Integral Kernel for Binned Data

Anonymous Author(s)
Affiliation
Address
email

Abstract

1 Introduction

It is often necessary to perform regression over binned data. Either because the original data is binned or because one wishes to use the aggregation of data as a method of optimisation (for storage or processing) or privacy protection. The examples in this paper cover some of these use cases, but it is likely other motivations exist for such data aggregation.

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: to preserve anonymity[1], the dataset is aggregated into the means over age ranges: e.g. aged 24 to 36 months average 90cm, aged 36 to 48 months, 92cm, 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.

We propose an elegant method for performing Gaussian Process (GP) regression given binned data. Gaussian Processes are a principled probabilistic method for performing regression given uncertain training data inputs. 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.

The analytical method described only applies when the boundaries of the integral are indpendent (i.e. the volume to be integrated over is a cuboid). This often applies (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. To demonstrate the method in higher dimensions we'll consider citibike data... [etc].

[input is BMI but inputs are Height and Weight] if one portion of a tract experienced housing development one may wish to include that region in [sensor geometry in a particle accelerator?] - citibike example (with locations combined into census or postal areas) - note: optimise by grid-search and x-val -> as ML estimate will be underestimating the latent function's complexity?

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

[1] Aggregation alone is not enough to protect individual privacy as two noise free aggregated values could be combined to reveal data that was supposed to be private.

Submitted to 31st Conference on Neural Information Processing Systems (NIPS 2017). Do not distribute.

to make predictions for this latent function. To proceed via Gaussian process regression (see e.g. **?**) we assume that there is some *latent function*, $f(t)$, that represents the values of weights 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 its integral is also *jointly* drawn from the same Gaussian process. This allows us to analytically map between the aggregated measure and the observation of interest.

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

A Gaussian process assumption for a function specifies that any finite realisation of points from that function should be jointly distributed as a Gaussian density with particular mean and covariance matrix. A Gaussian density has the property that any linear combination of its samples will, in turn, be jointly distributed as Gaussian with the original density. Similarly for a Gaussian process, any *linear operator* (such as integration) applied to the original function will lead to a joint Gaussian process over the result of that linear operator and the original function. In other words there will be a joint Gaussian process between 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 (although non-zero mean functions are easily incorporated into the framework) but 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 $f(t')$ at different times (t and t'), values of $F(s,t)$ and $F(s',t')$, i.e. the covariance between two integrals, and the 'cross covariance' between the the latent function $f(t')$ and the output of the integral $F(s,t)$.

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

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

where α is the scale of the output and ℓ is the (currently) one-dimensional length-scale.[2] To compute the covariance for the integrated function, $F(\cdot)$ we integrate the original EQ form over both its time variables,

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

substituting in our EQ kernel, and integrating;

$$k_{FF} = \frac{1}{2}\sqrt{\pi} l \alpha \left((s'-s)\,\mathrm{erf}\left(\frac{s-s'}{l}\right) + (s-t')\,\mathrm{erf}\left(\frac{s-t'}{l}\right) + s'\,\mathrm{erf}\left(\frac{s'-t}{l}\right) + t\,\mathrm{erf}\left(\frac{t-s'}{l}\right) + (t-t')\,\mathrm{erf}\left(\frac{t'-t}{l}\right)\right.$$

We rewrite this as,

$$k_{FF}((s,t),(s',t')) = \alpha\frac{\ell^2}{2}\left[g\left(\frac{t-s'}{l}\right) + g\left(\frac{t'-s}{l}\right) - g\left(\frac{t-t'}{l}\right) - g\left(\frac{s-s'}{l}\right)\right] \quad (1)$$

where we defined $g(z) = z\sqrt{\pi}\mathrm{erf}(z) + e^{-z^2}$ and erf$(\cdot)$ is the Gauss error function.

Similarly we can calculate the cross-covariance between F and f,

$$k_{Ff}((s,t),(t')) = \frac{\sqrt{\pi} l}{2}\left(\mathrm{erf}\left(\frac{t-t'}{l}\right) + \mathrm{erf}\left(\frac{t'-s}{l}\right)\right).$$

Finally, for the numerical optimisation of the kernel parameters we need the gradient of K_{FF} wrt l and α. Defining $h(z) = \frac{z\sqrt{\pi}}{2}\,\mathrm{erf}(z) + e^{-z^2}$. We can write the gradient as

$$\frac{\delta k_{FF}}{\delta l} = \alpha l \left[h\Big(\frac{t-s'}{l}\Big) + h\Big(\frac{t'-s}{l}\Big) - h\Big(\frac{t-t'}{l}\Big) - h\Big(\frac{s-s'}{l}\Big)\right]$$

[2] Note that there is a $\sqrt{2}$ difference between our length-scale and that normally defined, this is for convenience in later integrals.

and $\frac{\delta k_{FF}}{\delta \alpha}$ is simply the expression for k_{FF} in equation 1 with the initial α removed.

