV-VC model trained on the simulated data using $m = 10$ basis functions. The rightmost histograms are the empirical densities of errors. Figures (a) and (b) have similar scales, but note the scale difference in (c).




(a) 10 Basis Functions

(b) 100 Basis Functions

**Figure 5.** $\Delta z$ as a function of training size for all the methods using a simple model ($m = 10$) and a complex model ($m = 100$) trained on the simulated data.

**Table 5.** Performance measures of training the GP-VC model using $m = 10$ basis functions and different weighting schemes trained on the simulated survey.

| | $\Delta z$ | $\Delta z_{\rm norm}$ | $\max_z$ | $\max_{\rm norm}$ | $\mu_z$ | $\mu_{\rm norm}$ | $\sigma_z$ | $\sigma_{\rm norm}$ | $\text{out}_z$ | $\text{out}_{\rm norm}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Normal | 0.0500 | 0.0337 | 0.6128 | 0.6008 | −0.0017 | −0.0016 | 0.0500 | 0.0336 | 0.0507 | 0.0507 |
| Balanced | 0.0500 | 0.0324 | **0.4933** | 0.3419 | **0.0007** | **0.0001** | 0.0500 | 0.0324 | 0.0510 | 0.0510 |
| Normalized | 0.0500 | **0.0280** | 0.6389 | **0.2862** | 0.0008 | −0.0005 | 0.0500 | **0.0280** | **0.0458** | **0.0498** |

outperforms the normal sum of squares objective on the test sample and has lower maximum errors, although the differences are generally small.

### 6.4 Size of the training sample

Thus far, we have only considered the case of having a large (80 per cent of the total data) training sample. In practice, it is likely that the training-validation-test will be substantially smaller than the data set for which photometric redshifts are required. Therefore, in this section the generalization performance of the models are tested by limiting the training sample size to different percentages of the data set. The validation and test samples were fixed to the same samples used in previous experiments to ensure consistent reporting on the same test sample. The models were trained using various percentages from 5 to 80 per cent, once using a small number of basis functions ($m = 10$) and a second experiment using a larger number of basis functions ($m = 100$) and the results are shown for both in Figs 5(a) and (b) respectively. GP-VC consistently outperforms the other models across the training range using both simple and complex models. ANNz on the other hand performs poorly and quickly overfits in the complex version of the test. It is worth noting that, unlike the other models, ANNz consistently shows an unsteady performance in all of the parameter tuning tests for the simulated data set. However, we note that on real noisy data this problem diminishes (Section 7).

### 6.5 Size of the basis set

Until now, we have limited the number of basis functions to 10, except for the last section to test the generalization performance of the models. In practice, the only limitation on the number of basis functions used for training the GP is computing resources. In this section we investigate how the accuracy of the photometric redshifts depends on the number of basis functions.

We cross-compare all of the models by varying the number of basis functions $m$ from 5 to 200 by an increment of 5 to study the relationship between accuracy and complexity. $\Delta z$ as a function of $m$ for all the models are shown in Fig. 6, the $y$-axis is shown on a log scale for the purpose of visualization. The STABLEGP method exhibits the worst performance across the board, especially when the number of basis functions is small. On the other hand, GP-VC consistently outperforms the rest, and most significantly when trained with few basis functions. ANNz outperforms GP-GL and GP-VL, but it does not scale well with complexity as it starts to overfit after $m = 30$. All the models were trained using a sum of squared errors objective with no cost-sensitive learning or a prior mean function optimization in this experiment.

In Fig. 7 we show the values of $\Delta z$ and $\Delta z_{\rm norm}$ for the GP-VC approach using an extended range of basis functions of 5, 10, 25, 50, 100, 200, 400, 800 and 1600 with joint linear optimization, using both the normalized weights and normal sum of squares. With the GP-VC we obtain $\Delta z_{\rm norm} = 0.0295$ with just $m = 5$ basis functions, and when using $m = 1600$ we obtain $\Delta z_{\rm norm} = 0.0026$ and a maximum normalized error $\Delta z_{\rm norm} = 0.0652$. We note that although the *training* complexity costs require effort for large numbers of basis functions, once all parameters are inferred we enjoy effectively a linear basis model performance running over unseen (test) data. We therefore consider the performance for a realistic, yet large, number of functions. For an in depth analysis of feature selection, magnitude cuts and training sample size on photometric redshift the reader is referred to Hoyle et al. (2015).

