Non-parametric Bayesian updating and windowing with kernel density and the kudzu algorithm

Robert L Grant

BayesCamp Ltd

80 High Street, Winchester, SO23 9AT, United Kingdom

September 7, 2021

Abstract

The concept of "updating" parameter estimates and predictions as more data arrive is an important attraction for people adopting Bayesian methods, and essential in big data settings. Implementation via the hyperparameters of a joint prior distribution is challenging. This paper considers non-parametric updating, using a previous posterior sample as a new prior sample. Streaming data can be analysed in a moving window of time by subtracting old posterior sample(s) with appropriate weights. We evaluate three forms of kernel density, a sampling importance resampling implementation, and a novel algorithm called kudzu, which smooths density estimation trees. Methods are tested for distortion of illustrative prior distributions, long-run performance in a low-dimensional simulation study, and feasibility with a realistically large and fast dataset of taxi journeys. Kernel estimation appears to be useful in low-dimensional problems, and kudzu in high-dimensional problems, but careful tuning and monitoring is required. Areas for further research are outlined.

Keywords: Bayesian data analysis, big data, density estimation trees, kernel density estimation, non-parametric statistics, streaming data.

1 Background

In Bayesian analysis, the prospect of updating posterior distributions as new data arrive is a practically and philosophically attractive one. Yesterday's posterior becomes today's prior. It might not be practicable to analyse Big Data, including real-time streaming data, other than in batches or windows. Rapid updates of models and predictions could be beneficial, for example, in epidemic forecasting, microeconomic forecasting, pricing models, algorithmic trading, humanitarian aid logistics, and cybersecurity.

Bayesian updating is widely discussed in introductory Bayesian teaching, but almost never practised. The reason may be the difficulty in conceiving of a parametric multivariate distribution for the full collection of parameters and other unknown values such as missing data, a distribution that may combine discrete, bounded continuous and unbounded continuous marginal distributions.

This paper investigates non-parametric approaches: a kernel density estimation (KDE) approach, two potential methods to reduce computational burden, a sampling importance resampling (SIR) implementation, and a novel algorithm called *kudzu*, which smooths over density estimation trees.

Non-parametric Bayesian updating can be used to update within a moving window of recent data (1), by subtracting the estimated density of an old prior sample. Note that this will remove the influence of all data prior to that point, and the original priors, so the original priors should be added back in. Alternatively, a collection of "sliding windows" can be updated in parallel.

1.1 Parametric updating

In general, Bayesian updating has the following form. Data arrive in batches $(\mathbf{X}_1, \mathbf{X}_2 \ldots \mathbf{X}_B)$, of $n_1, n_2 \ldots n_B$ observations, and $n = \sum_{b=1}^{B} n_b$. There is a probabilistic model for the data, with a likelihood $P(\mathbf{X}_b|\theta)$ for the bth batch, where $\theta \in \mathbb{R}^p$ is a vector of parameters. Before the first batch is analysed, there is a prior distribution $P(\theta)$ for the unknown values. For what follows, it does not matter what philosophical interpretation of the original prior probability is used.

The posterior distribution after the first batch is:

$$P(\theta|\mathbf{X}_1) = C_1 P(\theta) P(\mathbf{X}_1|\theta) \tag{1}$$

for some normalising constant C_1, chosen so that the posterior integrates to 1 over its domain. In practice, the posterior is estimated by simulation, using one of various sampling algorithms such as Markov chain Monte Carlo (MCMC), sequential Monte Carlo (SMC), Metropolis-adjusted Langevin algorithm (MALA) or piecewise deterministic Markov processes (PDMPs) (2) (3). This means that the constants C_i need not be calculated and that the output of analysis is a sample from the posterior distribution. This sample comprises m draws, each of which is a vector θ of parameter values.

The posterior from analysing the first batch is used as the prior when analysing the second batch, which is equivalent to an all-data analysis that multiplies the original prior by both batches' likelihoods (Equation 2).

$$\begin{aligned} P(\theta|\mathbf{X}_1, \mathbf{X}_2) &= C_2 P(\theta|\mathbf{X}_1) P(\mathbf{X}_2|\theta) \\ &= C_2 C_1 P(\theta) P(\mathbf{X}_1|\theta) P(\mathbf{X}_2|\theta) \\ &\propto P(\theta) P(\mathbf{X}_1|\theta) P(\mathbf{X}_2|\theta) \end{aligned} \tag{2}$$

Data can be accumulated indefinitely in this way (Equation 3).

$$
\begin{aligned}
P(\theta|\mathbf{X}_1, \mathbf{X}_2, \dots \mathbf{X}_b) &\propto P(\theta) \prod_{i=1}^{b} P(\mathbf{X}_i|\theta) \\
\therefore \quad log(P(\theta|\mathbf{X}_1, \mathbf{X}_2, \dots \mathbf{X}_b)) &\propto log(P(\theta)) + \sum_{i=1}^{b} log(P(\mathbf{X}_i|\theta))
\end{aligned}
\tag{3}
$$

This offers an advantage in the age of big data (both in terms of volume and velocity) (1), allowing suitably complex Bayesian models to be updated with each batch of data, while only calculating likelihoods for recent data, and thus obtaining updated inferences for θ quickly.

There will be some settings where the data in batch i are not independent of those in batch $i-1$, and this basic updating formula will not apply, unless the model is somehow designed to take this dependency into account, for example with time-series models.

The parametric form of updating specifies a probability density/mass function for the prior at each update, the moments of which are determined by the previous batch's posterior sample. In some distributions, such as the multivariate normal, we can use the previous update's posterior summary statistics as plug-in estimates of the next update's prior distribution hyperparameters. In others, the method of moments could be used. Thus, $P(\theta) \prod_{i=1}^{b-1} P(\mathbf{X}_i|\theta)$ is replaced with an estimated probability density function, $f(\cdot)$:

$$
P(\theta|\mathbf{X}_1, \mathbf{X}_2, \dots \mathbf{X}_b) \propto f(\theta|\mathbf{X}_1, \mathbf{X}_2, \dots \mathbf{X}_{b-1}) P(\mathbf{X}_b|\theta) \tag{4}
$$

The core challenge of Bayesian updating is to estimate $f(\cdot)$ efficiently and accurately.

1.2 Updating in Markov chain Monte Carlo and sampling importance resampling

Algorithm 1 shows parametric updating within the context of MCMC.

Calculation of the posterior requires calculation of a likelihood contribution (a probability density or mass) from each of the n observations, and a prior distribution contribution from each of the p components of θ, so Bayesian sampling algorithms usually have a computational cost of order $\mathcal{O}(n), \mathcal{O}(p)$. Some algorithms, such as MCMC's Hamiltonian Monte Carlo (HMC) variant, MALA and PDMPs, calculate not just the posterior density but the posterior gradients (Jacobian), and use this in some way that more than compensates for the time required to calculate it (2). HMC, implemented in the open source software, Stan (4), is used in this paper. Algorithms using the gradient require smooth priors and likelihoods, and so cannot accommodate discrete parameters (5).

Aside from MCMC, Bayesian updating is also amenable to sampling importance resampling (SIR) (6). SIR samples θ from a tractable distribution $f(\theta)$, and obtains inferences for a challenging but desired distribution $h(\theta)$ by resampling each draw θ_i with probability proportional to weights:

$$\frac{h(\theta_i)/f(\theta_i)}{\sum_i h(\theta_i)/f(\theta_i)} \tag{5}$$

In the Bayesian updating setting, $f(\cdot)$ is the prior distribution, and $h(\cdot)$ is the desired posterior, so the numerator in the weight is proportionate to the likelihood. We can omit the denominator as it is constant.

In practice, we can select a prior draw at random, and then store a number of posterior draws that arise from it in some way yet to be specified, that number being proportional to the likelihood at the prior draw.

