
Preprint:
Smith, M.T., Álvarez, M.A. and Lawrence, N.D. (Submitted: 2019) Gaussian process regression for binned data. [Preprint] (Submitted)

Noname manuscript No.
(will be inserted by the editor)

Gaussian Process Regression for Binned Data

Michael Thomas Smith · Neil D Lawrence* · Mauricio A Álvarez

Received: date / Accepted: date

Abstract Many datasets are in the form of tables of binned data. Performing regression on these data usually involves either reading off bin heights, ignoring data from neighbouring bins or interpolating between bins thus over or underestimating the true bin integrals.

In this paper we propose an elegant method for performing Gaussian Process (GP) regression given such binned data, allowing one to make probabilistic predictions of the latent function which produced the binned data.

We look at several applications. First, for differentially private regression; second, to make predictions over other integrals; and third when the input regions are irregularly shaped collections of polytopes.

In summary, our method provides an effective way of analysing binned data such that one can use more information from the histogram representation, and thus reconstruct a more useful and precise density for making predictions.

Keywords Regression · Gaussian Process · Integration

This work has been supported by the Engineering and Physical Research Council (EPSRC) Project EP/N014162/1. We thank Wil Ward and Fariba Yousefi for their assistance & suggestions.

*Work conducted while at the University of Sheffield.

Michael Thomas Smith
Department of Computer Science
University of Sheffield
E-mail: m.t.smith@sheffield.ac.uk

Neil D Lawrence
Department of Computer Science
University of Sheffield
E-mail: neil@sheffield.ac.uk

Mauricio A Álvarez López
Department of Computer Science
University of Sheffield
E-mail: mauricio.alvarez@sheffield.ac.uk

1 Introduction

Consider the following problem. You want to use a dataset of children's ages and heights to produce a prediction of how tall a child of 38 months will be. The dataset has been aggregated into means over age ranges: e.g. those aged 24 to 36 months have an average height of 90cm, those aged 36 to 48 months, 98cm, etc.

A naive approach would be to simply read off the age range's mean. A slightly more advanced method could interpolate between bin centres. The former method fails to use the data in the neighbouring bins to assist with the prediction, while the latter will produce predictions inconsistent with the dataset's totals. Ideally we would have access to the original dataset, however binning such as this is ubiquitous, sometimes for optimisation (for storage or processing, for example hectad counts in ecology, annual financial reports, traffic counts), sometimes as an attempt to preserve privacy (for example geographical and demographic grouping in the census) and sometimes due to the data collection method itself (camera pixels or fMRI voxels; survey selection, as in section 5.5; or rain-gauge measurements taken each hour). The examples in this paper cover some of these use cases, although many others exist. We also demonstrate how this method can be combined with differential privacy (DP), to provide a simple method for performing DP-regression.

This problem is a particular example of symbolic data analysis (SDA); in which a latent function or dataset (micro-data) is aggregated in some way to produce a series of symbols (group level summaries). In SDA inference is then conducted at the symbol-level (Beranger et al, 2018). It often ignores the underlying likely distributions, often assuming that the data lies

uniformly within the histogram bins (Le-Rademacher and Billard, 2017) for example.

In this paper, we propose a method for performing Gaussian Process (GP) regression given such binned data. Gaussian Processes are a principled probabilistic method for performing regression. Put simply they provide a way of describing how one believes data across input space is correlated, and thus make predictions using previously given training data. We show how one can find the correlation between a cuboid region's integral and a single point. In essence, the histogram we are working with can be considered as a simple form of density estimation. We attempt to extract as much information as possible from the histogram representation, and reconstruct a more useful and precise density. We will later refer to the analytically derived kernel as the *integral kernel*.

The analytical method described only applies when the boundaries of the integral are independent of one another (i.e. the volume to be integrated over is a cuboid) which often occurs in real datasets, for example in population surveys one might bin people into age ranges and income ranges. However there are many cases where the bins are non-cuboid. An obvious example is that of census tracts, which often follow complicated paths. Kyriakidis (2004) handles this by approximating the volumes with sets of points, over which the covariance is computed. We briefly re-examine this method (including the problem of point placement and hyperparameter optimisation), extending it to use a set of hyperrectangles to approximate the volumes instead.

The analytical derivation is based on work by Alvarez et al (2009) in which dynamical systems are modelled using a Gaussian process. A related article is that of Ažman and Kocijan (2005) in which derivatives of a function are observed (rather than the function itself). This was later used to enforce monotinicity in Riihimäki and Vehtari (2010). A similar constraint was described in Gosling et al (2007) who use the observation that a GP and its derivative are jointly Gaussian. In the current paper we operate in the opposite direction, and assume we have observations of a collection of definite integrals of a latent function. Oakley and O'Hagan (2007) follow a similar line of reasoning to this paper and use this to sample the posterior using MCMC. Our method has the advantage that it has a closed form solution, but the use of MCMC allowed them to integrate over the model's hyperparameters and enforce other constraints (such as non-negative latent functions, for example if they are describing a probability density function). Our method operates over higher dimensions and its speed means it can be used as part of an approximation for non-rectangular regions, as in Section 4.1. The work of Calder and Cressie (2007) is also related, as it focuses on the creation of new kernels though integration. In their case however it was to apply convolution to the latent function.

Note that, unlike probabilistic integration (O'Hagan, 1991), we are not trying to integrate a function, but rather have been given the integrals of an unknown 'latent' function and wish to reconstruct this unknown function.

There is a slight relationship with the combining of basis functions in functional analysis. In Ramsay (2006, equation 16.3), the authors describe how a new kernel is created by summing over the basis functions of two one-dimensional bases, somewhat like how we integrate over the (effectively infinite, for a GP) basis functions that lie over the domain being integrated.

We derive both the analytical and approximate forms of the kernel, then demonstrate the methods on a series of simulated and real datasets and problems.

2 Analytical Derivation

To begin we consider the analytical formulation in which we believe that there is a latent function that has been integrated to provide the outputs in the training data. We assume, for now, that we want to make predictions for this latent function. To proceed via Gaussian process regression (Williams and Rasmussen, 2006) and continuing with our child-height example, we assume that there is some *latent function*, $f(t)$, that represents the values of height as a function of age. The summary measures (average over age ranges) can then be derived by integrating across the latent function to give us the necessary average. Importantly, if the latent function is drawn from a Gaussian process then we can construct a Gaussian process from which both the latent function and its integral are *jointly* drawn. This allows us to analytically map between the aggregated measure and the observation of interest.

To summarise, we assume that a second function, $F(s,t)$, describes the integral between the ages s and t of $f(\cdot)$ and we are given observations, $y(s,t)$, which are noisy samples of $F(s,t)$.