We also generated photometric redshifts from a committee of five neural networks using a two-layer architecture, each layer with twice the number of hidden units as the number of filters as

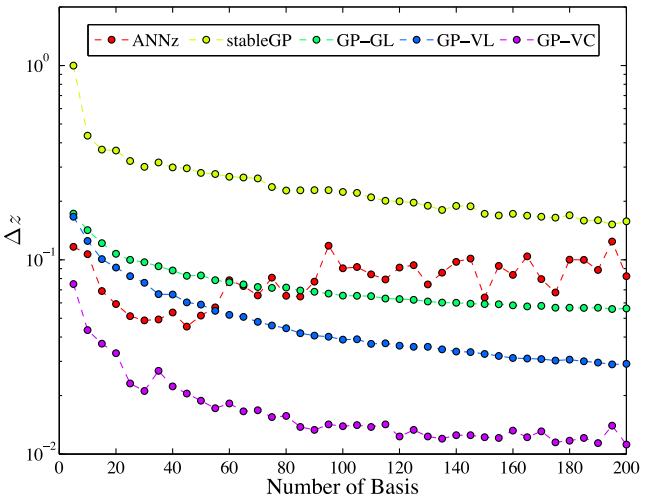



**Figure 6.** $\Delta z$ as a function of the number of basis functions for all the methods.

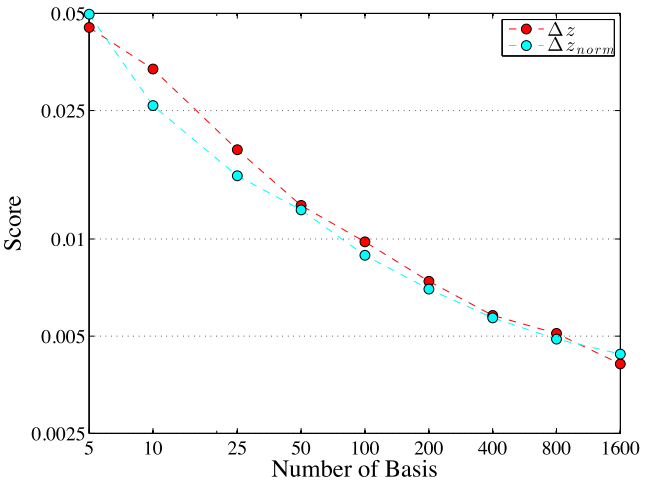
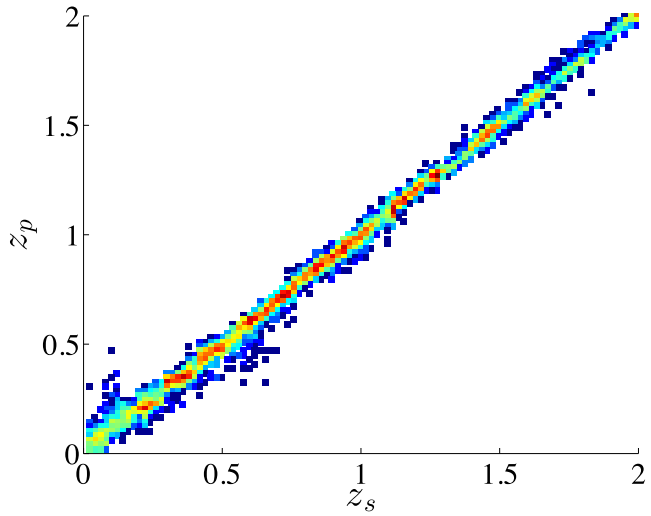


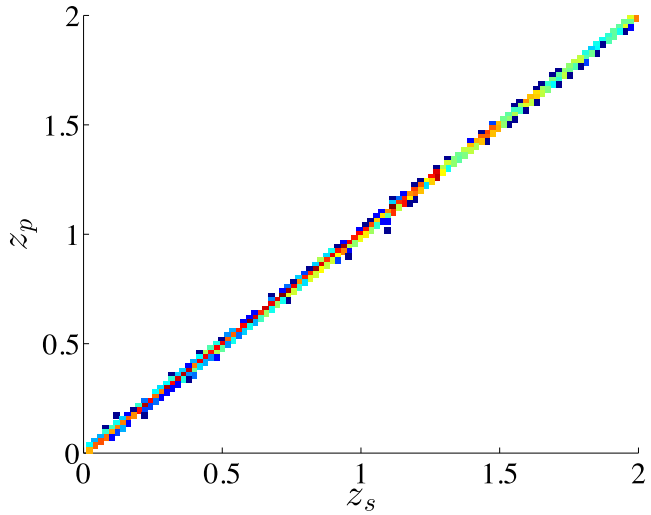

**Figure 7.** A log–log plot reporting the $\Delta z$ and $\Delta z_{\rm norm}$ scores after training GP-VC models on the simulated data with 5, 10, 25, 50, 100, 200, 400, 800 and 1600 basis functions with joint linear optimization. The results for the $\Delta z$ were optimized using normal sum of squares whereas the results for the $\Delta z_{\rm norm}$ were optimized using normalized weights.