Algorithm 1: Parametric Bayesian updating within Markov Chain Monte Carlo

Input : the number of the current update $i \in \mathbb{N}$,
$m \times p$ matrix of prior draws $\mathbf{d}_i$,
n_i-row matrix of data in the ith batch $\mathbf{x}_i$,
a p-vector of initial parameter values θ_0,
a prior probability density function $P(\theta|\xi)$ with hyperparameters ξ,
a function `estimateHyperparameters`($\mathbf{d}$), which returns a vector ξ (for example, by the method of moments),
a function `proposal`($\theta_{j-1}, logPosterior_{j-1}, \nabla logPosterior_{j-1}$) to generate the jth proposal for the p-vector of parameters θ_j
a Boolean function `accept`($\theta_j, logPosterior_j, \theta_{j-1}, logPosterior_{j-1}$) returning TRUE iff θ_j is to be added to the posterior sample (for example, the Metropolis-Hastings accept-reject formula)

Output: $m \times p$ matrix of posterior draws $\mathbf{d}_{i+1}$

$\mathbf{d}_{i+1} = [\,]$
$j = 0$
$\xi_i =$ `estimateHyperparameters`($\mathbf{d}_i$)
while $rows(\mathbf{d}_{i+1}) < m$ **do**
  $logPosterior_j = logLikelihood_j = logPrior_j = 0$
  `// likelihood evaluation`
  for $h \leftarrow 1$ **to** n_i **do**
    $logLikelihood_j \mathrel{+}= log_e(P(\mathbf{x}_i[h,]|\theta_j))$
  end for
  `// prior evaluation`
  $logPrior_j = log_e(P(\theta_j|\xi_i))$
  $logPosterior_j = logPrior_j + logLikelihood_j$
  if $j > 0$ **then**
    `// accept-reject step`
    if `accept`($\theta_j, logPosterior_j, \theta_{j-1}, logPosterior_{j-1}$) **then**
      $\mathbf{d}_{i+1}.appendRow(\theta_j)$
    end if
    else
      $\theta_j = \theta_{j-1}$
    end if
  end if
  $\theta_{j+1} =$ `proposal`($\theta_j, logPosterior_j, \nabla logPosterior_j$)
  $j \mathrel{+}= 1$
end while
return $\mathbf{d}_{i+1}$

SIR is faster than MCMC because the full likelihood is only evaluated for a smaller number of prior draws, not for proposed values of θ which are then rejected by the Metropolis-Hastings formula. There is no need to calculate the prior density or gradients of the posterior.

In typical SIR, $h(\cdot)$ would be approximated by a collection of replicates of draws from $f(\cdot)$, but in this setting we want to use some method, such as KDE, to provide smoothing of the prior. This smoothing can take place before or after likelihood evaluation, which we call SIR-B and SIR-A respectively, for Before and After. In SIR-B, we will *oversample* the prior by adding new draws $\mathbf{d}_f \sim f(\theta|\mathbf{d}_i)$ according to the smoothed prior density, and then evaluating the likelihood (Algorithm 2). This offers an opportunity to add more draws in the region of high likelihoods (local oversampling). This will be especially useful when the likelihood has much higher curvature than the prior, such as early in the updating process. We will consider how to define *local* in the next section.

1.3 Kernels and density estimation trees

Given that it may be impossible, or too time-consuming or error-prone, to find a suitable parametric multivariate form for $f(\cdot)$, non-parametric alternatives would be useful. At heart, this is a density estimation problem. For decades, kernel density estimation has been widely used, but it has not previously been assessed in this context of Bayesian updating. Neufeld and Deutsch developed related methods in geostatistics, but different in two ways: their method is for the purpose of combining datasets rather than updating over time, and does not use posterior draws to estimate the prior density (7).

Kernel density estimation (KDE) produces a density estimate from a

Algorithm 2: Sampling importance resampling updating with a generic prior density estimator

Input : $m \times p$ matrix of prior draws $\mathbf{d}_i$,
a batch of data $\mathbf{X}_i$,
number of prior draws in SIR-B, $m_{sir} > m, m_{sir} \in \mathbb{N}$,
a function $f(\theta|\mathbf{d}_i)$ that estimates the prior density,
a function $h(\theta|\mathbf{X}_i)$ that returns the likelihood of θ,
a pair of natural numbers $[m_{min}, m_{max}]$, the number of posterior draws per prior draw, at zero and maximum likelihood

Output: $m \times p$ matrix of posterior draws $\mathbf{d}_{i+1}$

$\mathbf{d}_{i+1} = [\,]$ `// to hold the posterior draws`
$loglik = [\,]$ `// to hold log-likelihoods`
for $k \leftarrow 1$ **to** m **do**
  $loglik.append(h(\theta = \mathbf{d}_i[k,]|\mathbf{X}_i))$ `// likelihood evaluation`
end for
`// Oversampling, if required, and likelihood evaluation`
if *SIR-B* **then**
  for $k \leftarrow m + 1$ **to** m_{sir} **do**
    $\mathbf{d}_f \sim f(\theta|\mathbf{d}_i)$ `// oversample one prior draw`
    $\mathbf{d}_i.append(\mathbf{d}_f)$
    $loglik.append(h(\theta = \mathbf{d}_f|\mathbf{X}_i))$
  end for
end if
for $k \leftarrow 1$ **to** $nrows(\mathbf{d}_i)$ **do**
  `// find number of posterior draws`
  $m_{draws} = (exp(loglik_k)/max(loglik))(m_{max} - m_{min}) + m_{min}$
  for $j \leftarrow 1$ **to** m_{draws} **do**
    if *SIR-B* **then**
      $\mathbf{d}_j = \mathbf{d}_i[k,]$ // repeat prior draw
    end if
    else
      `// scatter around prior draw`
      $\mathbf{d}_j \sim f(\theta|\mathbf{d}_i[k,])$
    end if
    $\mathbf{d}_{i+1}.append(\mathbf{d}_j)$
  end for
end for
`// reduce posterior or resample as required`
$\mathbf{d}_{i+1} =$ `sample`$(\mathbf{d}_{i+1}, m)$
return $\mathbf{d}_{i+1}$

sample by adding together the densities of a collection of smooth kernel distributions, one centred on each draw in the sample. The sum is then normalised to integrate to one. There is also a body of work on automatically determining the bandwidth, or spread of the kernel, in order to best estimate the distribution (8).

Unfortunately, it is known that kernel density estimation performs poorly in higher-dimensional problems. This is due to the Euclidean distance between points, which increases with dimensionality. Two points, one at the origin $(0, 0, 0, \ldots, 0)$ and the other at $(1, 1, 1, \ldots, 1)$ in p dimensions will be separated by 1 in any single dimension but by $\sqrt{p}$ in total. So, for all but low-p problems, setting a kernel bandwidth that is wide enough to join the kernels together into a smooth density estimate will have the undesired side-effect of enlarging the marginal variances.

High-dimensional density estimation is a field of active research. Most proposed methods, such as normalising flows (9), deep learning (10), or diffusions (11), are either computationally expensive, or too difficult for most data analysts to implement, or both. Density estimation trees (DETs) (12) use the same algorithm as classification and regression trees to orthogonally partition the bounding box (the smallest hypercuboid containing all prior draws) and allocate a density estimate to each "leaf" in the tree. This has advantages over KDE of contiguity and parsimony, but not smoothness. DETs are easy to implement, and the underlying tree algorithm is well-understood and computationally highly efficient. Below, we will propose a *kudzu* algorithm to create a smooth distribution from a DET.

This paper has three objectives:

1. to examine the distortion that KDE in HMC, and two variants which are potentially more efficient, KDE in SIR, and kudzu, introduce to

three illustrative prior densities in the absence of likelihoods ($p = 2, n = 0$)

2. to evaluate the long-run performance of KDE and its two variants, implemented in HMC and SIR, via a simulation study ($p = 3, n = 200$)

3. to explore the feasibility of KDE in SIR and kudzu in a realistically large real-life application ($p = 50, n = 2,657,081$)

2 Methods

Subsequent algorithms focus on the estimation of $f(\cdot)$, replacing only the prior evaluation part of MCMC and the oversampling or posterior scattering parts of SIR-B and SIR-A respectively.

2.1 Kernel density estimation

The m previous posterior draws can be used as the next prior by adding and normalising kernel densities centred on each draw, and using this as the prior density. The simplest approach is to apply an unbounded multivariate kernel (for example, multivariate normal) to each prior draw.

The bandwidth can be determined automatically; we used the method of Sheather-Jones independently on each dimension of the prior draws and considered increasing or decreasing the resulting vector (13). In general, a bandwidth that is too large over-estimates marginal variances, while one that is too small induces multimodality, which can lead to non-convergence of Bayesian sampling algorithms.

We assessed multimodality by Hartigans' dip test (14) (in the R package `diptest`), and each dimension's Sheather-Jones bandwidth was iteratively multiplied by 1.5 while its dip test p-value was less than 0.05. The p-value is

used here as a summary statistic of incompatibility with a unimodal distribution, and does not conflict with the Bayesian approach taken elsewhere. This approach is called *multimodality expansion* in this paper. However, in some common statistical models, such as regularized regression, multimodality is a true characteristic of the posterior; this is explored below.