A Gaussian process assumption for a function specifies that for a set of random variables, the outputs are jointly distributed as a Gaussian density with a particular mean and covariance matrix. The integration operator effectively adds together an infinite sum of scaled covariances. In summary there will be a Gaussian process with a covariance which describes individually and jointly, the two functions $f(t')$ and $F(s,t)$. Such a Gaussian process is specified, *a priori*, by its mean

function and its covariance function. The mean function is often taken to be zero. It is the covariance function where the main interest lies.

To construct the joint Gaussian process posterior we need expressions for the covariance between values of $f(t)$ and $f(t')$, values of $F(s,t)$ and $F(s',t')$ (i.e. the covariance between two integrals) and the 'cross covariance' between the latent function $f(t')$ and the output of the integral $F(s,t)$. Where t, t', s and s' specify input locations.

For the underlying latent function we assume that the covariance between the values of the latent function $f(\cdot)$ is described by the exponentiated quadratic (EQ) form,

$$k_{ff}(u,u') = \alpha\, e^{-\frac{(u-u')^2}{l^2}},$$

where $\sqrt{\alpha}$ is the scale of the output and l is the (currently) one-dimensional length-scale.[1] We are given training points from the integral $F(s,t) = \int_s^t f(u)du$. Reiterating the above, if $f(u)$ is a GP then $F(s,t)$ is also a GP with a covariance we can compute by integrating the covariance of $f(u)$,

$$k_{FF}((s,t),(s',t')) = \int_s^t \int_{s'}^{t'} k_{ff}(u,u')\, \mathrm{d}u'\mathrm{d}u.$$

Substituting in our EQ kernel, and integrating,

$$\begin{aligned} k_{FF}((s,t),(s',t')) = \frac{1}{2}\sqrt{\pi}l\alpha \Bigg[& (s'-s)\,\mathrm{erf}\left(\frac{s-s'}{l}\right) \\ & + (s-t')\,\mathrm{erf}\left(\frac{s-t'}{l}\right) + s'\,\mathrm{erf}\left(\frac{s'-t}{l}\right) \\ & + t\,\mathrm{erf}\left(\frac{t-s'}{l}\right) + (t-t')\,\mathrm{erf}\left(\frac{t'-t}{l}\right) \\ & + \frac{l}{\sqrt{\pi}}\left(-e^{-\frac{(s-s')^2}{l^2}} + e^{-\frac{(s-t')^2}{l^2}} + e^{-\frac{(s'-t)^2}{l^2}} - e^{-\frac{(t-t')^2}{l^2}}\right)\Bigg], \end{aligned} \tag{1}$$

where $\mathrm{erf}(\cdot)$ is the Gauss error function. For ease of interpretation and later manipulation we rewrite this as,

$$\begin{aligned} k_{FF}((s,t),(s',t')) = \alpha\frac{l^2}{2} & \\ \times \Bigg[g\left(\frac{t-s'}{l}\right) + g\left(\frac{t'-s}{l}\right) & \\ - g\left(\frac{t-t'}{l}\right) - g\left(\frac{s-s'}{l}\right)\Bigg] & \end{aligned} \tag{2}$$

where we defined $g(z) = z\sqrt{\pi}\mathrm{erf}(z) + e^{-z^2}$.

Because we are interested in computing a prediction for the latent function (i.e. the density) that's been integrated, it would be useful to have the cross-covariance between F and f. If we assume that the joint distribution of F and f is normal, we can calculate the cross-covariance,

$$\begin{aligned} k_{Ff}((s,t),(t')) = \alpha\frac{\sqrt{\pi}l}{2} & \\ \times \left(\mathrm{erf}\left(\frac{t-t'}{l}\right) + \mathrm{erf}\left(\frac{t'-s}{l}\right)\right). & \end{aligned} \tag{3}$$

When using this 'integral kernel' in a real GP regression problem we are likely to need to select appropriate hyperparameters. Typically this is done using gradient descent on the negative log marginal-likelihood, L, with respect to the hyperparameters. In this case, we need the gradient of k_{FF} wrt l and α (respectively, the length-scale and variance of the latent EQ function).[2] Defining $h(z) = \frac{z\sqrt{\pi}}{2}\,\mathrm{erf}(z) + e^{-z^2}$, we can write the gradient as

$$\begin{aligned} \frac{\partial k_{FF}((s,t),(s',t'))}{\partial l} = \alpha l & \\ \times \Big[h\Big(\frac{t-s'}{l}\Big) + h\Big(\frac{t'-s}{l}\Big) & \\ - h\Big(\frac{t-t'}{l}\Big) - h\Big(\frac{s-s'}{l}\Big)\Big]. & \end{aligned} \tag{4}$$

Similarly we can compute the gradient of the hyperparameters with respect to the cross-covariance (k_{Ff}). Defining another support function $d(z) = \frac{z\sqrt{\pi}}{2}\mathrm{erf}(z) - ze^{-z^2}$, we can show that the gradient is

$$\frac{\partial k_{Ff}((s,t),(s'))}{\partial l} = \alpha \times \Big[d\Big(\frac{t-t'}{l}\Big) + d\Big(\frac{t'-s}{l}\Big)\Big]. \tag{5}$$

We need the gradients of the hyperparameters of the latent function's kernel k_{ff}, we will not state these here as they are already well known.

For each kernel above we can compute the gradient with respect to α simply by returning the expression for the appropriate kernel with the initial α removed.

The same idea can be used to extend the input to multiple dimensions. If we specify that each dimension's kernel function contains a unique lengthscale parameter, with a bracketed kernel subscript index indicating these differences, we can express the new kernel as the product of our one dimensional kernels,

$$k_{FF}((\boldsymbol{s},\boldsymbol{t}),(\boldsymbol{s}',\boldsymbol{t}')) = \prod_i k_{FF(i)}((s_i,t_i),(s_i',t_i')), \tag{6}$$

[1] There is a $\sqrt{2}$ difference between our length-scale and that normally defined, this is for convenience in later integrals. Note that other kernels could be substituted, with associated work to integrate the kernel's expression. The supplementary contains a demonstration using the exponential kernel instead.

[2] These gradients are then multiplied by $\frac{\partial L}{\partial k_{FF}}$ by the GP framework to give the gradients $\frac{\partial L}{\partial l}$ and $\frac{\partial L}{\partial \alpha}$.

with the cross covariance given by

$$k_{Ff}((\boldsymbol{s},\boldsymbol{t}),(\boldsymbol{t}')) = \prod_i k_{Ff(i)}((s_i,t_i),(t_i')).$$

3 Non-negative latent function constraint

It is common for the latent function to describe a feature which is known to be non-negative. Examples include house prices, people's heights and weights, populations, etc. We therefore may wish to constrain the model to only produce non-negative predictions. To address this problem we use as a basis the work of Riihimäki and Vehtari (2010) who constrain a GP posterior mean to be approximately monotonic by adding 'virtual points'. We could use a similar mechanism by adding virtual points that specify observations of our latent function instead. The likelihood of these new points is no longer Gaussian. Instead we use a probit function (as in the reference) with probability approaching zero if negative, and probability approaching one if positive. This non-Gaussian likelihood fails to remain conjugate with the prior. We therefore compute an approximate posterior by applying the expectation propagation (EP) algorithm, as suggested in the reference.