**Table 6.** Performance measures for the final ANNz model using a committee of five networks with 8:16:16:1 architectures and the final GP-VC model trained on the simulated survey using $m = 1600$ basis functions with a jointly optimized linear function on the simulated survey.

| | $\Delta z$ | $\Delta z_{\rm norm}$ | $\max_z$ | $\max_{\rm norm}$ | $\mu_z$ | $\mu_{\rm norm}$ | $\sigma_z$ | $\sigma_{\rm norm}$ | $\text{out}_z$ | $\text{out}_{\rm norm}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| ANNz | 0.0262 | 0.0180 | 0.3696 | 0.3391 | −0.0004 | −0.0007 | 0.0262 | 0.0180 | **0.0433** | **0.0406** |
| GP-VC | **0.0041** | **0.0026** | **0.0764** | **0.0652** | **0.0000** | **0.0000** | **0.0041** | **0.0026** | 0.0480 | 0.0460 |




(a) ANNz




(b) GP-VC

**Figure 8.** The density scatter plot for (a) the final ANNz model using a committee of five networks with 8:16:16:1 architectures and (b) the final GP-VC model trained using $m = 1600$ basis functions with a jointly optimized linear mean function on the simulated survey.

recommended in Collister & Lahav (2004) and has become a standard for most ANNz users. The models were trained using the same training and validation samples, and the quoted results were calculated on the test sample. The results of the final GP-VC and ANNz models are summarized in Table 6 and the density scatter plots for the final models are shown in Fig. 8 for comparison. We find that both learn the underlying (simulated) relationship, GP-VC however provides a factor of approximately seven improvement in the accuracy of $\Delta z$ and $\Delta z_{\rm norm}$ over the commonly-used ANNz architecture for the simulated data set.

## 7 RESULTS ON SDSS DATA

In this section we compare ANNz with GP-VC on data from the SDSS 12th Data Release. The data used for training was selected from all the galaxies in the data base where photometric and spectroscopic data are available and any sources with missing data was excluded from training. The `modelMag` magnitudes were used with their associated error estimates. The following SQL statement was used to extract the data from the SDSS DR12 data base using the CasJobs service provided by SDSS.[2]

```
SELECT
p.objid,
p.modelMag_u, p.modelMag_g,
p.modelMag_r, p.modelMag_i,
p.modelMag_z, p.modelMagerr_u,
p.modelMagerr_g, p.modelMagerr_r,
p.modelMagerr_i, p.modelMagerr_z,
s.z as zspec, s.zErr as zspecErr,
s.survey as survey
INTO
mydb.modelmag_data set
FROM
PhotoObjAll as p, SpecObj as s
WHERE
p.SpecObjID = s.SpecObjID AND
s.class = 'GALAXY' AND
s.zWarning = 0 AND
p.mode = 1 AND
dbo.fPhotoFlags('PEAKCENTER') != 0 AND
dbo.fPhotoFlags('NOTCHECKED') != 0 AND
dbo.fPhotoFlags('DEBLEND_NOPEAK') != 0 AND
dbo.fPhotoFlags('PSF_FLUX_INTERP')!=0AND
dbo.fPhotoFlags('BAD_COUNTS_ERROR')!=0AND
dbo.fPhotoFlags('INTERP_CENTER') != 0
```

We use similar image flags to the ones used in Brescia et al. (2014). To target the original spectroscopic limit of the SDSS, the models are also tested with a cut-off of $r < 17.7$ applied. Four different data sets were created from the retrieved data:

(i) SDSS: This data set includes only sources from the SDSS but not the BOSS survey (817 604 sources).