Algorithm 3: Unbounded kernel updating

Input : $m \times p$ matrix of prior draws $\mathbf{d}_i$,
proposed parameter vector θ,
a function `bandwidth`$(\mathbf{d}_i)$, which returns a p-vector σ_{bw},
a kernel $f_k(\theta|\mathbf{d}_i[k,], \sigma_{bw})$ for the kth prior draw

Output: estimated prior density $f(\theta|\mathbf{d}_i)$

$\sigma_{bw} =$ `bandwidth`$(\mathbf{d}_i)$
$f(\theta|\mathbf{d}_i) = F(\theta|\mathbf{d}_i) = 0$
for $k \leftarrow 1$ **to** m **do**
    $f(\theta|\mathbf{d}_i) \mathrel{+}= f_k(\theta|\mathbf{d}_i[k,], \sigma_{bw})$
    `// this integral is evaluated once, not for every MCMC proposal or SIR prior draw:`
    $F(\theta|\mathbf{d}_i) \mathrel{+}= \int_{\mathbf{\Theta}} f_k(\theta|\mathbf{d}_i[k,], \sigma_{bw})$
end for
$f(\theta|\mathbf{d}_i) = f(\theta|\mathbf{d}_i)/F(\theta|\mathbf{d}_i)$
return $f(\theta|\mathbf{d}_i)$

Algorithm 3 calculates, for each proposed parameter vector θ, n_b likelihoods and m kernel densities, and is $\mathcal{O}(n_b), \mathcal{O}(m)$ for each batch. However, as p grows, the distance between prior draws and kernels grows too, requiring higher m, so using a reduced set of prior kernels to estimate such a density, with a reasonable approximation, might help to keep computation time practicable.

We consider two approaches. Firstly, the prior draws can be aggregated in a grid, and a kernel evaluated at the centre of each occupied hypercuboid, weighted by the count of draws within the hypercuboid (Algorithm 4). Secondly, a bounded kernel (which is non-zero only within some distance of its centre) can be used and evaluated only for draws that are neighbours of the

proposed parameter vector θ (Algorithm 5).

There will inevitably be a tradeoff between speed and accuracy with the hypercuboid grid. Also, the number of occupied hypercuboids will rapidly increase as the number of parameters, p, increases. If there are 10 bins in each dimension, there will be 10^p hypercuboids. It is common practice to store thousands of posterior draws, so even with realistically highly correlated posteriors, this approach is unlikely to reduce computation time much unless $p < 4$, in which case unbounded KDE is likely to be fast and accurate.

The bounded kernel updating (Algorithm 5) is implemented in this paper by transformed $Beta(8, 8)$ distributions. These resemble normal distributions but decrease smoothly to zero density at the boundaries, which are [0,1] prior to transformation. They are made multivariate and uncorrelated by adding their log-densities over all dimensions. They are centred on the prior draws and scaled according to the bandwidth.

Prior draws around the current proposed θ are regarded as neighbours if their L-∞ norm of distances, divided by bandwidth, is less than three. That is, in all dimensions of parameter space, they lie within three bandwidths of θ_j. The challenge for this algorithm is in efficient neighbour identification, which must be done at each proposed θ and therefore must run inside the favoured sampler.

An updating algorithm that used both hypercuboids and bounded kernels would imply contradictory bandwidths: narrow for identifying neighbours and simultaneously wide for smoothing the hypercuboid centres, and so we do not pursue it further.

Algorithm 4: Hypercuboid unbounded kernel updating

Input : $m \times p$ matrix of prior draws $\mathbf{d}_i$,
proposed parameter vector θ,
number of bins in each dimension, $q \in \mathbb{N}$,
a function `getCentres`$(\mathbf{d}_i)$, which returns a $q^p \times p$ matrix $\widetilde{\mathbf{d}}_\mathbf{i}$ of hypercuboid centres,
a function `getCounts`$(\mathbf{d}_i)$, which returns a q^p-vector $\ddot{\mathbf{d}}_\mathbf{i}$ of hypercuboid counts,
a kernel $f_k(\theta|\widetilde{\mathbf{d}}_\mathbf{i}[k], \sigma_{bw})$ for the kth hypercuboid

Output: estimated prior density $f(\theta|\mathbf{d}_i)$

$\sigma_{bw} =$ `bandwidth`$(\mathbf{d}_i)$
$\widetilde{\mathbf{d}}_\mathbf{i} =$ `getCentres`$(\mathbf{d}_i)$
$\ddot{\mathbf{d}}_\mathbf{i} =$ `getCounts`$(\mathbf{d}_i)$
`// remove empty hypercuboids`
for $k \leftarrow q^p$ **to** 1 **do**
  if $\ddot{\mathbf{d}}_\mathbf{i}[k] = 0$ **then**
    $\widetilde{\mathbf{d}}_\mathbf{i}.removeRow(k)$
    $\ddot{\mathbf{d}}_\mathbf{i}.removeRow(k)$
  end if
end for
`// evaluate prior`
$f(\theta|\mathbf{d}_i) = F(\theta|\mathbf{d}_i) = 0$
for $k \leftarrow 1$ **to** m **do**
  $f(\theta|\mathbf{d}_i) \mathrel{+}= \ddot{\mathbf{d}}_\mathbf{i}[k] f_k(\theta|\widetilde{\mathbf{d}}_i[k,], \sigma_{bw})$
  `// this integral is evaluated once, not for every MCMC proposal or SIR prior draw:`
  $F(\theta|\mathbf{d}_i) = \ddot{\mathbf{d}}_\mathbf{i}[k] \int_\mathbf{\Theta} f_k(\theta|\widetilde{\mathbf{d}}_i[k,], \sigma_{bw})$
end for
$f(\theta|\mathbf{d}_i) = f(\theta|\mathbf{d}_i)/F(\theta|\mathbf{d}_i)$
return $f(\theta|\mathbf{d}_i)$

Algorithm 5: Bounded kernel updating

Input : $m \times p$ matrix of prior draws $\mathbf{d}_i$,
proposed parameter vector θ,
a function `getNeighbours`$(\mathbf{d}_i, \theta)$, which returns a $\check{m} \times p$ matrix $\check{\mathbf{d}}_i$ of draws in $\mathbf{d}_i$, which are neighbours to θ

Output: estimated prior density $f(\theta|\mathbf{d}_i)$

$\sigma_{bw} =$ `bandwidth`$(\mathbf{d}_i)$
$\check{\mathbf{d}}_i =$ `getNeighbours`$(\mathbf{d}_i)$
$\check{m} = nrows(\check{\mathbf{d}}_i)$
$f(\theta|\mathbf{d}_i) = 0$
`// prior evaluation`
for $k \leftarrow 1$ **to** $\check{m}$ **do**
  $f_{logbeta}(k) = 0$ `// log-kernel for one neighbour prior draw`
  `// loop over parameters (univariate betas)`
  for $j \leftarrow 1$ **to** p **do**
    $f_{logbeta}(k) += log_e(Beta(0.5 + (\theta[j] - \check{\mathbf{d}}_i[k, j])/6\sigma_{bw}[j]|8, 8))$
  end for
  $f(\theta|\mathbf{d}_i) += e^{f_{logbeta}(k)}$
end for
`// Normalising by the integral is not necessary because the Beta is a proper density function`
$f(\theta|\mathbf{d}_i) = f(\theta|\mathbf{d}_i)/\check{m}$
return $f(\theta|\mathbf{d}_i)$

2.2 Sampling importance resampling

Implementation of SIR follows Algorithm 2, where we use some function $f(\theta|\mathbf{d}_i)$ that estimates the prior density locally around θ to smooth our prior draws $\mathbf{d}_i$, either before or after likelihood evaluation.

SIR is typically faster than MCMC but relies on $\mathbf{d}_i$ to fully explore the parts of parameter space where posterior density is highest. In high dimensions, we may find that only one prior draw contributes any posterior draws, and the posterior becomes one instance of $f(\theta|\mathbf{d}_i)$. When this is an uncorrelated multivariate kernel, it has the effect of eliminating correlations in the posterior and setting its standard deviation to the kernel bandwidth.

In this paper, we experimented with locally oversampling SIR-A in a hypercuboid in parameter space, bounded by ± 3 standard errors around the maximum likelihood estimate. We used a truncated kernel density estimate within this for $f(\theta|\mathbf{d}_i)$.

It should also be noted that many implementations of MLE for statistical applications use algorithms such as Broyden-Fletcher-Goldfarb-Shanno (BFGS), which, although they are faster than Bayesian sampling, invert an estimate of the Hessian matrix (and are hence $\mathcal{O}(p^2)$ in the parameters), because the inverse Hessian is used for asymptotic inference (15). A derivative-free optimisation algorithm, such as Nelder-Mead, may be preferable if the inverse Hessian is not required (16).

The computational complexity of SIR is $\mathcal{O}(n_b), \mathcal{O}(p), \mathcal{O}(m_{local} \leq m)$, but has unknown costs of maximum likelihood estimation, local oversampling, and a relatively small cost of sampling pseudo-random numbers to generate posterior draws.