We refrain from reproducing the full derivation of the EP site parameters as the full details are in Riihimäki and Vehtari (2010), however to summarise, we have two types of observation; integrals over the latent function and virtual observations of the latent function itself. For the former the likelihood remains Gaussian, for the latter we use a probit likelihood. The posterior is approximated using EP. We have a joint Gaussian process that describes the latent function, f and its definite integrals, F over hyperrectangles. We use the same expression of Bayes' rule as in Riihimäki and Vehtari (2010),

$$p(F,f|\boldsymbol{y},\boldsymbol{z},\boldsymbol{X},\boldsymbol{V}) = \frac{1}{Z}p(F,f)p(\boldsymbol{y}|F,\boldsymbol{X})p(\boldsymbol{z}|f,\boldsymbol{V})$$

but here $\boldsymbol{y}$ are the observations of the definite integrals (at $\boldsymbol{X}$) of the latent function, $\boldsymbol{z}$ is a placeholder vector representing the latent function's non-negative status at the virtual point locations, $\boldsymbol{V}$. The two likelihood terms are,

$$p(\boldsymbol{y}|F,\boldsymbol{X}) = \prod_{i=1}^{N} \mathcal{N}\left(y_i|F(\boldsymbol{x_i}),\sigma^2\right)$$

$$p(\boldsymbol{z}|f,\boldsymbol{V}) = \prod_{j=1}^{M} \Phi\left(\frac{f(\boldsymbol{v}_j)}{\nu}\right)$$

The normalisation term is,

$$Z = \int p(F,f)p(\boldsymbol{y}|F,\boldsymbol{X})p(\boldsymbol{z}|f,\boldsymbol{V})\, dFdf$$

We then proceed with the EP algorithm to compute a Gaussian approximation to the posterior distribution,

$$q(F,f|\boldsymbol{y},\boldsymbol{z},\boldsymbol{X},\boldsymbol{V}) = \frac{1}{Z_{EP}}p(F,f)p(\boldsymbol{y}|F,\boldsymbol{X})\prod_{i=1}^{M} t_i(\tilde{Z}_i,\tilde{\mu}_i,\tilde{\sigma}_i),$$

where t_i are scaled Gaussian, local likelihood approximations, described by the three 'site parameters'. Thus the posterior in this approximation is again Gaussian and a mean and covariance can be computed, using the EP algorithm (iteratively updating the site parameters and normalising term until convergence).

A final step, once the latent function's mean and variance has been computed is to use the probit link function to generate our posterior prediction, specifically, given the distribution of the latent function prediction $p(f_*|\boldsymbol{X},\boldsymbol{y},\boldsymbol{x}_*,\boldsymbol{V})$ we produce a final prediction fed through the probit link, $\int \Phi(f_*)p(f_*|\boldsymbol{X},\boldsymbol{y},\boldsymbol{x}_*,\boldsymbol{V})df_*$.

Finally a quick note on the placement of the virtual points. The original paper discusses a few possible approaches; for low-dimensional inputs we can space these points evenly over a grid. For higher dimensions one could restrict oneself to placing these points in locations with high probability of being negative. In the examples in this paper where they are used, the dimensionality of the data set is low enough that using a grid of virtual points remains tractable.

4 Arbitrary Polygon Shapes

The product of kernels (6) assumes that we integrate between t_i and t_i' for each dimension i, giving a Cartesian product of intervals. This constrains us to regions consisting of rectangles, cuboids or hyperrectangles. Thus if our input regions are described by polytopes[3] that are not hyperrectangles aligned with the axes, then the above computation is less immediately tractable, as the boundaries of the integral kernels will interact. For specific cases one could envisage a change of variables, but for an arbitrary polytope we need a numerical approximation. Classical methods for quadrature (such as Simpson's method, Bayesian Quadrature, etc) are not particularly suited for this problem, either because of the potential high-dimensionality, or the non-alignment with the axes. If one considered Bayesian Quadrature

[3] A polytope is the generalisation of a polygon to arbitrary numbers of dimensions.

(O'Hagan, 1991) for example, one is left with an analytically intractable integral, with a function with discontinuities describing the boundary of the polytope. We instead follow the more traditional approach described by Kyriakidis (2004) who propose a numerical approximation that mirrors the exact analytical methods in this paper. Specifically they find an approximation to the double integral (2) of an underlying kernel (equation 5 in the reference). Given a uniformly random set of locations ($\boldsymbol{X}$ and $\boldsymbol{X'}$) in each polygon, one sums up the covariances, $k_{ff}(\boldsymbol{x}_i, \boldsymbol{x}'_i)$, for all these pairings. Then to correct for the volumes of the two regions one divides by the number of pairings (NN') and multiplies by the product of their areas/volumes (A and A') to get an approximation to the integral,

$$k_{FF}(\boldsymbol{X}, \boldsymbol{X'}) \approx \frac{AA'}{NN'} \sum_{i=1}^{N} \sum_{j=1}^{N'} k_{ff}(\boldsymbol{x}_i, \boldsymbol{x}'_j).$$

Note that an advantage of this numerical approximation is the ease with which alternative kernels can be used. Their paper does not address the issue of point placement or hyperparameter optimisation. We decided the most flexible approach was to consider every object as a polytope. Each object is described by a series of S simplexes, and each simplex is described by $d+1$ points (each consisting of d coordinates). Selecting the simplexes is left to the user, but one could build a 3d cube (for example) by splitting each side into two triangles and connecting their three points to the cube's centre, thus forming 12 simplexes, requiring $12 \times 4 \times 3 = 144$ input values. Next, for every input polytope we place points. We summarise a method for point placement in Algorithm 1 which describes how one might select points distributed uniformly within each polytope. This method guarantees points will be placed in the larger simplexes that make up the set of polytopes (if the expected number of points within that simplex is greater than one) which means that the points will be placed pseudo-uniform-randomly, aiding the approximation as this offers a form of randomised quasi-Monte Carlo sampling. We compared this to a simple Poisson-disc sampling combined with the simplex sampling to further reduce discrepancy.[4] Finally, for each pair of points between each pair of polytopes we compute the covariance and the gradient of the kernel with respect to the hyperparameters, θ. To compute the gradient of the likelihood, L, with respect to the hyperparameters, we need to compute the gradients for all the $N \times N'$ point pairings, using the kernel, $k_{ff}(\cdot, \cdot)$, of the latent function, and average (taking into account the areas (A and A') of the two polygons);

$$\frac{\partial L}{\partial \theta} = \frac{AA'}{NN'} \sum_{i=1}^{N} \sum_{j=1}^{N'} \frac{\partial k_{ff}(\boldsymbol{x}_i, \boldsymbol{x}'_j)}{\partial \theta} \frac{\partial L}{\partial k_{ff}(\boldsymbol{x}_i, \boldsymbol{x}'_j)}.$$

[4] Future work might also wish to compute an equivalent to the Sobel sequence for sampling from a simplex.