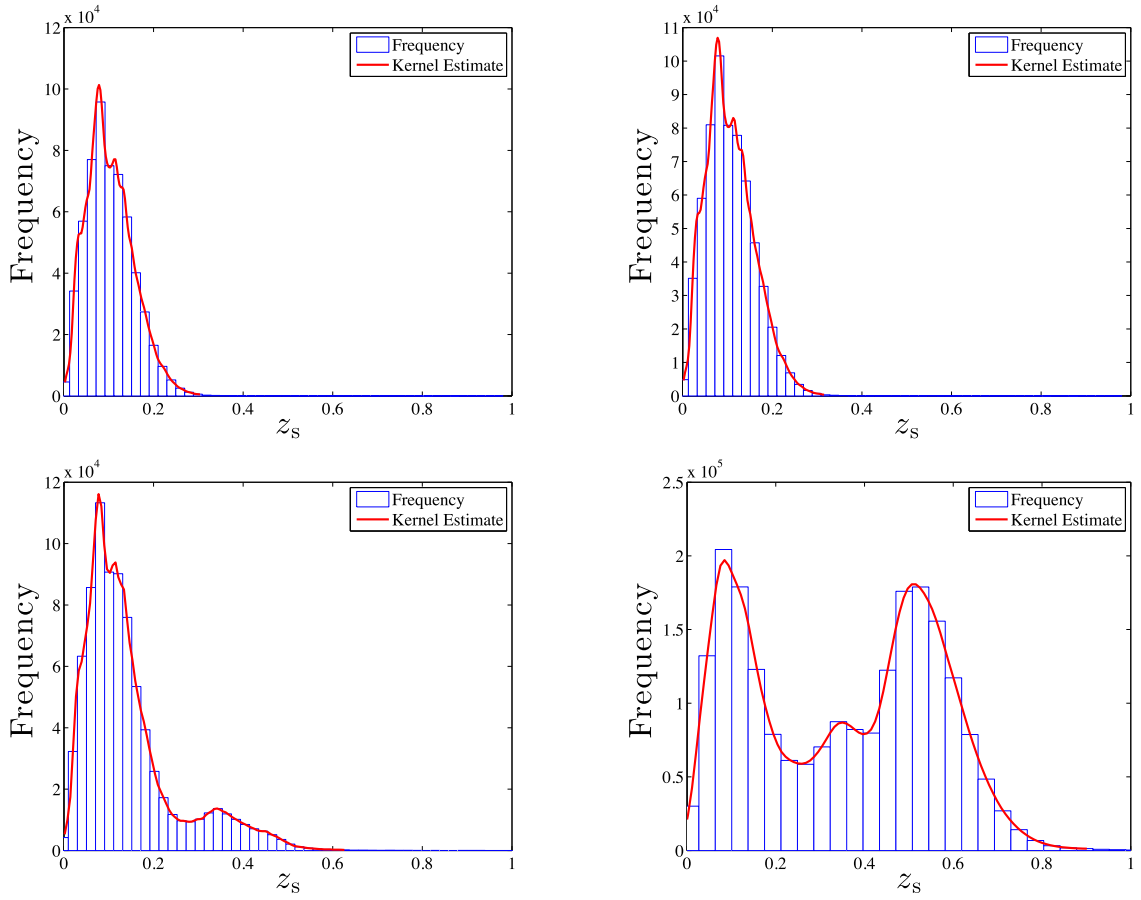
(ii) SDSS with cut: This data set includes only the sources from the SDSS data set with $r < 17.7$ (577 725 sources).

(iii) SDSS+BOSS: This data set includes all sources from the BOSS and the SDSS surveys (2 120 465 sources).

(iv) SDSS+BOSS with cut: This data set includes only the sources from the SDSS+BOSS data set with $r < 17.7$ (629 117 sources).

The distribution of the spectroscopic redshifts of the data sets are shown in Fig. 9. Similar to the simulated data setup, we used 80 per cent for training, 10 per cent for validation and 10 per cent for testing in each data set.

[2] casjobs.sdss.org




(a) SDSS (b) SDSS+BOSS

**Figure 9.** The distribution of the spectroscopic redshift in the (a) SDSS and (b) SDSS+BOSS data sets. The top figures show the distributions with the $r < 17.7$ cut while the bottom figures show the distributions without the cut.

### 7.1 Varying the degree of freedom

In this experiment, we compare ANNz to GP-VC's performance based on the number of free parameters. This is achieved by setting the number of basis functions in GP-VC such that the number of free parameters are approximately equal to the number of free parameters in a two-layer neural network using the following assignment:

$$\mathcal{D}_g = m_g \left(d^2 + d\right) + d + 1, \tag{25a}$$

$$\mathcal{D}_z = m_z^2 + dm_z + 3m_z + 1, \tag{25b}$$

$$m_g \approx \frac{m_z^2 + dm_z + 3m_z - d}{d^2 + d}, \tag{25c}$$

where $\mathcal{D}_g$ is the degree of freedom in GP-VC, with joint prior mean function, $\mathcal{D}_z$ is the degree of freedom in ANNz, $m_g$ is the number of basis functions in GP-VC and $m_z$ is the number of hidden units in ANNz. The number of hidden units is set to be equal in both layers. Setting the number of basis functions in GP-VC according to (25c), ensures that $\mathcal{D}_g \approx \mathcal{D}_z$.