2.3 Kudzu algorithm

Ram and Gray proposed density estimation trees (DETs) for high-dimensional densities (12). Using the same algorithm as classification and regression trees (17), but with a different loss function, DETs predict the density instead of a dependent variable. By exhaustively partitioning the bounding box (smallest hypercuboid containing all prior draws) in parameter space, a density estimate can be produced for any proposed parameter vector. This is in contrast to KDE, where gaps between kernels may produce misleading low densities, and SIR, where very few prior draws (even after local oversampling) might lie at a high likelihood location. However, for a prior to be useful in Bayesian samplers like HMC, the density estimate must be smooth, yet DET returns a sum of non-overlapping uniform distributions.

We therefore replace each discontinuous edge of each leaf with a smooth ramp function, which is sometimes called convolution. Anderlini recently investigated a similar, but not smooth, convolution in the context of high-energy physics, where data are so plentiful that only density estimates can be stored (18).

In this paper, we used the logistic function:

$$g(x) = \frac{e^x}{e^x + 1} = \frac{1}{e^{-x} + 1} \tag{6}$$

x is the midpoint and inflection point of the function, where $g(x) = 0.5$. This is the cumulative density function of the logistic distribution, and its derivative, the probability density function, is $\frac{d}{dx}g(x) = g(x)g(-x)$. So, the logistic distribution is the product of two logistic ramps, facing in opposite directions and sharing a common midpoint.

We can also form a smooth univariate density for a leaf as the product

of two ramp functions with different midpoints. In one dimension, θ is the parameter, μ_b the desired midpoint on the bottom (lower) edge, and μ_t likewise on the top edge. The desired bandwidth (steepness of the ramp) is σ, which is common to bottom and top. The bottom logistic function is $g(\frac{\theta-\mu_b}{\sigma})$ and there is a reversed $g(\frac{\mu_t-\theta}{\sigma})$ at the top. Each leaf has a DET-estimated joint density $f_{DET}(\ell)$, which implies a marginal density for the jth dimension of:

$$f_{DET}(\ell, j) = f_{DET}(\ell) \prod_{1 \leq k \leq p, k \neq j} (\mu_{tk} - \mu_{bk}) \tag{7}$$

where μ_{tk} indicates the top edge in the kth dimension. The univariate convolved density is:

$$f_{kudzu}(\theta, \ell, \sigma, j) = \frac{f_{DET}(\ell, j)}{\left(e^{\frac{\mu_b-\theta}{\sigma}} + 1\right)\left(e^{\frac{\theta-\mu_t}{\sigma}} + 1\right)} \tag{8}$$

In general, DET might not place leaf edges at locations where the density is halfway between the DET-predicted densities of neighbouring leaves, and so the midpoints of the ramps might have to be translated by some distances δ_b and δ_t to obtain accurate prior variances and covariances:

$$f_{kudzu}(\theta, \ell, \sigma, \delta_b, \delta_t, j) = \frac{f_{DET}(\ell, j)}{\left(1 + e^{\frac{\mu_b+\delta_b-\theta}{\sigma}}\right)\left(1 + e^{\frac{\theta-\mu_t-\delta_t}{\sigma}}\right)} \tag{9}$$

A smoothed leaf will only return a true density from Equation 9 (such that the sum of all leaves' densities integrates to 1) if all δs are zero. Otherwise, the integral must be used to scale the density function correctly. In

one dimension:

$$
\begin{aligned}
&F_{kudzu}(\theta, \ell, \sigma, \mu_t, \delta_t, \mu_b, \delta_b) \\
&= \int f_{kudzu}(\theta, \ell, \sigma, \delta_b, \delta_t) d\theta \\
&= f_{DET}(\ell, j) \frac{\mathrm{e}^{1/\sigma} \ln\left(\mathrm{e}^{\frac{\mu_b+\delta_b-\theta}{\sigma}}+1\right) - \mathrm{e}^{1/\sigma} \ln\left(\mathrm{e}^{\frac{\mu_t+\delta_t-\theta}{\sigma}}+1\right)}{\left(\mathrm{e}^{\frac{1}{\sigma}}-1\right)/\sigma} + f_{DET}(\ell, j) \\
&= f_{DET}(\ell, j) \sigma \frac{e^{1/\sigma}}{e^{1/\sigma}-1} \ln\left(\frac{e^{\frac{\mu_b+\delta_b-\theta}{\sigma}}+1}{e^{\frac{\mu_t+\delta_t-\theta}{\sigma}}+1}\right) + f_{DET}(\ell, j)
\end{aligned}
\tag{10}
$$

The penultimate line of Equation 10 is preferable for computation, while the last line links to Equations 11 and 12.

Because the logistic ramp becomes very close to zero and one at $\pm 6\sigma$ from the midpoint, we evaluated the definite integrals over this range. The final term in Equation 10, $+f_{DET}(\ell, j)$, which is the constant of integration, cancels out. Let $w_\ell = \mu_t + \delta_t - \mu_b - \delta_b$ be the width of the leaf, and $z_\ell = w_\ell/\sigma$ the width in units of σ in the current dimension.

Equivalently, set $\sigma = 1, \mu_b + \delta_b = 0, \mu_t + \delta_t = z_\ell$ and evaluate the integral over a generic distance $\pm\phi\sigma$ from the midpoints: between $\theta = -\phi$ and $\theta = z_\ell + \phi$.

$$
\begin{aligned}
&\Big[F_{kudzu}(\theta, \ell, \phi, z_\ell, w_\ell)\Big]_{\theta=-\phi}^{\theta=z_\ell+\phi} \\
&= f_{DET}(\ell, j) \sigma \frac{e^{\frac{z_\ell}{w_\ell}}}{e^{\frac{z_\ell}{w_\ell}}-1} \ln\left(\frac{e^{0-z_\ell-\phi}+1}{e^{z_\ell-z_\ell-\phi}+1}\right) - \ln\left(\frac{e^{0+\phi}+1}{e^{z_\ell+\phi}+1}\right) \\
&= f_{DET}(\ell, j) \sigma \frac{e^{\frac{z_\ell}{w_\ell}}}{e^{\frac{z_\ell}{w_\ell}}-1} \ln\left(\frac{(e^{z_\ell+\phi}+1)(e^{-z_\ell-\phi}+1)}{(e^{\phi}+1)(e^{-\phi}+1)}\right)
\end{aligned}
\tag{11}
$$

When there is a plateau between the ramps (practically, $\mu_t + \delta_t - \mu_b - \delta_b > 12\sigma$), the ramps can be regarded as not interacting with each other and

the integral simplifies either by appeal to the symmetry of $g(\cdot)$ and the rectangular DET density or, more formally, by limits:

$$\lim_{z_\ell \to \infty, \phi \to \infty} \Big[F_{kudzu}(\theta, \ell, \phi, z_\ell, w_\ell) \Big]_{\theta=-\phi}^{\theta=z_\ell+\phi} = f_{DET}(\ell, j) w_\ell \tag{12}$$

In p dimensions, we can simply multiply all univariate kudzu densities together for a leaf, because they are independent, and likewise for their integrals. Then, the complete kudzu density is the sum of the leaves' densities. In Equation 13, the numerator is drawn from Equation 9 and the denominator from either Equation 11 or Equation 12. The $f_{DET}(\ell, j)$ in each will cancel out.

$$\begin{aligned} &f_{kudzu}(\theta | \sigma, \mu_{\mathbf{t}}, \delta_{\mathbf{t}}, \mu_{\mathbf{b}}, \delta_{\mathbf{b}}, \phi) \\ &= \sum_{\ell=1}^{L} f_{DET}(\ell) \prod_{j=1}^{p} \frac{f_{kudzu}(\theta, \ell, \sigma, \delta_b, \delta_t)}{\Big[F_{kudzu}(\theta, \ell, \sigma, \mu_t, \delta_t, \mu_b, \delta_b) \Big]_{\theta=\mu_b+\delta_b-\phi\sigma}^{\theta=\mu_t+\delta_t+\phi\sigma}} \end{aligned} \tag{13}$$

We found that δ_b and δ_t should move the ramp midpoints towards leaves of higher density in order to produce accurate variances. We calculated the centre of each leaf $\mathbf{c}_\ell$ and used the centre of the highest-density leaf as an approximate mode $\mathbf{c}_{\ell^*}$, where $\ell^* = \underset{\ell}{\operatorname{argmax}} f_{DET}(\ell)$. We then set the δs to move edges toward $\mathbf{c}_{\ell^*}$, proportionate to the sine of the smallest angle between each edge and $\mathbf{c}_{\ell^*}$.