4.1 Hyperrectangle Numerical Approximation

One obvious proposal is to combine the numerical and analytical methods. We also generalise the above method to handle the covariance between a pair of sets of polytopes. Specifically, rather than approximate a set of polytopes with points, one could, conceivably achieve a higher accuracy by replacing the points with the same number of hyperrectangles, placed to efficiently fill the polytopes. As with the point method, but with hyperrectangles; we compute the covariance k_{FF} between all pairings of hyperrectangles from the different sets of polytopes and then sum these to produce an estimate for the covariance between the two sets of polytopes (potentially correcting for the volume of the two sets of polytopes if the two sets of hyperrectangles do not completely fill them). Specifically, we compute,

$$k_{FF}(\boldsymbol{X}, \boldsymbol{X'}) \approx \sum_{i=1}^{N} \sum_{j=1}^{N'} \frac{A_i A'_j}{a_i a'_j} k_{FF}(\boldsymbol{x}_i, \boldsymbol{x}'_j),$$

where A_i refers to the volume of the polytope associated with hyperrectangle i (note other hyperrectangles may also be associated with that polytope), and a_i is the sum of the volumes of all the hyperrectangles being used to approximate the same polytope. Thus their ratio gives us a correction for the hyperrectangle's volume shortfall.

The placement of the hyperrectangles is a more complex issue than the placement of the points in the previous section. For the purposes of this paper we use a simple greedy algorithm for demonstration purposes. Other work exists on the time complexity and efficient placement of rectangles to fill a polygon, although many either allow the rectangles to be non-axis-aligned or requires the polygon to be an L shape (Iacob et al, 2003) or orthogonal, or are only for a single rectangle (Daniels et al, 1997) in a convex polygon (e.g. Knauer et al, 2012; Alt et al, 1995; Cabello et al, 2016). We found the straightforward greedy algorithm to be sufficient.

5 Results

We illustrate and assess the above methods through a series of experiments. We start, in Section 5.1 with a simple one-dimensional example in which we have noisy

Algorithm 1 Pick a random point inside a polytope.

Require: $\mathcal{T}$, the polytope we want to fill with samples - described by a list of $d \times n$ matrices defining simplexes. d spatial dimensions and $n = d + 1$ vertices.
Require: ρ, density of points (points per unit volume)

function GetUniformSamples($\mathcal{T}$, ρ)
  for Simplex, $\mathcal{S}$ `in` $\mathcal{T}$ **do**
    $V \leftarrow$ CalcVolume($\mathcal{S}$)
    for $0 \leq i < V\rho$ **do**
      $P \leftarrow P \cup$ SimplexRandomPoint($\mathcal{S}$)
    end for
  end for
end function

function CalcVolume($\mathcal{S}$ made of vertices $\boldsymbol{v}_0 ... \boldsymbol{v}_{n-1}$)
  ▷ modified from Stein (1966)
  return $\left|\frac{1}{d!} \det\left[\boldsymbol{v}_1 - \boldsymbol{v}_0, \boldsymbol{v}_2 - \boldsymbol{v}_0, \ldots, \boldsymbol{v}_{n-1} - \boldsymbol{v}_0\right]\right|$
end function

function SimplexRandomPoint($\mathcal{S}$)
  ▷ Algorithm duplicated from Grimme (2015)
  z $\leftarrow [1] +\!\!+ \texttt{uniform}(d) +\!\!+ [0]$ ▷ see footnote†
  $l_i \leftarrow z_i^{1/(n-i)} \quad 1 \leq i \leq n$
  return $\sum_{i=1}^{n} (1 - l_i)(\prod_{j=1}^{i} l_j)\boldsymbol{v}_i$
end function

†uniform(d) selects d uniformly random numbers. $+\!\!+$ is the concatenation operator.

observations of a series of definite integrals and we want to estimate the latent function. In Section 5.2 we use another synthetic dataset to illustrate the non-negative virtual point constraints on the posterior. In Section 5.3 we use a real dataset describing the age distribution of a census tract, with the individuals providing the data made private through the differential privacy framework (Dwork and Roth, 2014). We demonstrate how the method can support inference on noisy, differentially private data and test the non-negative constrained integral. In Section 5.4 we consider another histogram example, but this time with a higher dimensional input, of the durations of hire bike users, given the start and finish station locations. In Section 5.5 we extend the method to predict other integrals (not just densities). Finally in Section 5.6 we consider non-rectangular input volumes and compare numerical approximations for GP regression. In these later sections the latent function output is far from zero, thus the non-negative constraint had no effect (and is not reported).

5.1 Speed Integration Example

Before looking at a real data example, we illustrate the kernel with a simple toy example. We want to infer the speed of a robot that is travelling along a straight line. The distance it has travelled between various time points has been observed, as in Table 1. A question we might ask, how fast was the robot moving at 5 seconds? We enter as inputs the four integrals. We select the lengthscale, kernel variance and Gaussian noise scale by maximising the log marginal likelihood, using gradient descent (Williams and Rasmussen, 2006, Section 5.4.1). We now can make a prediction of the latent function at five seconds using standard GP regression. Specifically the posterior mean and variances are computed to be,

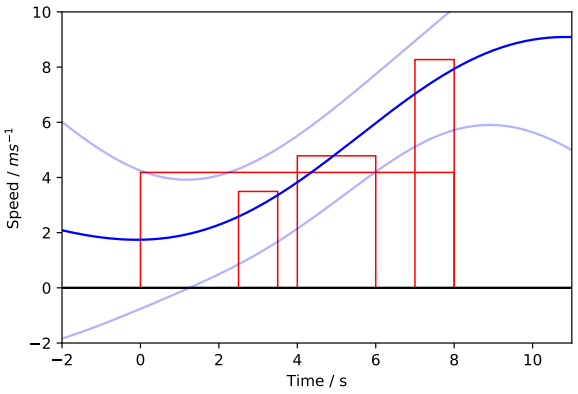

Fig. 1 Illustration of how the robot's speed can be inferred from a series of observations of its change in location, here represented by the areas of the four rectangles. The blue lines indicate the posterior mean prediction and its 95% confidence intervals.

$$\bar{f}_* = \boldsymbol{k}_{F*}^\top (\boldsymbol{K}_{FF} + \sigma^2 I)^{-1} \boldsymbol{y} \tag{7}$$

$$\mathrm{V}[f_*] = k_{**} - \boldsymbol{k}_{F*}^\top (\boldsymbol{K}_{FF} + \sigma^2 I)^{-1} \boldsymbol{k}_{F*}, \tag{8}$$

where $\boldsymbol{K}_{FF}$ is the covariance between pairs of integrals, $\boldsymbol{k}_{F*}$ is the covariance between a test point in latent space and an integral. σ^2 is the model's Gaussian noise variance. $\boldsymbol{y}$ are the observed integral outputs and k_{**} is the variance for the latent function at the test point.