In this test, we trained a two-layer ANNz architecture using 10–100 hidden units and a matching GP-VC model based on equation (25c) with joint mean optimization. The number of hidden units was set to be equal in both layers but only a single network was used to generate the predictions not a committee of five networks. Both models were trained on the SDSS data set and the results on the test set are shown in Fig. 10. The results are consistent with the results from the simulated data, the performance of ANNz degrades

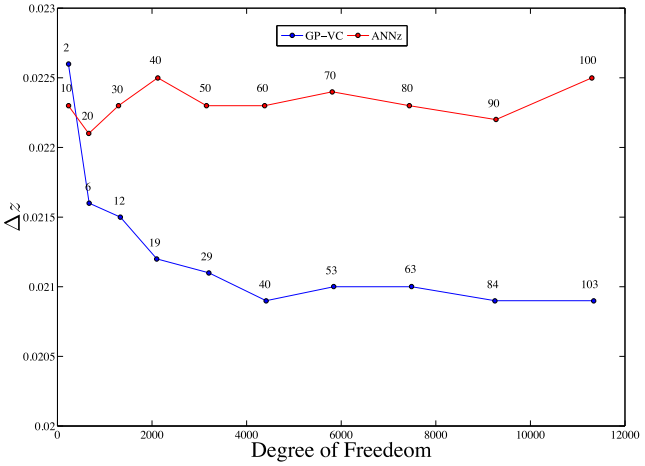
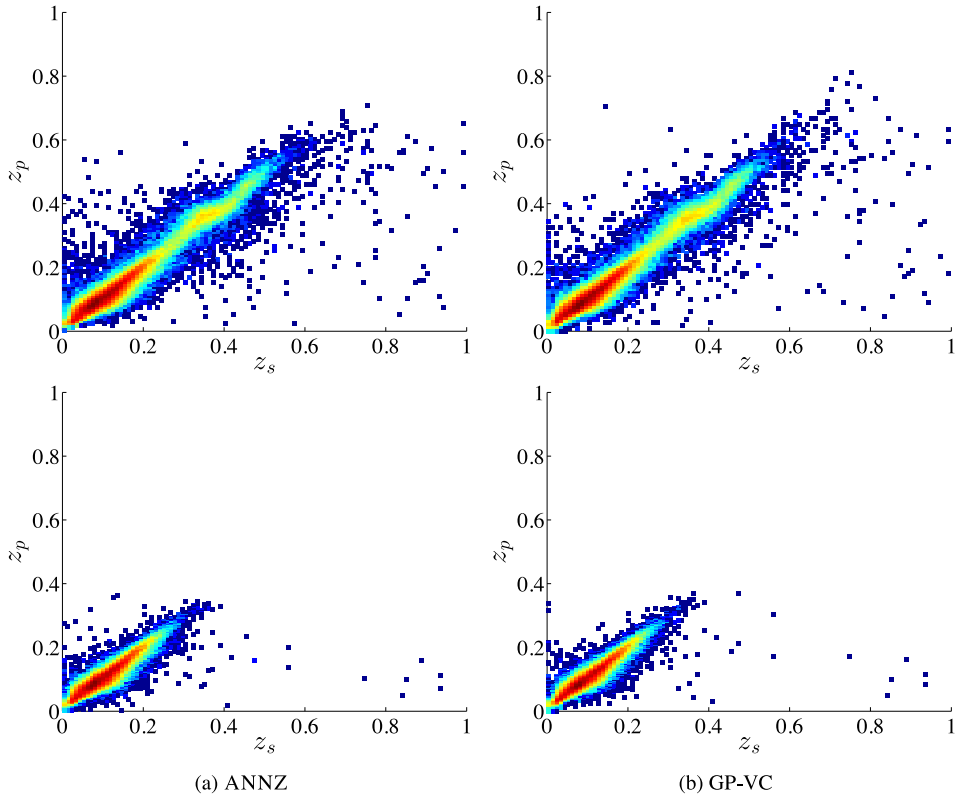



**Figure 10.** A comparison between GP-VC and ANNz with two layers on the SDSS data set using various degrees of freedom. The numbers on top of each point is the equivalent number of basis functions in GPVC and the number of hidden units for each layer in ANNz.