Experimentation showed that DET is best suited to uncorrelated samples, so we transform the prior sample $\mathbf{d}_i$ by principal components analysis: centring by subtracting a vector of the means $\mathbf{m}_{d_i}$, scaling by dividing (elementwise, also known as Hadamard division) by a vector of the standard deviations $\mathbf{s}_{d_i}$, and rotating by premultiplication with the eigenvectors $\mathbf{R}_{d_i}$ of the correlation matrix. Finally, we rescale the principal component scores

Algorithm 6: Kudzu algorithm, pre-sampling

Input : $m \times p$ matrix of prior draws $\mathbf{d}_i$,
proposed parameter vector θ,
edge translation as a proportion of a leaf width δ,
logistic ramp bandwidth σ

Output: estimated prior density $f(\theta|\mathbf{d}_i)$

`// normalise, rotate and normalise again`

$\mathbf{m}_{d_i} = \mathbf{s}_{d_i} = \mathbf{s}_{PCA_i} = [\,]$

$\mathbf{d}_{norm_i} = \mathbf{d}_i$ `// this matrix will be altered`

for $j \leftarrow 1$ **to** p **do**

  $\mathbf{m}_{d_i}[j] = \texttt{mean}(\mathbf{d}_i[,j])$

  $\mathbf{s}_{d_i}[j] = \texttt{sd}(\mathbf{d}_i[,j])$

  $\mathbf{d}_{norm_i}[,j] = (\mathbf{d}_i[,j] - \mathbf{m}_{d_i}[j])/\mathbf{s}_{d_i}[j]$

end for

$\mathbf{R}_{d_i} = \texttt{eigenvectors}(\mathbf{d}_{norm_i})$

$\mathbf{d}_{PCA_i} = \mathbf{d}_{norm_i}\mathbf{R}_{d_i}$

for $j \leftarrow 1$ **to** p **do**

  $\mathbf{s}_{PCA_i}[j] = \texttt{sd}(\mathbf{d}_{PCA_i}[,j])$

  $\mathbf{d}_{PCA_i}[,j] = \mathbf{d}_{PCA_i}[,j]/\mathbf{s}_{PCA_i}[j]$

end for

`// get list describing` L `leaves from DET of prior sample`

$\{tops, bottoms, densities\} = \texttt{DET}(\mathbf{d}_{PCA_i})$

`// 'tops' and 'bottoms' are` $L \times p$ `matrices, densities is an` L`-vector`

$\mathbf{c} = tops - bottoms/2$

$mode = \mathbf{c}[\texttt{which}(densities == \texttt{max}(densities)),]$

`// delta-translate edges`

for $\ell \leftarrow 1$ **to** L **do**

  for $j \leftarrow 1$ **to** p **do**

    $edge = \mathbf{c}[\ell,]$

    $edge[j] = tops[\ell, j]$

    $sineEdge = (mode[j] - edge[j])/\|(mode - edge)\|_2$

    $tops[\ell, j] += sineEdge\delta(tops[\ell, j] - bottoms[\ell, j])$

    $edge[j] = bottoms[\ell, j]$

    $sineEdge = (mode[j] - edge[j])/\|(mode - edge)\|_2$

    $bottoms[\ell, j] += sineEdge\delta(tops[\ell, j] - bottoms[\ell, j])$

  end for

end for

return $\{tops, bottoms, densities, \sigma, \mathbf{s}_{PCA_i}, \mathbf{m}_{d_i}, \mathbf{s}_{d_i}, \mathbf{R}_{d_i}^{-1}, L\}$

by dividing (elementwise) by their standard deviation vector $\mathbf{s}_{PCA_i}$. This final step is not part of principal components analysis but assists Stan and other samplers by having all dimensions on similar scales.

$$\mathbf{d}_{PCA_i} = (\mathbf{R}_{d_i}((\mathbf{d}_i - \mathbf{m}_{d_i}) \circ \mathbf{s}_{d_i}^{\circ -1})) \circ \mathbf{s}_{PCA_i}^{\circ -1} \tag{14}$$

$\mathbf{d}_{PCA_i}$ is supplied to DET, then we find the kudzu δ translations, and pass to the sampler the translated edges $(\mu_b + \delta_b, \mu_t + \delta_t)$, bandwidths (σ) and DET densities $(f_{DET}(\ell))$ to the sampler, along with data for the likelihood $(\mathbf{m}_{d_i})$, the vectors $\mathbf{s}_{PCA_i}, \mathbf{m}_{d_i}, \mathbf{s}_{d_i}$, and inverse of the matrix $\mathbf{R}_{d_i}$. The sampler computes the log-prior density on the transformed parameter space according to Equation 13, back-transforms to the model parameter space, and combines the proposed model parameters with the data for the log-likelihood.

One remaining challenge is that the likelihood may require the posterior to move to outside the prior sample bounding box, which could be a region of near-zero and almost flat prior density. To provide some density and gradient outside the DET $(\pm\phi\sigma)$, we mix the kudzu density with a small proportion (we used 0.02) of a multivariate normal probability density function, with standard deviation twice that of the largest principal component standard deviation.

This algorithm is named after the notorious Kudzu vine (*Pueraria lobata*), which grows rapidly, smoothing over the shapes of trees.

Within the sampler, equation 13 is calculated for any proposed parameter vector θ. This involves $2Lp$ logistic ramps, and because L changes with p, the computational cost of the prior is not precisely known. In the experiments in this paper, we found L was approximately linearly proportionate to p, making kudzu approximately $\mathcal{O}(p^2)$ in the parameters, and $\mathcal{O}(n_b)$, with

fixed cost from m via the DET algorithm, which is very fast.

We used Ram and Gray's DETs as implemented in the MLPACK C++ library, via the command line tool `mlpack_det` (12)(19). We adopted the default settings.

We are not aware of any previous attempt to use DETs for Bayesian updating, nor any proposed smooth convolutions of DETs.

2.4 Examining distortion on illustrative priors without likelihood

This addresses objective 1 of this paper. To understand the distortions that the methods might introduce, and how this is affected by KDE bandwidth, we updated three exemplar bivariate prior distributions ten times without any likelihood. Initial prior samples of 4000 draws were generated from:

1. a multivariate normal $\begin{bmatrix} x \\ y \end{bmatrix} \sim MVN\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1 \end{bmatrix}\right)$,
2. a banana distribution, conditionally specified as:

$$
\begin{aligned}
u &\sim N(0,1) \\
v &\sim N(0,0.7) \\
w &= -u^2 + v \\
\begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} cos(0.25) & -sin(0.25) \\ sin(0.25) & cos(0.25) \end{bmatrix} \begin{bmatrix} u \\ w \end{bmatrix}
\end{aligned}
$$

3. a bimodal "lasso" distribution typical of regularized regression posteriors (20):

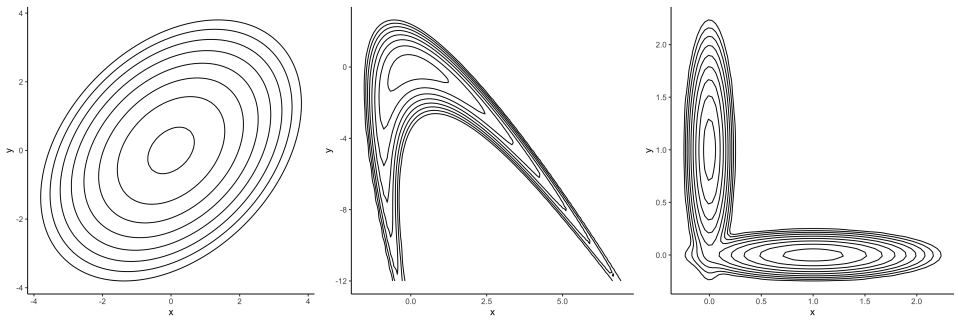

Figure 1: Illustrative contour plots of (left to right) the multivariate normal, banana and lasso distributions.

$$\begin{bmatrix} x \\ y \end{bmatrix} \sim \frac{1}{2} MVN\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0.3 & 0 \\ 0 & 0.06 \end{bmatrix} \right) + \frac{1}{2} MVN\left(\begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0.06 & 0 \\ 0 & 0.3 \end{bmatrix} \right)$$

The multivariate normal is an easy base case because of its convexity; it is widespread in applied analyses as an asymptotic sampling distribution for many statistics. The banana distribution is widely used as a challenging test case in Bayesian methodology because of the lacuna between the two arms, the skew in the marginal distributions, and the diagonal alignment. The "lasso" distribution is a common example of multimodal posteriors that arises in everyday practice, such as after LASSO or other regularized regressions.

The expected variation in these distributions attributable to sampling error was quantified by generating a further 1000 random samples, and comparing each with the original using the 20-sliced 2-Wasserstein distance (21). This is a scalar statistic summarising the difference between two multivariate distributions.

Ideally, the moments of the distribution would be preserved across up-

dates, while still allowing individual draws to move. This would appear as random variation of the 20-sliced 2-Wasserstein distances within their sampling distributions. Wasserstein distances that increasingly exceed the upper bound of the sampling distribution would indicate accumulating distortion, while Wasserstein distances that do not randomly move throughout the sampling distribution, or are consistently below its lower bound, indicate inadequate smoothing.