The optimal hyperparameters that maximise the log marginal likelihood, are for the kernel to have variance of $12.9\mathrm{m}^2\mathrm{s}^{-2}$ and lengthscale 7.1s, model likelihood Gaussian noise, $0.6\mathrm{m}^2\mathrm{s}^{-2}$.

Figure 1 illustrates the four observations as the areas under the four rectangles, and shows the posterior prediction of the GP. To answer the specific question above, the speed at $t = 5$s is estimated to be $4.87 \pm 1.70\mathrm{ms}^{-1}$ (95% CI). We constructed the synthetic data with a function that increases linearly at $1\mathrm{ms}^{-2}$, with added noise. So the correct value lies inside the prediction's CIs.

5.2 Non-negative constraint

As a simple demonstration of the non-negative constraint in operation, we consider a synthetic one di-

Start location / m	End location / m	Time
0	8	33.47
2.5	3.5	3.49
4	6	9.56
7	8	8.27

Table 1 Simulated observations of robot travel distances. Figure 1 illustrates these observations with rectangle areas.

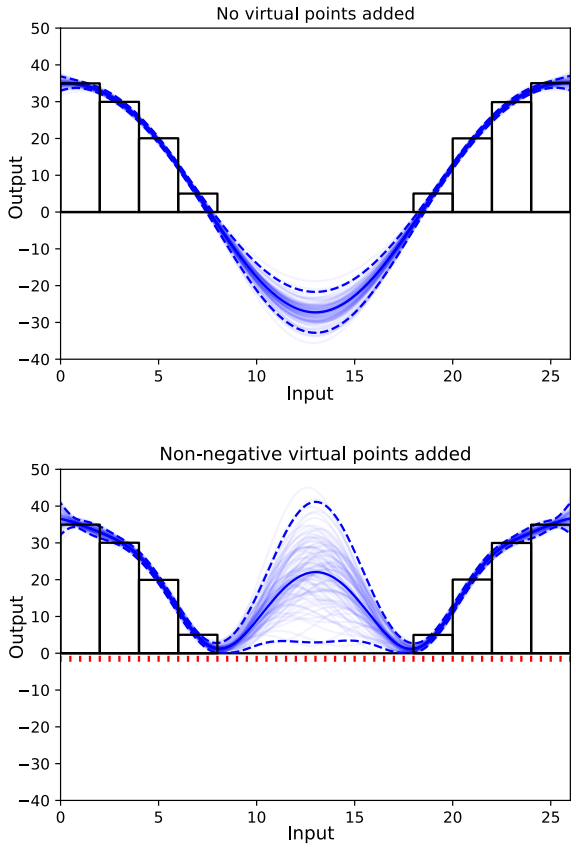

Fig. 2 Synthetic dataset demonstrating the use of virtual points (locations indicated by red ticks below axis) to enforce non-negativity. Mean, solid blue line; 95%-CI, dashed blue line. The upper figure uses a simple Gaussian likelihood without virtual points. The lower figure has a grid of virtual points and a probit likelihood function for these observations. Note that the latent posterior mean and uncertainty is fed through this link function to produce the mean and CIs plotted.

mensional dataset of eight observations arranged to encourage the posterior mean to have a negative region. We then place a regular grid of fifty-three virtual points over the domain. Figure 2 illustrates both the standard Gaussian-likelihood prediction and the result with these probit-likelihood virtual points. There are no observations between output locations eight and eighteen leaving the function largely unconstrained thus there is large uncertainty in this part of the domain. The reader may notice that the uncertainty in this part of the domain is greater for our constrained model. This is not directly a result of the constraints, but rather due to shorter lengthscales. When the GP hyperparameters were optimised for the constrained example, the lengthscales chosen by the ML procedure were significantly shorter (4.36 instead of 10.37), one can see that this is necessary, as any function that both fits the data but also avoids becoming negative requires a relatively steep change in gradient (around eight and eighteen in the plot).

5.3 Differentially Private Age Data

We consider the age distribution of 255 people from a single output area (E00172420) from the 2011 UK census.[5] We also make this histogram differentially private, to demonstrate the improved noise immunity of the new method. We group the people into a histogram with equal ten year wide bins, and add differentially private noise using the Laplace mechanism (Dwork and Roth, 2014, section 3.3). Specifically we take samples from a scaled Laplace distribution and add these samples to the histogram's values. The Laplace noise is scaled such that the presence or absence of an individual is provably difficult to detect, using the ε-DP Laplace mechanism. One can increase the scale of the noise (by reducing ε) to make it more private, or increase ε, sacrificing privacy for greater accuracy. The aim is to predict the number of people of a particular age. We use four methods; (i) simply reading off the bin-heights, (ii) fitting a standard GP (with an EQ kernel) to the bin centroids, (iii) using a GP with the integral kernel or (iv) using the integral kernel, constrained to be non-negative.

Figure 3 demonstrates these results. Note that the GP with an integral kernel will attempt to model the area of the histogram, leading to a more accurate prediction around the peak in the dataset. The figure also indicates the uncertainty quantification capabilities provided by using a GP. Not all applications require or will use this uncertainty, but we have briefly quantified the accuracy of the uncertainty by reporting in Table 2 the proportion of the original training data that lies outside the 95% CI (one would expect, ideally, that this should be about 5%).

To explore the interaction of the methods with the addition of noise to the data, we manipulate the scale of the DP noise (effectively increasing or decreasing the scale of the Laplace distribution) and investigate the effect on the RMSE of the four methods. Remember

[5] a peak of students at age 18 was removed, so the graph only includes the permanent residents of the area.

ε	Simple	Centroid	Integral	Non-Neg	95% CI
0.01	15.505	6.165 [16%]	5.236 [7%]	5.282 [54%]	0.024
0.10	5.242	2.139 [25%]	2.158 [9%]	2.826 [35%]	0.051
0.20	5.087	1.916 [22%]	1.735 [8%]	1.966 [12%]	0.046
0.50	5.032	1.874 [19%]	1.652 [7%]	1.715 [11%]	0.012
1.00	5.033	1.835 [16%]	1.611 [7%]	1.696 [12%]	0.005
not-DP	5.030	1.962 [7%]	1.604 [7%]	1.690 [12%]	0.000

Table 2 RMSE for all 100 age bins, for the simple (directly read off histogram), centroid (EQ GP fit to bin centres), integral method and the integral method with the non-negative constraint, for various levels of differential privacy. Computed RMSE using 30 DP samples. 10,000 bootstrap resamples used to compute 95% CI estimate (to 1 significant figure), value quoted is largest of four columns for simplicity. In [brackets] we have recorded the percentage of predictions that lay outside the 95% CI of the posterior. Bin size, 10 years.