**Table 7.** Performance measures for the final ANNz model using a committee of five networks with 5:10:10:1 architectures and the final GP-VC model using $m = 103$ basis functions with a jointly optimized linear function on all the data sets.

| | $\Delta z$ | $\Delta z_{\rm norm}$ | $\max_z$ | $\max_{\rm norm}$ | $\mu_z$ | $\mu_{\rm norm}$ | $\sigma_z$ | $\sigma_{\rm norm}$ | $\text{out}_z$ | $\text{out}_{\rm norm}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | (a) SDSS | | | | | |
| ANNz | 0.0308 | 0.0249 | 0.8689 | **0.4478** | 0.0000 | −0.0007 | 0.0308 | 0.0249 | 0.0327 | 0.0378 |
| GP-VC | **0.0302** | **0.0244** | **0.8678** | 0.5017 | 0.0000 | **−0.0006** | **0.0302** | **0.0244** | **0.0316** | **0.0365** |
| | | | | | (b) SDSS with cut | | | | | |
| ANNz | 0.0221 | 0.0190 | 0.8710 | 0.4489 | 0.0002 | **−0.0002** | 0.0221 | 0.0190 | 0.0397 | 0.0469 |
| GP-VC | **0.0209** | **0.0179** | **0.8562** | **0.4413** | **0.0001** | −0.0003 | **0.0209** | **0.0179** | **0.0386** | **0.0456** |
| | | | | | (c) SDSS+BOSS | | | | | |
| ANNz | 0.0539 | 0.0386 | **1.3113** | **0.7457** | **−0.0000** | −0.0015 | 0.0539 | 0.0386 | 0.0393 | 0.0346 |
| GP-VC | **0.0513** | **0.0366** | 1.4284 | 0.8601 | −0.0001 | **−0.0013** | **0.0513** | **0.0366** | **0.0385** | **0.0340** |
| | | | | | (d) SDSS+BOSS with cut | | | | | |
| ANNz | 0.0222 | 0.0188 | **0.8523** | **0.4421** | −0.0000 | −0.0004 | 0.0222 | 0.0188 | **0.0367** | 0.0445 |
| GP-VC | **0.0207** | **0.0178** | 0.9831 | 0.5043 | −0.0000 | **−0.0003** | **0.0207** | **0.0178** | 0.0376 | 0.0445 |




(a) ANNZ (b) GP-VC

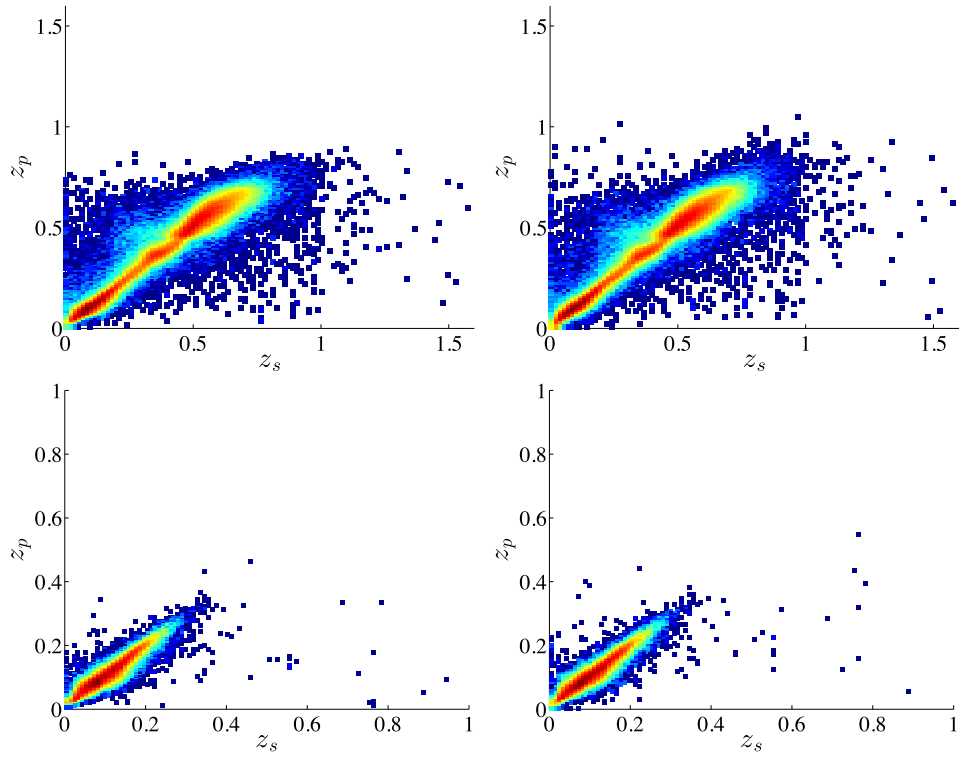
**Figure 11.** The density scatter plot for (a) the final ANNz model using a committee of five networks with 5:10:10:1 architectures and (b) the final GP-VC model trained using $m = 103$ basis functions with a jointly optimized linear mean function. The top figures show the plots for the SDSS data set with out the $r < 17.7$ cut, while the bottom figures show the plots for the SDSS with cut.

as we increase the complexity of the network, whereas GP-VC is more robust and shows a steady improvement.