The same idea can be used to extend the input to multiple dimensions. Each kernel function contains a unique lengthscale parameter, with the bracketed kernel subscript index indicating these differences. In conclusion we can express the new kernel as the product of our one dimensional kernels:

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

with the cross covariance given by

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

2.1 Experiment on Age Data

We apply the approach to the age distribution of 255 people from a single output area (E00172420) from the 2011 UK census. We group the people into a histogram with equal ten year wide bins. Our implementation is based on the GPy software (**?**), here we fixed the lengthscale we use GPy's optimization algorithm to select the noise variance and kernel scale parameter (for both kernels) which maximize the log likelihood of the data.

Figure **??** illustrates the improvement over the EQ kernel, while figure **??** shows a GP fit to the histogram data. To aid intuition, figure **??** illustrates the same data binned into 20 year wide domains, and illustrates EQ and integral kernel GP means in which the sample noise is fixed at zero, and lengthscale fixed at 10 years. Note how, without noise, the EQ kernel would pass through the top-centre of each histogram bin, while the integral kernel will ensure the area under its curve is equal to the area of the bin.

Figure **??** shows that for all values of the lengthscale parameter, the integral kernel produces a more accurate estimate of the original data. We believe that, in general, this kernel will be superior for regressing binned or histogram datasets. Both GP regression results were, with the right lengthscale, superior to the use of the binned means.

Integral Predictions: Audience Size Estimation

We may also wish to produce predictions for new bins. A motivating, real, example is as follows. Imagine you are 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 either gender. They had 320 replies. In February a second survey required that the respondents were aged between 25 and 35 and female, and had 210 replies. In March their third survey targetted males aged 20 to 30. How many respondents might they expect? The latent function is population density across the age and gender axes, while the outputs are population counts. We can use the same expressions 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).

3 Arbitrary Polygon Shapes

If the polygons aren't hypercuboids aligned with the axes, then this computation is less immediately tractable.

We propose the following numerical approximation.

To reiterate, we are interested in finding an approximation to the double integral of an underlying kernel (which describes the covariance of a latent function) integrating over every pair of locations in the two polygons. We could simply pick a reasonable set of locations in each polygon, sum up the covariances for all these pairings, then divide by the number of pairings and multiply by the product of their areas/volumes to get an approximation to the integral:

$$\frac{AA'}{N} \sum_{i=1} N k(x_i, x'_i)$$

A brief note on describing the d-dimensional (hyper)volumes. 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.

Algorithm **??** in the supplementary describes how one might select points distributed uniformly within each polytope.

Finally for every input polytope we place points. Then 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. To compute the gradient of the likelihood wrt the hyperparameters, we compute;

$$\sum_{i=1}^{N} \sum_{j} \frac{dk_{ij}}{dl}$$

Algorithm 1 Selecting samples uniformly at random

Require: $\mathcal{T}$, the polytope we want to fill with samples - described by a list of simplexes
Require: d, density of points (points per unit volume)

function GETUNIFORMSAMPLES($\mathcal{T}$)
  for SIMPLEX, $\mathcal{S}$ `in` $Polytope$ **do**
    $V \leftarrow$ CALCVOLUME($\mathcal{S}$) ▷ comment
    for $0 \leq i < Vd$ **do** ▷ V, Volume of simplex; d, density of points
      $P \leftarrow P \cup$ SIMPLEXRANDOMPOINT($\mathcal{S}$)
    end for
  end for
end function
function CALCVOLUME($\mathcal{S}$ made of vertices $v_0 ... v_d$) ▷ Implements algorithm in **?**[TODO READ REFERENCE!!!] **return** $\left|\frac{1}{n!} \det\left[v_1 - v_0, v_2 - v_0, \ldots, v_d - v_0\right]\right|$
end function
function SIMPLEXRANDOMPOINT($\mathcal{S}$) ▷ Implements algorithm in **?** [TODO!!!!]
end function

3.1 Delete? Arcs

I'm not sure if this method will be quicker (it might be at higher dimensions??)

Consider a (hyper)sphere centred upon each point in A. For one of these (hyper)spheres, compute the area (or volume) that lies within the triangle (or tetrahedron). All the points within this will have the same value for a stationary kernel.

Regarding the number of dimensions d, the number of points one will need to reasonably describe such a shape probably scales by $O(a^d)$, thus the number of pairings will scale by $O((a^d)^2) = O(a^{2d})$. Using the above method, this will remove that, although doing the computation of the area within polygon will probably scale at least by d, $O(da^{2d})$??? At high dimen

4 Results

Citibike

Audience Estimation

5 Discussion

http://www2.stat.duke.edu/ fei/samsi/Readings/Derivatives/der7.pdf