This was replicated for six KDE bandwidths (the Sheather-Jones vector σ_{SJ}, $\frac{1}{4}\sigma_{SJ}$, $\frac{1}{2}\sigma_{SJ}$, $2\sigma_{SJ}$, $4\sigma_{SJ}$, and finally the multimodality expansion of σ_{SJ}). The resulting posterior samples were plotted, described qualitatively, and their 20-sliced 2-Wasserstein distances were plotted.

All code and visualisations are available online(22). This part of the study was run on a MacBook Pro laptop (4-core CPU, 2.7GHz, 8GB RAM) in R and `rstan`.

2.5 Evaluating long-run performance by simulation study

This addresses objective 2 of this paper. Datasets of 2000 observations were simulated for a logistic regression model:

$$
\begin{aligned}
x_1 &\sim N(0,1) \\
\epsilon &\sim N(0,0.08) \\
x_2 &= \frac{x_1}{1.8} + \epsilon \\
\pi &= \frac{e^{(2+0.5x_1-1.5x_2)}}{1+e^{(2+0.5x_1-1.5x_2)}} \\
y &\sim Binomial(\pi)
\end{aligned}
$$

There are thus three parameters, with true population values: $\beta_0 =$

$2, \beta_1 = 0.5, \beta_2 = -1.5$.

For each dataset, parametric updating (with the correct forms of prior / posterior distribution), the KDE algorithms in HMC, and unbounded KDE in SIR-A, were fitted with ten batches of data of 200 observations each. Four thousand posterior draws were retained each time and passed to the next update as prior draws. The orginal priors were: $\beta_i \sim N(0, 4); \quad i \in \{0, 1, 2\}$. Four hundred simulations were run because an initial experiment using 1000 simulations showed a standard deviation across simulations of about 0.2 in each of the parameters. We aim to detect shifts of 0.01 in the location of the posteriors as a primary objective, and so have 2 SDs less than this.

KDE algorithms in HMC were fitted in Stan software (4), and KDE in SIR ran in an R script(23). Kudzu was not evaluated in this simulation study with 3 parameters, as it is intended for high-dimensional applications. All code is available online (22). This part of the study ran in a cloud computing facility on a 16-core Linux virtual machine with 32GB RAM, running four R instances in parallel, each running `rstan` on four cores.

2.6 Exploring feasibility of KDE in SIR and kudzu: New York taxis case study

This addresses objective 3 of this paper. We analysed a large real-life dataset, with the aim of learning about feasibility and practical implementation. We evaluated two algorithms: unbounded KDE in SIR-A with local oversampling based on the MLE, and kudzu. Data on New York City yellow taxi journeys in 2013 were used, as published online by Whong(24). We considered all journeys in the first week of 2013 after basic data cleaning ($n = 2,657,081$), with the logarithm of the total cost of the journey as the dependent variable, and sought to fit a linear regression model with the

following independent variables:

- 19 indicator variables for 20 "neighbourhoods", dividing Manhattan below 110th Street into a grid aligned to Central Park with nine divisions in a south-south-west to north-north-east direction and one division perpendicular to this
- 7 indicator variables for 7 days of the week
- 23 indicator variables for 24 hours of the day

Our goal was to provide a daily update to the model parameters using each day's journeys. We also fitted all-data analyses after 1, 3, 5 and 7 January, and compared the posterior means, standard deviations and correlations between these and the SIR/kudzu updates

Initially, the model was fitted with $N(0, 2)$ priors for all parameters in Stan, using the first 30,000 journeys from each day. Two thousand posterior draws were retained from each of two chains, after discarding the first 1000 of each as warm-up (4). This provided a pump-priming prior sample.

In the SIR implementation, we used local oversampling, within a hypercuboid of 2 standard errors around the maximum likelihood estimate, with the aim of avoiding degeneracy. For kudzu, we examined standard deviations of the posteriors from a kudzu update after 1 January, and an all-data analysis, and chose $\delta = w_\ell/3$ and $\sigma_\ell = w_\ell/18$ to correct for variance inflation.

All code is available online(22). This part of the study was run in a cloud computing facility on two Linux virtual machines: one with 4 cores and 8GB RAM running R and `rstan` for the SIR and kudzu updates, and one with 8 cores and 64GB RAM using R and `cmdstanr` for the all-data

comparison analyses. `cmdstanr` allows for larger analyses by running Stan outside R and writing results to disk as they emerge.

3 Results

3.1 Examining distortion on illustrative priors without likelihood

Wasserstein distances show evidence of increasing distortion with successive updates, regardless of method and distribution (visualisations online (22)). This is not necessarily a problem, as we are effectively updating with a uniformly distributed likelihood, and would expect the distribution to spread out. In the case of the lasso distribution, two particular challenges are that one of the modes can disappear, and that the bandwidths that are suitable for one mode are wholly unsuitable for the other, yet our methods must select one bandwidth per parameter.

Examination of scatterplots of the successive posterior draws explains the results further (Figure 2).

When the bandwidth is small, as with one-quarter of Sheather-Jones, the unbounded kernel method immediately shrinks the distributions, and within two updates they have degenerated to a few points. The hypercuboid method degenerates immediately. The bounded kernel method survives for two or three updates before serious distortion happens. SIR remains almost unchanged between all updates, including preserving random fluctuations in density seen in the original prior draws.

Imagining the extreme case of a zero bandwidth clarifies this phenomenon. The three methods using Hamiltonian Monte Carlo will return zero posterior densities unless the proposed θ_j happens to coincide with a prior draw, so

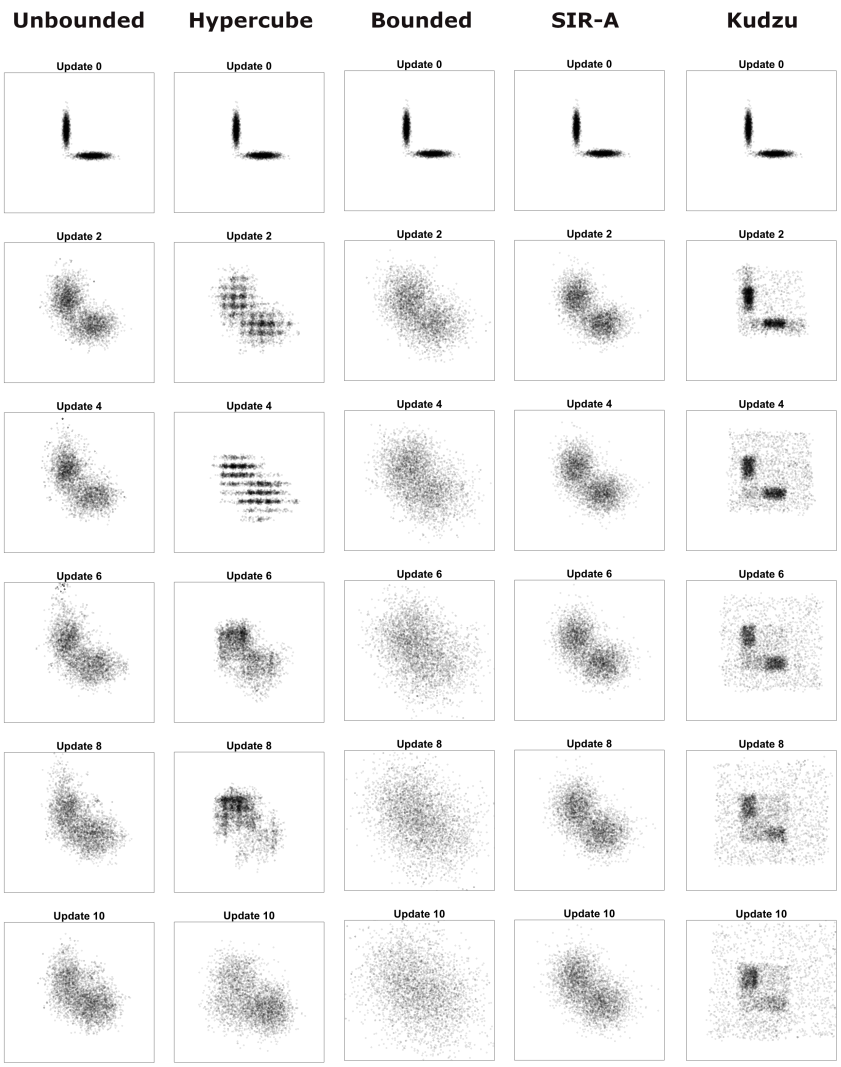

Figure 2: Prior-only updates of the lasso distribution

the posterior will be a sample of points from the prior. With no gradient to inform the Hamiltonian between prior draws, HMC will fail to fully explore the posterior. When this is used in the next update, it will contract further. SIR, on the hand, simply replicates the prior draws with all posterior draws being in identical locations; the distribution cannot drift to follow the likelihood.