### 7.2 Final results

In this experiment, photometric redshifts were generated using a committee of five two-layer architectures, each layer with twice the number of hidden units as the number of filters and compared to the best performing GP-VC model from the previous test. The experiment was carried out on the four data sets, the performance measures are reported in Tables 7a, 7b, 7c and 7d for the SDSS, SDSS with cut, SDSS+BOSS and SDSS+BOSS with cut respectively. The density scatter plots for the SDSS and SDSS+BOSS data sets are shown in Figs 11 and 12, respectively.

Although not as stark as for the simulated data set, our GP-VC algorithm consistently outperforms ANNz on the important metrics ($\Delta z$ and $\Delta z_{\rm norm}$) and from examining the plots, the outliers are less extreme in the GP-VC case, with an increase in accuracy of around 5 per cent for the full SDSS+BOSS data set. However, we note that given the overall performance of GP-VC on the simulated data set, we expect the real power of our GP-VC to algorithm to be shown in the sparse data regime, and we reserve such an analysis to a future paper.




(a) ANNZ (b) GP-VC

**Figure 12.** The density scatter plot for (a) the final ANNz model using a committee of five networks with 5:10:10:1 architectures and (b) the final GP-VC model trained using $m = 103$ basis functions with a jointly optimized linear mean function. The top figures show the plots for the SDSS+BOSS data set with out the $r < 17.7$ cut, while the bottom figures show the plots for the SDSS+BOSS with cut.

## 8 SUMMARY AND FUTURE WORK

In this paper a sparse GP framework is presented and applied to photometric redshift estimation. The framework is able to out perform ANNz, sparse GP parametrized by a set of global hyperparameters and low rank approximation GP. The performance increase is attributed to the handling of distribution bias via a weighting scheme integrated as part of the optimization objective, parametrizing each basis function with bespoke covariances, and integrating the learning of the prior mean function to enhance the extrapolation performance of the model. The methods were applied to a simulated data set and SDSS DR12 where the proposed approach consistently outperforms the other models on the important metrics ($\Delta z$ and $\Delta z_{\rm norm}$). We find that the model scales linearly in time with respect to the size of the data, and has a better generalization performance compared to the other methods even when presented with a limited training set. Results show that with only 30 per cent of the data, the model was able to reach accuracy close to that of using the full training sample. Even when data were selectively removed based on *RIZ* magnitudes, the model was able to show the best recovery performance compared to the other models. The cost-sensitive learning component of the framework regularizes the predictions to limit the effect caused by the biased distribution of the output and allows for direct optimization of the survey objective (e.g. $z_{\rm norm} = |z_{\rm s} - z_{\rm p}|/(1 + z_{\rm s})$). Again, the algorithm consistently outperforms other approaches, including ANNz and STABLEGP, in all reported experiments. We also investigate how the size of the training sample and the basis set affects the accuracy of the photometric redshift prediction. We show that for the simulated set of galaxies, based on the work of Jouvel et al. (2009), we are able to obtain a photometric redshift accuracy of $\Delta z_{\rm norm} = 0.0026$ and $\max_{\rm norm} = 0.0652$ using 1600 basis functions which is a factor of seven improvement over the standard ANNz implementation. We find that GP-VC outperformed ANNz on the real data from SDSS-DR12, with an improvement in accuracy of ∼5 per cent, even when restricted to have the same number of free parameters. In future work we will test the algorithm on a range of real data, and pursue investigations of how the algorithm performs over different redshift regimes and for different galaxy types.

## ACKNOWLEDGEMENTS

IAA acknowledges the support of King Abdulaziz City for Science and Technology. MJJ and SNL acknowledge support from the UK Space Agency. The authors would like to thank the reviewers for their valuable comments.