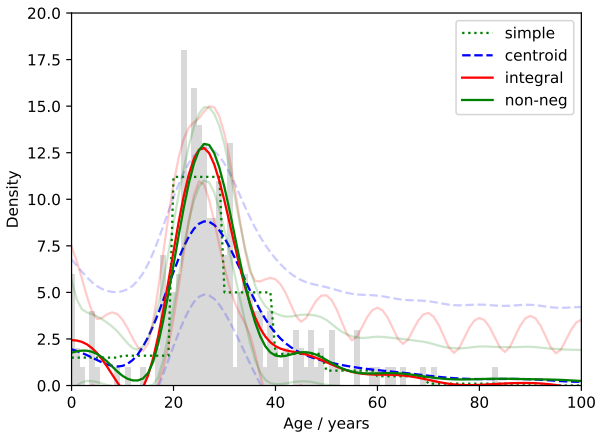

Fig. 3 Various fits to an age histogram of 255 people in which the data has been aggregated into ten-year wide bins (in this case we have not added DP noise), with the original data plotted as grey bars. The dotted, green line uses the bin heights to make predictions directly. The dashed, blue line fits an EQ GP to the centroids of the bins. The red solid line is the prediction using the integral kernel. The solid green line is the integral kernel constrained to be positive. The 95% confidence intervals are indicated with fainter lines.

that decreasing the value of ε makes the prediction more private (but more noisy). This is not cross-validated leave-one-out, as the use-case includes the test point in the aggregation.

Table 2 illustrates the effect of the DP noise scale on the RMSE. We find that the integral method performs better than the others for all noise-scales tested. Intriguingly the simple method seems to be less affected by the addition of DP noise, possibly as the two GP methods effectively try to use some aspect of the gradient of the function, an operation which is vulnerable to the addition of noise. The integral method seems particular useful in the most commonly used values of ε, with meaningful reductions in the RMSE of 13%. The non-negative method does not perform as well. We have set to zero all the negative training data (by default adding the DP noise will make some of the training points negative). If one considers the portion of the domain over 60 years, one can see that the mean of the non-negative-constrained integral-kernel posterior is a little above the others. This occurs as, if one imagines the effect of the non-negative constraint on all possible functions, they will all lie above zero, thus the constraint pushes the mean upwards, i.e. even if the mean without the constraint was non-negative, this mean would have included negative examples. The effect is a worsening of the RMSE/MAE, as many of the training points are zero (which is now further from the posterior mean). The proportion that fall inside the 95% CI is also low as many of the test points are zero, and this model's 95% CI typically will not quite include zero itself.

To describe the DP noise (and potentially differences in the expected noise if the integrals represent means, for example) we added a white noise heteroscedastic kernel to our integral kernel. This effectively allows the expected Gaussian noise for each observation to be specified separately. One could for example then specify the Gaussian noise variance as σ^2/n_i where n_i is the number of training points in that histogram bin (if finding the mean of each bin). If the histogram is the sum of the number of items in each bin we should use a constant noise variance.

5.4 Citibike Data (4d hyperrectangles)

The above example was for a one-dimensional dataset. We now consider a 4d histogram. The New York based citibike hire scheme provides data on the activities of its users. Here we use the start and end locations (lat and long) to provide the four input dimensions, and try to predict journey duration as the output. To demonstrate the integral kernel we bin these data into a 4-dimensional grid, and find the mean of the data points that lie within each bin. To investigate the integral kernel's benefits we vary the number of bins and the number of samples in these bins. As before we compare against an alternative

method in which we fit a GP (using an EQ kernel) to the centroids of the bins. Note that bins that contain no datapoints were not included as training data (as their mean was undetermined). We chose the two models' hyperparameters using a grid search on another sample of citibike data and assess the models by their ability to predict the individual training points that went into the aggregation.

Table 3 illustrates these results. One can see obvious features; more samples leads to more accurate predictions and small numbers of bins causes degradation in the prediction accuracy. However, most interesting is how these interact with the two methods. We can see that for low numbers of bins the integral method does better than the centroid method. With, for example, $5^4 = 625$ bins, the integral method provides no additional support as the data is spread so thinly amongst the bins. The integral kernel will simply act much like the latent EQ kernel. Specifically, these first two experiments suggest that when there are many data points the two methods are fairly comparable, but the integral kernel is of greatest utility when there either are few samples (as shown in Table 3), or they contain considerable noise (as shown in Table 2, for low values of ε).

5.5 Audience Size Estimation (predicting integrals not densities)

We may also wish to produce predictions for new bins. A motivating, real, example is as follows. Imagine you work for a market research company and have a cohort of citizens responding to your surveys. Each survey requires a particular audience answers it. For example the company's first survey, in January, required that respondents were aged between 30 and 40 of any income. They had 320 replies. In February a second survey required that the respondents were aged between 25 and 35 and earned at least \$30k, and had 210 replies. In March their third survey targeted those aged 20 to 30 with an income less than \$40k. How many respondents might they expect? The latent function is population density across the age and income axes, while the outputs are population counts. We can use expressions (2) & (3) as described at the start of Section 2 but use k_{FF} instead of k_{Ff} when making predictions (the inputs at the test points now consist of the boundaries of an integral, and not just the location to predict a single density). Figure 4 illustrates this with a fictitious set of surveys targeting sub-groups of the population.

We simulated a population of 5802 survey-takers by sampling from the US census bureau's 2016 family income database. We distribute the start dates randomly, with a skew towards younger participants. For the example in Figure 4 we computed a prediction for the test region using the integral kernel. We compared this to a model in which the counts had been divided by the volumes of the cuboids to estimate the density in each, and used these with the centroid locations to fit a normal GP (with an EQ kernel) to estimate the density (and hence count) in the test cuboid. For this case we found that both methods underestimated the actual count (of 1641). The centroid method predicted 1263 (95% CI: 991-1535), while the integral method predicted 1363 (95% CI: 1178-1548). The shortfalls are probably due to the skew in the participant start times towards the older portion. The previous training cuboids would have had lower densities, leading to the underestimates here. Intriguingly the integral method still produces a more accurate prediction.

To test this more thoroughly, we simulate 1998 sets of surveys (between 6 and 19 surveys in each set) over this data, and compare the RMSE (and MAE) of the two methods when predicting the number of respondents to a new survey. Table 4 shows that the integral method produces more accurate results in this simulated dataset.

5.6 Population Density estimates (2d non-rectangular disjoint inputs)

In earlier sections we assumed rectangular or cuboid input regions in the training set. However many datasets contain more complicated shapes. In this section we briefly apply the numerical approximation devised by Kyriakidis (2004) and extended in section 4 to use hyperrectangles to fill the polytopes. In this example we use the population density of areas from the UK census. In particular those output areas lying within a 16km^2 square, centred at Easting/Northing 435/386 km (Sheffield, UK). We assume, for this demonstration, that we are given the total population of a series of 40 groupings of these output areas (the output areas have been allocated to these sets uniformly and randomly). This simulates a common situation in which we know the aggregate of various subpopulations. The task then is to predict the population density of the individual output areas that make up the aggregates. Figure 5 demonstrates example placement results, while Table 5 demonstrates the effect of changing the number of points/rectangles on the MAE. For the lowest numbers of approximating points the hyperrectangle approximation had a lower MAE for an equal number of approximating points. Significantly more points were needed (approximately 3-4 times as many) when using points to approximate the MC integration than when using the hyperrectangles, to reach the same accuracy.