When the bandwidth is large, as with double Sheather-Jones, the unbounded, bounded and SIR methods all show steady inflation of variances and deflation of correlations; somewhere between one and four updates are possible before the distribution is unrecognisable. The hypercuboid method immediately causes severe distortion. SIR has the best fidelity to the original prior distribution.

Overall, the multimodality expansion achieves the best results for KDE in the MVN and banana distributions, while all KDE bandwidths inflated marginal variances in the lasso distribution for each mode's respective minor axis. This demonstrates that a single bandwidth cannot always be applied across one parameter without distortion. Often, these inherently multimodal posteriors can be anticipated based on the type of model, such as regularized regression or artificial neural networks. In these settings, local bandwidth algorithms may help; this is not evaluated in this paper(25).

Kudzu shows artefacts of the rectangular leaves in the DET. It performs best on the lasso distribution but no better than KDE on the others. Importantly, the lasso has two components, each of which is aligned with the axes. As with any trees, kudzu will perform best on uncorrelated data.

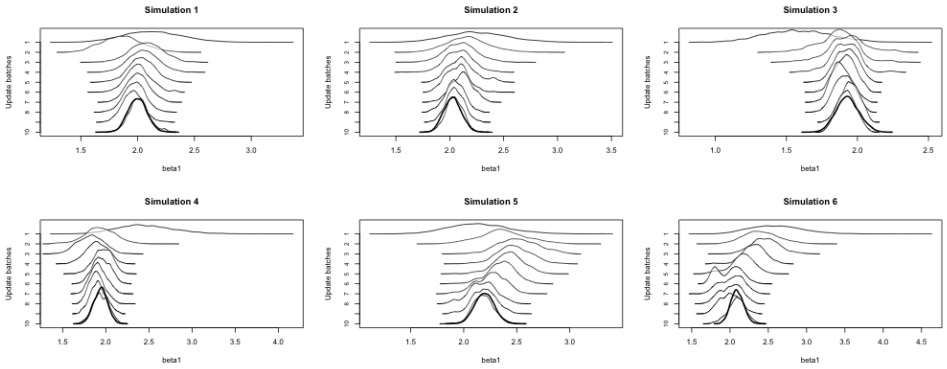

Figure 3: Posterior densities of β_0 from the first six simulations of unbounded KDE (Algorithm 3). Ten updates are arranged from top to bottom. The thicker density superimposed on the tenth update is the posterior sample obtained from analysing all data together, which may be regarded as ground truth for comparison.

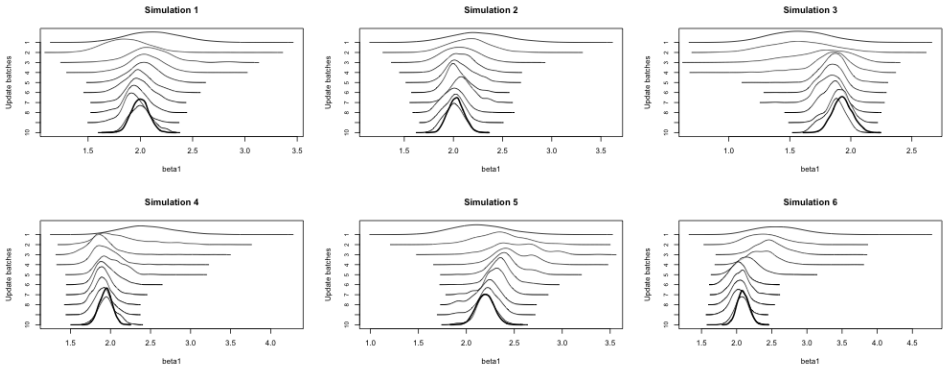

Figure 4: Posterior densities of β_0 from the first six simulations under unbounded KDE in SIR-A (Algorithm 2). Ten updates are arranged from top to bottom. The thicker density superimposed on the tenth update is the posterior sample obtained from analysing all data together, which may be regarded as ground truth for comparison.

3.2 Evaluating long-run performance by simulation study

The long-run properties of bias, coverage and mean posterior standard deviation are shown in Table 1. Biases were close to the threshold of 0.01 which this study aimed to detect, and were almost identical between algorithms, except that SIR was slightly larger. Unbounded KDE achieved the best coverage, because of its slightly smaller posterior standard deviations. The joint posterior distribution is convex and two-dimensional.

TABLE 1: Bias, coverage, and posterior standard deviation in the simulation study

		β_0	β_1	β_2
Unbounded KDE	Bias	0.012	0.010	0.015
	Coverage	93.8%	95.8%	95.8%
	Posterior SD	0.085	0.086	0.103
Hypercuboid KDE	Bias	0.013	0.010	0.015
	Coverage	97.5%	97.3%	97.8%
	Posterior SD	0.101	0.102	0.121
Bounded KDE	Bias	0.012	0.010	0.015
	Coverage	96.5%	96.8%	97.5%
	Posterior SD	0.094	0.095	0.114
SIR-A	Bias	0.015	0.011	0.017
	Coverage	97.5%	97.0%	97.0%
	Posterior SD	0.101	0.103	0.122

The mean times taken for nine updates (the first batch having been analysed with the original parametric priors) were: 512 seconds with unbounded KDE, 70 seconds with hypercuboid KDE, 713 seconds with bounded KDE, and 6 seconds with SIR-A. Plots of all combinations of outputs are available online (22).

3.3 Exploring feasibility of KDE in SIR and kudzu: New York taxis case study

The model was fitted in Stan to the first 30,000 journeys from each day to generate the pump-priming prior sample, which took 4 hours and 13 minutes, using two chains, discarding the first 1000 draws from each as warm-up and passing 4000 draws to the first update.

When using SIR for the 1 January update, only one prior draw generated posterior draws, and local oversampling around the MLE did not resolve this, increasing the prior draws to 100,000. Subsequent updates drew from 2, 4, 3, 847, 2814 and 4000 prior draws each, but the distortion introduced at 1 January caused persistent loss of correlations. This problem is particularly evident at the first few updates because the curvature of the likelihood ($n > 300,000$) exceeds that of the prior density ($n = 210,000$), and the high dimensionality of the parameter space.

Table 2 shows the time (in hours) taken for kudzu updates, the number of DET leaves (L), and compares this with all-data analyses every two days. Times are means across all chains, for 2500 warmup and 1000 sampling draws. Daily model updates using all data each time would not be possible after 6 January, as all-data time exceeds 24 hours.

Kudzu produced posterior means and correlations that closely matched those from all-data analyses (more graphics are available online (22)), but variances increased, though not monotonically with updates.

TABLE 2: Daily updates by kudzu, compared to all-data analyses

Update	n_b	kudzu time	all-data time	L	median relative bias	median relative efficiency
1 Jan	316,446	4.3	5.2	66	1.02	1.19
2 Jan	315,907	5.8	—	57	—	—
3 Jan	365,437	7.2	10.7	49	0.99	1.35
4 Jan	401,651	7.2	—	59	—	—
5 Jan	392,836	8.2	13.5	54	0.98	1.76
6 Jan	321,985	8.8	—	69	—	—
7 Jan	332,819	7.1	31.1	62	1.00	1.27

Several feasibility lessons can be drawn from this exercise:

- Unlike MCMC-based approaches, SIR can only evaluate the prior draws provided, so if the prior has lower curvatures ($\frac{\partial^2 f(\cdot)}{\partial \theta^2}$) than the likelihood, and/or there are a moderately large number of parameters, there may be degeneracy. This is most likely to affect early batches.

- Local oversampling is limited in high-dimensional parameter spaces. The number of samples needed simply to occupy all orthants around the MLE or prior draw with highest likelihood is 2^p (in the taxi application, $2^{50} > 10^{15}$). Also, we are sampling from a hypercuboid in p-dimensional space, knowing that there will be a region (convex in many statistical models) within this which would produce posterior draws. The proportion of the hypercuboid which is wasted by examining empty corners increases rapidly with p.

- Although we can theoretically use posterior subtraction for Bayesian windowing, it may accumulate noise across multiple kernels. If the updating algorithm is fast enough, we should prefer to keep a collection of sliding windows(1). These can be fitted in parallel, requiring only more hardware.

- Manual tuning and monitoring will be essential for all methods.

4 Discussion

The investigations in this paper show that density estimates of prior distributions show some promise as a way of obtaining rapid updates to Bayesian models in settings where data accumulates quickly. However, the results are sensitive to the choice of bandwidths and δ-translation, especially in multimodal or non-convex distributions. Distortion can accumulate over multiple updates. Analysts must therefore monitor the fidelity of an updating method to a particular prior and likelihood before adopting it in practice. Periodic checkpointing, by running full-data analyses and adopting the resulting posterior draws, will also help.