## REFERENCES

Abdalla F. B., Banerji M., Lahav O., Rashkov V., 2011, MNRAS, 417, 1891
Alam S. et al., 2015, ApJS, 219, 12
Ball N. M., Brunner R. J., Myers A. D., Strand N. E., Alberts S. L., Tcheng D., 2008, ApJ, 683, 12
Bolzonella M., Miralles J.-M., Pelló R., 2000, A&A, 363, 476
Bonfield D. G., Sun Y., Davey N., Jarvis M. J., Abdalla F. B., Banerji M., Adams R. G., 2010, MNRAS, 405, 987
Bonnett C. et al., 2015, preprint (arXiv:1507.05909)
Brammer G. B., van Dokkum P. G., Coppi P., 2008, ApJ, 686, 1503
Brescia M., Cavuoti S., Longo G., De Stefano V., 2014, A&A, 568, A126
Colless M. et al., 2003, preprint (astro-ph/0306581)

Collister A. A., Lahav O., 2004, PASP, 116, 345
Cunha C. E., Lima M., Oyaizu H., Frieman J., Lin H., 2009, MNRAS, 396, 2379
Driver S. P. et al., 2011, MNRAS, 413, 971
Drucker H., Burges C. J. C., Kaufman L., Smola A. J., Vapnik V., 1997, in Mozer M., Jordan M., Petsche T., eds, Advances in Neural Information Processing Systems 9. MIT Press, Cambridge, MA, p. 155
Feldmann R. et al., 2006, MNRAS, 372, 565
Firth A. E., Lahav O., Somerville R. S., 2003, MNRAS, 339, 1195
Foster L. et al., 2009, J. Mach. Learn. Res., 10, 857
Geach J. E., 2012, MNRAS, 419, 2633
Gibbs M., MacKay D. J. C., 1997, Techincal report, Efficient implementation of Gaussian processes. Cavendish Laboratory, Cambridge, UK
Hildebrandt H. et al., 2010, A&A, 523, A31
Hogan R., Fairbairn M., Seeburn N., 2015, MNRAS, 449, 2040
Hoyle B., Rau M. M., Zitlau R., Seitz S., Weller J., 2015, MNRAS, 449, 1275
Ilbert O. et al., 2006, A&A, 457, 841
Jolliffe I. T., 1986, Principal Component Analysis. Springer-Verlag, New York
Jouvel S. et al., 2009, A&A, 504, 359
Kind M. C., Brunner R. J., 2013, MNRAS, 432, 1483
Laureijs R. et al., 2011, preprint (arXiv:1110.3193)
Le Fèvre O. et al., 2013, A&A, 559, A14
Le Fèvre O. et al., 2015, A&A, 576, A79
LeCun Y., Bottou L., Orr G. B., Mueller K.-R., 1998, Lecture Notes Comput Sci., 1524, 9
Lilly S. J. et al., 2009, ApJS, 184, 218
Lima M., Cunha C. E., Oyaizu H., Frieman J., Lin H., Sheldon E. S., 2008, MNRAS, 390, 118
Mercer J., 1909, Phil. Trans. R. Soc. A, 209, 415
Nocedal J., 1980, Math. Comput., 35, 773
Rasmussen C., Williams C., 2006, Gaussian Processes for Machine Learning, Adaptative Computation and Machine Learning Series. University Press Group Limited, MIT Press, Cambridge, MA
Rau M. M., Seitz S., Brimioulle F., Frank E., Friedrich O., Gruen D., Hoyle B., 2015, MNRAS, 452, 3710
Roberts S., Osborne M., Ebden M., Reece S., Gibson N., Aigrain S., 2013, Phil. Trans. R. Soc. A, 371, 20110550
Sánchez C. et al., 2014, MNRAS, 445, 1482
Schmidt M., 2005, Minfunc: Unconstrained Differentiable Multivariate Optimization in Matlab. Available at: http://www.cs.ubc.ca/schmidtm/Software/minFunc.html
Snelson E., Ghahramani Z., 2006, in Weiss Y., Schölkopf B., Platt J., eds, Advances in Neural Information Processing Systems 18, MIT Press, Cambridge, p. 1257
Tipping M. E., 2001, J. Mach. Learn. Res., 1, 211
Tsiligkaridis T., Hero A., 2013, IEEE Trans. Signal Process., 61, 5347
Vanzella E. et al., 2004, A&A, 423, 761
Way M. J., Foster L. V., Gazis P. R., Srivastava A. N., 2009, ApJ, 706, 623
Weiss G., McCarthy K., Zabar B., 2007, in DMIN, Stahlbock R., Crone S. F., Lessmann S., eds., Cost-Sensitive learning vs. Sampling: Which is best for Handling Unbalanced Classes with Unequal Error Costs, CSREA Press, MIT Press, Cambridge, MA, p. 35
Zhang Y., Leithead W. E., Leith D. J., 2005, in 44th IEEE Conf. Decision and Control and European Control Conference (CDC-ECC 2005). IEEE, Seville Spain, p. 3711

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