	Number of bins									
	2^4		3^4		4^4		5^4		6^4	
Samples	Centroid	Integral	Centroid	Integral	Centroid	Integral	Centroid	Integral	Centroid	Integral
80	487.4	**465.1**	494.9	**482.8**	485.9	**469.9**	478.5	**452.8**	**471.3**	474.7
160	487.4	**475.5**	**471.1**	481.6	**451.0**	464.2	**425.5**	435.7	**406.9**	408.9
320	473.1	**466.2**	453.4	**438.3**	422.4	**388.9**	**408.9**	431.9	378.8	**374.0**
640	468.6	**465.5**	449.5	**435.5**	413.4	**374.7**	**372.5**	379.9	366.7	**365.2**
1280	469.8	**467.5**	450.7	**446.3**	418.1	**375.2**	**382.1**	385.3	375.6	**370.1**
2560	468.9	**467.7**	456.9	**436.4**	420.3	**373.1**	363.1	**360.3**	**353.0**	359.3

Table 3 Mean Absolute Error in predictions of journey duration (in seconds) for the citibike dataset using the integral and centroid methods, over a variety of sample counts and bin counts. 1000 randomly chosen journeys were used in the test set, experiment performed once for each configuration. Bold highlights best of each pair.

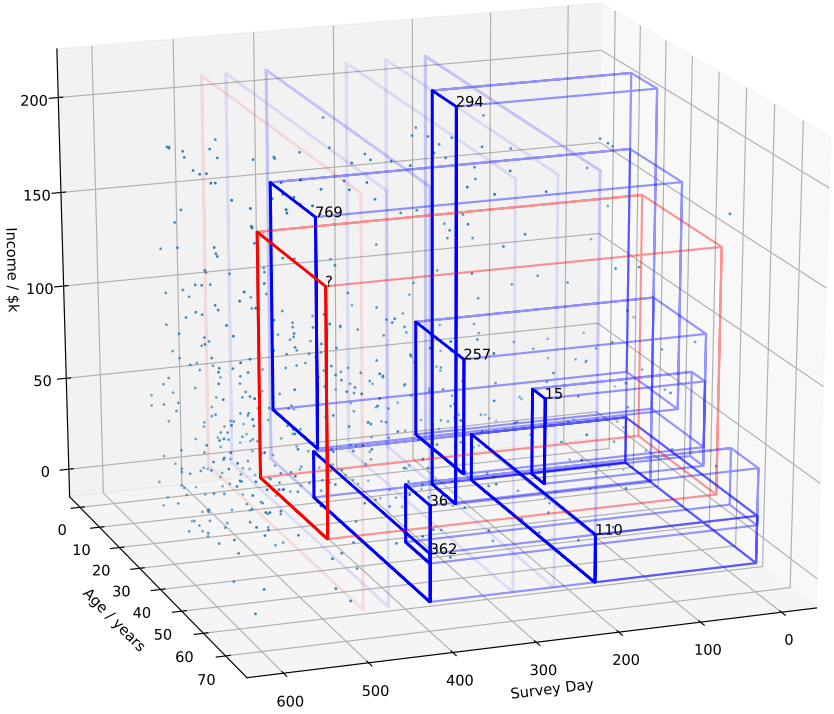

Fig. 4 A demonstration of the audience-size estimation problem. Within the 3d volume lie the individuals that make up the population subscribed by the company. Their location in 3d specified by the date they joined, their income and age. Seven previous surveys (in blue) have been performed over a growing group of clients. Each survey is indicated by a rectangle to indicate the date it occurred and the age/income of participants recruited. All the participants within the cuboid projected backwards from the rectangle are those that had already registered by the date of the survey and so could have taken part. Each volume is labelled with the number of people which took part in each survey. In red is a new survey we want to estimate the count for.

	Method			
	Integral		Centroid	
RMSE	126.3	± 10.74	223.1	± 14.88
MAE	73.1	± 4.52	143.5	± 7.51

Table 4 RMSE and MAE for 1998 randomly generated audience survey requests. 95% CIs for these statistics was calculated using non-parametric Monte Carlo bootstrapping with 100,000 samples with replacement.

The lower-discrepancy sampling did not appear to significantly improve the results of the point approximation.

As another example we look at the covariance computed between three sets of polygons illustrated in Figure 5. We test both the point- and hyperrectangle- approximations. Table 6 shows the MAE when computing the covariance between these three sets of polygons. Using the rectangle approximation reduces the error by

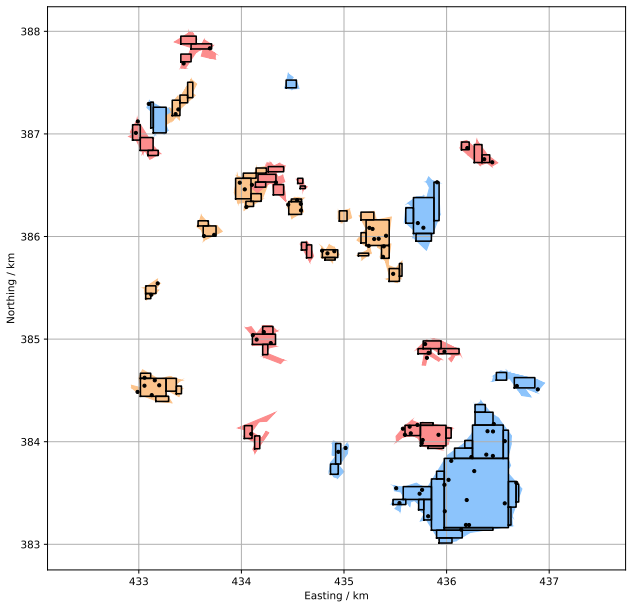

Fig. 5 Example of both rectangular and point approximation to three sets of polygons (from the census output areas of Sheffield). With approximately 30 rectangles or points used for each set.

Number of approximating points	Mean Abs. Error points	low-disc	hyperrects.	Std. error
2	242.3	242.0	198.0	2.1
4	209.2	207.7	183.8	2.5
8	194.4	193.6	185.1	1.8
16	187.3	189.4	187.1	1.1
32	186.1	187.6	185.7	0.9
64	185.5	185.9	185.4	0.4

Table 5 Number of integration approximation features per input for points, lower-discrepancy points and hyperrectangle shape integral methods, and the effect this has on the MAE of the output area density predictions (population density, people per km^2). Reported MAE based on average of twenty point placement iterations. Maximum std. error for each row shown (computed from 14 runs of each). Lengthscale = 160m. kernel variance = 160. Gaussian likelihood variance = 1, variances originally in units of people^2 but the outputs were normalised.

approximately 4 times, for the same number of training points/rectangles.