In this paper, we have evaluated a simple application in Hamiltonian Monte Carlo (HMC; Stan software) using unbounded multivariate kernels on the prior draws, and explored two variants that might have reduced the computational burden: coarsening the draws to a grid of hypercuboid centres, and evaluating only neighbouring draws with a bounded kernel distribution. The two variants have some problems: the hypercuboids are only helpful in problems with very few parameters and coarse grids, and the bounded kernels require computer-intensive identification of neighbours.

A quite different sampling approach is SIR, which was much faster than HMC and of comparable, and sometimes superior, fidelity, though it introduces new problems of degeneracy. This paper implemented a simple form of local oversampling around the maximum likelihood estimate, which did not perform well, largely because of the problem of sparsity in high-dimensional spaces.

If any overall recommendation can be made from this study, it is that low-p applications can benefit from unbounded KDE in MCMC or SIR (with Sheather-Jones bandwidths or multimodality expansion, and regular check-

pointing). For higher-p, kudzu should be considered. Each update should be monitored by an experienced analyst.

The analyses in this paper have some limitations. We have examined only continuous, unbounded parameters.

In this simulation and case study, priors and likelihoods overlap considerably. When this is not the case, the tails of the kernel density will become crucial. We also only considered updating with one method exclusively. There may be merit in a strategy that mixes batch sizes and algorithms (such as kudzu in early updates, then SIR).

This paper also raises the prospect of non-parametric windowing, to remove the likelihood contributions of older batches of data, analogous to a weighted moving average. This can account for the problem of data drift — changes in the population with time — in arbitrarily complex Bayesian models. The theory for this, and what inferences from such windowed posteriors mean, should be developed further.

Further research to refine these methods should focus on selection of kernels and bandwidths for KDE, DET pruning and δ-translation for kudzu. There may be advantages to kudzu over ensembles of DETs. Better local oversampling strategies may extend the utility of SIR into higher-p; isotropic and anisotropic transformations of parameter space may be useful prior to oversampling, or explorations of hulls (not necessarily convex) of the likelihood surfaces and oversampling within these.

One problem yet to be resolved is an efficient way of growing the hull in non-convex or multimodal likelihoods, where the region of interest will not be bounded by neighbours of the maximum-likelihood prior draw. It may be possible to use piecewise deterministic Markov processes (PDMPs), for example the zig-zag sampler (26), on the likelihood, starting from the

maximum-likelihood prior draw, storing the reflection points, from which a hull can be approximated, although it is not clear what advantage this would offer over using PDMPs on the prior kernel density and the likelihood together. Another refinement may be to explore the boundary between zero and one posterior draw in the manner of the hug-and-hop algorithm (27).

Datasets of this size, accumulating in real time, are a common challenge in contemporary data science. Although Bayesian models have the benefit of flexible specification, inference with uncertainty around estimates, and ease of interpretation, analysts often have to use machine learning "black boxes" instead because of the delay that Bayesian analysis would incur. We believe that full Bayesian modelling, with quantification of uncertainty and probability-based interpretation, is preferable and needs to be made available to settings where data arrive rapidly and models must be updated to account for this.

References

[1] Akidau T. *Streaming 101: the world beyond batch* O'Reilly (2015). https://www.oreilly.com/radar/the-world-beyond-batch-streaming-101/

[2] Green PJ, Łatuszyński K, Pereyra M, Robert CP. Bayesian Computation: A Summary of the Current State, and Samples Backwards and Forwards. *Statistics and Computing* (2015);25:835–862.

[3] Fearnhead P, Bierkens J, Pollock M, Roberts GO. Piecewise Deterministic Markov Processes for Continuous-Time Monte Carlo. *Statistical Science* (2018);33(3):386–412.

[4] Stan Development Team. Stan Modeling Language Users Guide and

Reference Manual, `rstan` version 2.21.2 (2021) https://mc-stan.org (accessed 25 January 2021)

[5] Neal RM. MCMC Using Hamiltonian Dynamics. In *Handbook of Markov Chain Monte Carlo*, editors Brooks S, Gelman A, Jones GJ, Meng XL. CRC Press, Boca Raton, 2011.

[6] Givens GH, Hoeting JA. *Computational Statistics*, 2nd edition, section 6.3.1. Wiley, Hoboken, 2013.

[7] Neufeld C., Deutsch CV. Data Integration with Non-Parametric Bayesian Updating. *Centre for Computational Geostatistics Report 8* (2006): 105. University of Alberta, Canada. http://www.ccgalberta.com/ccgresources/report08/2006-105-non_parametric_bu.pdf

[8] Zambom AZ, Dias R. A Review of Kernel Density Estimation with Applications to Econometrics. *International Econometric Review* (2013);5(1):20–42.

[9] Papamakarios G, Pavlakou T, Murray I. Masked Autoregressive Flow for Density Estimation. *Advances in Neural Information Processing Systems* (2017);30:2335–2344. https://www.research.ed.ac.uk/en/publications/masked-autoregressive-flow-for-density-estimation

[10] Liu Q, Xu J, Jiang R, Wong WH. Density Estimation Using Deep Generative Neural Networks. *Proceedings of the National Academy of Sciences* (2021);118(15):e2101344118. https://doi.org/10.1073/pnas.2101344118

[11] Botev ZI, Grotowski JF, Kroese DP. Kernel Density Estimation Via Diffusion. *The Annals of Statistics* (2010);38(5):2916–2957.

https://arxiv.org/pdf/1011.2602.pdf

[12] Ram P, Gray AG. Density Estimation Trees. *KDD '11: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining* (2011): 627–635. https://doi.org/10.1145/2020408.2020507

[13] Sheather SJ., Jones MC. A Reliable Data-Based Bandwidth Selection Method for Kernel Density Estimation. *Journal of the Royal Statistical Society, Series B* 1991;53(3):683–690.

[14] Hartigan JA, Hartigan PM. The Dip Test of Unimodality. *Annals of Statistics* (1985);13:70–84.

[15] Press WH, Teukolsky SA, Vetterling WT, Flannery BP. *Numerical Recipes*, 3rd edition, section 10.9. Cambridge University Press, Cambridge, 2007.

[16] Givens GH, Hoeting JA. *Computational Statistics*, 2nd edition, section 2.2.4. Wiley, Hoboken, 2013.

[17] Hastie T, Tibshirani R, Friedman J. *The Elements of Statistical Learning*, 2nd edition, section 9.2. Springer, New York, 2009.

[18] Anderlini L. Density Estimation Trees as Fast Non-Parametric Modelling Tools. *17th International workshop on Advanced Computing and Analysis Techniques in physics research* (ACAT2016), Valparaiso, Chile, 2016. https://indico.cern.ch/event/397113/contributions/1837849/attachments/1213965/1771772/main.pdf

[19] Curtin RR, Edel M, Lozhnikov M, Mentekidis Y, Ghaisas S, Zhang S. mlpack 3: A Fast, Flexible Machine Learning Library. *Journal of Open Source Software* (2018);3:26.

https://mlpack.org/static/pub/2018mlpack.pdf

[20] Gelman A, Carlin JB, Stern HS, Dunson DB, Vehtari A, Rubin DB. *Bayesian Data Analysis*, 3rd edition, section 14.6. CRC Press, Boca Raton, 2014.

[21] Kolouri S, Nadjahi K, Şimşekli U, Badeau R, Rohde GK. Generalized Sliced Wasserstein Distances. *33rd Conference on Neural Information Processing Systems* (NeurIPS 2019), Vancouver, Canada. https://proceedings.neurips.cc/paper/2019/file/f0935e4cd5920aa6c7c996a5ee53a70f-Paper.pdf

[22] Grant RL. *Non-parametric Bayesian updating*, Personal code repository (2021). http://robertgrantstats.co.uk/code/non-parametric-bayes-updating.html

[23] R Core Team. R: A language and environment for statistical computing, version 4.0.3 (2020). R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/ (accessed 25 January 2021)

[24] Whong C. *FOILing NYC's Taxi Trip Data*, Chris Whong blog (2014). https://chriswhong.com/open-data/foil_nyc_taxi/

[25] Breiman L, Meisel W, Purcell E. Variable Kernel Estimates of Multivariate Densities. *Technometrics* (1977);19(2):135–144.

[26] Bierkens J, Fearnhead P, Roberts G. The Zig-Zag process and super-efficient sampling for Bayesian analysis of big data. *Annals of Statistics* (2019);47(3):1288–1320.

[27] Ludkin M, Sherlock C. Hug and Hop: a discrete-time, non-reversible Markov chain Monte Carlo algorithm. Archived pre-print. ArXiv (2019). https://arxiv.org/abs/1907.13570