Number of points or rectangles	Mean Abs Error Points	Rectangles
16	0.0197	0.0049
32	0.0084	0.0018
64	0.0007	0.0002
128	0.0004	< 0.00015

Table 6 Mean Absolute Error in estimates of the covariance matrix values between the three sets of polygons illustrated in Figure 5. The estimated 95% error is ± 0.0001 due to uncertainty in true covariance. Isotropic EQ kernel, lengthscale = 1km.

We experimented briefly at higher dimensions, looking at the estimates of the covariance between a pair of 4-dimensional hyperspheres of radii one and two placed with centres three units apart, so just touching. Using an isotropic EQ kernel (lengthscale=2.0) we compared ten points to ten rectangles in each sphere and found that the estimated covariance using points (instead of hyper-rectangles) had roughly double the MAE (specifically the correct value was 314, with MAEs for points and rectangles were 50.0 and 25.1 respectively).

6 Discussion

In this paper we have derived both an analytical method for inference over cuboid integrals and an approximate method for inference over arbitrary inputs consisting of arbitrary sets of polytopes. In all the experiments, the integral kernels were able to improve on widely used alternatives. However, the improvement was most pronounced when the training data was binned into relatively few bins. The first example, using age data from a census area, demonstrated most clearly why this method may perform more accurately than the 'centroid' alternative; when the dataset has a peak or trough, the centroid method will fail to fully explain the bin integrals, and will have shallower responses to these changes than the data suggests is necessary. Using the method to predict integrals (Section 5.5) was particularly effective, when compared to the centroid alternative. One immediate use case would be estimating the number of young adults from the age histogram, for example, for making local-shop stocking decisions, etc; the centroid method would massively underestimate the number of people in their mid-20s.

In some of the examples we model count data, this typically is non-negative, so we incorporate the work of Riihimäki and Vehtari (2010) to enforce a non-negative latent function. This changes the posterior considerably and the ML estimates of the hyperparameters, thus influencing the entire domain. The practical utility of this operation probably depends on the dataset, for the

example we used, the less-principled Gaussian-likelihood-only method performed slightly better.

Other kernels could be substituted for the EQ. Although this requires some analytical integration work, we have found for other popular kernels the derivation straightforward. The supplementary contains an example of the exponential and linear kernel.

Finally, in Section 4, we looked at approximation methods for non-cuboid, disjoint input regions. First we implemented the point-based approximation of Kyriakidis (2004). Although it did not achieve a particularly practical RMSE on the census dataset, it beat the centroid alternative, and provides a principled method for handling such data. However it is likely to be restricted to lower-dimensional spaces due to the increasing number of approximation points required in higher dimensions. We then replaced the approximation built of points with one built of rectangular patches, and used the covariance computed using the integral kernel. We found we needed considerably fewer rectangles than points to achieve similar accuracies. It is important to note though that the benefit from reduced numbers of training points is likely to be cancelled by the complexity of the integral kernel's covariance function, specifically the computation of four erfs in (2) and (3). However the relative advantages depend on the shape being approximated. Clearly an L shape will probably be more efficiently approximated by two rectangles than by many randomly placed points. Further improvements are possible for more complex shapes, as we have not used the most efficient rectangle placement algorithm. The rectangles could extend beyond the shape being approximated. One could introduce rectangles that contribute a negative weight, to delete those outlying regions, or cancel out patches where two rectangles have overlapped. We leave such enhancements for future researchers.

In this paper we have proposed and derived principled and effective methods for analytical and approximate inference over binned datasets. We have tested these methods on several datasets and found them to be effective and superior to alternatives. This provides an easy, useful and principled toolkit for researchers and developers handling histogrammed or binned datasets, who wish to improve their prediction accuracies.

References

Alt H, Hsu D, Snoeyink J (1995) Computing the largest inscribed isothetic rectangle. In: Canadian Conference on Computational Geometry, pp 67–72

Alvarez M, Luengo D, Lawrence N (2009) Latent force models. In: Artificial Intelligence and Statistics, pp 9–16

Ažman K, Kocijan J (2005) Comprising prior knowledge in dynamic Gaussian process models. In: Proceedings of the International Conference on Computer Systems and Technologies, vol 16

Beranger B, Lin H, Sisson SA (2018) New models for symbolic data analysis. arXiv preprint arXiv:180903659

Cabello S, Cheong O, Knauer C, Schlipf L (2016) Finding largest rectangles in convex polygons. Computational Geometry 51:67–74

Calder CA, Cressie N (2007) Some topics in convolution-based spatial modeling. Proceedings of the 56th Session of the International Statistics Institute pp 22–29

Daniels KL, Milenkovic VJ, Roth D (1997) Finding the largest area axis-parallel rectangle in a polygon. Computational Geometry 7:125–148

Dwork C, Roth A (2014) The algorithmic foundations of differential privacy. Foundations and Trends in Theoretical Computer Science 9(3-4):211–407

Gosling JP, Oakley JE, O'Hagan A, et al (2007) Nonparametric elicitation for heavy-tailed prior distributions. Bayesian Analysis 2(4):693–718

Grimme C (2015) Picking a uniformly random point from an arbitrary simplex. `https://www.researchgate.net/profile/Christian_Grimme/publication/275348534_Picking_a_Uniformly_Random_Point_from_an_Arbitrary_Simplex/links/553a08800cf247b858815a6b.pdf`, University of Münster [Online; accessed 31-July-2018]

Iacob P, Marinescu D, Luca C (2003) Covering with rectangular pieces. Analele Stiintifice ale Universitatii Ovidius Constanta 11(2):75–86

Knauer C, Schlipf L, Schmidt JM, Tiwary HR (2012) Largest inscribed rectangles in convex polygons. Journal of discrete algorithms 13:78–85

Kyriakidis PC (2004) A geostatistical framework for area-to-point spatial interpolation. Geographical Analysis 36(3):259–289

Le-Rademacher J, Billard L (2017) Principal component analysis for histogram-valued data. Advances in Data Analysis and Classification 11(2):327–351

Oakley JE, O'Hagan A (2007) Uncertainty in prior elicitations: a nonparametric approach. Biometrika 94(2):427–441

O'Hagan A (1991) Bayes–Hermite quadrature. Journal of statistical planning and inference 29(3):245–260

Ramsay JO (2006) Functional data analysis, Chapter 16. Wiley Online Library

Riihimäki J, Vehtari A (2010) Gaussian processes with monotonicity information. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp 645–652

Stein P (1966) A note on the volume of a simplex. The American Mathematical Monthly 73(3):299–301
Williams CK, Rasmussen CE (2006) Gaussian processes for machine learning. the MIT Press